Investigation into Indexing XML Data Techniques

Alhadi Klaib, Joan Lu
Department of Informatics
University of Huddersfield
Huddersfield, UK

Abstract- *The rapid development of XML technology improves the WWW, since the XML data has many advantages and has become a common technology for transferring data cross the internet. Therefore, the objective of this research is to investigate and study the XML indexing techniques in terms of their structures. The main goal of this investigation is to identify the main limitations of these techniques and any other open issues. Furthermore, this research considers most common XML indexing techniques and performs a comparison between them. Subsequently, this work makes an argument to find out these limitations. To conclude, the main problem of all the XML indexing techniques is the trade-off between the size and the efficiency of the indexes. So, all the indexes become large in order to perform well, and none of them is suitable for all users' requirements. However, each one of these techniques has some advantages in somehow.*

Keywords: XML Data; indexing XML data techniques; indexing techniques; XML database.

1 Introduction

XML has become a common technology for the transformation of data across the WWW. It was recommended by the World Wide Web Consortium (W3C) in 1998, and has become the standard medium for data presentation and exchange over the WWW. Indexing plays a major role in enhancing XML data queries operation. The relational database is a robust and mature technology and thus more reliable compared with XML. However, the XML data has some advantages compared with the relational model, which are the following: 1) the structure of an XML document is integrated with the data, whereas the structure in a relational model is separate. Thus, it is better to use XML as a medium for transforming data on the Web; 2) XML data has the advantage of flexibility for querying languages, a feature that is not available in SQL; 3) XML data is flexible for adapting to the development of the data structures[1, 2].

1.1 Research questions: this research considers the following questions:

a) What are the main XML data indexing techniques and structures that have been developed and investigated?
b) What are the key results and findings on the XML indexing techniques and structures?
c) What problems, limitations, and challenges that this research area faces?

1.2 Research aims: Broadly speaking, the main goal of this research is to study XML indexing techniques and structures in order to ascertain the key advantages and disadvantages for each of these techniques. In more detail, the following points are the objectives:

a) Investigate the main indexing techniques and schemes.
b) Consider some common comparison criteria for these indexing techniques.
c) Compare these techniques and determine the limitations.
d) Identify the issues and open problems of XML indexing techniques.

1.3 Motivations: The motivation of this research is basically the significance of XML data for database management systems and web technology. Moreover, XML database indexing is a major factor in the efficiency and reliability of XML data technology. This indexing is also a critical demand nowadays due to the growth of XML data usage, XML database size, and the number of users during the last decade. Therefore, it is important to investigate the performance of XML indexing techniques. Thus, this research considers and studies the most popular XML indexing techniques in order to find out the advantages and limitations for each kind of them [3-5].

1.4 Paper organization: The remainder of the paper is structured as follows: Section 2 discusses the research method. The literature review is presented in Section 3. Section 4 discusses the results. Section 5 describes the discussion and the analysis. Section 6 discusses the conclusion, with future work presented in Section 7.

2 Research method

The research method used is a comparison review with a consideration of the research questions, identification of research area, selection process, comparison criteria, and evaluation. The research discusses these stages in the following:

2.1 Review Plan: the first step is to make a plan for the review. This plan considers the method and the process that will be applied. The goal of the study is to review the studied indexing XML data techniques and answer the research questions.

2.2 Identification of Research: a comprehensive and fair search is an important factor to obtain relevant materials such as articles and conference papers, and then to achieve a good review. The start of this search was to identify the search keywords and terms. Different keywords were used in order to obtain as many papers and other materials as possible. The table 1 shows examples of these keywords.

TABLE 1: SEARCH KEYWORDS AND TERMS

XML data	*Indexing XML*
▪ XML principles. ▪ XML data and database. ▪ XML data and semi-structured. ▪ XML Database. ▪ Query XML. ▪ Query optimization. ▪ XML applications. ▪ XML documents.	▪ Indexing XML techniques. ▪ XML indexes classification. ▪ XML Schemes. ▪ XML index structures. ▪ Evaluation of XML indexes.

All possible keywords and terms of the indexing and XML technologies were tried in the search. The Summon (Huddersfield University search engine) and Google Scholar are the most used search engines for the search. The following electronic databases and libraries are the most used resources: ACM Digital Library, Google Scholar database, Science Direct, Springer Link, and Huddersfield University library.

2.3 Selection process: after each search operation, a number of identified articles and papers were ignored since they were irrelevant or duplicate titles. To identify which articles to select, the selection process went through a number of stages as follows:

1- Check the title to find those related to the review.

2- Read the abstract in order to find more details about the articles and exclude all articles not relevant. After conducting this criterion, the number of articles was reduced to about 100.

3- These articles were scanned and considered, as a final check and to exclude any not related ones.

2.4 Quality assessment and classification: the identified papers and articles need to be classified into categories. There are many kinds of classifications for XML indexing techniques, each of which depends on the aims and aspects of the study. The next section gives more detail about the classification and criteria used in this research. With respect to quality, the articles are divided into two classes, namely, development and innovation studies, and analysis and review studies. The criterion for any study being considered as development and innovation study is that the article has proposed or developed a new contribution; whereas the criterion for the analysis and review study is that the article has a survey or a review of others.

2.5 Techniques for classification and review: XML indexing techniques can be classified into three categories according to the ways and aspects in which they are evaluated. First, the indexing techniques are evaluated on the basis of the structural relationships in the index. The classification of this category is usually divided into three classes, as follows: 1) node indexes; 2) path indexes; 3) sequence indexes [6, 7]. Second, this classification is based on the position or location of the index residence. The position can be either in the main memory as a temporary index or in the hard disk, called a disk-based index. The former has the advantage of fast response as it avoids the input and output expenses. However, it has the disadvantage that it lacks scalability for large index files.

Third, in this category, classification is based on the type of document that is indexed. There are two classes, namely, the index data-centric document, and the index document-centric XML database [8].

This research focuses on structural relationships indexes as this category is the most relevant to the objectives of the research [9, 10].

2.6 Common criteria for the evaluation of indexing techniques: the ideal method for evaluating a XML indexing technique is to compare it with other techniques using some of the criteria that are suitable for all such techniques. There are a number of common criteria that are used to compare indexing techniques. This research uses some of them in order to evaluate the three structural indexes, namely, node index, graph index and sequence index. These criteria were chosen as the most helpful ones for users to select the best technique for their requirements. The selection is carried out by determining the features that these indexes support, for instance: precision, response time, and completeness. The following are these used criteria:

a) **Retrieval power:** this means the precision and completeness of the result, and the type of queries supported.

b) **Processing complexity**: this step covers a few issues, such as the requirement of structural joins – in order to improve the performance of a query operation; there is a need to minimize the number of joins. Other issues include the processing cost, and the need to compute the relationships between elements.

c) **Scalability**: large indexes involve many input and output operations. Thus this increases the query processing time.

d) **Update cost**: basically, there are two kinds of updates, namely, inserting a node and inserting a subtree. The nodes in a tree index need to be kept organized in a particular way to reflect all kinds of relationships. These relationships have to be preserved if a new node is inserted into the tree. Thus, the index has to reflect its location with respect to these relationships, which makes the case more complex, especially if the scheme has no spaces for the new node.

3 Literature review

XML data is considered to be a semi-structured data since the schema and the data are mixed in the semi-structured data. Thus, this feature provides flexibility for the semi-structured data. Moreover, XML and semi-structured data have more similar features such as simple models, flexibility, self-descriptive models, and readable models for both humans and computers. The semi-structured data is able to represent XML data from other models [1, 11-13].

3.1 XML data indexing techniques: this research focuses on the XML indexes of structural relationships as this category is the most relevant to the objectives of this research. These indexing techniques are evaluated on the basis of the structural relationships in the index. The classification of this category is usually divided into three classes, which are the following: 1) node indexes; 2) path indexes; 3) sequence indexes.

3.1.1 Node indexes: basically, node indexes retain values in the XML tree structure. Each value reflects the location of a node in the tree. These values are used to find a certain node's parents, child, sibling, ancestor and descendent. These values are represented by numbers and used to

resolve simple and twig queries. Generally, the most commonly used schemes are the interval (also known as region) labelling scheme and the prefix (also known as path) scheme. Further details about these schemes can be found in the following sections [14-17] [18-21] [22].

3.1.1.1 Prefix scheme: this type of scheme generates code containing two fragments which are the *prefix* part and the *actual-code*. The prefix part encodes the previous node code. The actual-code encodes the order of the node in the tree. There are many examples of this scheme, and the most popular one is *Dewey* labelling. Dewey labelling has two parts in each node except in the root node. This code is called the *Dewey* code [23-25]. The code at each node has two parts. The first part is an increasing number that reflects the location of the node. The second part is the Dewey code which is the parent's code. These parts are separated by a dot like this ".". The root code has only one part since it has no parent. Figure 1 shows an example of a Dewey labelling scheme [16, 24, 26-28].

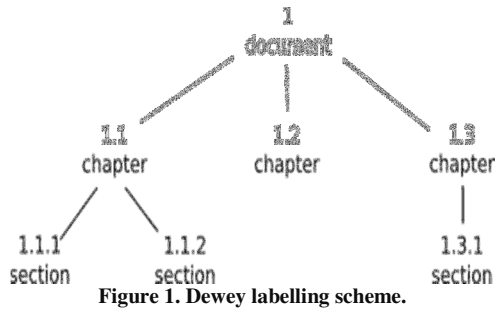


Figure 1. Dewey labelling scheme.

A number of researchers [17] developed a dynamic labelling method that can be used with Dewey labels with identifiers of size 0(log n) where "n" is the size of the database. All labelling schemes including Dewey need 0(n) bits per label [7]. Different kinds of prefix labels are suggested. The following are examples of some of them: Duong and Zhang [29] developed Labelling Scheme for Dynamic XML Data called (LSDX). In this method the numbers and letters are combined. This scheme supports the ancestor/descendent relationship and the sibling between nodes. Lu and Ling [23] propose a labelling scheme that contains two parts. The first part is the group ID. The second part is the group prefix. This scheme is called GRoup base Prefix (GRP). Both the LSDX and GRP schemes are firm and immutable as the label sizes of these schemes can reach 0(n) bits per label. The ORDPATH labelling scheme was developed by a number of researchers [30].

3.1.1.2 Interval labelling Scheme: this is also known as Region-based Encoding. Basically, the idea of this scheme is to attach two values to each node – the startID and the endID. The startID is used to save the node ID for first element or attribute. The endID is used to store the end of the attribute. There are some examples of this labelling scheme such as the (Beg, End) and (Pre, Post) labelling schemes. The (Beg, End) labelling scheme allocates two numbers to each node, based on its sequential traversal order as follows: the mechanism gives a "Beg" number to all elements starting from the root, and each element, attribute of an element, value of an attribute, and value of an element, according to the sequential location in the XML document. When we reach the end of an attribute or an attribute value then the allocated value is the "End" number [31-33].

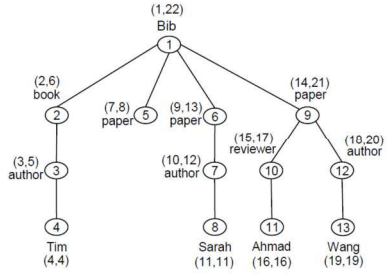


Figure 2. (Beg, End) labelling Scheme [34].

The scheme (Beg, End) can be used to answer both twig query and path query by making use of the relational database management system. This can be achieved by using "structural joins" [35]. To answer a query, the relationships between any couple of nodes in a path in this query is investigated individually as the granularity of this indexing scheme is determined at the level of each node and therefore this provides a precise and complete answer for the query. XML queries shred XML documents into tables of relational databases with the fixed schema (Label, Beg, End, Level, Flag, and Value) [8]. Table 2 represents the relational table of shredded XML document of the above node tree.

TABLE 2: A NODE TABLE OF THE XML DATA IN FIGURE 2 [34]

Label	Beg	End	Level	Flag (Type)	Value
Book	2	6	2	Element	Null
Author	3	5	3	Value	Tim
Paper	7	8	2	Element	Null
.....	...	...	....	.....	.....
Paper	14	21	2	Element	Null
Reviewer	15	17	3	Attribute	Ahmad

A number of researchers such as Silberstein et al. and Chen et al. [33] have developed dynamic labelling schemes for interval indexes. The dynamic labelling schemes allow relabeling of the schemes. The interval labelling scheme is the most used scheme for fixed encoding. Such examples propose leaving spaces between the values in order to add new nodes. In case of adding new nodes, there is a need for re-numbering or other solution.

Cohen et al. [7] approved that persistent labelling needs 0(n) bits per label where n is the size of the tree. The interval labels size is used to measure the complexity as this size determines the total size of the index. It is preferable to keep the used number of bits small as this can allow the index to reside in the main memory.

Li and Moon [18] developed the (*Order, Size*) labelling scheme. Each *Order* and *Size* has a certain job. The *Order* one is based on a traversal of pre-order, whereas the *Size* part is an estimation of the number of child or descendent nodes for a given node. The advantage of this mechanism is that this labeling scheme leaves space for any case of adding or inserting nodes in order to avoid relabeling of the data-tree as relabeling can cause delay.

3.1.1.3 Summary: to conclude, both the prefix scheme and interval labelling scheme perform well in XML operations. The interval labelling scheme is better than the prefix scheme in terms of the storage space cost. Thus, some

enhancements have been added by reducing the comparison costs and other features in interval labelling scheme. Nevertheless, this scheme is costly in terms of updates [35, 36] [37, 38].

3.1.2 Graph indexing scheme (Path scheme): this type of indexing scheme is also known as a summary index and is described as a structural path summary. It is used to enhance query efficiency by producing a path summary for XML data in order to accelerate the process of query evaluation. However, it can also be used to solve twig queries, but with the extra cost of multiple joins operations. There are a number of graph indexes such as DataGuide [39, 40]; Index Fabric [41]; APEX [42]; D(K)-index [43]; $(F+B)^K$-index [44]; and F&B-index [1, 8]. Graph indexes are classified in different categories according to the number of criteria [41, 42, 45, 46].

Graph indexes have been classified into different types of categories on the basis of different criteria. Examples of these classifications are the following: Polyzotis and Garofalakis classified the graph indexes according to exactness [47]. This classification divided the schemes into exact schemes such as strong Data Guide, 1-index, disk-based F&B-index, Index Fabric, and F&B-index, and approximate schemes. Examples of approximate schemes are A(K)-index, approximate Data Guide, D(K)-index, and $(F+B)^K$-index [47].

There is another category that classifies schemes into two classes, namely, path indexes (aka P-index), which are suitable for simple path queries such as DataGuide and 1-Index, and twig indexes (aka T-index), suitable for twig queries such as F&B-index [8].

Another category [34] considers a classification that categorizes the graph schemes into determinism and bisimilarity. In more detail, in determinism, the paths of the tree are considered to be deterministic paths. The other category, bisimilarity, has two sub classes – forward and backward. According to the determinism and bisimilarity classification, for more detail, graph indexes are categorized into the following classes:

3.1.2.1 Deterministic graph indexes: in this index, every path is listed once in the summary graph. Each path in a summary graph has at least one identical path in the data graph. There are some indexing schemes that are considered as deterministic graph indexes such as Strong DataGuide, Approximate DataGuide, and Index Fabric [39, 40, 43].

The Strong DataGuide was proposed by Goldman and Widon [39]. Strong DataGuides have the ability to give complete and precise results for both simple parent/child path queries and ancestor/descent path. Regarding the twig queries, Strong DataGuides are complete but not precise [44].

The Approximate DataGuide (ADG) has solved the problem of the large size of the Strong Data Guide since this scheme shows large size in some cases [40].

Cooper et al. proposed the Index Fabric in order to solve the problem of scalability [41]. The Index Fabric is theoretically like the Strong Data Guide as the size might enlarge dramatically. Moreover, the Index Fabric is complete for both path and twig queries. However, it is precise for the path but not for the twig [34].

3.1.2.2 Non-deterministic graph indexes with backward bisimilarity: there are a number of these indexing schemes such as: 1-index, A(K) index, and D(K) index. These indexes are divided based on backward bisimilarity. The 1-index was proposed by Milo and Suciu [48] in order to decrease the size of a structural summary. The 1-index divides the data nodes of a document into similar classes based on their backward bisimilarity. The 1-index and DataGuide are the same in the case of s simple XML data tree.

Kaushik et al. [44] proposed the A(K)-index mostly to solve the size cost. The size of an A(K)-index is small compared with Strong DataGuide and 1-index. The A(K)-index is usually complete but not always precise.

Chen, Lim, et al. propose the D(K)-index [43] in order to select the most appropriate value of "*k*" which is a big problem for the A(K)-index. Thus, the D(K)-index is better than the A(K)-index with respect to processing time and storage size. Apart from this, the A(K)-index and D(K)-index are similar schemes in terms of scalability, completeness and precision [34].

3.1.2.3 Non-deterministic graph index with forward and backward bisimilarity: this is the only kind of graph index that has the ability to support twig queries: the F&B-index, $(F+B)^K$-index, and the disk based F&B-index [49] [44]. The F&B-index was proposed by Abiteboul et al. [1]. It is different from the A(K)-index and D(K)-index as this scheme is based on the incoming and outgoing paths' bisimilarity for all nodes. Thus, it is a twig structural index scheme. [50] developed the $(F+B)^k$-index. This scheme is an improved release of the F&B-index. The $(F+B)^k$-index scheme controls the size of the F&B-index by identifying the value of the "*K*" [8].

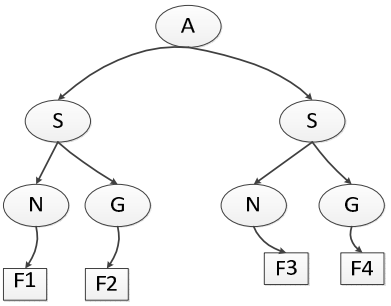
The Disk-based F&B-index was proposed by [51]. This index scheme has provided additional properties and criteria. The Disk-based F&B-index is an integration of 1-index and F&B-index in a new clustered Disk-based F&B-index which is then saved on the disk. [34].

3.1.3 Sequence indexing scheme: this kind of index converts both XML documents and queries into structure sequences. Sequence indexes put the values and the structures of XML data together into an integrated index structure. This new structure is used to evaluate both path and twig queries efficiently: answering a query, making a string sequence that matches the sequence of the data with the query. This technique reduces the need for joins to evaluate twig query [27]. Basically, the sequence schemes are classified into two types according to the importance of tree mapping direction, which are as follows: top-down sequence indexing schemes, and bottom-up sequence indexing schemes.

Wang, Park, Fan and Yu (2003) proposed the ViST. This scheme is based on B+ tree [52]. The ViST (Virtual Suffix Tree) is an example of top-down sequence indexes. The ViST scheme is based on the B+ tree. In addition, the ViST has the disadvantage of weakening the query operations due to the large number of nodes being checked. This is due to the ViST being used as a top-down sequence. Thus, the size of the index becomes very large when dealing with large XML documents since the top elements are added into the sequence. This is the main disadvantage of the ViST scheme. The PRIX (Prufer sequence for indexing

XML) is an example of a bottom-up sequence index. This indexing technique does a good job in decreasing the query processing time. Since the ViST has a problem of scalability as mentioned above, Rao and Moon propose the PRIX as another method that uses a bottom-up sequence to solve the scalability problem with the ViST [53].

Some studies show that these two indexing schemes have a weakness in terms of precision, recall, and processing complexity [52, 54]. However, the sequence indexing techniques have some advantages such as 1) the ability to expect the results of the query; 2) using the complete query tree as one component in order to avoid the cost of the joint operations. Figure 3 is an example illustrates how this indexing technique works.



Label	*Map*
Students	A
PGR	S
Name	N
Group	G
Omar	F1
XDIR	F2
James	F3
SE	F4

Figure 3: Sequence-based indexing examples

From Figure 3, the XML tree is changed into a sequence as follows: T= (A), (S,A), (N,AS), (F1,ASN), (G,AS), (F2,ASG), (S,A), (N,AS), (F3,ASN), (G,AS), (F4,ASG).

Despite the sequence indexing technique having the advantage of speeding the query pre-evaluation and getting rid of the increase of structural joins, this technique also has two disadvantages, namely, 1) the sequence of XML produced by one of these technique algorithms has to be reconstructed quite often; 2) this technique uses a hashing algorithm to encode the textual values, which makes the hash list increase very fast, therefore the indexing becomes very slow [53, 55-60].

4 Results

Using the framework that was defined in the Methods section, the articles were classified accordingly. Table 3 shows this classification and the statistics.

The indexing techniques have been categorized into three classes. Node indexes, graph indexes, and sequence indexes. The investigation in this research has found the following consequences:

1- To the best of our investigation, there is no single indexing technique that is ideal for all users' requirements. Therefore, choosing a suitable indexing technique depends on the user's requirements.

2- With respect to the retrieval power and processing complexity, the node indexing technique is a good one for precision of the results.

3- Regarding scalability, it is clear that most of the indexing techniques suffer from this problem. The problem that these techniques always face is the trade-off between size and efficiency.

4- Update cost is also a common limitation and all investigated techniques have this disadvantage.

TABLE 3: CLASSIFIED ARTICLES

indexing techniques / Articles classifications	Node index	Graph index	Sequence index	Sum	Other articles
Development and Innovation	5	9	7	21	16
Analysis and Review	3	2	2	7	
Sum of each class	8	11	9	28	

4.1 Limitations and open issues of the indexing techniques: there are number of limitations and open issues that will be discussed in the following:

4.1.1 Limitations: having investigated and studied most indexing techniques, there are three key limitations and problems. These limitations are given as follows:

1- Index size: most of the studied indexing techniques have a problem of scalability. Some indexing techniques are considered to be main memory indexes. Thus, owing to the large size of indexes they cannot reside in the main memory. Furthermore, other approaches that use disk-based indexing techniques also have a problem with the scalability of the indexes as the processing time is affected by the index capacity and its performance.

2- The cost of computation problem: there is a high cost with respect to construction of indexes and the query evaluations procedure. Regarding the relational XML indexes, the disadvantage is that there is a need for a complex computation process in the query evaluation by working out the elements' relationships. The updating of indexes: this problem is a common one among XML indexing techniques.

4.1.2 Open issues and challenges: many researches have been carried out on the indexes of XML data. However, there are still many challenges and open issues that need to be considered. Perhaps the key challenge for XML indexes is the irregularity of structure and data. XML data is considered as semi-structured data. This means that data may be not be complete or may be irregular, and the structure may change quickly and randomly.

5 Discussion and analysis

The XML database systems are a relatively new research area and not as mature as the DBMS. However, many previous studies considered the classifications and evaluations of XML indexing and structures. Each of them has a different investigation and results, depending on the aims and the methodology of the study. Furthermore, XML indexing data is a key factor in enhancing the XML query.

Although heavy research in XML indexing data has been carried out, most of these techniques still face a lack of efficiency. This research has found there is no single technique that is perfect for all types of queries. However, each kind of XML indexing technique has some advantages in different aspects. Node indexes are the most inefficient with regard to structural joins since they need joins for both path and twig queries, whereas graph indexes need no structural joins to support path queries. But for the evaluation of the twig queries, the structural joins are required. Regarding the sequence indexes, they are the most efficient as they encode the structure within the sequence. To summarize, Table 4 shows a summary

evaluation for the four techniques using the mentioned criteria.

TABLE 4: COMPARISON BETWEEN THE INDEXING TECHNIQUES USING THE CRITERIA

No.	Criteria	Node index technique	Graph index technique	Sequence index technique
1	**Retrieval power (precision)**	Yes	Yes/no (yes for path & no for Twig)	No
2	**Processing complexity**	No	Yes/no (join required)	Yes
3	**Scalability**	Yes	Yes	Yes
4	**Update cost**	Yes	Yes	Yes

6 Conclusion

To conclude, The XML database systems are not as mature as the relational database management systems which have been studied heavily for decades. The indexing of XML data plays a major role in enhancing the performance of queries. In addition, since XML has a great deal of advantages in terms of data transformation in the WWW, this research has investigated and studied the XML indexing data techniques and structures. The issue is that, to index XML data, it has to represent and reflect the structure, so an efficient path can be made. Moreover, the main problem for indexing techniques is the trade-off between the size and efficiency of the indexes. Therefore, all the indexes become large in order to perform well. The analysis of this research has been carried out on the basis of factors that influence the performance, such as the retrieval power, processing complexity, scalability and update cost. The initial findings show that no one of all the indexing techniques is ideal for all cases and user's requirements.

7 References:

[1] S. Abiteboul, P. Buneman, and D. Suciu, *Data on the Web: from relations to semistructured data and XML*: Morgan Kaufmann, 2000.

[2] R. W. P. Luk, H. V. Leong, T. S. Dillon, A. T. S. Chan, W. B. Croft, and J. Allan, "A survey in indexing and searching XML documents," *Journal of the American Society for Information Science and Technology,* vol. 53, pp. 415-437, 2002.

[3] S. H. Simon, *XML*: McGraw-Hill, 2001.

[4] S. St. Laurent, *XML: a primer*: MIS: Press, 1997.

[5] A. Vakali, B. Catania, and A. Maddalena, "XML data stores: emerging practices," *IEEE Internet Computing,* vol. 9, pp. 62-69, 2005.

[6] N. Bruno, N. Koudas, and D. Srivastava, "Holistic twig joins: optimal XML pattern matching," pp. 310-321.

[7] C. Edith, K. Haim, and M. Tova, "LABELING DYNAMIC XML TREES," *SIAM Journal on Computing,* vol. 39, p. 2048, 2010.

[8] G. Gang, R. Chirkova, and R. Chirkova, "Efficiently Querying Large XML Data Repositories: A Survey," *IEEE Transactions on Knowledge and Data Engineering,* vol. 19, pp. 1381-1403, 2007.

[9] S. Al-Khalifa, H. V. Jagadish, N. Koudas, J. M. Patel, D. Srivastava, and W. Yuqing, "Structural joins: a primitive for efficient XML query pattern matching," pp. 141-152.

[10] M. Atay, A. Chebotko, D. Liu, S. Lu, and F. Fotouhi, "Efficient schema-based XML-to-Relational data mapping," *Information Systems,* vol. 32, pp. 458-476, 2007.

[11] G. Thom, "Data on the Web: From Relations to Semistructured Data and XML," vol. 125, ed: Media Source, 2000, p. 112.

[12] B. Catania, A. Maddalena, and A. Vakali, "XML document indexes: a classification," *IEEE Internet Computing,* vol. 9, pp. 64-71, 2005.

[13] D. DeHaan, D. Toman, M. Consens, and M. Özsu, "A comprehensive XQuery to SQL translation using dynamic interval encoding," pp. 623-634.

[14] M. El-Sayed, K. Dimitrova, and E. Rundensteiner, "Efficiently supporting order in XML query processing," pp. 147-154.

[15] M. Maghaydah and M. A. Orgun, "Labeling XML nodes in RDBMS," pp. 122-126.

[16] T. Härder, M. Haustein, C. Mathis, and M. Wagner, "Node labeling schemes for dynamic XML documents reconsidered," *Data & Knowledge Engineering,* vol. 60, pp. 126-149, 2007.

[17] D. Fisher, F. Lam, W. Shui, and R. Wong, "Efficient ordering for XML data," pp. 350-357.

[18] Q. Li and B. Moon, "Indexing and querying XML data for regular path expressions," in *VLDB*, 2001, pp. 361-370.

[19] M. Al-Badawi, S. North, and B. Eaglestone, "Classifications, Problems Identification and a New Approach," 2007.

[20] M. Al-Badawi, S. North, and B. Eaglestone, "The 3D XML benchmark." vol. 1, ed, 2010, pp. 13-20.

[21] M. Al-Badawi, H. A. Ramadhan, S. North, and B. Eaglestorne, "A performance evaluation of a new bitmap-based XML processing approach over RDBMS," *International Journal of Web Engineering and Technology,* vol. 7, pp. 143-172, 2012.

[22] S. Mohammad, P. Martin, and W. Powley, "Relational universal index structure for evaluating XML twig queries," pp. 116-120.

[23] J. Lu and T. W. Ling, "Labeling and querying dynamic XML trees," in *Advanced Web Technologies and Applications*, ed: Springer, 2004, pp. 180-189.

[24] M. L. Scott and M. L. SCOTT, *Dewey decimal classification*: Libraries Unlimited, 1998.

[25] D. K. Fisher, F. Lam, W. M. Shui, and R. K. Wong, "Dynamic labeling schemes for ordered XML based on type information," in *Proceedings of the 17th Australasian Database Conference-Volume 49*, 2006, pp. 59-68.

[26] I. Tatarinov, S. D. Viglas, K. Beyer, J. Shanmugasundaram, E. Shekita, and C. Zhang, "Storing and querying ordered XML using a relational database system," *Proceedings of the ACM SIGMOD International Conference on Management of Data,* pp. 204-215, 2002.

[27] W. Wei, J. Haifeng, L. Hongjun, and Y. Jeffrey Xu, "PBiTree coding and efficient processing of containment joins," pp. 391-402.

[28] J. Lu, T. W. Ling, C.-Y. Chan, and T. Chen, "From region encoding to extended dewey: On efficient processing of XML twig pattern matching," in *Proceedings of the 31st international conference on Very large data bases*, 2005, pp. 193-204.

[29] M. Duong and Y. Zhang, "LSDX: a new labelling scheme for dynamically updating XML data," in *Proceedings of the 16th Australasian database conference-Volume 39*, 2005, pp. 185-193.

[30] P. O'Neil, E. O'Neil, S. Pal, I. Cseri, G. Schaller, and N. Westbury, "ORDPATHs: insert-friendly XML node labels," pp. 903-908.

[31] P. Dietz, "Maintaining order in a linked list," pp. 122-127.

[32] D. D. Kha, M. Yoshikawa, and S. Uemura, "An XML indexing structure with relative region coordinate," pp. 313-320.

[33] A. Silberstein, H. He, K. Yi, and J. Yang, "BOXes: Efficient maintenance of order-based labeling for dynamic XML data," in *Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on*, 2005, pp. 285-296.

[34] S. Mohammad and P. Martin, "XML structural indexes," Citeseer2009.

[35] C. Zhang, J. Naughton, D. DeWitt, Q. Luo, and G. Lohman, "On supporting containment queries in relational database management systems," *Proceedings of the ACM SIGMOD International Conference on Management of Data,* pp. 425-436, 2001.

[36] X. Wu, M. L. Lee, and W. Hsu, "A prime number labeling scheme for dynamic ordered XML trees," in *Data Engineering, 2004. Proceedings. 20th International Conference on*, 2004, pp. 66-78.

[37] Y. Chen, G. Mihaila, R. Bordawekar, and S. Padmanabhan, "L-Tree: a dynamic labeling structure for ordered XML data," in *Current Trends in Database Technology-EDBT 2004 Workshops*, 2005, pp. 209-218.

[38] B. Yang, M. Fontoura, E. Shekita, S. Rajagopalan, and K. Beyer, "Virtual cursors for XML joins," pp. 523-532.

[39] R. Goldman and J. Widom, "Dataguides: Enabling query formulation and optimization in semistructured databases," 1997.

[40] R. Goldman and J. Widom, "Approximate dataguides," in *Proceedings of the Workshop on Query Processing for Semistructured Data and Non-Standard Data Formats*, 1999, pp. 436-445.

[41] B. F. Cooper, N. Sample, M. J. Franklin, G. R. Hjaltason, and M. Shadmon, "A fast index for semistructured data," in *VLDB*, 2001, pp. 341-350.

[42] C.-W. Chung, J.-K. Min, and K. Shim, "APEX: An adaptive path index for XML data," in *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, 2002, pp. 121-132.

[43] Q. Chen, A. Lim, and K. W. Ong, "D (k)-index: An adaptive structural summary for graph-structured data," in *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, 2003, pp. 134-144.

[44] R. Kaushik, P. Bohannon, J. Naughton, and H. Korth, "Covering indexes for branching path queries," pp. 133-144.

[45] "XRel: a path-based approach to storage and retrieval of XML documents using relational databases," *ACM Transactions on Internet Technology (TOIT),* vol. 1, pp. 110-141, 2001.

[46] M. Yoshikawa, T. Amagasa, T. Shimura, and S. Uemura, "XRel: a path-based approach to storage and retrieval of XML documents using relational databases," *ACM Transactions on Internet Technology,* vol. 1, pp. 110-141, 2001.

[47] N. Polyzotis and M. Garofalakis, "Structure and value synopses for XML data graphs," in *Proceedings of the 28th international conference on Very Large Data Bases*, 2002, pp. 466-477.

[48] T. Milo and D. Suciu, "Index structures for path expressions," in *Database Theory—ICDT'99*, ed: Springer, 1999, pp. 277-295.

[49] H. Su-Cheng and L. Chien-Sing, "Evolution of Structural Path Indexing Techniques in XML Databases: A Survey and Open Discussion," pp. 2054-2059.

[50] R. Kaushik, P. Shenoy, P. Bohannon, and E. Gudes, "Exploiting local similarity for indexing paths in graph-structured data," in *Data Engineering, 2002. Proceedings. 18th International Conference on*, 2002, pp. 129-140.

[51] W. Wang, H. Jiang, H. Wang, X. Lin, H. Lu, and J. Li, "Efficient processing of XML path queries using the disk-based F&B index," in *Proceedings of the 31st international conference on Very large data bases*, 2005, pp. 145-156.

[52] H. Wang, S. Park, W. Fan, and P. Yu, "ViST: a dynamic index method for querying XML data by tree structures," pp. 110-121.

[53] P. Rao and B. Moon, "PRIX: Indexing and querying XML using prufer sequences," in *Data Engineering, 2004. Proceedings. 20th International Conference on*, 2004, pp. 288-299.

[54] H. Wang and X. Meng, "On the sequencing of tree structures for XML indexing," in *Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on*, 2005, pp. 372-383.

[55] S. Mohammad, "INDEX STRUCTURES FOR XML DATABASES," Dissertation/Thesis, 2011.

[56] P. Rao and B. Moon, "Sequencing XML data and query twigs for fast pattern matching," *ACM Transactions on Database Systems (TODS),* vol. 31, pp. 299-345, 2006.

[57] J.-M. Bremer and M. Gertz, "Integrating document and data retrieval based on XML," *The VLDB Journal,* vol. 15, pp. 53-83, 2006.

[58] X. Meng, Y. Jiang, Y. Chen, and H. Wang, "XSeq: an indexing infrastructure for tree pattern queries," in *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, 2004, pp. 941-942.

[59] K. H. Prasad and P. S. Kumar, "Efficient indexing and querying of XML data using modified Prüfer sequences," in *Proceedings of the 14th ACM international conference on Information and knowledge management*, 2005, pp. 397-404.

[60] B. Stein, "Principles of hash-based text retrieval," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, 2007, pp. 527-534.