Indonesian Journal of Electrical Engineering and Computer Science
Vol. 18, No. 3, June 2020, pp. 1539~1550
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v18.i3.pp1539-1550

Performance evaluation on structural mapping choices for data-centric XML documents

Su-Cheng Haw, Emyliana Soong
Faculty of Computing and Informatics, Multimedia University, Malaysia

Article Info

Article history:

Received Sep 24, 2019
Revised Nov 26, 2019
Accepted Dec 10, 2019

Keywords:

Data-centric XML
Mapping scheme
Structural mapping
Transformation
XML query

ABSTRACT

eXtensible Mark-up Language (XML) has been widely used as the de facto standard for data exchange over the Web. It is crucial to ensure that the data can be mapped correctly into the underlying data storage format, that is, without any lost of information. The two mapping strategies are structural-based and model-based. The structural-based mapping involves the presence of Data Type Definition (DTD) for schema mapping while the model-based mapping does not require the present of DTD or any schema for the mapping purpose. The structural-based mapping is good especially for data-centric type of data, i.e., data which is structured and can be binded into certain schema. As such, this paper evaluates and compares the performances of two selected existing structural-based mapping via simulation. Two main evaluations are: (i) storing the XML data into relational database (RDB), and (ii) querying the XML data from the RDB. The time taken for each respective process will be recorded and compared. From the experimental results, it is observed that the s-XML approach outperformed the SAX approach in terms of storing and query evaluations for most of the test cases conducted.

Copyright © 2020 Institute of Advanced Engineering and Science.
All rights reserved.

Corresponding Author:

Su-Cheng Haw,
Faculty of Computing and Informatics,
Multimedia University, Malaysia
Email: sucheng@mmu.edu.my

1. INTRODUCTION

Internet is powerful since it works as communication platform to support any documents transmission globally such as e-transaction. However, internet needs eXtensible Markup Language (XML) for data representation because it has the ability of flexible structure on communication over World Wide Web (WWW). On the other hand, Relational Database (RDB) plays an important role in storing data for the back-end database at most of the organization. Unfortunately, the RDB has drawback of processing data independently on its context. In that case, mapping and querying XML through RDB is crucial especially on resolving the conflict between the hierarchical structure of XML and the flat structure of RDB [1, 2].

The two main mapping approaches are structural-based mapping (schema-based mapping) and model-based mapping (schema-less mapping) [3]. The structural-based mapping is good especially for data-centric type of data, i.e., data which is structured and can be binded into certain schema. Thus, in the structural-based mapping approach, the relational schema design involves the supports of Data Type Definition (DTD) or XML schema (XSD) to determine the number of relations required after the shredding process. However, structural-based mapping is not suitable to store dynamic and unstructural variant of XML documents. This is because for any new updates in the XML document, it requires re-loading of data in the relations created. Nevertheless, in structural-based mapping, the approaches can support the relationship among nodes (Parent-Child (P-C), Ancestor-Descendant (A-D), sibling and level) effectively [4].

On the other hand, model-based mapping approach involves in a fixed relational schema which is built to store XML document without any support of DTD or XSD [5]. In this approach, they support every XML applications either in static and dynamic. This is the advantage of model-based mapping which can support any variety of XML documents in WWW.

The focus of this paper is on the structural-based mapping approach since our focus is on the data-centric XML document, whereby the structural schema of the XML is not affected even if there is a change in the data. The rest of the paper is organized as follows. Section 2 review on the three selected state-of-art structural-based mapping approaches. Section 3 discusses on the experimental setup, results and analysis on the comparison between the selected approaches. Lastly, Section 4 summarizes the paper and suggested some future works.

2. LITERATURE REVIEW

Figure 1 shows the illustration sample of XML which will be used throughout the paper.

```
1 <?xml version = "1.0" ?>
2 <!DOCTYPE  university SYSTEM "university.dtd">
3 <university >
4       <student id = "1140">
5          <name> Amirah </name>
6          <enrollment>
7             <intake> October 2014 </intake>
8             <detail>
9                <faculty> FCI </faculty>
10               <course> Computer Science </course>
11               <specialization> Information System </specialization>
12            </detail>
13            </enrollment>
14      </student>
15      <student id = "1141">
16         <name> Emyliana </name>
17         <enrollment>
18            <intake> October 2014 </intake>
19         </enrollment>
20      </student>
21 </university>
```

Figure 1. XML illustrative example

2.1. DOM-based Mapping Approach

The first approach for structural-based mapping choices is based on Document Object Model (DOM). Atay et. al. proposed the Ordered XML Insert (OXInsert) [6], which is based on the idea that inlinable elements will have exactly one parent node during the schema mapping step. However, the inlinable of XML element is unknown from the DOM tree; this information is referred on the DTD. The OXInsert algorithm inserts XML document into RDB, which schema is generated from the DTD input in the prior step. OXInsert is an enhancement of their previous work, XInsert [7] with taking ordered nature into account. The schema mapping functions are as follows.

a) Use σ (e) that maps the elements into relational table.
b) Use θ (e) that maps the XML attributes to relational attribute.
c) Use δ (e) that maps leaf element to relational attribute.

In the DOM approach, Atay et al. [7] models the XML document as an ordered element tree. The tree consists of node which represents element and edge that represent as P-C relationship. Basically, each node has attributes and values. The element of the tree has a few notations which are e.name represents as XML element's name, e.EID represents as global ID of XML based on pre order tree traversal, e.endID represents as its largest descendent, e.attribute represents as set of XML attributes, e.value represents as the element value, e.parent represents as the parent node of the element and e.children represents as the ordered sequence of the child node. The e.value only exists when e is a leaf node.

Atay et al. [7] solved the problem of varying document structure. Hence, the value of missing node will have appeared as null in the columns. In some cases, for the elements that have same type and the tree structures are varied, OXInsert processes the descendants using σ mapping. They have also conducted an experimental evaluation to compare DOM-based to Simple API for XML (SAX) based approach. The result revealed that the DOM-based approach has better performance than the SAX-based approach up to 75 MB size document.

2.2. SAX-based Mapping Approach

The second approach is according to SAX-based approach which solves a linear schema mapping problem. Atay et al. [7] proposed the SAX-based Data Mapping algorithm, called SDM for short. This algorithm is event driven and only makes one run scanning of whole document. SDM deals in sequential scan of overall document that triggers few events such as startElement (), characters () and endTag () which indicates start tag, character data and end tag respectively.

Triggering the start tag under procedure startElement () generates sequential global ID (called GID for short) to ensure the order of XML document in RDB is maintained for the reconstruction of RDB into XML if any. However, there are two conditions for the element which are inlinable and non-inlinable to the parent element.

If element encounter non-inlinable element, SDM proceeds in creating new tuple, t of table σ(type (e)) and fills the fields with the information from the element, while pushing element type e and GID onto GST known as global stack, tuple t will be pushed onto stack ST σ(type (e)). This procedure applies to all non–inlinable elements until all descendants are processed.

If the element is inlinable to the parent element, no new tuple is created. However, GID and attributes values of e are updated for the tuple which is on top of the stack ST σ(type (e)). Then, the element, e and GID are pushed onto stack GST.

2.3. Simple XML (s-XML)

Good labeling is efficient to make sure the labels given to the XML nodes are uniquely identified. Hence, for the third approach of structural-based data mapping is s-XML [8], which is based on persistent labeling scheme [9]. Simple XML or better known as s-XML in short [8], utilised the persistent labeling to support the update labeling function without reconstructing the labels if any new nodes of insertion and deletion exist to the original XML document.

The idea behind the persistent labeling is to encourage quick determination of relationship between a pair of nodes and support four basic structural hierarchical relationship which are A-D, P-C, sibling and level relationship. The persistent labeling labels the node as (l, [np, dp], [n, d]) which l represents as level of node in the tree, [np, dp] represents as the self-label of parent node and [n, d] represents as the local label. The labeling pair [n, d] where the n denotes the position of the node among siblings while d is assigned as 1 in the idea of static labeling and [n, d] represents the n/d rational.

Refering to Figure 2, the root node is always at level node zero and there is no parent node existed to the root node. This applies to the node 1 as the level node is zero and since the node has no sibling, the self label is labeled as [1, 1]. Applying the rule based on the XML document, university is labeled as (0, [1, 1]) since university node is a root node. For the non-root node, there exists parent node to each element node. Based on Figure 2, the node 2 and node 3 have the allocation of parent label node of [1, 1] as for node 1's local label.

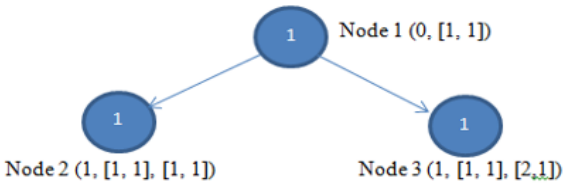


Figure 2. Assigning [n_p, d_p] and [n, d] by s-XML approach

s-XML shreds all the nodes from the tree into two respective tables which are *ParentTable* and *ChildTable*. The parent table store all internal nodes information such as IdNode, PName, CName, Level, LParent and SelfLabel which identify the node uniquely, stores the parent node's name, the child's name which is the self node's name, level of the tree information, stores the parent node's label as referencing to

the IdNode of the parent from the parent table and the self node's label. In addition, the value attribute only apply on child table to store the leaf node's value respectively.

While in the child nodes, there exists information as IdNode, Level, PName, SelfLabel, LParent and Value. For the IdNode, it uniquely identifies the current node, while for the level, it stores the tree level of the nodes. PName stores the parent node's name, the SelfLabel stores the label of the current node, LParent stores the parent node's label as referencing to the IdNode of the parent from the parent table and Value as to store the value of the current node since it is a leaf node. Table 1 and Table 2 depict the mapped result of *ParentTable* and *ChildTable* respectively.

Table 1. Sample Data of s-XML for ParentTable

ID Node	pName	cName	Level	LParent	SelfLabel
1	-	university	0	-	[1,1]
2	university	student	1	&1	[1,1]
3	university	student	1	&1	[2,1]
4	student	enrollment	2	&2	[2,1]
5	student	enrollment	2	&3	[2,1]
6	enrollment	detail	3	&5	[2,1]

Table 2. Sample Data of s-XML for ChildTable

ID Node	Level	pName	SelfLabel	LParent	Value
1	2	student	[1,1]	&2	'Amirah'
2	2	student	[1,1]	&3	'Emyliana'
3	3	enrollment	[1,1]	&5	'October 2014'
4	3	enrollment	[1,1]	&7	'October 2014'
5	4	detail	[1,1]	&9	'FCI'
6	4	detail	[2,1]	&9	'Computer Science'
7	4	detail	[3,1]	&9	'Information System'

Subramaniam et al. [8] found that s-XML has good performance because of the simple mapping approach usage and all data are distributed fairly among sufficient number of tables. In addition, the number of tables and format of the tables are fixed regardless of the XML document complexity. Other than that, with persistent labeling utilized as the labeling scheme, this data mapping approach supports structural queries retrieval efficiently and have great support for dynamic updates.

2.4. Summary of Review Approaches

Table 3 shows the comparison on the three selected mapping schemes. The DOM-based represents the XML document as a DOM tree. The main idea of the OXInsert algorithm is the algorithm takes ordered encoding while mapping the XML document into RDB with the presence of DTD. However, DOM-based approach needs two runs to complete the document scanning: the first run of the scanning is for constructing the DOM tree while the second run is accessing the DOM tree to process. Lastly, the DOM-based approach is a space concerning since the approach needs spaces in the main memory to fit in the DOM tree.

SAX-based approach needs the DTD for the schema mapping process. The SAX-based approach uses SDM algorithm to map the XML document into RDB efficiently. SDM requires only one scan to process all the information. SAX-based does not suffer to any space concerning as it does not need to construct any tree for the data mapping.

s-XML takes advantage of the persistent labeling in labeling the tree nodes for data mapping. s-XML needs only one run to construct the tree nodes with information. This approach does not require a DTD for the data mapping. The s-XML also does not suffer in any space limitation because the tree construction is simple.

Table 3. Comparison between Approaches

Features	DOM-based	SAX-based	s-XML
Require DTD	Yes	Yes	No
Scanning run	Two runs	One run	One run
Updating labelling	No	No	Yes
Speed concerning	Yes	No	No
Require tree	Yes	No	Yes
Ordered mapping	Yes	Yes	No
Algorithm	OXInsert	SDM	Mapping & Query Retrieval Algorithms

2.5. Other Recent Approaches

Lim et al. [10] evaluated on SAX and DOM approaches in their studies. They concluded that DOM is more efficient when it could fit the DOM tree in the main memory, while the SAX is better in performance when dealing with huge XML document.

Qtaish et al. [11] performed a comprehensive review on some mapping approaches. From their review, it was pointed out that the number of join operations in the translated SQL does affect the query processing. On another study, Gamal et al. [12] discovered that s-XML is most efficient in term of storage space and data retrieval as compared to relational DTD, Edge and Attribute approaches. The approach performed better in processing complex chain query during data mapping into RDB and data retrieval regardless the dataset sizes. Conversely, Machkour et al. [13] proposed a method to convert a DTD into object-relational model by preserving structural and semantic constraints. Their approach supports reversible conversion from object-relational model into XML.

Ahmad and Samad [14] proposed using normalization based on functional dependencies to map XML into RDB. They proposed XtoR algorithm, which consists of three main components: (1) defining the functional depencies for XML, (2) constructuring inference rules, and (3) the mapping function. In their paper, although there is no experimental proof, they have provided two motivating examples to demonstrate their proposed method.

Mao and Ye [15] proposed a bidirectional mapping algorithm between relational schema and XML schema. They build an intermediate object tree to transfer the data information while preserving the data structure, referential integrity and semantic constraints. On another separate research, Molnar et al. [16] proposed utilizing Conceptual Graphs to provide a graphical representation for logic which is able to support human reasoning and computer tractable. In addition, their proposed system is able to generate XML queries for users with little knowledge about XML and XQuery.

Hamad [17] proposed a middleware relational storage for converting between XML and RDB based on path-based relational storage approach and DOM Model. In addition, the proposed technique applies 1-index method to reduce the storage size. Lyamin and Chereposvskaya [18] proposed rules-driven method to map XML into RDB based on production rule system. Some of their defined rules are on the *relations, attribute, ancestors, descendants, convert (value, type), getRelation* and so on. Their proposed system has been implemented in a University for importing and exporting data presented in XML files into RDB that contains many educational materials.

More recently, Combi et al. [19] proposed XHyb, which represent simple logic for specifying features of XML documents with respect to common integrity and reference constraints of DTD. Similarly, Martens et al. [20] proposed BonXai, an alternative XML specification language, which is as easy to use as DTD, and yet contains the expressiveness and features of XML Schema such as use of types, and key constraints.

On the other hand, Yaghmazadeh et al. [21] proposed using instances of the input XML document to map into RDB instead of based on structural or model mapping. They designed the tool named MITRA, which is a novel tree-to-table transformation Domain-Specific Language (DSL) that can express a rich class of mapping programs. On top of that, they also presented the Deterministic Finite Automata (DFA) approach for learning column transformation, and subsequently, predicate learning to filter out irrelevant tuples in the intermediate table.

3. THE ARCHITERCTURE OF EVALUATION ENGINE

Figure 3 shows the architecture of the simulation engine, which have the following procedures:

a. Connecting to the database.
b. Inserting DTD file to build the database schema.
b) Inserting XML file for data mapping process and storing into database.
c) Executing user queries for retrieve data from the database.
d) Calculating time taken for storing data and querying evaluation.

The main interface is depicted in Figure 4, which consists of three main parts: (1) Database Configuration – to setup database connection, (2) Storing and Retrieval screen – to select the input files and storage methods, and (3) Query Processing – to evaluate the query and displaying the evaluation results.

Figure 5 depicts the two algorithms of our simulation engine. The first algorithm describes the storing procedure of data mapping from XML into RDB while the second algorithm explains the querying process of XML data from RDB. The storing algorithm begins with mapping function which serves different types of storing based on two data mappings which are SAX and s-XML. Line 1 to 2 describes the input and the expected output of the algorithm. Line 4, 8 and 9 are for time calculating purpose, that is, to evaluate the efficiency of the data mapping. Line 6 and 7 are calling functions based on respective data mappings. Line 11 to 19 depicts the function for SAX approach, while Line 20-28 depicts the s-XML approach. SAX mapping needs DTD file to generate the database schemas in RDB before data mapping processing begins (see Line 12, which subsequently call the ParseDTD function in Line 29 to 34). s-XML mapping does not need DTD file for schema mapping which it directly creates the database schema into two tables.

On the other hand, the second algorithm describes on querying part of the XML data from RDB. The time taken to answer the query is displayed on the screen window. Line 4, 8 and 9 are for calculating the time to evaluate the efficiency of querying based on queries complexity. Since each data mapping has varied ways of storing, then each data mapping resulted in varied way of querying from database.

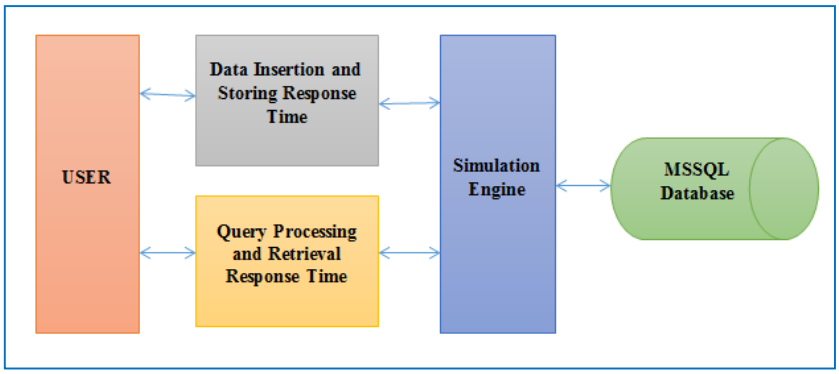


Figure 3. The architecture diagram

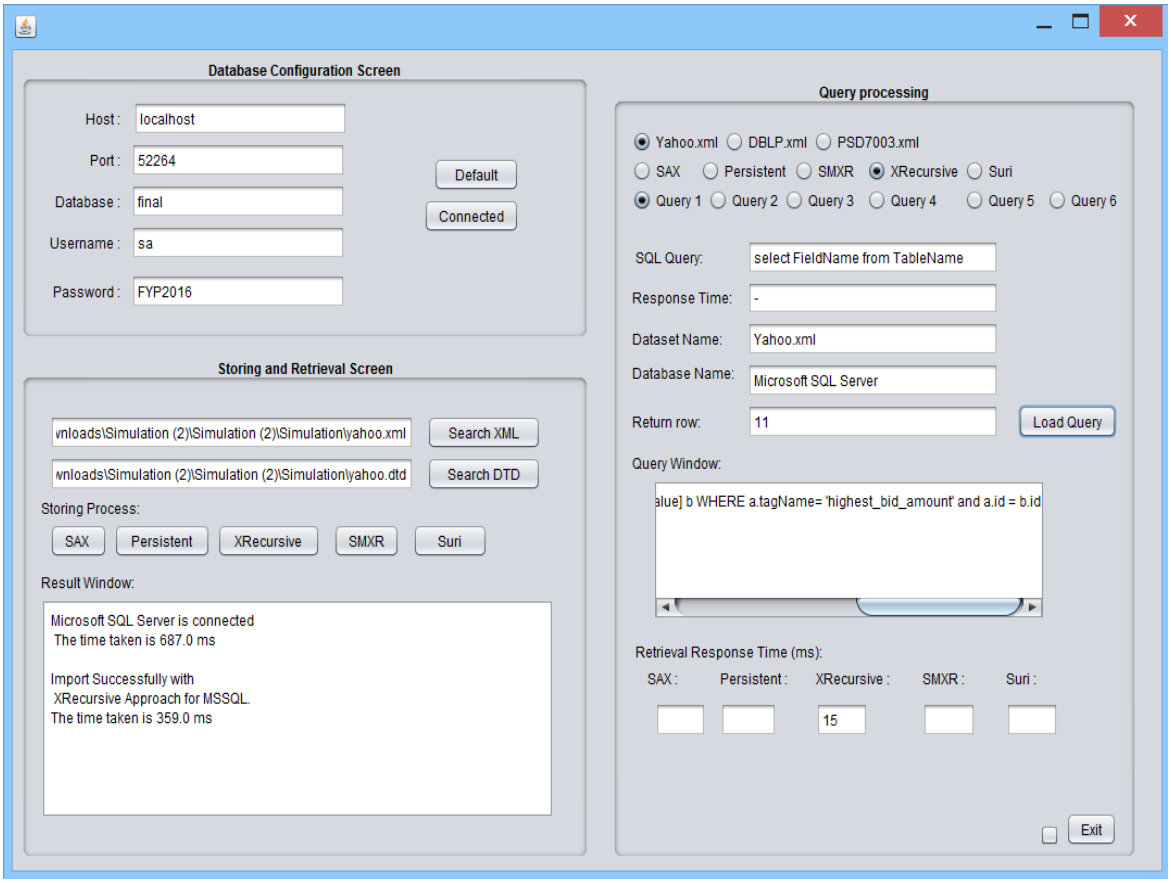


Figure 4. User interface for performance evaluation

```
Algorithm: Storing XML data into relational database in structural based
1 Input: XML file, DTD file
2 Output: Data in relational database
3 Functional mapping() {
4       Long time1 = System.currentTimeMillis()
5       Set connection to MSSQL database
6       If mapping := SAXX then SAXX() End If
7       If mapping := Persistent then Persistent() End If
8       Long time2 = System.currentTimeMillis()
9      Time taken to store = time2 – time1
10 }
11 Function SAXX () {
12       ParseDTD ()
13       While element.is.NotEmpty () do
14             Trigger start element
15             Trigger character
16             Trigger end element
17             Insert element into database schema generated by DTD file
18      End While
19 }
20 Function Persistent () {
21      Create two tables in database
22      While element.isNotEmpty () do
23      If node := root node then insert into ParentTable with Null parentLabel End If
24      If node := first level node then insert into ParentTable with parentLabel: = idNode End If
25      If node := sibling node then insert into ParentTable with ChildNext node data End If
26      If node := leaf node then insert into ChildTable with ChildData values End If
27      End While
28 }
29 Function Parse DTD () {
30       Parse DTD file
31       Create database schema based on DTD file
32 }

Algorithm: Querying XML data from relational database
1 Input: Data in relational database
2 Output: Result set in screen window
3 Function Querying {
4         Long time3 = System.currentTimeMillis ()
5         If query := query1 or query 2 then execute simple query from desired database End If
6         If query := query3 or query 4 then execute average query from desired database End If
7         If query := query5 or query 6 then execute complex query from desired database End If
8         Long time4 = System.currentTimeMillis ()
9         Time taken for query = time4 – time 3
10 }
```

Figure 5. Pseudocode of the proposed approach

4. PERFORMANCE EVALUATION: RESULTS AND DISCUSSION

Figure 1 shows the illustration sample of XML which will be used throughout the paper.

4.1. Experimental Setup

The XML datasets were obtained from University of Washington website [22] as this is the standard datasets for benchmarking. The selected datasets for the experimental evaluation is based on two file sizes as depicted in Table 4. All the testing activities were performed on i5-3630QM 2.40GHz processor with 16GB of RAM running on Windows 7.

Table 4. XML Datasets for Evaluation

Dataset	File name	File size	Category
DBLP	Dblp	130.73MB	Medium size
Protein	Psd7003	722.59MB	Large size

4.2. Performance Evaluation

4.2.1 Storing Evaluation

Table 5 shows the database creation and storing time for the two structural mapping approaches. From the result, s-XML approach has better performance evaluation on insertion of data compare to SAX approach.

Table 5. Database Creation and Storing Time

Approaches	Database Storing Time (ms)	
	DBLP	Protein
SAX	4654833 (1.3 Hr)	23581170 (6.6 Hr)
s-XML	1595387 (0.4 Hr)	13202258 (3.7 Hr)

4.2.2 Retrieval Evaluation

Query evaluation plays a crucial criterion to determine if one storage method is better than the other [23, 24, 25]. In evaluating the retrieval, six queries were prepared and set in the simulation engine. The simulation engine provides six function buttons to represent six different types of queries for the retrieval evaluation. Through clicking on the respective query button, the simulation engine processes the query from the RDB. Table 6 depicts the query pattern used in evaluation process. Q1 to Q3 are path queries (simple queries with P-C, A-D and mixed) while Q4 to Q6 are twig queries (complex or branching queries with P-C, A-D and mixed).

Table 6. Query Pattern Table

Query	Query Pattern
Query1	Path query with P-C relationship
Query2	Path query with A-D relationship
Query3	Path query with mixed relationship
Query4	Twig query with P-C relationship
Query5	Twig query with A-D relationship
Query6	Twig query with mixed relationship

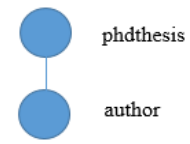
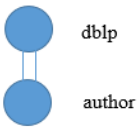
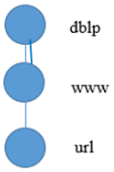
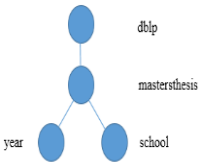
The performance evaluation on retrieval is based on three consecutive testing times and the average result of three is the final result for the query retrieval. The reason being to ensure the cache memory does not contain any unnecessary data that can affect the response time. The total number of rows returned from each query is recorded to check for the correctness. It can be observed that the response times for each query are varied because of the queries complexities and number of returned results.

A. DBLP Dataset

Table 7 depicts the six queries retrieval evaluation, focusing on the query descriptions and query node representations, the number of return results, and the average time for each query to retrieve data from RDB on SAX and s-XML approaches.

From Query1 to Query3, the query retrieval time for SAX has longer time as compared to the s-XML approach. s-XML has faster response time because it does not involve any expensive join of tables as SAX approach. Query on SAX approach takes a lot of multiple joins of tables since every element node has its own table. Thus, to track the relationship of the node to the ascendant involves in joining few tables. However, in s-XML, all the non–leaf elements are stored in parent table while the leaf nodes are stored in child table.

Table 7. Summary of the Query Retrieval Results on DBLP Dataset

Query No	Description	Query Node	Number of Return Results	Retrieval Time (ms) SAX	s-XML
Query1	List out all the information that consists of phdthesis with any author node.		72	91	72.33
Query2	List out all the information that consists of dblp with any author node.		716,488	650.33	210.66
Query3	List out all the information that consists of www with its respective immediate url node.		38	232.33	72
Query4	List out all the information that consists of dblp with its immediate node mastherthesis, which has immediate branching node of year and school.		10	151.66	139
Query5	List out all the information that consists of dblp with any title or year node.		657,690	980.66	263
Query6	List out all the information that consists of dblp with its respective immediate node www and any node which has author.		716,526	1183.33	272.66

For Query4, the query has only 10 returned rows. However, s-XML is still a better approach in retrieving the data than SAX. This is due to the reason of the number of joins involved. SAX used three tables to process the query in order to track the targeted node with its ascendant node. Thus, SAX needs to match selfID and parent ID within three tables. From Query5, the response retrieval time of s-XML is also faster than SAX based on the similar reason that has been discussed earlier which is due to the left join involvement. The number of returned rows is 657,690. For Query6, the query is the longest and the most complex among the six queries on DBLP dataset. However, s-XML is still better than SAX approach with 716528 returned rows because in SAX query, there is eleven left join functions in two select statements involved.

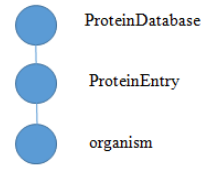
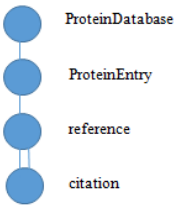
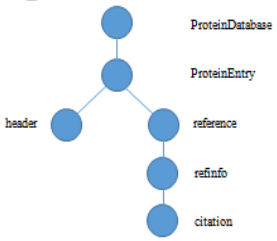
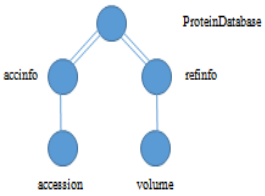
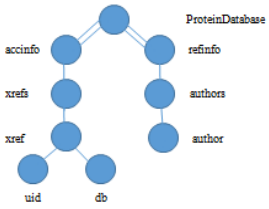
B. Protein Dataset

Table 8 depicts the summary of the six queries retrieval evaluation results of the six queries on PSD7003 dataset. From Table 8, it can be observed that the s-XML approach outperform the SAX approach for all the cases. In Query2, it involves four left joins as compared to Query1, which has only two left joins.

For Query3, the number of returned rows is 314,789 rows which are slightly more than returned rows in Query2 by 26 rows. However, the response retrieval time from both tables differ quite a lot.

The query design for both tables is nearly the same. The difference are it involves right joins and the number of tables involved in the joins, which in Query3 has five tables involvement while in Query2 has four tables involvement.

Table 8. Summary of the Query Retrieval Results on PSD7003 Dataset

Query No	Description	Query Node	Number of Return Results	Retrieval Time (ms) SAX	s-XML
Query1	List out all the information that consists of ProteinEntry with its respective immediate organism node		262,525	335	173.33
Query2	List out all the information that consists of ProteinEntry with its respective immediate refinfo node.		314,763	574.33	233
Query3	List out all the information that consists of ProteinEntry with its respective immediate reference node, which consists of any citation node.		314,789	920.33	314
Query4	List out all the information that consists of ProteinEntry with both its immediate header node, and reference node, which consists of refino with its immediate citation node		627,295	1,132.66	277.33
Query5	List out all the information that consists of ProteinDatabase with both nodes named accinfo and refinfo, which has their immediate child named accession and volume node respectively.		365,416	1,293.33	367
Query6	List out all the information that consists of ProteinDatabase with both nodes named accinfo and refinfo, which has their immediate child named xrefs and authors node, whereby Xrefs has immeidate xref node with its immeidate uid and db nodes, which authors has immediate author node.		806,824	9,136.33	430.66

In Query4 and Query5, both of these tables involve right joins in the query design. The returned rows for Query4 are 365,416 rows while for Query5 are 627,295 rows. Query5 has lesser returned rows but has higher response retrieval time than Query4. We observed the SAX query in Query5 involves in two right joins and four left joins while in Query4 involves only one right join and five left joins.

For Query6, s-XML approach has a better result than SAX approach in query retrieval with 806,824 returned rows. The query design in SAX that involves in expensive joins within three select statements while in s-XML does not involve in any expensive joins of tables. However, both of the retrieval time have the highest result due to the complex query among all queries on PSD7003 and need more time to fetch rows.

4.3. Discussion

From the evaluation thus far, it is obvious that the s-XML approach outperformed the SAX in terms of storing evaluation as this approach directly create and map the data into two tables in RDB while for SAX approach, it uses DTD to create the table and map the data into more than two tables depends on the number of elements in the XML file.

As for the retrieval evaluation, s-XML does not require any expensive joins like SAX approach to preserve the relationships among the elements node. It can be seen that s-XML approach is scalable and proven to be out-performed even in largert dataset. For instance, the retrieval time for complex queries (Query6) on PSD dataset is about 21 times faster than SAX approach. Hence, we concluded that s-XML approach has the best storing and query retrieval performance evaluation.

5. CONCLUSION AND FUTURE WORK

In this paper, we have reviewed some recent approaches based on structural mapping in terms of how each approach works, the advantages and disadvantages. Performance evaluations have also been carried out to compare the performance of SAX and s-XML approaches in term of storing and querying evaluations. From the experimental results, it is shown that s-XML has the best storing and retrieval performance. This is due to the fact that its table structure is simple, hence, much lesser joins are required as compared to SAX, which uses multiple tables and subsequently, many expensive joins are required. In future, in terms of experimental evaluation, the size of the datasets should be tested for much larger datasets especially with the emergence of Big Data technology.

REFERENCES

[1] S.C. Haw, et al., "XMapDB-Sim: Performance Evalaution on Model-based XML to Relational Database Mapping Choices", *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 7(2), pp. 551-566, 2017.

[2] H. Zhu, H. Yu, G. Fan, and H. Sun, "Mini-XML: An efficient mapping approach between XML and relational database", *International Conference on Computer and Information Science*, 2017, pp. 839-843.

[3] A. Qtaish, and K. Ahmad, "XAncestor: An efficient mapping approach for storing and querying XML documents in relational database using path-based technique", *Knowledge-Based Systems*, vol. 114, pp. 167-192, 2016.

[4] N. Ferro, and G. Silvello, "Descendants, ancestors, children and parent: A set-based approach to efficiently address XPath primitives", *Information Processing & Management*, vol. 52(3), pp. 399-429, 2016.

[5] E. Song, et al., "*Handling XML to Relational Database Transformation Using Model-Based Mapping Approaches*", IEEE Conference on Open Systems, 2018, pp. 65-70.

[6] M. Atay, *et al.*, "Efficient schema-based XML-to-Relational data mapping", *Journal Information Systems*, vol. 32(3), pp. 458-476, 2005.

[7] M. Atay, *et al.*, "*Mapping XML data to relational data: a DOM-based approach*", International Conference on Internet and Multimedia Systems and Applications, 2004, pp. 59–64.

[8] S. Subramaniam, *et al.*, "Bridging XML and relational databases: An effective mapping scheme based on persistent labelling", *International Journal of Electrical and Computer Engineering*, vol. 2(2), pp. 239-246, 2011.

[9] A. Gabillon, and M. Fansi, "A New Persistent Labelling Scheme for XML", *Journal of Digital Information Management*, vol. 4(2), pp. 112-116, 2006.

[10] C. Lim, *et al.*, "Storing, reasoning, and querying OPM-compliant scientific workflow provenance using relational databases", *Future Generation Computer System*, vol. 27, pp. 781-789, 2011.

[11] A. Qtaish, and K. Ahmad, "*Query Mapping Techniques for XML Documents: A Comparative Study*", International Conference on Electrical Engineering and Informatics, pp. 529-534, 2015.

[12] M.M. Gamal, *et al.*, "*A literature survey on mapping between fuzzy XML databases and relational or object oriented databases*", Third World Conference on Complex Systems, 2015, pp. 1-6.

[13] M. Machkour, et al., "*A Reversible Conversion Methodology between XML and Object-relational Models*," International Conference on Information and Communication Systems, 2016, pp. 270-275.

[14] K. Ahmad, and R. Samad, "*Semantic Based Mapping from XML to Relations*", International Conference on Information Science and Digital Content Technology, pp. 255-260, 2012.

[15] J. Mao, and X. Ye, "*Relational Schema and XML Schema Bidirectional Mapping Algorithm Based on the Intermediate Object Tree*", International Conference on Computer and communications, 2017, pp. 2380-2383.

[16] A.E. Molnar, *et al.*, "*Conceptual graph driven modeling and querying methods for RDMBS and XML databases*", IEEE International Conference on Intelligent Computer Communication and Processing, 2017, pp.55-62.

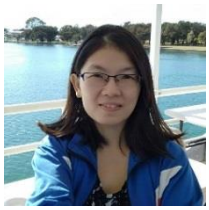
[17] H.A.L. Hamad, "RXML: Path-based and XML DOM Approaches for Integrating Between Relational and XML Databases", *International Journal of Database Management System*", vol. 9(5), pp.1-10, 2017.

[18] A.V. Lyamin, and E.N. Cherepovskaya, "XML-Relational Mapping using Production Rule System", IEEE Intelligent Systems Conference, pp. 422-429, 2017.

[19] C. Combi, *et al.*, "A hybrid logic for XML reference constraints", *Data & Knowledge Engineering*, vol. 115, pp. 94-115, 2018.

[20] W. Martens, et al., "Bonxai: combining the simplicity of DTD with the expressiveness of XML schema", *ACM Transaction on Database Systems*, vol. 42 (3), pp. 15:1–15:42, 2017.
[21] N. Yaghmazadeh, *et al.*, "*Automated Migration of Hierarchical Data to Relational Tables using Programmingby-Example*", Proceedings of the VLDB Endowment, vol. 11(5), pp. 580-593, 2018.
[22] XML Repository, University of Washigton. Retrieve from http://aiweb.cs.washington.edu/research/projects/xmltk/xmldata/www/repository.html
[23] H. Nassiri, *et al.*, "*Querying XML and Relational Data*", Procedia Computer Science, vol. 134, pp. 340–345, 2018.
[24] H. Nassiri, *et al.*, "*Integrating XML and Relational Data*", Procedia Computer Science, vol. 110, pp. 422–427, 2017.
[25] E. Seshatheri, and T. Bhuvaneswari, "Effective XQuery keyword using XML query processing", *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 14(1), pp. 450-454, 2019.

BIOGRAPHIES OF AUTHORS



Su-Cheng Haw is Associate Professor at Faculty of Computing and Informatics, Multimedia University, where she leads several funded research on the XML databases. Her research interests include XML databases, query optimization, data modeling, semantic web, ontology, data management, and data warehousing. She has published more than 120 articles in reputable journals and conferences.

Emyliana Soong is a postgraduate student at Faculty of Computing and Informatics in Multimedia University. She is currently doing a research on XML data mapping scheme in transforming XML document into relational database.