
Proceedings Paper:
Al-khazraji, S. and North, S.D. orcid.org/0000-0002-8478-8960 (2018) The Emergence Computation of Overflow in Dynamic XML Tree Based on Prefix and Interval Labelling Schemes. In: 2017 International Conference on Engineering and Technology (ICET). The International Conference on Engineering & Technology 2017, 21-23 Aug 2017, Akdeniz University, Antalya, Turkey. IEEE . ISBN 978-1-5386-1949-0

https://doi.org/10.1109/ICEngTechnol.2017.8308213

The Emergence Computation of Overflow in Dynamic XML Tree Based on Prefix and Interval Labelling Schemes

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

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

***Abstract*— Despite the fact that dynamic XML labelling schemes have been investigated widely, some challenges still need to be tackled. Dynamic XML documents are subject to change. An efficient dynamic labelling scheme is able to maintain the node relationships throughout continuous changes to the XML tree structure. Such a scheme generates labels for new nodes to avoid the need to relabel the whole tree. The main problem for dynamic XML is *overflow* that occurs when the label length of the new node is over the reserved space limit. There has not been sufficient analysis to determine the class of labelling scheme which faces this problem in the early stages of update. To this end a series of experiments were performed when updating the Nasa XML database, which contains real data. Five sets of new nodes (50, 100, 400, 800, 1200) were inserted into this dataset using two versions of XML node indexing system: a Prefix and an Interval labelling scheme. It was found that Interval falls victim to the problem of overflow after the insertion of only 100 nodes whereas Prefix has no problem even when adding 1200 nodes.**

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

I. Introduction

The extensive exploitation to XML documents for data storage and exchange in different applications [1; 2; 3; 4] has attracted researchers to search for techniques to manage the increased data [5]. Some conventional database management systems known as XML-enable databases support XML documents such as Oracle XDK and Microsoft SQL Server [6; 7]. To store XML data in these databases, a mapping process is needed to transfer the data from an XML tree into a table format of rows and columns. Another kind of XML database called Native-XML databases XSD has a similar structure to XML and eliminates the need for the mapping process and this kind of XML databases is the center of this study [6; 7].

Relational database management systems have a mature indexing system to process user queries efficiently and effectively. However, this indexing system is not suitable for a data that has an hierarchical structures such as XML [5]. XML documents can be represented as a tree through different relationships: parent-child P-C, ancestor-descendant A-D, and sibling relationships [8; 2]. To access the intended node in this hierarchical tree structure, an indexing system is required which is capable of representing the node relationships and as result improves query processing [9].

Node labelling schemes are used as the indexing system for XML documents by assigning a single label to each node and this label represents its relationships in the XML tree [10; 11]. XML query languages such as XPath and XQuery have the same structure as XML documents and a comparison of the structures of the node labels and the query can speed up the query processing [9; 12].

Labelling schemes are evaluated in three terms: they should be *Compact* the label length should be small in order to fit in computer memory. *Flexible*, the scheme should be able to represent the different kinds of node relationships, and *Dynamic* the issue analysed in this paper, where the scheme should be able to assign labels dynamically during the update without a relabelling process [13; 14; 15].

Node labelling schemes are employed to assign a single label to a node that clarifies the structure of the node and its relationships in the document. This class of scheme is useful for the static XML databases. However, XML databases which are regularly updated cause problems for these schemes [16; 17; 18; 19; 20]. Labelling schemes for dynamic documents need to keep the node relationships during update to maintain the effectiveness of query processing [16]. Dynamic labelling schemes can preserve query processing efficiency through dynamic label generation but the label length will increase with increasing file size and that can have a detrimental effect the scheme performance because it needs a lot of storage [21].

Researchers in the XML labelling scheme domain have proposed a number labelling schemes which permit the addition of more data without the need for relabelling. In this study, the *Vector Order-Centric* labelling scheme designed by [22] was analysed for its ability to update XML documents dynamically. As was addressed in [22] the issue of XML node insertion occurs

in two forms: *uniform insertion*, where the node is inserted between random pairs of successive nodes and *skewed insertion* which is split into two sub-categories: *order skewed insertion* is a repeated insertion before or after a specific node and *random skewed insertion* is repeatedly inserting between two random nodes.

The paper will be organised as follows: section (II) will explain three kinds of labelling scheme. Section (III) will define the problem of overflow. Section (IV) includes the experiments results and discussion and Section (V) concludes the paper.

II. THE RELATED WORKS

XML mark up was proposed to provide flexibility in designing document structure [23]. This characteristic made XML a global technology for data transmission and representation in various applications [8], such as Bioinformatics [24], Geography [25], Mathematical Markup Language MathML [26], Distributed Learning Transferring ADL [27]. In order to enable these structures to be queried efficiently labelling schemes have been devised to provide an index to the tree's structure [28].

In this paper, a number of labelling schemes will be discussed. They are classified as: Interval *Labelling Scheme*, *Prefix Labelling scheme*, and *Multiplicative Labelling Scheme*.

A. interval-based labelling schemes

The labels of an Interval Labelling schemes represent the node location in the XML tree as a pair of numbers assigned during preorder and postorder tree traversal [29; 30]. This scheme is known as an Interval labelling scheme because the interval between the two numbers that form the label identifies the node's parent, ancestor or descendant node relationships ([30] as cited in [5]).

The earliest XML labelling scheme is called Pre-Post labelling scheme and was designed by [31]. It generates labels to represent the node relationships in the tree using two integers. For instance, in the Figure 1 the node School is the parent of Student because the Student's label is (2,3) which in the interval of the School's label (1,7). Moreover, School in the Figure 1 is the ancestor node of student's ID because the label of ID is (3,1) which is in the range of School label (1,7).

In [22], it was reported that a Pre-Post labelling scheme is able to represent A-D relationships but not P-C

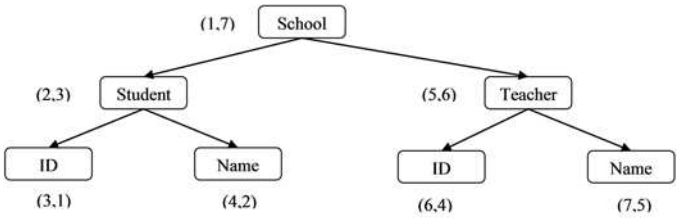

Figure 1: Preorder/Postorder-Based Labelling Scheme.

relationships explicitly. So [32] proposed a new interval labelling scheme which is dubbed the *Containment* labelling scheme. The suggested scheme allocates labels for each node which consist of three sections: Start, End, and Position to define the node correlations as shows in Figure 2.

It can be seen that the node level of School in Figure 2 is a higher level of the node Student, therefore, a P-C relationship must exist between them.

In [9], it was reported that the Interval labelling schemes produce labels sequentially in depth first tree traversal which enables node relationships definition. However, these schemes do not support dynamic XML documents. To cover this drawback, space is reserved for node insertions but this approach has a disadvantage. With extensive use of the XML database, the allocated space may be insufficient for the number of nodes inserted and this situation is known as overflow. It means that relabelling of the whole tree is required [33; 29].

Therefore, a group of researcher adopted another technique for static and dynamic label generation as will be explained in the next section.

B. Prefix Labelling Scheme

Prefix labelling schemes are based on a system used by librarians called *Dewey Decimal Coding* [34] and it can be used to define the structural relationships through node labels [13]. Prefix labelling schemes assign labels during a depth first traversal of the XML tree. Each label consists of sections and are separated by delimiters ',' or '.'. The prefix section of the node's label is the parent label which itself contains its parent's label and so on starting from the root [35; 13; 16]. The Figure 3 illustrates *Dewey Encoding* which is the most known Prefix Labelling Scheme and was designed by [35].

Many schemes have been proposed based on Dewey Encoding to provide labels for the dynamic update of XML documents. However, these schemes required a great deal of storage space for deep trees [29]. So another group of researchers adopted the mathematical operations to assign labels that define the relationships effectively as will be explained in the next section.

C. Multiplicative Labelling Schemes

Schemes of this class use integers as labels and employ mathematical operations such as modulus [36], division, multiplication and the Chinese reminder operation [30]. Another group of researches exploited graph vectors to define the node relationships through label encoding.

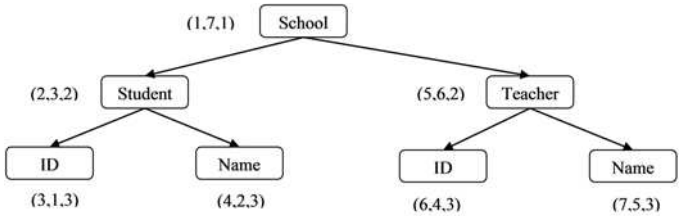

Figure 2:Containement labelling scheme.

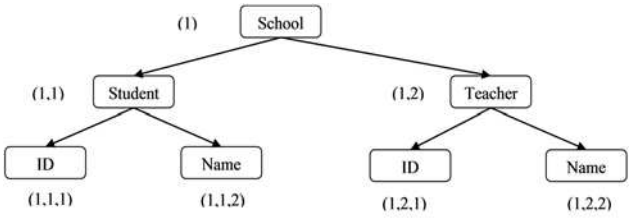

Figure 3: Dewey Encoding labelling scheme.

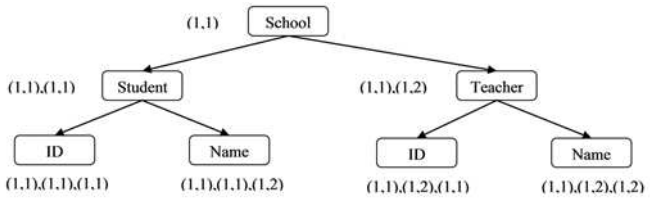

Figure 5: Vector order-centric based on Dewey Encoding labelling scheme.

and designed a new dynamic labelling scheme known as *vector order-centric* [22]. Vector order-centric schemes can encode labels produced by the static schemes, such as Interval (Containment) as can be seen in Figure 4 and Prefix (Dewey Encoding) as illustrated in **Error! Reference source not found.** .

In spite of the benefits of vector order-centric labelling scheme [22], it was reported that it has a weakness [13; 21] which is the target of this paper.

III. The Overflow problem in vector order-centric labelling scheme

Static labelling schemes cannot change the node labels when the structure of the XML tree is changed [11]. Therefore, [22] suggested a labelling scheme, called vector order-centric that employed graph vectors to maintain the node labels when the tree structure is changed. The labels were initially generated by one of the conventional static labelling schemes.

The tree update was addressed by [22] through looking at two insertion process: *uniform insertion* and *skewed insertion*. Uniform insertion is inserting a new node between two randomly chosen consecutive nodes. Skewed insertion was classified into two classes: *ordered skewed insertion*, is repeatedly inserting new nodes before or after a particular node. *Random skewed insertion*, is repeatedly adding new nodes between two nodes in random order.

In [22], it was explained that vector order-centric labelling is efficient when labelling XML trees dynamically without a relabelling process. However, [21] reported that the some results of vector order-centric are unavailable because it exploits UTF-8 mechanism for label representation. They did not give further information of the mechanism impact on the label representation which can assist the later studies.

To address the problem, a vector order-centric labelling scheme was implemented using both Prefix and Interval schemes. Dewey Encoding, an example of a Prefix labelling scheme was used for the initial labelling. *V-Dewey* will be used to show the resulting labels for Dewey Encoding within a vector order-centric scheme. From the Interval schemes, Containment was used for the initial labelling of the XML dataset. V-Containment is the result of Containment labelling encoded by the vector order-centric scheme.

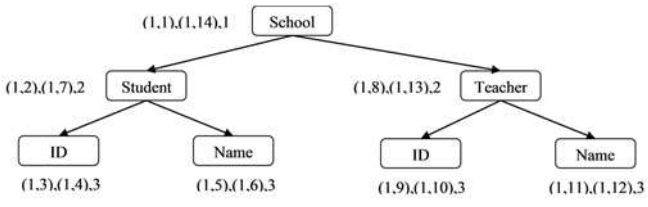

Figure 4: Vector order-centric based on Containment Labelling Scheme.

Two groups of experiments were executed using V-Containment and V-Prefix to insert 200, 500 and 1000 elements into the database based on ordered skewed insertion. The first group experiments were used to measure the time required to insert the three sets of elements into the document. The second series of experiments were used to evaluate the storage space needed to store the labels as will be explained in the next section.

IV. Experiments and results analysis

A. System Setup

The experiments were run using 'Release 4.4.0RC1' as an integrated development environment IDE to execute Java code on a computer has Intel (R) Core (TM) i5-3570t CPU 2.30 GHz, RAM 4 MB, and windows 7 Enterprise. V-Containment and V-Dewey were adopted to update Nasa XML dataset which is a real XML dataset. This database has a balance between the depth and width of its tree structure as shown in Table **1**. The Nasa dataset can be downloaded from the website of the University of Washington [37] for research purposes. In addition the statistical application SPSS was employed to investigate the results.

B. Discussion

Five groups of nodes: 50, 100, 400, 800, and 1200 were inserted into Nasa to analyse the impact of depth increasing on the label size using V-Containment and V-Dewey schemes. The time required to generate labels for the new nodes are displayed in Table 2 and they are illustrated in Figure 6.

As can be seen in Table 2, the mean time for encoding 50 new nodes using V-Dewey is 10,763 ms which is half of 32,829ms the mean time for encoding 100 nodes. In addition, the mean time for labelling 100 new nodes has doubled around 8 times for labelling 400 update nodes and time consumption has steeply with the increased number of new nodes.

Table 1: The Characteristics of Nasa XML Database.

XML Database	No. of Elements	Max Depth (Level)	File Size
Nasa	476646	8	23MB

Table 2: The statistical information of mean time for insertion five groups of nodes.

Scheme	50 nodes insertion	100 nodes insertion	400 nodes insertion	800 node insertion	1200 nodes insertion
V-Cont.	2247	4185	13204	24562	35233
V-Dewey	10763	32829	266425	811827	1439715

On the other hand, the mean time for encoding 50 new nodes using V-Containment is 2,247ms and it is about doubled for the insertion of 100 new nodes. Moreover, the mean time for generating labels for the new 400 nodes is 13,204ms, three times the mean tome for 100 nodes. The mean time gradually declined after insertion of 800 new nodes starting from 24,562ms.

To investigate this case, another set of experiments were conducted to study the storage space requirement for updated the same set of nodes.

The storage space needed to store these groups of new nodes is shown in Table 3 which clarifies the differences of time consumptions for labels generating. The storage required for storing 50 labels produced by V-Containment is 86 Byte and it increased to 87 Byte for the new 100 labels. After that, the number is reduced to 13 Byte for the over 100 new nodes. On the contrary, the storage required for storing new node labels generated using V-Dewey rose from 22,009 Byte for 50 new nodes to 528,009 Byte for the 1200 labels as shown in Figure 7.

It can be shown Figure 7, the scale of storage for labels generated by V-Dewey is very large in comparison to the scale of V-Containment. 528,009 Byte is needed for 1,200 inserted labels produced by V-Dewey in contrast to 13 Byte for labels generated by V-Containment. As a result, the storage spaces of V-Containment cannot be seen in Figure 6, therefore, logarithm was employed to amplify the values of storage capacities as shows in Figure 8. So, it is clear that the space size for over 100 elements has reduced from 87 byte to be steady on 13 byte. Whilst, the storage space for the new labels which were generated by V-Dewey has increased sharply from 44,009 Byte to 528,009 Byte for over 100 nodes.

V-Dewey is based on Prefix labelling scheme which generates labels linearly and the label size depends on the number of label sections which in turn relies on the node's depth in the tree [35].

Table 3: space required for insertion five groups of nodes.

Scheme	50 nodes insertion	100 nodes insertion	400 nodes insertion	800 node insertion	1200 nodes insertion
V-Cont.	86	87	13	13	13
V-Dewey	22009	44009	176009	352009	528009

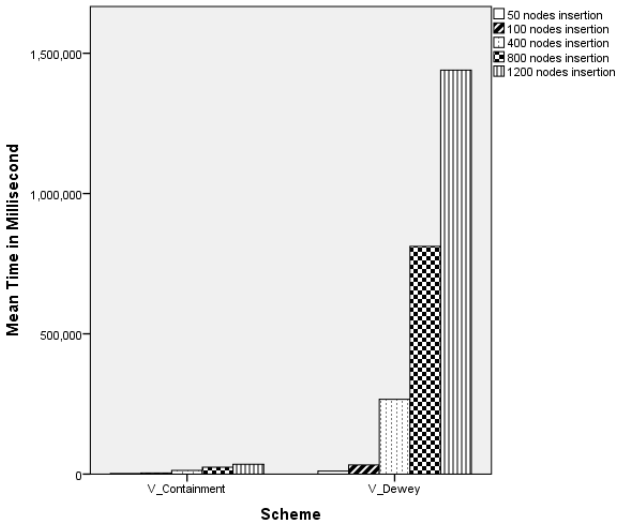

Figure 6: The mean time for insertion five grpups.

Based on the node's depth, the time range in V-Dewey has scaled up with the increase on the node numbers to represent the structural relationships from the root [38].

However, the view of V-Containment is different. It is based on an Interval labelling technique which generates labels exponentially that consists of a fixed number of sections and it is calculated based on the Node's parent label [32]. The label size will rise and so will the time consumption with the increase of the tree's depth. The decline of the storage space over 100 nodes insertion due to the label size has become over the ability of the mechanism for label representation which is known as *overflow* problem and relabelling process is inevitable [21].

Vector order-centric scheme uses UTF-8 mechanism [39] which is qualified to represent node label up to 2^{31} bits [21] which is less than the label size over 100 nodes. So, Figure 8 shows that the storage space of 400 elements and over is steady at around 1.11394 which is the natural log of 13 and time consumption has decreased with increase of the updated nodes as illustrated in Figure 6.

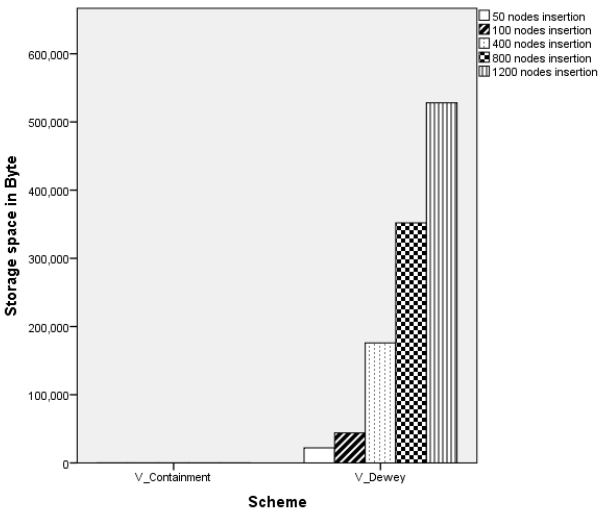

Figure 7: The Space required in KB to add three groups of elements in Nasa.

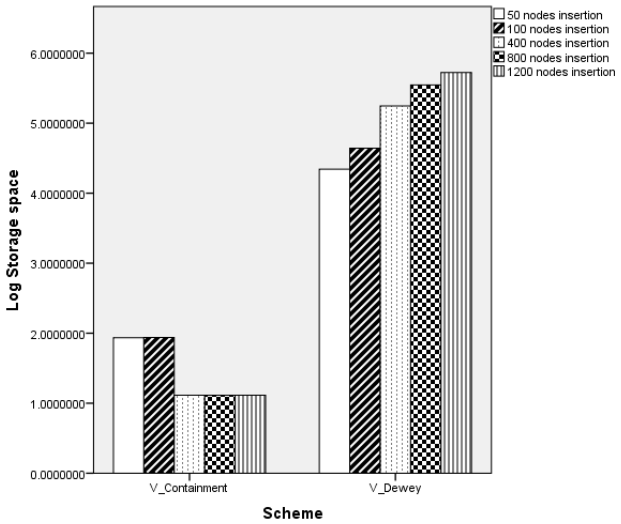

Figure 8: Logarithm of space vlause.

V.CONCLUSION

A common problem of dynamic XML tree update known as overflow was investigated in this paper. This problem has significant effect on the performance of dynamic XML trees. The node relationships need to be maintained during changes to the XML without the relabelling the existing nodes. However, the problem's emergence using different labelling schemes has not been addressed sufficiently. The Prefix and Containment labelling schemes were employed to define the nodes relationships of Nasa XML database. In addition, the vector order-centric technique [38] was employed to preserve the relationships during XML update. A number of experiments were conducted to insert three groups of elements into the dataset to identify the overflow problem using V-Containment and V-Prefix. It was found that the problem emerged after 100 node insertions using V-Containment. The representation of the node's context through labels needs more analysis to improve the dynamic labelling schemes.

ACKNOWLEDGMENT

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

VI.BIBLIOGRAPHY

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. **Tim Bary, Jeaqn Paoli, C.M. Sperberg-McQueen, Eva Maler, François Yergeau, John Cowan.** Extensible Markup Language (XML) 1.1 (Second Edition). *W3C.* [Online] W3C, 29 September 2006. [Cited: 31 January 2016.] https://www.w3.org/TR/xml11/.

5. **Almelibari, Alaa.** *Labelling Dynamic XML Documents: A GroupBased Approach.* Sheffield : University of Sheffield, 2015.

6. *Asia-Pacific Web Conference: efficient native XML storage system.* **Win, Khin-Myo and Ng, Wee-Keong and Lim, Ee-Peng.** s.l. : Springer, 2003. 59-70.

7. **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.

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. *Labeling dynamic XML trees.* **Cohen, Edith and Kaplan, Haim and Milo, Tova.** 5, s.l. : SIAM Journal on Computing, 2012, Vol. 39. 2048--2074.

12. *Query processing and optimization for regular path expressions.* **Wang, Guoren and Liu, Mengchi.** s.l. : Springer, 2003. 30--45.

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

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

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

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

17. *Efficient updates in dynamic XML data: from binary string to quaternary string.* **Li, Changqing and Ling, Tok Wang and Hu, Min.** 3, s.l. : Springer, 2008, Vol. 17. 573--601.

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

19. *QED: A novel quaternary encoding to completely avoid re-labeling in XML updates.* **Li, Changqing and Ling, Tok Wang.** s.l. : ACM, 2005. 501--508.

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

21. *SCOOTER: A compact and scalable dynamic labeling scheme for XML updates.* **O'Connor, Martin F and Roantree, Mark.** s.l. : Springer, 2012. 26--40.

22. *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.

23. **Harold, E.R. and Means, W.S.** *XML in a Nutshell.* s.l. : O'Reilly Media, 2004. 9781449379049.
24. *XML, bioinformatics and data integration.* **Achard, Frederic and Vaysseix, Guy and Barillot, Emmanuel.** 2, s.l. : Oxford Univ Press, 2001, Vol. 17. 115--125.
25. *The roles of geography markup language (GML), scalable vector graphics (SVG), and Web feature service (WFS) specifications in the development of Internet geographic information systems (GIS).* **Peng, Zhong-Ren and Zhang, Chuanrong.** 2, s.l. : Springer, 2004, Vol. 6. 95--116.
26. **Carlisle, D., Ion, P., Miner, P.** Mathematical Markup Language (MathML) Version 3.0. *W3C.* [Online] W3C, 10 4 2014. [Cited: 21 5 2016.] http://www.w3.org/TR/MathML/.
27. SCORM 2004 4th Edition Version 1.1 Overview. *ADL.* [Online] ADL. [Cited: 23 5 2016.] http://www.immagic.com/eLibrary/ARCHIVES/TECH/US_DOD/A090814O.pdf.
28. *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.
29. *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.
30. *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.
31. *Maintaining order in a linked list.* **Dietz, Paul F.** s.l. : ACM, 1982. 122--127.
32. *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.
33. *Labeling and querying dynamic XML trees.* **Lu, Jiaheng and Ling, Tok Wang.** s.l. : Sprunger, 2004. 180--189.
34. *Prefix based numbering schemes for XML: techniques, applications and performances.* **Sans, Virginie and Laurent, Dominique.** 2, s.l. : ACM, 2002, Vol. 1. 204--215.
35. *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.
36. *A modulo-based labeling scheme for dynamically ordered XML trees.* **Al-Shaikh, Raed and Hashim, Ghalib and BinHuraib, AbdulRahman and Mohammed, Salahadin.** s.l. : IEEE, 2010. 213--221.
37. Datasets, Details, and Download. *XML Data Repository.* [Online] Washington University. [Cited: 10 5 2016.] http://aiweb.cs.washington.edu/research/projects/xmltk/xmldata/www/repository.html.
38. *A Relevance Comparison between Interval and Prefix Labelling Schemes.* **Samer, Al-khazraji and Siobhán North.** s.l. : IEEE, 2017. 1-6.
39. *UTF-8, a transformation format of ISO 10646.* **Yergeau, Francois.** 2003.