International Journal of Soft Computing 11 (5): 289-294, 2016
ISSN: 1816-9503

RL-Frag: A Framework for Query Processing over Fragmented XML Data Stream

Samini Subramaniam, Su-Cheng Haw, Lay-Ki Soon and Kok-Leong Koong
Faculty of Computing and Informatics, Multimedia University, 63100 Cyberjaya, Malaysia

Abstract: Document fragmentation has proven its ability to promise scalability and availability of information for Internet users. However, fragmenting XML document and processing queries over fragmented data has been the greatest challenge. This is due to the fact that fragmented XML documents must be implicitly tied together although they are actually disintegrated. It is also important for query optimization done locally on designated fragments to ensure the efficiency of the query processing technique. In this study, we present the conceptual ideal of a novel query processing technique over fragmented XML data, RL-Frag which adopted region-based labeling, ReLab$^+$ which will be used to preserve structural relationship between the data.

Key words: XML, technique, docoment, relationship, processing

INTRODUCTION

XML has become the de-facto standard for data exchange over World Wide Web (WWW) (Suma *et al*., 2014). Due to the flexibility of XML data presentation, this technology has been given higher attention for reliable data storage. Various techniques were proposed to optimize data processing over XML document. Due to the spike in online data retrieval, XML documents are often disintegrated and dispersed across the network to increase scalability. In a traditional database system, there are two strategies in managing data in distributed environment which are replication (Mazilu, 2010) and fragmentation or partitioning (Iacob, 2011). Replication simply means copies of data or document is replicated and distributed across the network.

On the other hand, fragmentation technique 'cuts' database into few segments. By doing this user queries will also be 'cut' into sub-queries which will be sent to fragments that hold results for sub-queries which will be processed in parallel compared to serial execution over the whole document.

This study focuses the latter approach as it is more practical in terms of storage consumption and speed of query processing. Generally, to process queries on fragmented stream of data involves three processes which are identification of fragmentation strategy, perform fragmentation while preserving its structural properties and query processing over fragmented data. Many of the existing system adopt the fragmentation theory from database systems which are Horizontal Fragmentation (HF) which divides relation into sets of disjoint tuples, Vertical Fragmentation (VF) that generates disjoint sets of columns or attributes except for the primary key and Mixed Fragmentation (MF) combines both HF and VF together. Figure 1 shows how HF and VF can be reflected in an XML document.

In conjunction with this, there are three correctness rules of fragmentation (Braganholo and Mattoso, 2014) that were proposed in the distributed database design which is indeed useful in managing distributed XML data. The rules are explained below in the context of XML document.

Rule 1 completeness: Decomposition of an XML document, XD into fragments F_{X1}, F_{X2}, F_{X3}, …, F_{Xn} is complete if each element in XD can be found in some F_{Xi}.

Rule 2 reconstruction: XD must be reconstructed from its fragments using some computations, Δ. For an example; $XD = F_{X1} \Delta F_{X2} \Delta F_{X3} \Delta, \ldots, \Delta F_{Xn}$.

Rule 3 disjointness: If an element, d_i appears in fragment, F_{Xj}, then it should not appear in any other fragment, F_{Xk}, $k \neq j$ (exception: primary key attribute for vertical fragmentation).

By adopting these rules as the basis of our framework, we propose conceptual idea of a novel query processing technique over fragmented data, RL-Frag which able to perform fragmentation while maintaining the structural properties between elements and easy query processing technique through the adaptation of a labeling scheme, ReLab$^+$.

Corresponding Author: Samini Subramaniam, Faculty of Computing and Informatics, Multimedia University, 63100 Cyberjaya, Malaysia

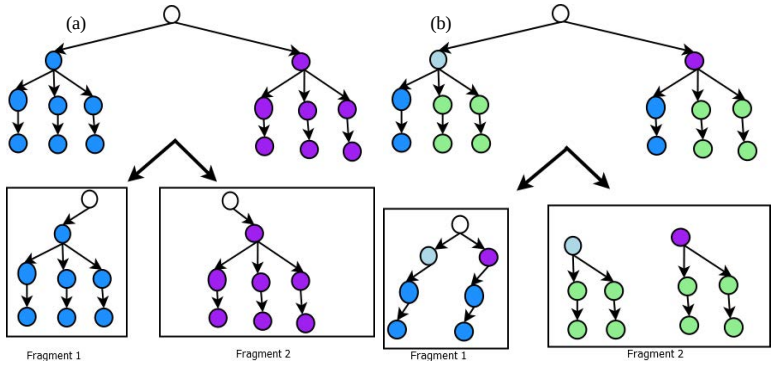


Fig. 1: Horizontal and vertical fragmentation on XML document: a) HF and b) VF

Literature review: In the past there were many studies were conducted on processing queries on fragmented data stream. A method called XFrag (Bose and Fegaras, 2005) was developed based on hole-filler model to describe structural relationship between the fragments. Each fragment contains holes which will be filled by another fragment which may contain hole. The chain relationship between the fragments using hole-filler concept enable the original document to be reconstructed from the fragments. This technique generates a tag structure which describes the structure of data that will be useful for fragmentation later. At minimum, tag structure document consists of tag element where it's occurrence is repeated for every element in an XML document. Each tag element will be qualified by distinct id, name (element name in the original XML document) and a type. The value of type is either 'FILLER' or 'EMBEDDED'. If the type if filler, then it is noted that the element will be fragmented separately and if the type is embedded, the element will be embedded in the fragment that holds the parent element. Structural relationship can be determined using the tag structure ID (tsid) which the unique identifier of a fragment. The fragment will only be processed if tsid matching the ID in tag structure.

Later, XFPro was proposed (Huo *et al*., 2006) to improve the technique proposed by the previous work. The main objective of XFPro was to overcome the space and time limitations by XFrag. XFPro presents technique to transform XPath expressions to optimized query plan. On top of that they proposed a pruning scheme to eliminate redundant operations after query reworking.

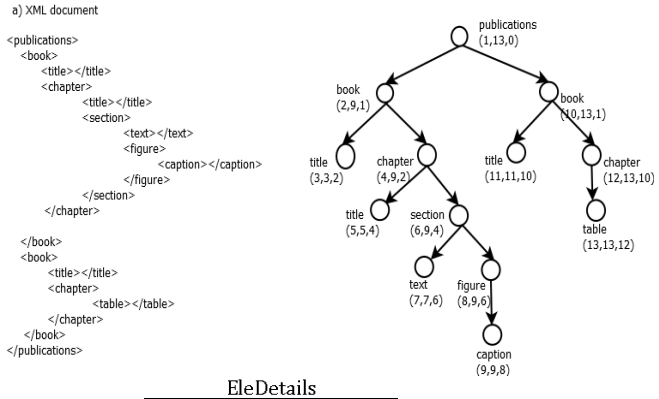
On the other hand, Lee *et al*. (2012) performed a study on the traditional hole-filler techniques and claimed they are inefficient in total memory consumption. To allow for better query processing, they integrated conventional labeling scheme such as Dewey labeling (Tatarinov *et al*., 2002) into their query processing algorithm, XFL. XFL uses the labeling scheme for Parent-Child (P-C) and Ancestor-Descendant (A-D) relationship determinations. The relationships between fragments were also supported using the labeling scheme adopted. XFL only requires fragment ID (FID) which can be used for fragment association and eliminate hole-filler altogether.

MATERIALS AND METHODS

Proposed model (RL_FRAG): Generally, RL-Frag can be divided into three main processes, XML node annotation using ReLab$^+$, the node annotator, fragmentation of original XML document into individual fragments which carries the properties of ReLab$^+$ and query processing over the fragmented data.

Node annotation using relab$^+$: Labeling mechanisms have proven their abilities in determining structural relationship among elements which is crucial to support queries with structural edges. In contrast with the researches by Lee *et al*. (2012) we have utilized a region-based labeling scheme for node and fragment identification which is more efficient in terms of space consumption. Every node in an XML document must be uniquely identified by labels generated using a labeling scheme. In this case, the nodes will be labeled using ReLab$^+$ which is an extension of ReLab (Subramaniam *et al*., 2014). Each node in the XML document will be labeled as (self, region, parent) where self refers to unique ID starts from 1 and increment by 1 as the tree is traversed using depth-first traversal. Region refers to the self-label of the right-most node in a subtree which will be useful to determine structural relationship between two or more nodes and parent label refers to the self-label of the parent node.

As soon as the nodes are annotated using ReLab$^+$, these information will be stored in a hash table, EleDetails, for further processing. Key column stores the self-label of



EleDetails

Key	Value
1	publications
2	book
3	title
4	chapter
5	title
...	...

Fig. 2: XML tree labeled using ReLab$^+$ and an example of EleDetails table

nodes and value table stores the element names of the nodes. Figure 2 shows an example of XML tree that was labeled using Relab$^+$ as well as a sample of EleDetails table.

Generation of strucuture.xml document: For each XML document, one structure.xml document will be created that contains the structure of data and details of fragmentation of original XML document. Each element that appears in the original XML document contains five attributes in structure document which is presented in the DTD document as:

- <!DOCTYPE Structure [
- <!ELEMENT Structure (Struct+)
- <!ATTLIST Struct ID #REQUIRED>
- <!ATTLIST Struct name #REQUIRED>
- <!ATTLIST Struct region #REQUIRED>
- <!ATTLIST Struct parent #REQUIRED>
- <!ATTLIST Struct ftype (EMBEDDED|TRUE) # REQUIRED]>

The details of each attribute in structure document are described as follows:

- ID-self-label of element in XML document which is generated using depth-first traversal over the XML document (using ReLab$^+$)
- Name-element name that appear in the XML document
- Region-region ID of element in XML document
- Parent0-self-label of parent node
- Ftype-value of ftype will be produced based on two rules

Rule 1: If a node has no child node or only one child node; the node will be embedded into parent fragment. The value of ftype is EMBEDDED.

Rule 2: If a node has more than one child nodes, the ftype of the node will be TRUE. Next, XML document will be fragmented based on the structure.XML document. If ftype is TRUE, then a new fragment will be created and if ftype is EMBEDDED, then the element will be added to the parent fragment. Based on strucuture.XML in Fig. 3b, the fragments below will be created as in Fig. 4.

Each fragment consist of two properties which are element name and FID which is fragment ID attribute where the value of this attribute is "self, region, parent" which was generated using ReLab$^+$ earlier. Once the fragments are generated, a hash table, FragDetails will be created which stores FID of first element in the fragment as the key and the FID of subsequent elements in the fragment as the value. Figure 5 shows an example of FragDetails.

RESULTS AND DISCUSSION

Query processing over fragmented data: In this segment, a brief description on how XPath query will be processed

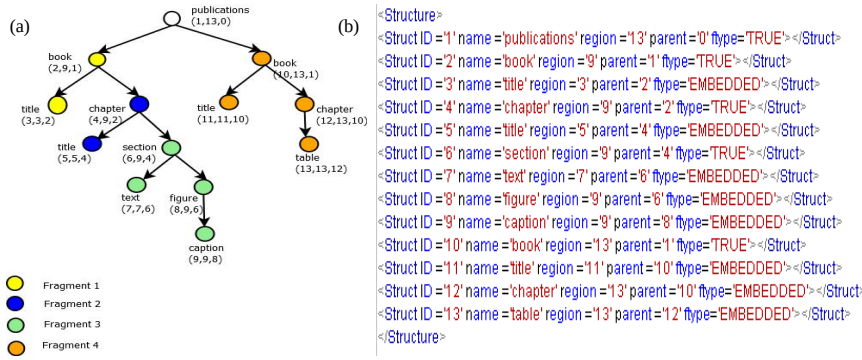


Fig. 3: a) shows how XML document will be fragmented based on ftype and b) structure.XML document

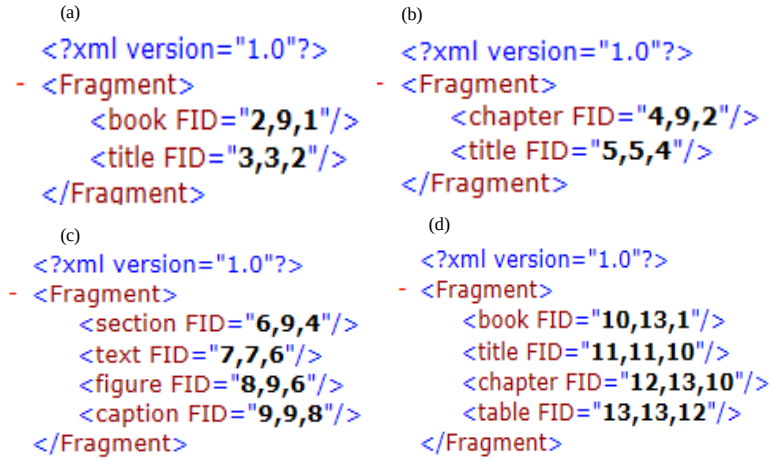


Fig. 4: Fragments from original XML document: a-d) Fragment 1-4

over XML fragment using RL-Frag is explained using an example. Based on the XML tree in Fig. 2, assume that XPath query is book/chapter/section. The first step is to obtain the self-label of all the elements in the query from EleDetails table.

- $Book_{(self\text{-}label)}$-> 2 and 10
- $Chapter_{(self\text{-}label)}$-4 and 12
- $Section_{(self\text{-}label)}$-6

Next is to identify the fragment in which the query nodes are located. This is done by matching the self-label with the FID in FragDetails which is shown in Fig. 6 and the same process will be done for every element in the query.

For each query node, a hash table will be generated which is used to store the result of P-C and A-D criteria fulfilment between two nodes. The key of the table stores the FID of the element and the value stores definer where its value can either be 'NS' (NOT SURE), 'T' (TRUE) or 'F' (FALSE) which means before the node is processed, meet relationship criteria or does not meet relationship criteria, respectively. Figure 7 shows an example of query processing pipeline.

Based on the sample query, Parent-Child relationship (P-C) is determined using the following rule; if $child_{parent}$ is equal to $parent_{self}$ then P-C exist. Otherwise, P-C does not exist. For book/chapter relationship, parent-label of chapter is compared with the self-label of book node. If the relationship is met, the value of definer is 'T' else it will be 'F'. The same method will be used to determine P-C between chapter/section as shown in Fig. 8.

Since, there is only one occurrence of section in the tree and the relationship is met, only the first chain of answer is considered. If all the definer is 'T' for each possible result, the answer will be moved to the final solution as shown in Fig. 9.

For queries with Ancestor-Descendant (A-D) edges, same method of query processing will be used except for the relationship determination, the following rule will be adopted. If $ancestor_{self}$<$descendant_{self}$< = $ancestor_{region}$ then A-D relationship exists between two nodes.

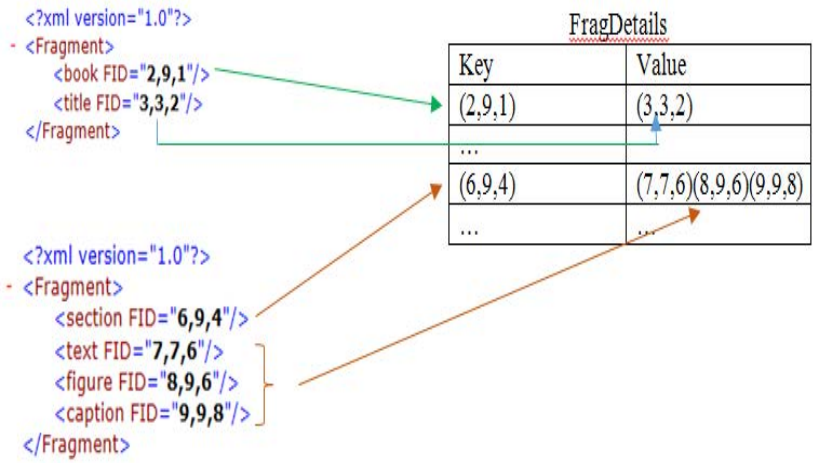


Fig. 5: FragDetails

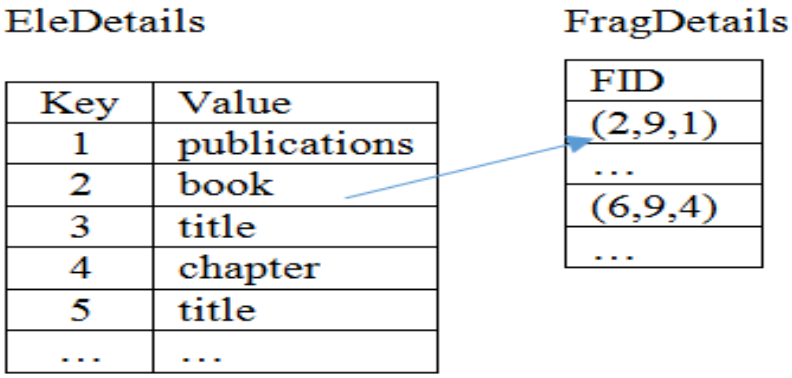


Fig. 6: Matching of self-label to FID in FragDetails table

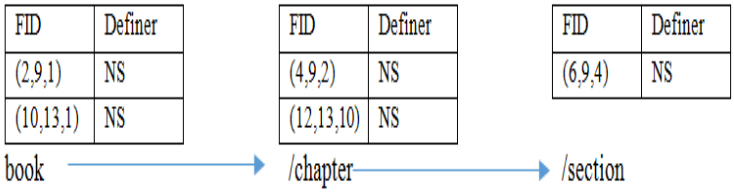


Fig. 7: Query processing 1

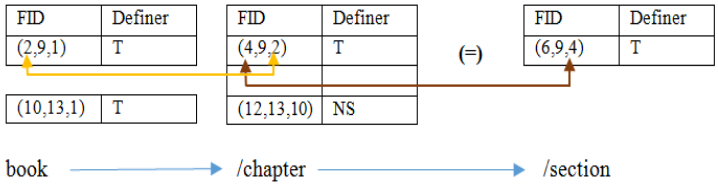


Fig. 8: Query processing 2

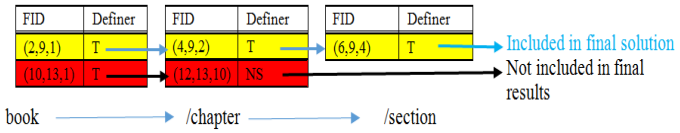


Fig. 9: Final output highlighted in yellow

Based on the procedures given, it can be seen that fragmentation of a document will not add any complexity to query processing if the structural relationships are maintained carefully. In the proposed method, ReLab$^+$ was used to generate labels in the simplest form which is used for query processing. It is also noted that only fragments relevant to query nodes are retrieved through the FragDetails table which is an added advantage to the proposed method.

CONCLUSION

It is vital for large document to be fragmented and distributed across network to ensure fast query processing and scalability. However, localization needs to be given great attention in processing queries in parallel over fragmented data. Our proposed method, RL-Frag provides mean to define structural relationship between fragments and within elements in a fragments in an efficient way through a simple node annotator, ReLab$^+$. The limitation of RL-Frag could be the excessive creation of fragments for skewed-structured XML document. In the future, Re-Frag will be implemented and evaluated in terms of its complexity and effectiveness in processing queries over fragmented data compared to the existing approaches.

REFERENCES

Bose, S. and L. Fegaras, 2005. XFrag: A query processing framework for fragmented XML data. Proceedings of the 8th International Workshop on the Web and Databases, June 16-17, 2005, Baltimore, Marylan, pp: 97-102.

Braganholo, V. and M. Mattoso, 2014. A survey on xml fragmentation. ACM SIGMOD Record, 43: 24-35.

Huo, H., G. Wang, X. Hui, R. Zhou, B. Ning and C. Xiao, 2006. Efficient Query Processing for Streamed XML Fragments. In: Database Systems for Advanced Applications, Tan, K.L. and V. Wuwongse (Eds.). Springer, New York, pp: 468-482.

Iacob, N., 2011. Fragmentation and data allocation in the distributed environments. Ann. Univ. Craiova-Math. Comput. Sci. Ser., 38: 76-83.

Lee, S., J. Kim and H. Kang, 2012. Memory-efficient query processing over XML fragment stream with fragment labeling. Comput. Inform., 29: 757-782.

Mazilu, M.C., 2010. Database replication. Database Syst. J., 1: 33-38.

Subramaniam, S., S.C. Haw and L.K. Soon, 2014. ReLab: A subtree based labeling scheme for efficient XML query processing. Proceedings of the 2nd International Symposium on Telecommunication Technologies, November 24-28, 2014, Langkawi, pp: 121-125.

Suma, D., U.D. Acharya, M. Geetha and M.R. Holla, 2014. XML information retrieval: An overview. Int. Global J. Eng. Res., 10: 26-32.

Tatarinov, I., S. Viglas, K.S. Beyer, J. Shanmugasundaram, E.J. Shekita and C. Zhang, 2002. Storing and querying ordered XML using a relational database system. Proceedings of the International Conference on Management of Data, June 3-6, 2002, Madison, Wisconsin, pp: