An Algorithm for Clustering XML Data Stream Using Sliding Window

Guojun Mao
College of Information
Central University of Finance and Economics
Beijing, China
maximmao@hotmail.com

Mingxia Gao, Wenji Yao
School of Computer Science
Beijing University of Technology
Beijing, China
gaomx@bjut.edu.cn;yaowenji@gmail.com

***Abstract*—This paper proposes an algorithm for clustering XML data stream using sliding window. It is a dynamic clustering algorithm based on XML structure. Firstly, we use level structure to represent XML document, which is based on temporal clustering feature. This structure is suitable for extracting information from XML document structure and calculating similarity between XML documents. Secondly, we use the sliding window technique, which adopts exponential histogram of XML cluster feature as a micro-cluster of it. By using the model, we can dynamically accept the new data and get rid of the old data thereby getting a better distribution feature of the current window. Finally, the experimental results based on real and synthetic XML datasets show that our algorithm not only achieves the real-time requirements of the online clustering, but also gains better clustering quality and faster processing speed.**

Keywords-XML data stream; sliding window

I. INTRODUCTION

With the development of the Web applications, large amount of information is created, exchanged and stored in the form of Extensible Markup Language (XML) format. XML has the nature of flexibility and self-describing. Users can use XML documents to represent data according to their own needs. Many modern applications, such as stock information, real-time news subscription and release detection, often generate a new data format, which we call XML data stream. Discovering useful knowledge from XML data stream is an important research topic and faces many challenges. In order to discover more useful knowledge, many researchers focused on clustering XML documents and proposed a number of algorithms. However, the existing algorithms of clustering XML documents are mainly aimed for static datasets and generally need to repeatedly scan and parse the documents many times. They did not pay much attention to the time-varying online clustering context. The methods of clustering XML documents can be divided into two categories: (1) Researches based on semantics that consider both the structure of each node and its semantic information [13]. (2) Researches based on structure of XML that do not take the semantic information of XML documents into consideration [1].

In this paper, we propose an algorithm of clustering XML data stream called SW-XSCLS, which is based on the second strategy stated above. Our algorithm uses sliding window technology and takes exponential histogram of cluster feature as its summary of the data structure. It can dynamically eliminate the outdated data and get a better understanding of the data distribution in the current window.

The contribution of our algorithm can be stated as follows: (1) we extend the conventional method of clustering XML documents to clustering the XML data stream; (2) we successfully apply sliding window technique to clustering XML data stream and create the SW-XSCLS algorithm; (3) the experimental results based on real and synthetic XML datasets show that our algorithm not only achieves the real-time requirements of online clustering, but also gains better clustering quality and faster processing speed.

The rest of this paper is organized as follows. Section 2 is about related work of processing XML documents and clustering data stream. Section 3 is the problem statement and Section 4 shows our method for clustering XML stream using sliding window. In Section 5, we show the experimental result and do some analysis from different aspect. Section 6 is about the conclusion and future work.

II. RELATED WORK

The traditional way of calculating the similarity between XML documents is the editing distance method. It computes the minimum cost for converting one tree to another tree. These operations include changing, inserting and deleting the node name. Costa et al. [1] use the idea of editing distance which converts each XML document to an ordered tree structure. They also use the "Jaccard" coefficient to calculate the similarity between XML documents. Wang et al. [2] define a standard for calculating the similarity distance. Their method need to analyze every XML document and use a directed graph to achieve the evaluation of similarity between XML documents. The accuracy of their method is not precise enough, because many XML documents that have different structures are possible to contain the same elements. So the standard has some limitations that may cause two different XML documents to be evaluated to have the same structure. Nayak [3] defines a level structure to represent the structure information of XML document and gives a similarity calculation criterion for this structure. But his method's calculation result is related to the input order of the XML documents. Changing the input order will change the result.

There is also some work related to XML schema analysis. Vincent et al. [12] address the problem of extending the definition of functional dependencies in XML documents. Balmin et al. [9] exhibit an incremental validation algorithm for XML schemas and they show it is a significant

 ISBN:978-1-61208-115-1

improvement over re-validation from scratch. Lu et al. [10] formalize the notion of the consistency of DTDs and propose a linear algorithm for checking the consistency. Elmasri et al. [11] introduce a design method for XML schemas based on well-understood conceptual modeling. They create XML schemas from a hierarchical point of view and generate SQL queries corresponding to the XML schemas.

In the clustering stream field, Zhang et al. [4] propose a method named BIRCH which incrementally and dynamically clusters incoming data points. Chang et al. [5] use two types of exponential histogram and sliding windows to cluster evolving data stream. Guha et al. [6] give constant-factor approximation algorithms for the k-Median problem and their method only need a single pass over the data. Zhou et al. [7] give a density based M-Kernel method for estimating data streams. Babcock et al. [8] present a novel technique for solving maintaining variance and k-median problems in sliding window model.

III. PROBLEM STATEMENT

A. XML Data Stream

In many researches XML data stream is defined as sequence of nodes obtained by the pre-order walk of the tree structure of XML. Processing XML data stream is analyzing these sequential nodes. But in this paper we define an XML data stream as follows:

Definition 1. An XML data stream is the sequence of XML documents (X_1, X_2 ... X_n ...) with timestamps. For any i ($i \geqslant 1$), X_i represents an XML document. The timestamp for each document is T_1 ... T_j ... and $T_i < T_j$, given i < j.

That is the arrival of each XML document is strictly time chronological. Since our study is based on the structural information, there is no restriction about the XML content.

B. Level Structure

This section mainly introduces the definition of the level structure of XML document and the process of analyzing XML data. The document is represented as an ordered tree with labels. Each tag (or element name) is denoted by a distinct integer according to their appearance order. By doing so, we eliminate the semantic information that can be inferred from the tag name and only concern about the structural information of the document. The level structure contains the hierarchy and context of the document. The multiple instances of values at same level are stored in an element since the occurrence number of an element is important for the clustering task.

The level structure of XML document contains the following information: the name, occurrence number and appear level of a node. Figure 1 shows an XML document and its level structure information.

Two level structures can be merged into a single structure. The principle is that merging the nodes at the same level. If multiple nodes with same name appear at the same level, we only save one of them. Figure 2 shows the merging process of two level structures.

An XML document can be represented by its level structure. The similarity between two XML documents is measured by the occurrence number of common elements in each corresponding level. Elements in different levels are assigned to different weights. The higher level has greater weight than the lower level. So documents with different root elements can be assigned to different clusters. Due to the page space limitation, the formula of calculating the similarity of two level structures can be referenced from Nayak [3].

C. Exponential histogram of cluster feature

Definition 2. Temporal Cluster Feature (TCF) is the collection of level structure of the XML documents in size n with the timestamp of T_1, T_2, ... T_n. TCF is denoted as (LS, n, T). The LS is the result after merging these level structures. The merging only occur between the same levels of two XML documents. The n is the number of level structure and T is the timestamp of the latest level structure in the temporal cluster feature.

The temporal cluster feature used in this paper is the extension of pseudo-time clustering features proposed in Chang et al [5].

Definition 3. Exponential Histogram of Cluster Feature (EHCF) is the collection of the TCFs. Given a set of documents X_1 ... X_n arrive at T_1 .. T_n , and let $T_i < T_j$ when i <j. According to the order of arrival, these documents are divided into several groups denoted by G_1, G_2 .... Given i <j, all documents in group G_i arrive earlier than the documents in group G_j.

Exponential histogram is the first method to generate the summary information in the sliding window model which builds buckets according to the appearance order of the elements. The capacity of the bucket at different level increases exponentially in base 2. The number of buckets at the same level is no more than the number of barrels of a pre-defined threshold.

 ISBN:978-1-61208-115-1

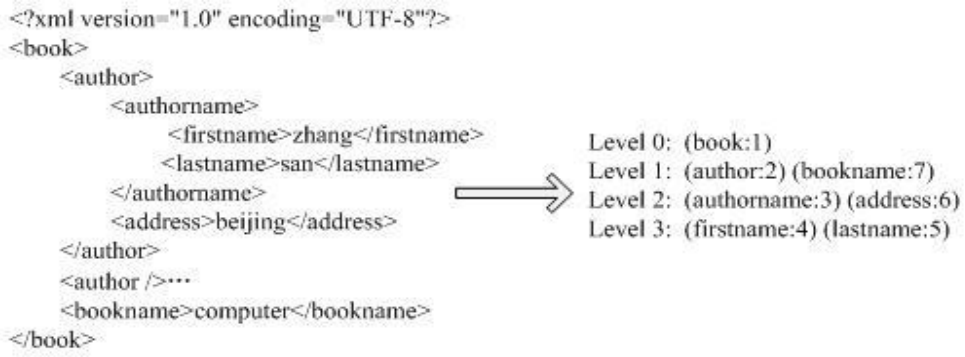

Figure 1. XML document and its level structure.

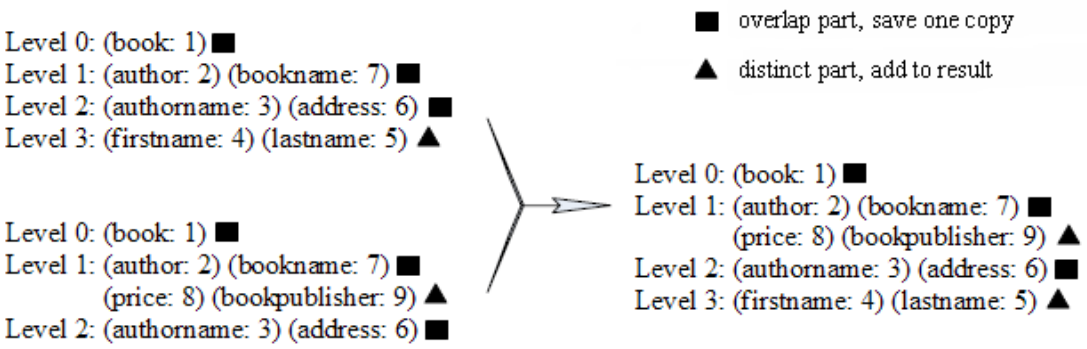

Figure 2. Merging two level structures.

IV. ALGORITHM FOR CLUSTERING XML DATA STREAM USING SLIDING WINDOW

This section focuses on the algorithm of clustering XML data stream using sliding window (SW-XSCLS). This algorithm can conduct clustering analysis for the XML data stream in the sliding window, which is maintained by a group of EHCF structures. This algorithm contains four parameters: DB is the XML data stream; W (0 <W <1) is the similarity coefficient; N is the size of the sliding window; NC is the maximum number of EHCFs maintained by one window.

In this algorithm, the count value represents the number of EHCFs maintained in the current window. For each record x in the data stream, the first step is calculating the similarity (mostSimilaryValue) between x and the EHCF (h), which has the minimum distance with x. Then we test whether the mostSimilaryValue is greater than the similarity threshold W. If it is greater than W, then x is added to the current EHCF. Otherwise we will create a new EHCF, which only contains the element x and increase count value by 1. If the number of the EHCFs reaches NC, we need to merge the most recent two EHCFs.

The next step is updating the EHCF which contains the expire record. Since the operation is executed at arrival of each new record, so at any time there is at most one EHCF contains the expired elements. As we know that EHCF is the collection of TCFs and each TCF contains a timestamp T, we can find out the expired data by using TCF. If the timestamp T of the TCF is not belonged to N, we can delete the TCF. If the last TCF in an EHCF is deleted, then the whole EHCF is deleted too. As a result, the count value, which is used to denote the number of EHCFs in the current window, will be decreased by 1.

We design two implementations for the calculating of the similarity between XML documents:

1. Merging all of the TCFs as a TCF in the EHCF (called allTCF) to calculate the similarity.

2. Using the latest TCF (called newestTCF) as the representative of the EHCF to calculate the similarity.

In both implementations, the first method merges all of the TCFs in the EHCF. So the accuracy of the calculating result is relatively high. The second method, which uses the latest TCF as a representative of the EHCF, has a faster calculating speed. In practice, using which kind of these two implementations is determined by the criteria that we concern about, that is the accuracy or the efficiency.

The whole process of the algorithm is described in Figure 3 as follows.

 ISBN:978-1-61208-115-1

```
DB: XML data stream
W(0 <W <1): similarity coefficient
N: window size
NC: the maximum EHCF number
GW-XSCLS (DB, W, NC, N)
begin
  initial count to 0;
  repeat each record x in DB
    if x is the first record
      generate an EHCF containing only TCF(x);
      add EHCF to current window;
    else
      get h, the most similar EHCF with TCF(x);
      get the mostSimilaryValue;
      if mostSimilaryValue > W
        Insert x into h;
      else
        generate an EHCF containing only TCF(x);
        increase count by 1;
        if count > NC
          merge the two similar EHCF;
          decrease count by 1;
        end
      end
    end
    if sum(x) > N
      get EHCF, the EHCF containing the oldest TCF;
      delete the Oldest TCF;
      if EHCF is null
        delete the EHCF;
        decrease count by 1;
      end
    end
  end
end
```

Figure 3. SW-XSCLS algorithm.

V. EXPERIMENTAL EVALUATION AND DISCUSSION

A. Datasets

We use both synthetic and real datasets in the experiments. The synthetic dataset is automatically generated by an XML generation tool called Oxygen. Because the number of real classes, the number of nodes and node levels of the artificial dataset can be controlled artificially, we can use it to test the processing time. The real dataset XMLFiles is the same as the dataset of static clustering algorithm XCLS in [3]. This dataset is composed of 460 XML documents from 23 natural areas, which includes 74 documents about films, 22 documents about universities, 208 documents about cars, 16 documents about literature, 38 documents about companies, 24 documents about accommodation, 10 documents about tourism, 10 documents about orders, 4 documents about auction, 2 documents about stipulation, 15 documents about pages, 2 documents about books, 20 documents about games, 12 documents about associations, 2 documents about health care and 1 document about nutrition. The labels of documents range from 10 to 100 and levels from vary from 2 to 15.

B. Evaluation criteria

The performance of clustering XML documents is evaluated using the standard criteria named intra- and inter-cluster similarity. They are internal cluster quality evaluation criteria.

The intra-cluster similarity measures the cohesion within a cluster, how similar two documents in a cluster are. This is calculated by measuring the level similarity between each pair of documents in the cluster. The intra-cluster similarity of a cluster C_i [3] is the average of all pair-wise level similarities (between two trees) within the cluster, where n is the number of documents in C_i.

$$IntraSim(C_i) = \frac{\sum_{i=1}^{n}\sum_{j=i+1}^{n} LevelSim_{i,j}}{0.5 \times n \times (n-1)}. \quad (1)$$

The intra-cluster similarity of a clustering solution in the window C= {C_1, C_2...C_k} [3] is the average of the intra-cluster similarities of all clusters taking into consideration the number of documents within each cluster, where n_i is the number of documents in C_i, N is the total number of documents and k is the number of clusters in the solution. The higher the intra-cluster similarity value is, the better the clustering solution is.

$$IntraSim = \frac{\sum_{i=1}^{k} IntraSim(C_i) \times n_i}{N}. \quad (2)$$

The inter-cluster similarity [3] measures the separation among different clusters. It is calculated by measuring the level similarity between two clusters. The inter-cluster similarity of the clustering solution is the average of all pair-wise level similarities of two clusters. The Level Similarity between two clusters is defined as similar to two documents, using the objects as clusters.

$$InterSim = \frac{\sum_{i=1}^{k}\sum_{j=i+1}^{k} LevelSim_{i,j}}{0.5 \times k \times (k-1)}. \quad (3)$$

In the experiment, unless specified, the parameters are set as follows: window size N = 100, similarity coefficient W = 0.8, the maximum number of histogram window NC = 50.

C. Experimental result and analysis

1) Quality evaluation: This section is the clustering quality comparison between our proposed algorithm (SW-

XSCLS) and the XCLS algorithm. Both algorithms use the real dataset. Since the clustering result of the algorithm XCLS is related to the order of the input, we execute the XCLS algorithm 5 times each with a different input order and calculate the average result as the evaluation criteria. Figure 4 shows the comparison of intra-cluster similarity. Figure 5 shows the comparison of inter-cluster similarity. From these figures we can see that the SW-XSCLS algorithm gets a better clustering result than the XCLS algorithm because the SW-XSCLS algorithm uses sliding window technology, which reduces the impact of outdated data in the result.

2) Parameters: Figure 6 shows the impact of execution time along with the similarity coefficient. The impact can be seen from the figure that with the increment of the similarity coefficient W, the processing time decreases. Is is easy to understand that the larger of the W value is, the less number of EHCFs maintained in the window is. So the amount of calculation will reduce and the processing time will become less also. But this has a little effect. We can see from the figure that when the similarity coefficient increases from 0.6 to 0.9, while the processing time is increased by only 20 seconds. This is a small percentage of the total processing time. Figure 7 shows the window size and the impact on the execution time.

3) Processing time: We use XML documents with different levels and different number of nodes to evaluate the processing time for our algorithm. In order to observe the processing time, we introduce two concepts: the average number of levels and the average number of nodes. Because the synthetic dataset is available to get any number of combinations of levels and nodes, so we generate a series of data sets. Figure 8 shows variation trend of the processing time of the algorithm tested on a series datasets which average number of levels changes and average number of nodes is fixed. Figure 9 shows variation trend of the processing time of the algorithm tested on a series datasets which average number of nodes changes and average number of levels is fixed.

VI. CONCLUSIONS

This paper proposed an algorithm for clustering XML data stream using sliding window. The algorithm is a dynamic algorithm based on level structure of XML documents. It can get a better clustering quality and a faster processing speed than traditional method. However, the existing clustering feature only takes into account the level structure information, ignores the semantic information of the data element. The effect is not good when processing XML documents with the same DTD or scheme. The future work will expand the existing expression on the type of data stream to meet the need for clustering data stream.

ACKNOWLEDGMENT

This research has been supported by Beijing Municipal Key Laboratory of Multimedia and Intelligent Software Technology and the National Science Foundation of China under Grants No.60873145.

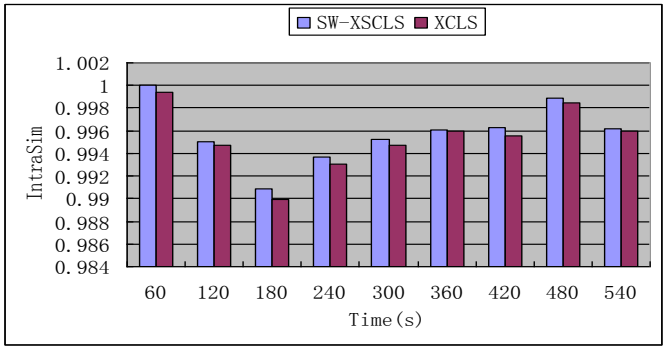

Figure 4. Intra-cluster similarity.

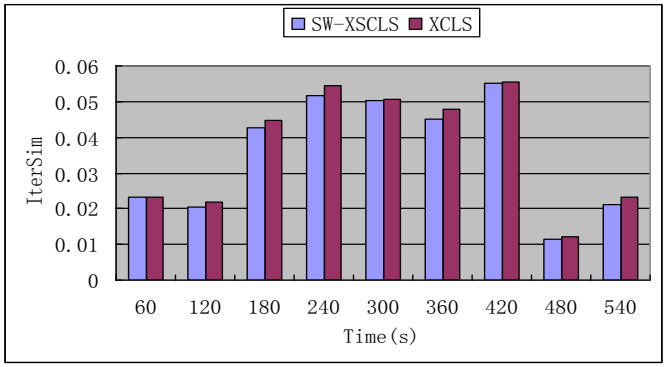

Figure 5. Inter-cluster similarity.

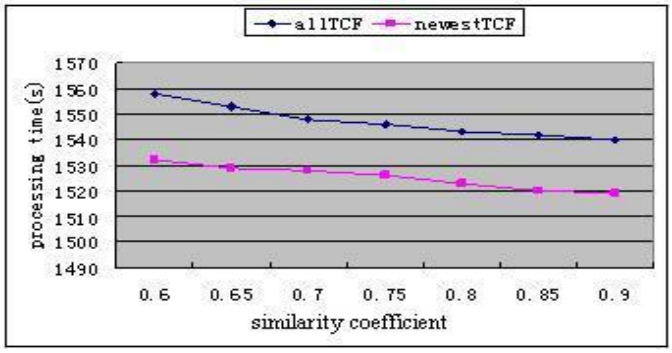

Figure 6. Changing the similarity coefficient.

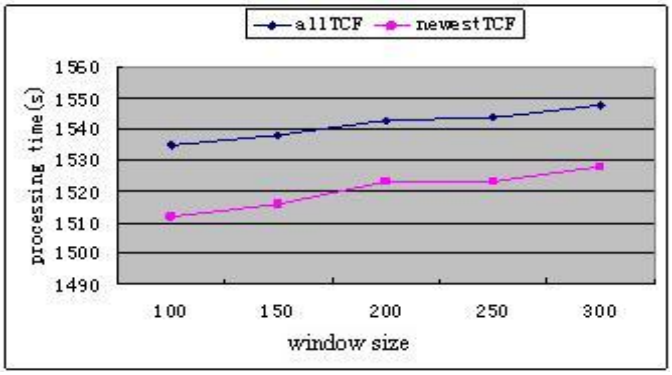

Figure 7. Changing the window size.

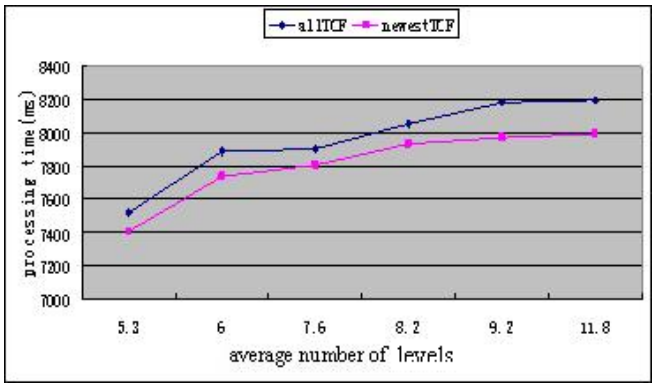

Figure 8. Processing time with number of levels.

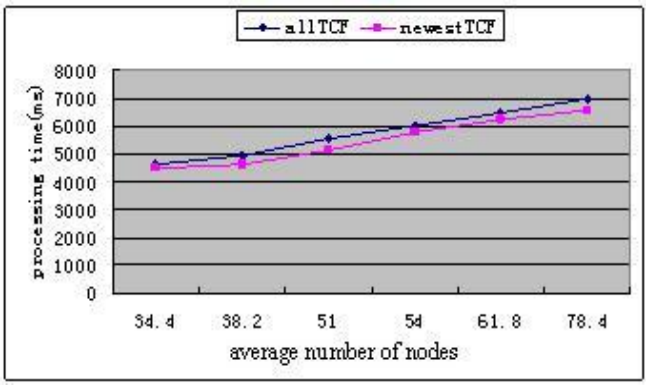

Figure 9. Processing time with number of nodes.

REFERENCES

[1] G. Costa, G. Manco, R. Ortale, and A. Tagarelli, "A tree-based approach to clustering XML documents by structure," Proc. The 8th European Conference on Principles and Practice of Knowledge Discovery in Databases, Springer-Verlag Press, Sept. 2004, pp. 137-148.

[2] L. Wang, W. Cheung, N. Marnoulis, and S.Yiu, "An efficient and scalable algorithm for clustering XML documents by structure," IEEE Transactions on Knowledge and Data Engineering, vol. 16, Jan. 2004, pp. 82-96, doi:10.1109/TKDE.2004.1264824.

[3] R. Nayak, "Fast and effective clustering of XML data using structural information," Knowledge and Information Systems, vol. 14, Feb.2008, pp. 197-215, doi:10.1007/s10115-007-0080-8.

[4] T. Zhang, R. Ramakrishnan, and M.Livny, "BIRCH: an efficient data clustering method for very large databases," Proc. ACM SIGMOD International Conference on Management of Data, ACM Press, vol. 25, Jun. 1996, pp. 103-114.

[5] J. Chang, F. Cao, and A. Zhou, "Clustering evolving data streams over sliding windows," Journal of Software, vo1. 18, Apr. 2007, pp. 905-918.

[6] S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan, "Clustering data streams," Proc. The 41st Annual Symposium on Foundations of Computer Science, IEEE Press, Nov.2000, pp. 359-366.

[7] A. Zhou, Z. Cai, L. Wei, and W. Qian, "M-kernel merging: towards density estimation over data streams," Proc. The 8th International Conference on Database Systems for Advanced Applications (DASFAA 03), IEEE Comput. Soc Press, Mar. 2003, pp. 285-292, doi:10.1109/DASFAA.2003.1192393.

[8] B. Babcock, M. Datar, R. Motwani, and L. O'Callaghan, "Maintaining variance and k-medians over data stream windows," Proc. The 22nd ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS 03), ACM Press, Jun. 2003, pp. 234-243.

[9] A. Balmin, Y. Papakonstantinou, and V. Vianu, "Incremental validation of XML documents," ACM Transactions on Database Systems, vol. 29, Dec.2004, pp. 710–751, doi:10.1145/1042046.1042050.

[10] S. Lu, Y. Sun, M. Atay, and F. Fotouhi, "On the consistency of XML DTDs," Data and Knowledge Engineering, vol. 52, Feb. 2005, pp. 231–247, doi:10.1016/j.datak.2004.05.007.

[11] R. Elmasri, Q. Li, J. Fu, Y. Wu, B. Hojabri, and S. Ande, "Conceptual modeling for customized XML schemas," Data and Knowledge Engineering, vol. 54, Jul. 2005, pp. 57–76, doi:10.1016/j.datak.2004.10.003.

[12] M. Vincent, J. Liu, and C. Liu, "Strong functional dependencies and their application to normal forms in XML," ACM Transactions on Database Systems, vol. 29, Sept. 2004, pp. 445–462, doi:10.1145/1016028.1016029.

[13] Y. Guo, D. Chen, and J. Le, "Clustering XML documents by combining content and structure," International Symposium on Information Science and Engineering (ISISE 08), IEEE Press, vol. 1, Dec. 2008, pp. 583-587, doi:10.1109/ISISE.2008.31.

 ISBN:978-1-61208-115-1