
Proceedings Paper:
Al-Khazraji, S. and North, S.D. orcid.org/0000-0002-8478-8960 (2018) A relevance comparison between interval and prefix labelling schemes. In: 2017 International Conference on Engineering and Technology (ICET). International Conference on Engineering and Technology 2017, 21-23 August 2017, Antalya, Turkey. IEEE . ISBN 978-1-5386-1948-3

https://doi.org/10.1109/ICEngTechnol.2017.8308211

A Relevance Comparison between Interval and Prefix Labelling Schemes

Samer Al-khazraji
Department of Computer Science
The Education College for Pure Science, Diyala University
Diyala, Iraq,
samerbaq@yahoo.com

Siobhán North
Department of Computer Science
The University of Sheffield
Sheffield, United Kingdom
s.north@sheffield.ac.uk

***Abstract*— Improving XML database management system has attracted researchers to consider whether the indexing system is equivalent to a relational database management system. The indexing system is based on labelling the nodes of the XML tree. Different types of labelling scheme have been proposed to label the document quickly and without consuming too much storage space. However, most the studies focused on evaluating the performance of new labelling schemes. The appropriateness of various existing schemes to the particular structure an XML document has not been addressed sufficiently. To investigate this aspect two common XML labelling schemes were employed: Prefix (Dewey Encoding) and Interval (Containment) to label three different examples of XML documents with very different structures. The time and storage space requirements were investigated to compare the relevance of each scheme to the structures of the documents. A number of experiments were conducted and it was found that Dewey Encoding and Containment techniques are relatively fast when labelling shallow tree structures. Dewey required little storage space to save labels of wide tree structures, however, Containment used less storage space when storing the labels of short trees.**

Keywords*— *XML labelling scheme, Prefix, Interval, Dewey, Containment.

I. INTRODUCTION

Undoubtedly XML has emerged as the de facto technology for data transmission and representation in a wide range of domains [1; 2; 3] and the need for a qualified management system to organise XML data storage is important [4].

XML databases are classified into two categories [5; 6] XML-enabled databases which use a conventional database management system such as Oracle XDK and Microsoft SQL Server that supports XML documents [7; 6]. To store XML data into this type of database, the data needs to be mapped into the traditional database management system which is a costly process. Native-XML databases NXD preserves the hierarchical structure of the XML document and eliminates the mapping process [7; 6] and this class of storage is the core of this research. Data is stored in the conventional database using tables which consist of rows and columns and accessing the required data can be achieved through an indexing system. However, this system cannot be used to query information which has been stored in a tree structure as XML documents are [4] as shown in Figure 1.

Data in an XML document tree demonstrates various kinds of structural relationships: parent-child P-C, ancestor-descendant A-D, and siblings' relationships [8; 2]. An indexing system is needed that has the ability to represent the nodes correlations in the XML databases and guide the query to the intended node effectively and efficiently [9]. Node labelling schemes can be used as an indexing system in XML document. They assign a unique label to each node that encodes the node relationships in the tree [10]. An XML query has a similar structure to XML documents and therefore the use of XML labelling schemes may increase the performance of query processing through matching the structures [11; 12].

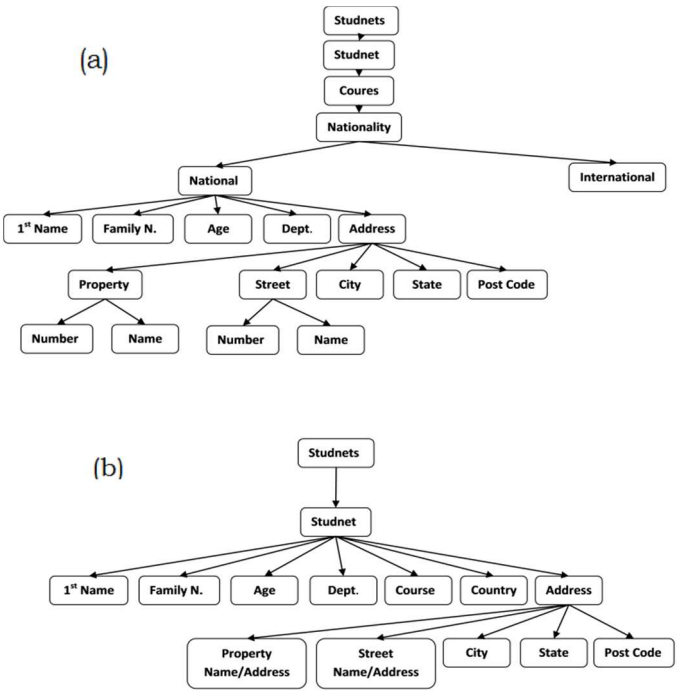

Figure 1: XML different tree structure. (a) Deep XML Tree. (b) Wide XML Tree.

It has been demonstrated that the time required to label XML documents relies on the XML document size and the number nodes [13]. However, existing work has rarely considered the XML tree structures as can be seen in Figure 1 where the tree in Figure 1 is deeper than that in Figure 1 and the latter tree is wider than the former. Depending on the tree structure, researchers have compared the performance of their schemes for labelling the documents with previous work on each individual XML dataset based on time and storage space. However, they did not investigate the performance of their approach with different tree structures.

XML labelling schemes have been categorised into: Interval labelling scheme, Prefix labelling scheme, Multiplicative labelling scheme, and Hybrid labelling scheme [14] cited by [15]. In this study, the performance of the Prefix and Interval schemes when labelling three different XML tree structures have been analysed based on time and storage space required.

The rest of the paper is organised as follows. In Section II a set of related work is reviewed. Section III investigates the relevance of XML schemes. Section IV includes experimental results as well as discussesing them and Section V concludes the paper.

II. THE RELATED WORKS

XML documents have been adopted in different domains for data representation and storage such as, data warehousing [16; 17; 18] cited in [19] mathematics (Mathematics Markup Language (MathML)) [20], healthcare [21]. The extensive use of XML documents will lead to an increase in the data produced. As a result, tree size will increase and this justifies the need for a technique able to define node relationships. This can be achieved through a labelling scheme [22; 23; 24]. XML labelling schemes function by describing the node's location in the tree and its relationships by a unique identifier, its label [22; 25].

A significant task of XML labelling schemes is to increase the performance of XML database management by improving query processing. A user query is written using one of the XML languages such as, XPath and XQuery which were designed to process the user queries in semi-structure documents such as XML. Effective and efficient query processing depends on the labelling scheme used to match the relationships between the nodes in the user query and the XML tree [9]. So, the performance of query processing depends on the efficiency of a scheme that is able to allocate a small label to each node in a tree quickly.

A two well-known labelling schemes will be explained in the next section, namely Interval and Prefix. Many schemes have been proposed based on these two fundamental techniques.

A. interval-based labelling schemes

Interval Based labelling schemes are named because the intervals between node labels are exploited to determine node relationships. They define the relationships between the parent or ancestor and its descendants nodes [26] as cited in [4]. The earliest labelling scheme for encoding XML nodes is Interval-Based labelling scheme designed by [27] and based on Pre and Post tree traversal order as cited by [28]. The node label structure of this scheme consists of two integer values depending on the node's location in the tree during *Preorder* and *Postorder* traversal of the tree [28]. For example, the node Library in Figure 2 is ancestor of the node Book because 1<2 in preorder tree traversal and 3<7 in postorder tree traversal.

However, it is not obvious that the node Library is the parent to the node Book.

To cover this drawback, an extension to this scheme was proposed by [29]. In their scheme a label consists of: (*Start, End, Position*), where, Start and End represent the range of labels of the descendant nodes and Position is the node's level in the tree; its distance from the root. In this scheme, the P-C relationship can be identified because the position of the child is one higher than that of the parent [30; 15].

For instance, the node Library in Figure 3 is the parent of the node Book because the level of Book is deeper than the level of Library.

$$Level_{Book} = Level_{Library} + 1$$

Moreover, the labels of descendant nodes are contained in the range of the parent label. This property called the *Containment* property [13].

The approach of [29] considered relationship representation in Interval labelling schemes. However, another group of researchers studied the simplicity of label generation based on interval labelling scheme as will be explained in the next approach.

In [15], a new labelling scheme was proposed to simplify the process of generating labels of [27]. The scheme assigns a unique label to each node which consist of (*level, ordinal, rID*) as illustrated in Figure 4. Where, Level is the node's level in the XML tree starting from level 0 the root's node level. Ordinal is a unique integer number assigned to the node during preorder tree traversal and rID is the ordinal of the right most sibling in its sub-tree.

Interval labelling schemes visit XML nodes twice to produce labels for each node.

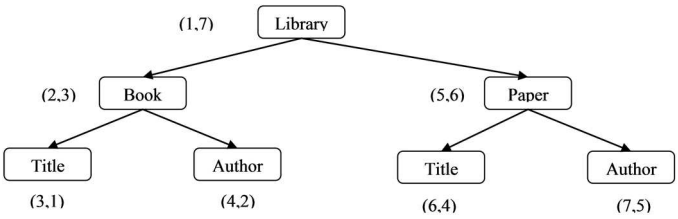

Figure 2: Preorder/Postorder-Based Labelling Scheme.

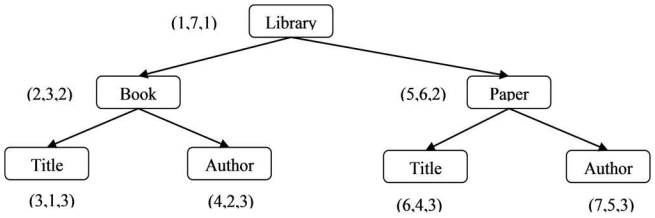

Figure 3: Containment Labelling Scheme

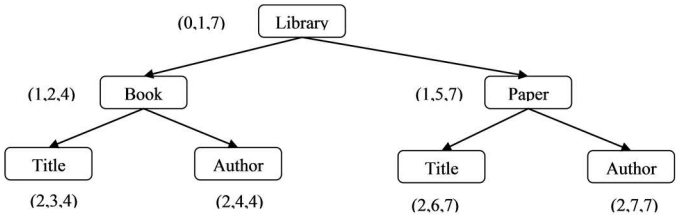

Figure 4: Relab Labelling Scheme

This is expensive in terms of space and is slow, furthermore the time increases exponentially as the tree grows. There was a need for a scheme that generates labels in linear time and storage space as the tree grows [31]. This kind of labeling scheme will be explained in the next subsection.

B. Prefix Labelling Scheme

Prefix labelling schemes are similar to a technique used by librarians called *Dewey Decimal Coding* [31]. In [22], it was argued that this class of scheme can represents different kinds structural relationships between nodes. In Prefix schemes the node's parent label is encoded as a prefix to the node's individual label. These labels are generated by depth-first search and they are separated by a delimiter, either ',' or '.' [32; 22; 33]. A popular Prefix labelling scheme proposed by [32] and known as *Dewey Encoding* and will be explained in the next section.

In [32], the labelling scheme was intended to answer order-sensitive queries such as [32; 26] *Preceding, Following*, *Preceding-sibling Following-Sibling*, and *Position = n*. The first class excludes the ancestor or descendant of the context nodes and is focused on if the node is before or after it. The second class will retrieve the preceding and following elements siblings in the XML tree. The last class of query will simply fetch the information of the intended node.

In [32], *Dewey Order* labelling was designed to be compatible with order sensitive queries. Their scheme is a combination of two numbering approaches *Global Order* and *Local Order*. *Global Order* assigns a label to each node based on the global order of the node in the XML tree as shows in Figure 5. *Local Order* allocates a label to the node based on the node's order among its siblings as shown in Figure 5. The Dewey Order labelling scheme combines these values and encodes the node's path from the root to its location in the tree as can be seen in Figure 5.

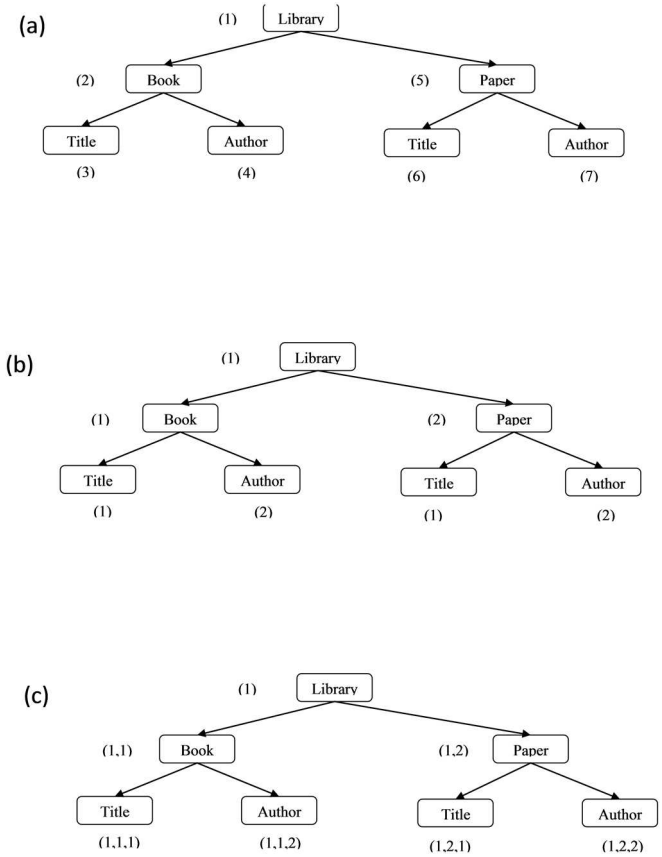

Figure 5: Prefix labelling scheme. (a) Global Order Labelling Scheme, (b) Local Order Labelling Scheme, (c) Dewey Labelling Scheme.

Dewey labelling is an expressive scheme which represents different kinds of structural relationships of XML nodes. From the Figure 5, Book and Paper are sibling nodes because their labels are sequential (e.g. 1 and 2) and the prefix of both is the same. Moreover, it is clear that Library is an ancestor of Author, where, the prefix of Author label starts with the label of Library as can be seen in Figure 5.

Many schemes have been proposed based on *Dewey Encoding* such as: ORDPATH [34], *Dynamic Float-Point Dewey* DFPD [35], *Labelling Scheme for Dynamic XML data* LSDX [36], *Compressed Dynamic Labelling Scheme* Com-D [37], OrderedBased [22], and etc. These labeling schemes were proposed as extensions for dynamic trees that support the update of the XML tree without relabelling. Schemes for dynamic labelling are outside the scope of this study and will not be discussed.

III. RELEVANCE INVESTIGATION OF XML LABELLING SCHEMES

XML labelling schemes were exploited to represent structural relationships whilst producing a unique label for each node in the document [10]. Consequently, labelling schemes can improve query processing efficiency by accessing the labels rather than the real document [11; 12].

Many attempts have been mad to identify the best labelling scheme, one which generates labels quickly and requires little storage space to save them. Most research effort has been spent in overcoming the drawbacks of previous work by suggesting new schemes that show an improvement over previous works. However, researchers do not analyze the applicability of their work to different structures of XML database.

To analyze the relationship between schemes and tree structure, two common labelling schemes were employed: Dewey Encoding and Containment. A number of experiments were executed to measure the performance of Dewey Encoding (Dewey in short) and Containment based on time and storage space. Three different real XML documents were employed: Nasa, DBLP, and Treebank-e all of which can be found on the Washington University website for research purpose [38]. Two sets of experiments were executed: the first were run to measure the time required to label each dataset using Dewey and Containment. The shortest time for labelling time a specific XML document structure will indicate its suitability for that document.

Another set of experiments were carried out to investigate the scheme which requires least storage space to store the labels of the XML dataset. Small label size can improve the query processing by reducing the comparison time between the structures of query and the node labels [9].

The run time and storage space required to label the three XML documents using these schemes were measured independently as will be explained in the next section.

IV. Experiments and Results Analysis

A. System Setup

A number of experiments were executed using Eclipse 'Release 4.4.0RC1' as an integrated development environment IDE to run Java code on a computer has Intel (R) Core (TM) i5-3570t CPU 2.30 GHz, RAM 4 MB, and windows 7 Enterprise. Moreover, SPSS20 which is a common statistical application was exploited to analyze the results. In these experiments, Containment and Dewey Encoding labelling schemes were employed to measure the performance with three XML databases: Nasa, DBLP, and Treebank-e. The characteristics of these datasets are shown in the Table 1.

B. Discussion

Figure 6 shows the statistical information in the Table 2. The x-axis represents the type of XML scheme and y-axis represents the time consumed for labelling the XML documents in millisecond.

Table 1: XML Databases

XML Database	No. of Elements	Max Depth (Level)	File Size
Nasa	476646	8	23MB
dblp	3332130	6	127MB
Treebank-e	2437666	36	82MB

Table 2: Time Consumed for Labelling Nasa, dblp, and Teebank-e using Dewey and Containement

Scheme	Nasa		Dblp		Treebank-e	
	Mean	*STD*	*Mean*	*STD*	*Mean*	*STD*
Dewey	262.96	14.716	1488.15	29.255	1175.09	42.050
Contain.	307.07	5.883	1911.89	24.688	1362.24	17.280

The statistical information in the Table 2 showed that the mean time to label using Dewey (1,488ms) is shorter than that for Containment (1,912ms) when encoding DBLP as illustrated in the Figure 6. In [28], it was explained that Containment generates a new label for each element that is between 1 and 2n, where 'n' is the number of elements. Therefore, this technique will require exponential time when labelling XML trees [31]. The same figure can be seen in labelling the same set of databases using Containment. However, the time consumption for labelling DBLP is the greatest because it has a largest number of nodes of all databases in the Table 1.

Another set of experiments were performed to evaluate the relevance of Dewey and Containment for labelling the same collection of XML documents based storage space.

Prefix produces labels sequentially and the label size depends on the node level in the tree. The depth of Treebank-e is 36 levels which means label of a node at the level 36 will

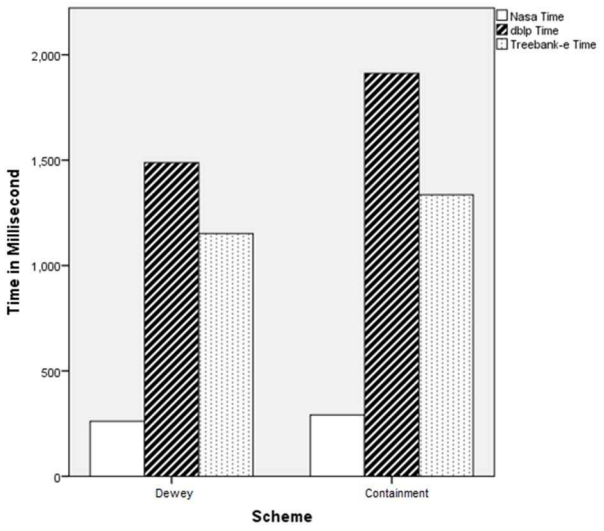

Figure 6: Time Required for Labelling XML Databases.

consist of 36 sections. The statistical information in the Table 3 shows that Treebank-e required the largest storage space 48,922 KB to store the labels generated by Dewey in comparison to other databases. The labels generated by Dewey are sequentially and the number of sections of the node label depends on the depth of the node in the XML tree [32].However, DBLP consumed 59,858 KB which is the

largest storage space required for storing labels and these were produced by the Containment scheme as shows in the Figure 7.

To validate our results, we compared them with the results of initial labelling of the three XML databases reported in [28]. It was observed that our results were consistent with the published results.

V. Conclusion

In this study, the problem of measuring the suitability of an XML labelling scheme for a particular XML document structure was studied. This issue has not been sufficiently addressed in the XML literature. It can potentially reduce effort in proposing a new scheme by revealing weaknesses of alternatives. This facilitates the design and optimisation of new schemes. To this end, three real XML databases were employed (Nasa, DBLP, Treebanck-e) and two common XML labelling schemes (Dewey Encoding and Containment) were used to label these databases. A set of experiments were carried out to analyse the appropriate scheme for labelling a particular XML document structure based on the time and storage space requirements.

Table 3: Space Required in KB for Saving the Labels of Nasa, dblp, and Treebank-e using Dewey and Containment Labelling Scheme.

Scheme	Nasa	dblp	Treebank-e
Dewey	7119.90	37664.14	48921.83
Contain.	7754.21	59858.03	44368.03

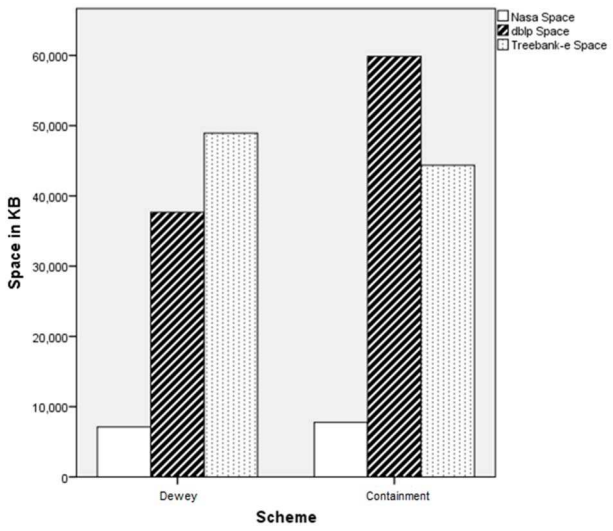

Figure 7 : Space Required in KB for Labelling XML Databases.

In case of the time measurement, it was observed that Dewey Encoding is suitable for a shallow XML tree structures and Containment better fits deep databases. On the other hand, both Dewey Encoding and Containment consumed little storage when saving the labels of the deep trees. For the future work, the relevance of other schemes to these different structures of XML databases should be investigated.

Acknowledgment

This research was supported in part by the Iraq Ministry of Higher Education and Scientific Research and the University of Diyala.

References

1. *XML and data integration.* **Bertino, Elisa and Ferrari, Elena.** 2001, IEEE, pp. 75--76.
2. *Element similarity measures in XML schema matching.* **Algergawy, Alsayed and Nayak, Richi and Saake, Gunter.** 2010, Elsevier, pp. 4975--4998.
3. *XML data clustering: An overview.* **Algergawy, Alsayed and Mesiti, Marco and Nayak, Richi and Saake, Gunter.** 2011, ACM, p. 25.
4. **Almelibari, Alaa.** *Labelling Dynamic XML Documents: A GroupBased Approach.* Sheffield : University of Sheffield, 2015.
5. **Bellahs`ene, Z.** *Database and XML Technologies.* Berlin : Springer, 2003.
6. **Kurt, Atakan and Atay, Mustafa.** *International Workshop on Databases in Networked Information Systems: An experimental study on query processing efficiency of native-XML and XML-enabled database systems.* s.l. : Springer, 2002. 268--284.
7. **Win, Khin-Myo and Ng, Wee-Keong and Lim, Ee-Peng.** *Asia-Pacific Web Conference: efficient native XML storage system.* s.l. : Springer, 2033. 59-70.
8. **Wilde, Erik.** *Wilde's WWW: technical foundations of the World Wide Web.* s.l. : Springer Science \& Business Media, 2012.
9. *Dynamic interval-based labeling scheme for efficient XML query and update processing.* **Yun, Jung-Hee and Chung, Chin-Wan.** 1, s.l. : Elsevier, 2008, Vol. 81. 56--70.
10. *An Analysis of Approaches to XML Schema Inference.* **Mlynkov'a, Irena.** s.l. : IEEE, 2008. 16-23.
11. *Dynamically updating XML data: numbering scheme revisited.* **Yu, Jeffrey Xu and Luo, Daofeng and Meng, Xiaofeng and Lu, Hongjun.** 1, s.l. : Springer, 2005, Vol. 8.
12. *Query processing and optimization for regular path expressions.* **Wang, Guoren and Liu, Mengchi.** s.l. : Springer, 2003. 30--45.
13. *A dynamic labeling scheme using vectors.* **Xu, Liang and Bao, Zhifeng and Ling, Tok Wang.** s.l. : Springer, 2007. 130--140.
14. *Data storage practices and query processing in XML databases: A survey.* **Haw, Su-Cheng and Lee, Chien-Sing.** 8, s.l. : Elsevier, 2011, Vol. 24. 1317--1340.
15. *ReLab: A subtree based labeling scheme for efficient XML query processing.* **Subramaniam, Samini and Haw, Su-Cheng and Soon, Lay-Ki.** s.l. : IEEE, 2014. 121--125.
16. *Change detection in hierarchically structured information.* **Chawathe, Sudarshan S and Rajaraman, Anand and**

Garcia-Molina, Hector and Widom, Jennifer. 2, s.l. : ACM, 1996, Vol. 25. 493--504.

17. *Comparing hierarchical data in external memory.* **Chawathe, Sudarshan S and others.** s.l. : VLDB, 1999, Vol. 99. 90--101.

18. *Detecting changes in XML documents.* **Cobena, Gregory and Abiteboul, Serge and Marian, Amelie.** s.l. : IEEE, 2002. 41--52.

19. *Structural similarity evaluation between XML documents and DTDs.* **Tekli, Joe and Chbeir, Richard and Yetongnon, Kokou.** s.l. : Springer, 2007. 196--211.

20. **Carlisle, D., Ion, P., Miner, P.** Mathematical Markup Language (MathML) Version 3.0. *W3C.* [Online] W3C, 4 10, 2014. [Cited: 5 21, 2016.] http://www.w3.org/TR/MathML/.

21. *Semantic and structural similarities between XML Schemas for integration of ubiquitous healthcare data.* **Thuy, Pham Thu Thu and Lee, Young-Koo and Lee, Sungyoung.** 7, s.l. : Springer, 2013, Vol. 17. 1331--1339.

22. *Orderbased labeling scheme for dynamic XML query processing.* **Assefa, Beakal Gizachew and Ergenc, Belgin.** s.l. : Springer, 2012. 287--301.

23. *Dynamic labelling scheme for XML data processing.* **Duong, Maggie and Zhang, Yanchun.** s.l. : Springer, 2008. 1183--1199.

24. *Triple Code: An Efficient Labeling Scheme for Query Answering in XML Data.* **Fu, Lizhen and Meng, Xiaofeng.** s.l. : IEEE, 2013. 42--47.

25. *A Dynamic Labeling Scheme Based on Logical Operators: A Support for Order-Sensitive XML Updates.* **Ghaleb, Taher Ahmed and Mohammed, Salahadin.** s.l. : Elsevier, 2015. 1211--1218.

26. *A prime number labeling scheme for dynamic ordered XML trees.* **Wu, Xiaodong and Lee, Mong-Li and Hsu, Wynne.** s.l. : IEEE, 2004. 66--78.

27. *Maintaining order in a linked list.* **Dietz, Paul F.** s.l. : ACM, 1982. 122--127.

28. *Labeling dynamic xml documents: an order-centric approach.* **Xu, Liang and Ling, Tok Wang and Wu, Huayu.** 1, s.l. : IEEE, 2012, Vol. 24. 100--113.

29. *On supporting containment queries in relational database management systems.* **Zhang, Chun and Naughton, Jeffrey and DeWitt, David and Luo, Qiong and Lohman, Guy.** 2, s.l. : ACM, 2001, Vol. 30. 425--436.

30. *Full tree-based encoding technique for dynamic XML labeling schemes.* **Zhuang, Canwei and Feng, Shaorong.** s.l. : Springer, 2012. 357--368},.

31. *Prefix based numbering schemes for XML: techniques, applications and performances.* **Sans, Virginie and Laurent, Dominique.** 2, s.l. : ACM, 2002, Vol. 1. 204--215.

32. *Storing and querying ordered XML using a relational database system.* **Tatarinov, Igor and Viglas, Stratis D and Beyer, Kevin and Shanmugasundaram, Jayavel and Shekita, Eugene and Zhang, Chun.** s.l. : ACM, 2002. {204--215.

33. *Dynamic labeling scheme for XML updates.* **Liu, Jian and Zhang, XX.** s.l. : Elsevier, 2016.

34. *ORDPATHs: insert-friendly XML node labels.* **O'Neil, Patrick and O'Neil, Elizabeth and Pal, Shankar and Cseri, Istvan and Schaller, Gideon and Westbury, Nigel.** s.l. : ACM, 2004. 903--908.

35. *Efficient labeling scheme for dynamic XML trees.* **Liu, Jian and Ma, ZM and Yan, Li.** s.l. : Elsevier, 2013, Vol. 221. 338--354.

36. *LSDX: a new labelling scheme for dynamically updating XML data.* **Duong, Maggie and Zhang, Yanchun.** s.l. : Australian Computer Society, Inc., 2005. 185--193.

37. *Dynamic labelling scheme for XML data processing.* **Duong, Maggie and Zhang, Yanchun.** s.l. : Springer, 2008. 1183--1199.

38. Datasets, Details, and Download. *XML Data Repository.* [Online] Washington University. [Cited: 5 10, 2016.] http://aiweb.cs.washington.edu/research/projects/xmltk/xmldata/www/repository.html.