International Journal of Technology 13(5) 1055-1063 (2022)
Received March 2022 / Revised August 2022 / Accepted August 2022

International Journal of Technology

http://ijtech.eng.ui.ac.id

Performance Evaluation of XML Dynamic Labeling Schemes on Relational Database

Su-Cheng Haw[1*], Aisyah Amin[1], Palanichamy Naveen[1], Kok-Why Ng[1]

[1]*Faculty of Computing and Informatics, Multimedia University, 63100 Cyberjaya, Malaysia*

Abstract. eXtensible Markup Language (XML), in its semi-structured format has been employed for the data exchange purpose over the Internet due to its expressivity, flexibility, and capability to accommodate both structured and unstructured data. Due to the vast amount of data being transacted and updated frequently, it is essential to have a solution that can efficiently store and query the data. Hence, a robust and persistent labeling scheme that can sustain the need to re-labeling the entire document is desirable. Relational Database (RDB) has emerged since the 1970s and has been widely used as back-end storage in most industries. Since XML and RDB are in different formats, an efficient mapping technique is required. Several labeling and mapping schemes have been proposed, yet, there is no comparison of the performance of these schemes implemented in the RDB storage. In this paper, we first review the dynamic labeling schemes such as ORDPath, ME Labeling, and ORD-GAP in addressing these two needs. Secondly, the XML annotated labeling schemes are transformed into RDB storage. Finally, the performance evaluations are carried out to determine which labeling scheme is more robust and efficient to support storage and query retrieval.

Keywords: Dynamic updates; Labeling scheme; Mapping scheme; Performance evaluation; XML database; XML query

1. Introduction

eXtensible Markup Language (XML) is the de facto standard for data transactions over the Internet in many application domains, such as document repositories, healthcare, banking, and business transactions (Hsu and Liao, 2020). With the massive growth of data transferred over the Internet, a solution capable of coping with this situation is crucial (Jittawiriyanukoon & Srisarkun, 2018). To fully exploit the full-featured XML, it is essential to support both assorted queries and dynamic updates over the XML content (Haw & Song, 2021). On the other hand, some labeling schemes require re-labeling the whole XML tree. As a result, it will increase the XML database size. As such, a persistent and robust labeling scheme susceptible to re-labeling is very much needed.

Since XML evolution up until now, many methods to transform XML into RDB have existed (Gupta & Narsimha, 2015). Yet, most existing methods only support static documents by assuming that the structural information will not change over time (Dhanalekshmi et al., 2021). This situation is impractical as the data transacted over the web is subject to frequent updates.

[*]Corresponding author's email: sucheng@mmu.edu.my, Tel.: +603-83125233; Fax: +603-83125264
doi: 10.14716/ijtech.v13i5.5871

This paper's first part concentrates on reviewing existing labeling schemes that support dynamic updates. The subsequent piece of the paper evaluates the performance of the chosen labeling schemes implemented on the RDB storage based on a path-based mapping approach. The path-based method tracks the hierarchical path information (Kapisha & Lakshmi, 2020) to map it into the parent table (path information on non-leaf nodes) and the child table (path information on leaf nodes) (Haw & Song, 2021).

Generally, labeling schemes can be grouped into multiplicative-based, region-based, path-based, and hybrid (Amin et al., 2018). The region encoding is based on assigning a unique integer to all nodes whereby the child nodes are within the parent node and ancestor node range, respectively (Pasnur et al., 2016). Prefix-based (path-based) is based on assigning the prefix of the parent node (as well as the ancestor) to the child (as well as the descendant). The multiplicative-based is based on some arithmetic operations to calculate a unique identifier for each node. At the same time, the hybrid-based may combine any group of labeling groups to overcome one scheme's weaknesses with the other scheme's features.

Alsubai and North (2017) proposed a multiplicative-based Child Prime Label (CPL) with the label of (start, end, level, CPL) using prime number assignment. Khanjari and Gaeini (2018) introduced using a binary bit for the FibLSS encoding scheme. The experimental evaluations demonstrated that FibLSS insertion is possible without the need for re-labeling. Taktek and Thakker (2020) proposed Pentagonal Scheme, which can handle several insertions on massive datasets. Ahn and Im (2020) proposed a MapReduce-based prefix-based labeling scheme by extending the dynamically compressed element labeling (DCL) (Ahn & Im, 2016). Using MapReduce, they can reduce the space incurred by dynamically adjusting the label based on the number of nodes. Azzedin et al. (2020) extended Dewey labeling (Tatarinov et al., 2002) and named the scheme RLP-Scheme. From their approach, the A-D can be identified easily, even for XML with many identical sub-trees.

Based on the review, we observed that prefix-based is the most diverse and widely adopted. Yet, this scheme label size grows with the length of the encoded path. In the worst case, its size is O(n). As such, in this paper, we have selected to focus on the evaluation of the path-based schemes, namely ORDPath (O'Neil et al., 2004), Multiplicative-Efficient (ME) labeling (Subramaniam & Haw, 2014) and ORD-GAP (Haw et al., 2021).

2. Methods

In this section, we first elaborate on the three selected labeling schemes in Section 2.1. Then, we will briefly outline the architecture of the proposed simulation engine in Section 2.2.

2.1. Background

2.1.1. ORDPath

O'Neil et al. (2004) proposed an ORDPath labeling scheme in the format of prefixing the nodes based on a hierarchical relationship of the particular node. ORDPath applies positive integer and odd numbers to assign the initial labeling schemes, as shown in Figure 1, where the entire line indicates the initial labeling. They reserved the even and negative integers for future insertion, i.e., the right-most, left-most, and in-between insertion. The left-most node insertion is computed by adding a -2 to the last ordinal of the first child in the same level. In contrast, the right-most node insertion is calculated by adding a +2 to the previous ordinal of the previous child. The in-between node insertion uses "careted in". For instance, a new odd element is inserted between an even ordinal and the last odd ordinal, and the odd element usually is 1. The dotted line in Figure 1 shows the insertion of right-most, in-between, and left-most nodes using the ORDPath scheme.

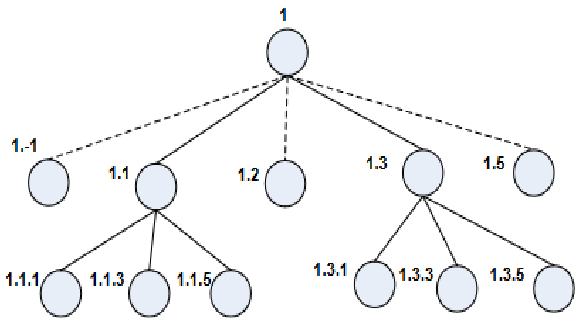


Figure 1 Left-most, right-most and in-between insertion

The Parent-Child (P-C) and Ancestor-Descendant (A-D) relationships can be determined in ORDPath. As an illustration, node 1.3 is the parent of node 1.3.5, while node 1 is the ancestor of node 1.3.5 by looking at the prefix, which is 1.3 and 1, respectively.

2.1.2. ME Labeling

ME labeling uses multiplication operation on odd numbers to compute the label of (level, [selflabel, ordinal]), whereby the level denoted the node's position in the tree's hierarchical depth, the parent is the self-label of the parent node. Ordinal is the order sequencing of the current node based on the formula 2n+1, while selfLabel is generated from the parent * ordinal label [14]. It started with the root node being annotated with 1. Subsequently, the first node of the ordinal at level 1 is 3 (calculated based on 2(1)+1), and the second node of ordinal at level 1 is 5 (calculated based on 2(2)+1). The third node of the ordinal at level 1 is 7 (calculated based on 2(3)+1).

The P-C relationship is the child label that is inherited from the parent label. It can be defined by using the formula selfLabel/ordinal. The parent node is usually one level below the child node. For the A-D relationship, it can be determined using all four rules shown in Figure 2.

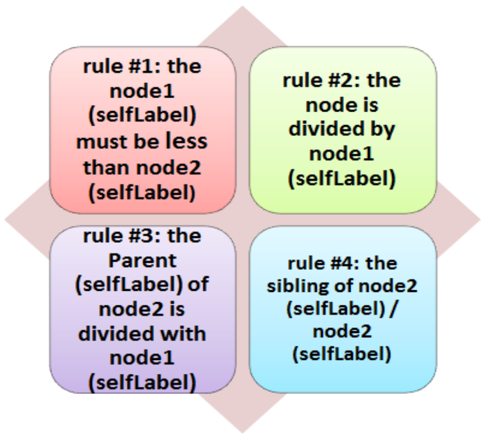

rule #1: the node1 (selfLabel) must be less than node2 (selfLabel)
rule #2: the node is divided by node1 (selfLabel)
rule #3: the Parent (selfLabel) of node2 is divided with node1 (selfLabel)
rule #4: the sibling of node2 (selfLabel) / node2 (selfLabel)

Figure 2 The four rules to determine A-D in ME labeling (Subramaniam & Haw, 2014)

Insertion of a new node in ME labeling can be inferred as the new node is going to be inserted in between NodeA and NodeB (see Figure 3). Presume that NodeC is the newly inserted node with selfLabel new self C and ordinal as new ordinal C The group of new self C, and new ordinal C for the Node C is as follows:

a) new self C = (self B)(ordinal A) + (self A)(ordinal B)
b) new ordinal C = new self C/parent of node A or Node B.

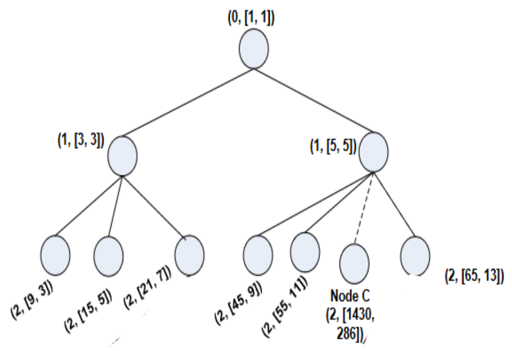


Figure 3 New node insertion in ME labeling

<u>2.1.3. ORD-GAP</u>

ORD-GAP (Haw et al., 2021) is a robust labeling scheme that assigns unique identifiers with dynamic gaps (calculated based on maximum fan-outs and maximum depth) on the initial labeling to allow future insertion. The nodes were labeled with *(s-e)l*, where the *s* and *e* are the start range and end range assigned based on depth-first traversal, while *l* is the level of the node. The *s* and *e* are computed based on the gap *g*, whereby *g* is calculated based $g = \Sigma(\max_{\text{fan-out}} + \max_{\text{depth}})$. Figure 4 depicts the partial labeling on the Sigmod dataset (UW, 2022). In this dataset, the $\max_{\text{fan-out}}$ is 4 and, the $\max_{\text{depth}}$ is 6; thus, *g* will be computed as 10. The XML tree is traversed based on depth-first traversal in the annotation process. As such, the value in the initial tree for node "issue" will be assigned with 11 (by adding the *s* value with *g* value, i.e., 1+10), followed by node "author" with 21. Next, the value *e* will be computed when a leaf node is reached. In this case, if the *s* label is 31 and is a leaf node, then the *e* value will be assigned with 41, followed by the node "issue" with 51. As for new insertions, the ORD-GAP adopted the ORDPath scheme for any future node insertions. ORD-GAP supports all three types of insertions, i.e., the left-most, right-most, and in-between.

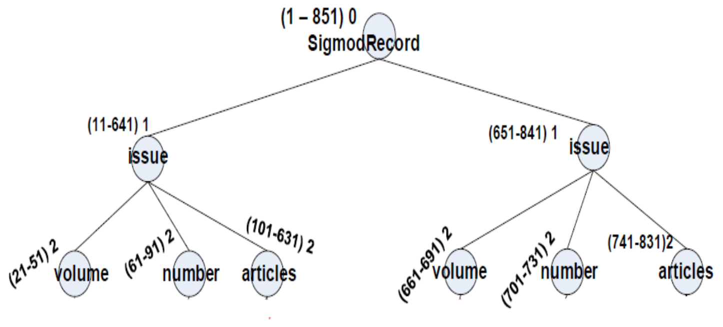


Figure 4 Partial view of SIGMOD dataset with ORD-GAP labeling

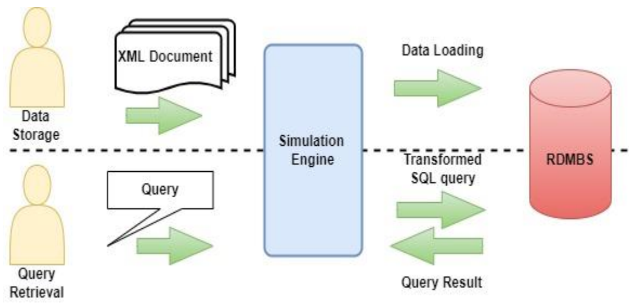


Figure 5 The architecture of the Simulation Engine

Figure 5 depicts the architecture of the simulation engine, which consists of two main processes: (i) Data Storage, and (ii) Query Retrieval. The loaded XML document will be annotated in the data storage process with the ORDPath, ORD-GAP, and ME labeling, respectively. The annotated XML document will be mapped into RDB storage. The query retrieval process will transform the user query (XPath) into the corresponding Structure Query Language (SQL). Then, the result will be returned to the user.

2.2. *Experimental Setup*

The PSD7003 dataset was obtained from the University of Washington repository (UW, 2022). PSD7003 is an annotated protein sequence database with a data size of 723 MB, a skewed structure with a min depth of 3, a max depth of 7 and an average depth of 5. This dataset is being selected for two reasons: (1) huge size and (2) skewed structure. We would like to see how the approaches performed under these two conditions. All tests were performed on Intel(R) Core (TM) i7-3770 CPU @ 3.4GHZ (64bit). The experiment carried out included the evaluation of data storing and loading time, storage space evaluation, and query retrieval evaluation. Figure 6 illustrates the evaluation process.

Data storing and loading time evaluation
Storage Space evaluation
query retrieval evaluation

Figure 6 The experimental evaluation process

3. Results and Discussion

3.1. *Data Storing and Loading Time*

In this evaluation, the XML document is mapped into RDB. As reported by (Al-Badawi et al., 2014) and (Haw & Song, 2021) respectively, since the structure of XML is semi-structured (ranging from skewed to flat design), the performance studies usually considered only the running time without considering other evaluations such as memory and CPU usage. As such, we will be evaluating on time taken to insert data.

The size of data being stored in RDB and the time taken for the storage were recorded. This paper is an extension of (Haw et al., 2021) on a larger dataset to identify which approach is more suitable to support a large dataset. The evaluations were executed four consecutive times, but only three executions were taken as the average time (see Table 1). The first execution may contain some buffer time, and thus we eliminated the result of the first run.

Insertion time on the PSD7003 dataset shows that ORD-GAP is the fastest, followed by ME labeling and ORDPath respectively. The labeling size of ORDPath and ME labeling have tremendous growth when the data insertion runs on a large dataset (Haw et al., 2021), (Khanjari & Gaeini, 2018).

Table 1 Data insertion based on various schemes

ORDPath (mins)	ME labeling (mins)	ORD-GAP (mins)
450.63	381.51	301.04

3.2. *Data Storage Space*

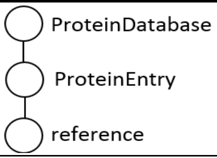
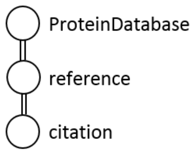
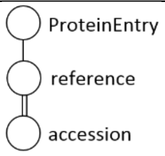
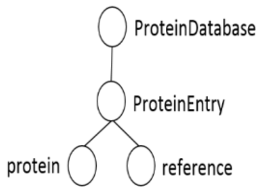
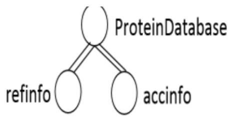
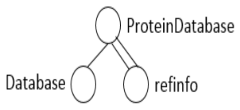
Mapping schemes is a technique where XML data is transformed into RDBs, which is row and column basis. Table 2 shows the summary of the overall database size evaluation. As can be observed, data storage for ORD-GAP produced a more significant size as compared to ORDPath. This is due to the ORD-GAP reserving more space “gap” for later insertion to support dynamic updates. However, when comparing ORD-GAP to ME labeling

ME labeling incurred more extensive storage as the label uses multiplication, which resulted in a large label size (Taktek & Thakker, 2020).

Table 2 Data storage based on various schemes

ORDPath (MB)	ME labeling (MB)	ORD-GAP (MB)
4019	4854	4483

Table 3 XPath Notation

Query	Query Node	XPath Notation
PQ1:		/ProteinDatabase/ProteinEntry/reference
PQ2:		//ProteinDatabase//reference//citation
PQ3:		//phdthesis/title
TQ4:		/ProteinDatabase/ProteinEntry[/protein/refe rence]
TQ5:		//ProteinDatabase[//refinfo//accinfo]
TQ6:		/ProteinDatabase/Database[//refinfo]

3.3. Query Retrieval Evaluation

Two types of queries are used in the SQL query retrieval experiment: the Path query (PQ) and the Twig query (TQ). Figure 7 depicts the types of query test cases. Table 3 displays the question expressed in XPath notation that is input as the test cases. These queries will be translated into the corresponding SQL statements. Table 4 depicts the two examples of SQL statements for the complex query for path query (PQ3) and twig query (TQ6) respectively. As can be observed from Table 4, both ME labeling and ORDPath require several joins to obtain the results (Qtaish & Alshudukhi, 2022). ME labeling needs several comparisons to determine if one relationship is in A-D, while for ORDPath, the relationship can be checked by using the prefix and node name comparison.

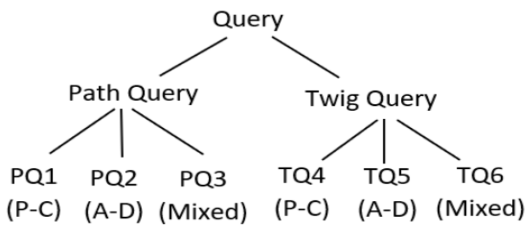


Figure 7 Query test cases (PQ1 to TQ6)

Table 4 SQL commands on various approaches

	Corresponding SQL commands		
Query	ORDPath	ME labeling	ORD-GAP
PQ3	Select SELFLABEL, (CHILDNAME) from [XMLDB].[dbo].[PARENTTABLE REED] where (CHILDNAME='accession') and SELFLABEL > any (select SELFLABEL from [XMLDB].[dbo].[PARENTTABLE REED] child where child.CHILDNAME = 'reference') and SELFLABEL < any (select SELFLABEL from [XMLDB].[dbo].[PARENTTABLE REED] c where c.CHILDNAME = 'reference' and PARENTNAME ='ProteinEntry')	Select SELFLABEL, (CHILDNAME) from [XMLDB].[dbo].[MEPARENTTABL E] where (CHILDNAME='accession') and SELFLABEL > any (select SELFLABEL from [XMLDB].[dbo].[MEPARENTTABL E] child where child.CHILDNAME = 'reference') and SELFLABEL < any (select SELFLABEL from [XMLDB].[dbo].[MEPARENTTABL E] c where c.CHILDNAME = 'reference' and PARENTNAME ='ProteinEntry')	Select start, (value) from XMLDB.dbo.itable where (Value='accession') and start > any (select START from XMLDB.dbo.itable child where child.Value = 'reference') and [end] < any (select [end] from XMLDB.dbo.itable c where c.Value = 'reference' and Pvalue ='ProteinEntry')
TQ6	Select SELFLABEL, CHILDNAME from [PSD].[dbo].[PARENTTABLE REED] where CHILDNAME ='Database' and PARENTLABEL in (Select IDNODE from [PSD].[dbo].[PARENTTABLE REED] c where c.CHILDNAME = 'ProteinDatabase') union Select SELFLABEL, (CHILDNAME) from [PSD].[dbo].[PARENTTABLE REED] where SELFLABEL > any (select SELFLABEL from [PSD].[dbo].[PARENTTABLE REED] child where child.PARENTNAME = 'ProteinDatabase') and SELFLABEL < any (select SELFLABEL from [PSD].[dbo].[PARENTTABLE REED] child where child.PARENTNAME = 'ProteinDatabase') and CHILDNAME = 'refinfo	Select SELFLABEL, CHILDNAME from [PSD].[dbo].[MEPARENTTABL E] where CHILDNAME ='Database' and PARENTLABEL in (Select IDNODE from [PSD].[dbo].[MEPARENTTABL E] c where c.CHILDNAME = 'ProteinDatabase') union Select SELFLABEL, (CHILDNAME) from [PSD].[dbo].[MEPARENTTABL E] where SELFLABEL > any (select SELFLABEL from [PSD].[dbo].[MEPARENTTABL E] child where child.PARENTNAME = 'ProteinDatabase') and SELFLABEL < any (select SELFLABEL from [PSD].[dbo].[MEPARENTTABL E] child where child.PARENTNAME = 'ProteinDatabase') and CHILDNAME = 'refinfo'	Select START, value from SIG.dbo.ITABLE where value ='Database' and PSTART in (Select IDNODE from SIG.dbo.ITABLE c where c.PVALUE = 'ProteinDatabase') union Select start, (value) from SIG.dbo.itable where START > any (select START from SIG.dbo.itable child where child.PVALUE = 'ProteinDatabase') and [END] < any (select [end] from SIG.dbo.itable child where child.PVALUE = 'ProteinDatabase') and value = 'refinfo'

Figure 8a depicts the path query evaluation, while Figure 8b illustrates the twig query evaluation results. ME labeling took a longer time to run PSD7003 as this dataset is unstructured data that requires multiple recursive joins. In addition, determining the A-D relationship requires several comparisons to confirm if the nodes are in the A-D relationship. ORDPath requires more time than ORD-GAP due to its traversal method based on breadth-first search, which travels level by level in the XML tree. As the number of siblings increased, the size label also increased.

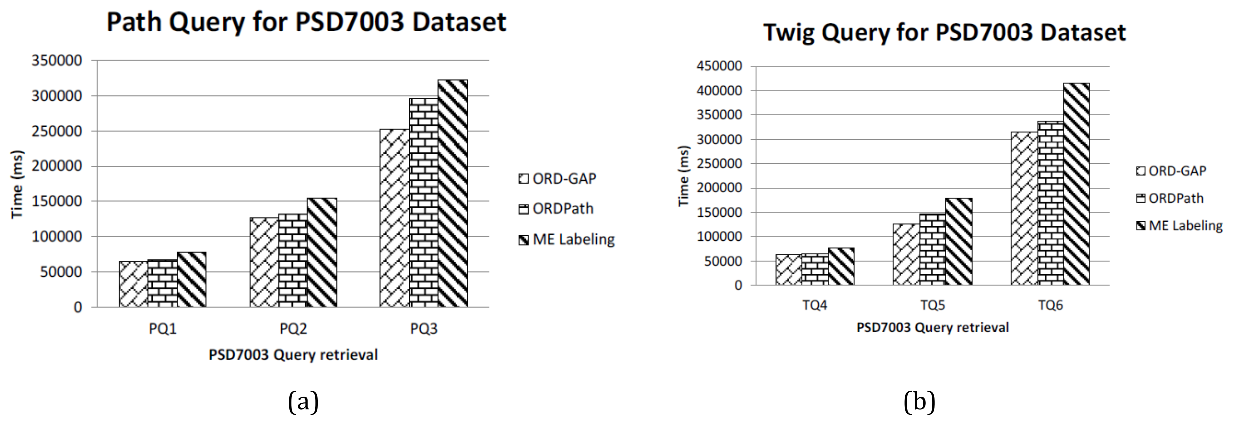


Figure 8 Evaluation of (a) path queries (b) twig queries retrieval

4. Conclusions

This paper discusses the performance evaluation of the three labeling schemes, i.e., ORD-GAP, ORDPath, and ME Labeling. From the observation, we noticed that ORD-GAP does not have the minimum storage size compared to ORDPath because it reserved a gap between nodes to maintain later insertion. In the query part, we evaluated the performance of the path query and twig query for ORD-GAP, ORDPath, and ME Labeling. It was observed that ME Labeling took a long time on both path and twig queries, especially for queries with A-D and mixed relationships. In our future work, we will evaluate dynamic updates regarding the time taken to insert sub-trees and nodes on the proposed approach. We will also look into the complexity analysis of each algorithm.

Acknowledgements

This research work has not received any grant from any funding agency. The authors are grateful to the University for granting them access to equipment to produce this work.

References

Ahn, J., Im, D.H., Kim H.G., 2016. A Mapreduce-Based Approach for Prefix-Based Labeling of Large XML Data. *In*: Joint International Semantic Technology Conference, pp. 83–98

Ahn, J., Im, D.H., 2020, Efficient Access Control of Large-Scale RDF Data Using Prefix-Based Labeling, *IEEE Access*, Volume 8, pp. 122405–122412

Al-Badawi, M., Al-Hamdani, A., Baghdadi, Y., 2014, A Comprehensive Evaluation of a Bitmapped XML Update Handler. *International Journal on Advances in Internet Technology*, Volume 7(1-2), pp. 1–16

Alsubai, S., North, S., 2017. A Prime Number Approach to Matching an XML Twig Pattern including Parent-Child Edges. *In:* 13th International Conference on Web Information Systems and Technologies, pp. 204–211

Amin, A., Haw, S.C., Subramaniam, S., Song, E., 2018. Labeling Schemes to Support Dynamic Updates on XML Trees: A Technical Review. *In:* Knowledge Management International Conference (KMICe) 2018, 25 –27 July 2018, Miri Sarawak, Malaysia

Azzedin, F., Mohammed, S., Ghaleb, M., Yazdani, J., Ahmed, A., 2020. Systematic Partitioning and Labeling XML Subtrees for Efficient Processing of XML Queries in IoT Environments. *IEEE Access*, Volume 8, pp. 61817–61833

Dhanalekshmi, G., Asawa, K., 2021. An efficient Prefix-Based Labelling Scheme for Dynamic Update of XML Documents. *International Journal of Advanced Intelligence Paradigm*, Volume 18(4), pp. 464–480

Gupta, S., Narsimha, 2015. Performance Evaluation of Nosql-Cassandra over Relational Data Store-Mysql for Bigdata. *International Journal of Technology*, Volume 6(4), pp. 640–649

Haw, S.C., Amin, A., Wong, C.O., Subramaniam, S., 2021. Improving the Support for XML Dynamic Updates Using a Hybridization Labeling Scheme (ORD-GAP). *F1000Research*, Volume 10(907), pp. 1–14

Haw, S.C., Song, E., 2021. Transforming Data-Centric Extensible Markup Language into Relational Databases Using Hybrid Approach. *Bulletin of Electrical Engineering and Informatics*, Volume 10(6), pp. 3256–3264

Hsu, W.C., Liao, I.E., 2020. UCIS-X: An Updatable Compact Indexing Scheme for Efficient Extensible Markup Language Document Updating and Query Evaluation. *IEEE Access*, Volume 8, pp. 176375–176392

Jittawiriyanukoon, C., Srisarkun, V., 2018. An Approximation Method of Regression Analysis in Concurrent Big Data Stream. *International Journal of Technology*. Volume 9(1), pp. 192–200

Kapisha, S.J., Lakshmi, G.V., 2020. Exploring XML Index Structures and Evaluating C-Tree Index-based Algorithm. *In:* 3rd International Conference on Intelligent Sustainable Systems (ICISS), pp. 212–218

Khanjari, E., Gaeini, L., 2018. A New Effective Method for Labeling Dynamic XML Data. *Journal of Big Data*, Volume 5, pp. 1–17

O'Neil, P., O'Neil, E., Pal, S., Cseri, I., Schaller, G., Westbury, N., 2004. ORDPATHs: Insert-friendly XML node labels. *In:* ACM SIGMOD International Conference on Management of Data, pp. 903–908

Pasnur, Arifin, A.Z., Yuniarti, A., 2016. Query Region Determination based on Region Importance Index and Relative Position for Region-based Image Retrieval. *International Journal of Technology*, Volume 7(4), pp. 654–662

Qtaish, A., Alshudukhi, J., 2022. An Efficient Prefix-Based Labeling Scheme for XML Dynamic Updates Using Hexagonal Pattern. *IEEE Access*, Volume 10, pp. 57107-57123

Subramaniam, S., Haw, S.C., 2014. ME Labeling: A Robust Hybrid Scheme for Dynamic Update in XML Databases. *In:* IEEE International Symposium on Telecommunication Technologies, pp. 126–131,

Taktek, E., Thakker, D., 2020. Pentagonal Scheme for Dynamic XML Prefix Labeling. *Knowledge-Based Systems*, Volume 209, p. 106446

Tatarinov, I., Viglas, S.D., Beyer, K., Shanmugasundaram, J., Shekita, E., Zhang, C., 2002, Storing and Querying Ordered XML using a Relational Database System. *In:* Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, Madison, Wisconsin

University of Washington (UW), 2022. XML Repository. Available online at https://aiweb.cs.washington.edu/research/projects/xmltk/xmldata/www/repository.html, Accessed on August 24, 2022