Model-based XML to Relational Database Mapping Choices

Emyliana Song, Su-Cheng Haw, Fang-Fang Chua

Abstract— Extensible Markup Language (XML) technology is widely used for data exchange and data representation in both online and offline mode. This structured format language able to be transformed into other formats and share information across platforms. XML is simple; however, it is designed to accommodate changes. For this paper, a study on transformation of XML document into relational database is conducted. Crucial part of this process is how to maintain the hierarchy and relationships between data in the document into database. Approaches that are discussed in this paper each uses own unique way of data storing technique and database design. Therefore, each algorithm is assessed with three datasets constitute of small, medium and large size XML file. The efficiency of the algorithms is being tested on time taken for data storing and query execution process. At the end of the evaluation, we discuss factors that affect algorithm performance and present suggestions to improve mapping scheme for future works.

Keywords—XML-to-RDB, Model mapping scheme, Transformation, XML Database, XML Labeling

I. INTRODUCTION

eXtensible Markup language (XML) is a markup language to encode data using plain text format. It simplifies data transformation, data exchange and data sharing between platforms and machines. XML is built to be extensible and self-descriptive. The XML document is both human and machine readable, which allows document to be easily searchable and optimized for machine processing and analyzing. XML syntax are case sensitive, each element or attribute contain value of the node in between start tag "< >" and end tag "</ >", The XML nodes are commonly divided into 4 type which are root, element, attribute and value node. Mapping algorithm is required to integrate XML into different format, what we are looking for in this paper is to amp XML document into relational database.

There are two types of mapping methods, which are structure-based and model-based. For this paper, the evaluation will be conducted only on model-based mapping approaches since model-based approaches support a wider range of web applications, as most applications does not usually come together with any schema [1]. One of the most obvious different between model-based and structure-based mapping approach is the requirement on having Document Type Definition file (DTD) or XML schema to define structure of XML document. For model-based mapping, DTD and XML schema is not needed.

The rest of the paper is organized as follows. Existing and related approaches on model-based mapping schemes are reviewed in section 2. Section 3 discussed the performance evaluation carried out in the experiment of selected approaches. Experimental results and analysis of the findings are presented in section 4. And lastly, Section 5 conclude the paper.

II. LITERATURE REVIEW

Throughout the years, numerous mapping schemes have been proposed to resolve issues on transforming XML to relational database structure. Each proposed method has its own unique technique that allows the algorithm to work better than previous existing mapping schemes.

Data loss may occur during transformation of XML document into relational database (RDB). However, there exist technology like XPath, Document Object Model (DOM) and Simple API for XML (SAX) to defines the properties, data manipulation and support seamless integration for the process.

SAX parser is less memory consuming and faster compare to DOM parser. This is because SAX is one-pass processing and event triggered method that send events permission to run whenever XML file has been recognized. On the other hand, DOM parser input all XML data into memory and only runs parser when full document is received. DOM have built in API function that allow node information like getnodename, getparentnode, getvalue and so on, ease application to identify and retrieve data.

SMX/R [2] annotates nodes in document using path-based labelling technique. Both Simple API for XML (SAX) and Document Object Model (DOM) parser are used in parsing process to validate the data. The authors used SAX parser to determine each start and end value of the elements and utilize XPath processing to navigate hierarchical structure of XML document based on path expression. SMX/R constructed two tables to stores the data, namely path table and path index table. Author did an experimental evaluation, which indicated that SMX/R performs better in term of number of joined operation to accomplish designated queries. Proposed approach uses less join operation and lesser number of interrelated tables compares to XRel, which uses all four constructed table to retrieve data.

Revised Manuscript Received on August 18, 2019

Emyliana Song, Faculty of Computing and Informatics, Multimedia University, Cyberjaya, Malaysia.
Su-Cheng Haw, Faculty of Computing and Informatics, Multimedia University, Cyberjaya, Malaysia.
Fang-Fang Chua, Faculty of Computing and Informatics, Multimedia University, Cyberjaya, Malaysia.

XRecursive [4] stores XML documents into RDB by assigning unique node label using recursive function. This approach identifies each path recursively by its parent node. In their approach, DOM parser is used to parse the XML document. XRecursive store XML data into two tables, tag structure (store the data of the element node) and tag value (store value of the node). On comparison results between XRecursive and SUXCENT [5], XRecursive uses much lesser storage space than SUXCENT approach. The reason is that SUXCENT uses five tables to store the XML data while only two tables are required by XRecursive.

Suri et al. [6] proposed an approach that uniquely identify each node with positive integers. XML document decomposed into an orderly depth first traversed tree, whereby each node then holds the node id (unique identifier) and node position (the path of node in the document). This information will be stored as attribute in Node table and Data table. Their approach adopted the DOM parser and utilize DOM API function in retrieving XML data in the document. Performance evaluations against XREL [3] and XPEV [7] indicated that their approach (herein after abbreviated as SS approach) uses the least storage space.

s-XML [8] used the Persistent labeling scheme with (level, local parent node, local node id) to annotate each XML node. Advantage of this labelling system is that it provides quick identification of parent-child relationship between nodes. Other than that, Ancestor-Descendant relationship can be trace by recursively search of parent-child relationship. This may require extensive recursive search especially for large dataset. Based on this approach, two tables namely, Parent table and Child table will be constructed respectively. Parent table stores data on non-leaf nodes while Child table store data on the leaf nodes. Nevertheless, it is observed that this approach is good to support complex chain and twig queries. From the performance test conducted, results revealed that s-XML performs better in tern of both time and storage consumption as against Edge [9] and Attribute [9]. Nevertheless, in recent work of Zhu et al [10] revealed that s-XML requires more time and space compared to Mini-XML. This is due to some redundant information in the tables.

Ying et al. [11] proposed a hybrid based labelling technique, combining path and node labelling technique. After the labeling process, the XML tree will be mapped into four tables, which are File, Path, LeafNode and InnerNode tables. File name and id of file is stored in File table, Path table stores XPath expression and uniquely assigned id of each non-leaf nodes. The InnerNode table stored the non-leaf node information, while the LeafNode table stored the leaf node information. Comparison test conducted by authors shows that proposed approach able to stores the data in XML document in shorter period and uses less storage consumption compared to XRel [3], XParent [12] and SUXCENT [5] approaches. Path table used in Ying et al. [11] approach support strong structural queries for twigs queries and as a result, it minimized the query response time caused by θ-joins.

Chikhale et al. [14] proposed mapping scheme that reduces time and space complexity in storing XML document into relational database. Proposed scheme uses batch stream and combine attributes objects into cluster after XML tree is created. K- mean algorithm is adapted in training the data into forming groups, optimal results might not be able to achieve for the first time, by running the process more than once helps to achieve optimal solution. Author also proposed system that helps user to stores XML file into relational database. XML file is file loaded and stored into the system. Then, document will be parsed, proposed algorithm and SQL queries then take place in converting document into relational database.

A new rule of mapping XML to relational database is introduced by Lyamin et al. [15]. Proposed method uses basic rules-driven principle of Artificial Intelligent and production rule system. This system defines the rules that helps in data representation. Authors includes categories that involves in mapping XML into RDB and built-in method applied in different database in the paper. The rules prerequisite an action with condition block “IF” and result of rule application come in condition block “THEN”. These rules are introduced to perform document transformation also, to support logical relations and control correspondence between hierarchical and relational structures.

III. IMPLEMENTATION

In this section, we will be discussing on implementation of the three selected mapping choices, namely SMX/R [2], SS [6] and XRecursive [4]. Each of the approaches will be tested against three sizes of datasets, ranging from 25KB to 722.59MB.

Evaluation test are performing on Intel i7-3630QM processor with 16 GB RAM running on Windows 8 64-bit machine. To start off, a simulation engine is created using Java language along with Microsoft SQL Server as the database management system. Performance evaluation conducted in this paper is divided into two sections, data mapping and storing process, and query execution process. Data mapping process first loads document into the simulation engine and starts to read the document by builder factory. XML data model parsed by parser then will be converted into RDB using the selected mapping algorithms. Simplified flow of the mapping process is shown in Fig. 1.

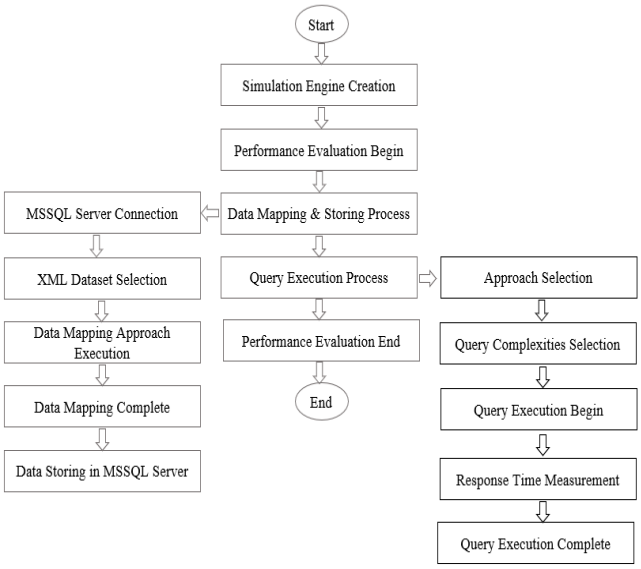

Fig. 1.The flow of mapping process

There are two main processes in the simulation engine. The first process is Data Mapping and Storing Process, while the second process is Query Execution Process. Fig. 2 shows the user interface of the simulation engine.

In the first process, the storing begins by selecting the database and creating a database environment to store tables. Process of inserting data from XML document to relational databases then began. The storing time of first node to last node stored into database is calculated and displayed onto simulation engine for user to record.

In the Query Execution Process, the processing process starts with selection of database. From the database selected, data that have been stored will be retrieved with the selected queries one at a time. Response time of querying data from tables will be recorded and display in the simulation engine.

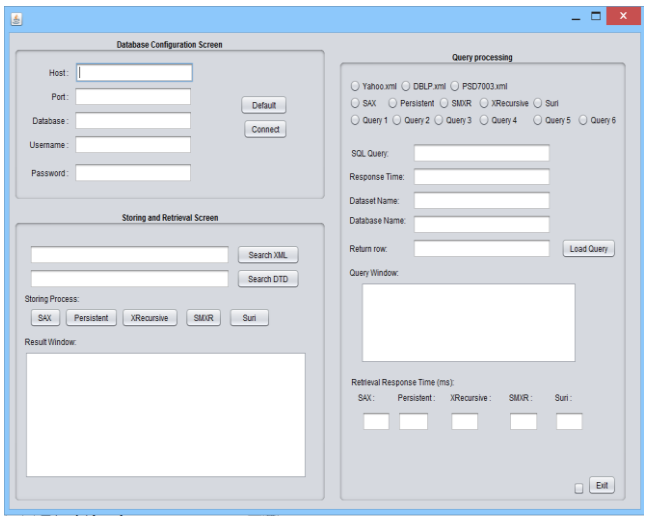

Fig. 2.The user interface of the simulation engine

Table 1 shows the details of three datasets used for the evaluation test. The response time for storing and retrieving of data using these datasets will be recorded. All these datasets come with varies sizes: yahoo.xml represents the small size dataset, dblp.xml represent medium size and lastly, psd7003.xml represent large scale document. Datasets were taken from University of Washington repository [13].

TABLE I. XML DATASETS DETAILS

Dataset	Document Name	File Size	Size
Yahoo	Yahoo.xml	25KB	Small
DBLP	DBLP.xml	130.73MB	Medium
PSD7003	PSD7003.xml	722.59MB	Large

IV. RESULTS AND DISCUSSION

Table 2 shows the storing time of all the three mapping approaches. Storing time is calculated from document parsing to mapping data into relational database, and the time used will be display on simulation engine in millisecond. The final time reading is an average of time taken for three consecutively run. Evaluation results show that XRecursive is the most efficient in data storing process compare to SS approach and SMX/R. XRecursive able to achieve the least storing time because each node is recursively retrieved from the document in orderly manner. No path expression or unique position labelling is used to identify nodes, besides, the database design by author is simple and contain only three columns in each table. Compare to the other two approaches (SS approach and SMX/R) uses six columns in one of the two tables, each complexity of each row increases the time taken for data to be stored.

All the approaches utilize DOM parser to parse the document except for SMX/R, SMX/R utilize both SAX and DOM parser. For PSD7003.xml, the heap size needs to be increased during data loading process, this is due to the file size of the dataset. From PSD7003 results, we can conclude that DOM parser is more suitable to be used with smaller dataset, because as the size of dataset increase, the amount of storing time required increases drastically.

TABLE II. DATA STORING TIME

	Storage time using different approach (ms)		
Dataset	**XRecursive**	**SS**	**SMX/R**
Yahoo	313	344	422
DBLP	1636409 (27 mins)	1794387 (30 mins)	1925692 (32 mins)
PSD7003	8506420 (2 hrs 21 mins)	1.3257736E7 (3 hrs 41 mins)	2.2799478E7 (6 hrs 20 mins)

Query execution process evaluates the retrieval time for data to be search in the newly created tables. The efficiency and effectiveness of this process are influenced by a few factors, such as, number of tables, the information stored, labelling technique and etc.

Query retrieval time is evaluated with six queries, each query is varied depending on the relationship between nodes. Similarly, each query runs three time consecutively and average of each will be used as reference for discussion.

Table 3 depicts the query pattern used in evaluation process. Generally, there are two main types of query, namely Path Query and Twig Query. Fig. 3 depicts the possible query patterns within path query, while Fig. 4 depicts the query patterns within twig query. As for our evaluation, Q1 to Q3 are path queries with Parent-Child (P-C) relationship, Ancestor-Descendant (A-D) relationship, and mixed relationship), while Q4 to Q6 are twig queries with P-C, A-D and mixed relationship.

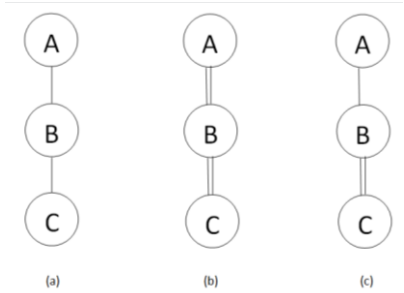

Fig. 3.Path Query with (a) P-C, (b)A-D, and (c) both P-C and A-D

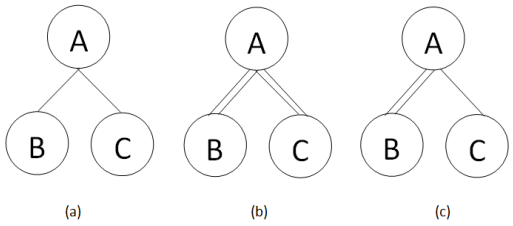

Fig. 4.Twig Query with (a) P-C, (b)A-D, and (c) both P-C and A-D

TABLE III. QUERY PATTERN TABLE

Query	Query Pattern
Q1	Path query with P-C relationship
Q2	Path query with A-D relationship
Q3	Path query with both P-C and A-D relationship
Q4	Twig query with P-C relationship
Q5	Twig query with A-D relationship
Q6	Twig query with both P-C and A-D relationship

Table 4 summarizes the overall query retrieval results, while Fig. 5 to Fig. 7 show the individual performance evaluation on the three selected datasets. For an overall view, we can see that SS approach works best for smaller dataset, but as the dataset grow larger, retrieval time for this approach became the least efficient.

TABLE IV. QUERY RETRIEVAL RESULT TABLE (MS)

Yahoo.xml	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**	**Average**
SS	16	26	27	16	15	15	**19.2**
XRecursive	15	21	47	33	16	36	28
SMX/R	47	67	52	16	37	35	42.3
Dblp.xml	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**	**Average**
SS	3536	1281	3844	1594	3375	3989	3603.2
XRecursive	2276	1243	1556	1794	3389	3378	**2272.7**
SMX/R	4764	1328	2753	1881	3988	3985	3116.5
psd7003.xml	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**	**Average**
SS	12193	12278	11988	24027	24094	46596	21862.7
XRecursive	8431	9035	8195	16566	16433	41039	16616.5
SMX/R	8715	9045	8711	17058	17470	36065	**16177.3**

From Fig. 5, it shows that SS approach able to execute SQL queries using the shortest time. This is due to the table design that enable minimal search between the tables, only necessary data is analyzed and retrieve. ParentID field in the tables minimizes the search space to identify the node relationship. Although SS approach can achieve the fastest result compare to the other approaches in small dataset, the result show otherwise when the data volume increases (see Fig. 7). SMX/R takes longer time because regardless the relationship between nodes, query still need to analyze the path expression to verify nodes hierarchy.

For medium sized dataset (DBLP.xml), SS approach efficiency begins to reduce compared to Yahoo dataset and XRecursive improved by 30% better than SS approach on total average result. When querying nodes that involve P-C relationship, XRecursive works better than SS approach, but both approached uses almost equally amount of time when it comes to A-D nodes. As for SMX/R approach, the result drastically improved when dealing with larger datasets. Reason for this will be further elaborated in the following paragraph (under the discussion for psd7003.xml dataset).

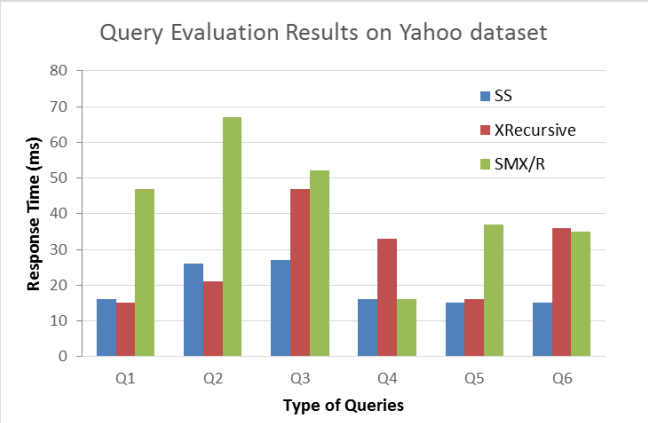

Fig. 5.Query Retrieval Results on Small Size Dataset

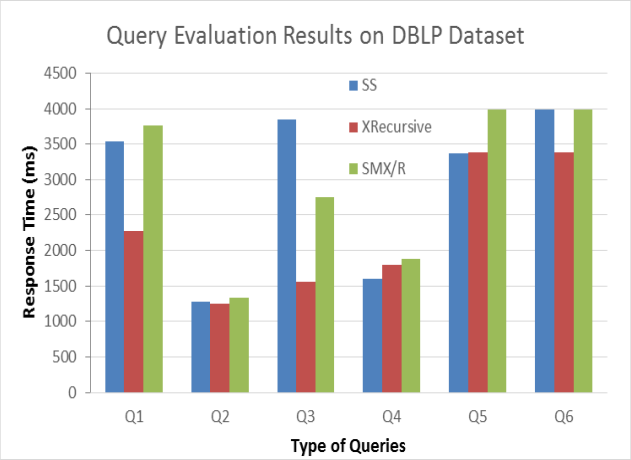

Fig. 6.Query Retrieval Results on Medium Size Dataset

From Fig. 7, it is observed that when it comes to retrieving data that involve both P-C and A-D relationship, SS approach and XRecursive takes up more time, while SMX/R require the least time in retrieving the query. This is since SMX/R does not require multiple joins operation in searching data between tables. The path expression used in path table is simply enough for algorithm to track the relationship (P-C and A-D) of a node. Hence, resulting minimal use of operations in the database. Oppositely, SS approach uses the node positioning system that increases the search space both in tag value table and between the two tables. Although XRecursive performed the best in data storing evaluation, result shows otherwise in query retrieval. The main cause of the contrast result is the technique used in XRecursive mapping algorithm; the data is recursively search by the parent node. Thus, it increases the number of join operation when it is dealing with larger dataset.

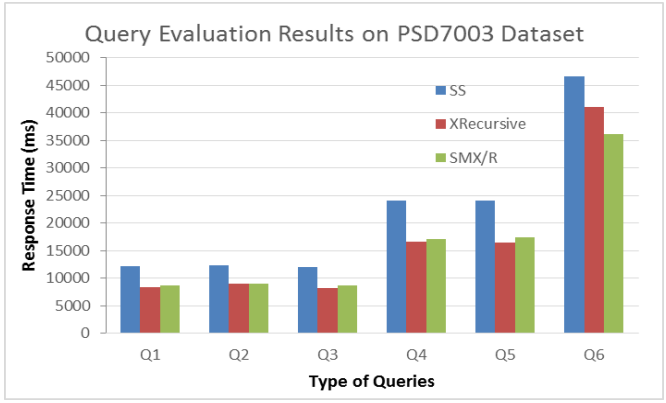

Fig. 7.Query Retrieval Results on Large Dataset

V. Conclusion and Future Works

From the paper, one may decide which approach is suitable to be adopted, depending on the nature of business. Approaches proposed may be used in different situation caused by the database design, type of information stored and mapping function. Good use of labelling technique and database design is crucial for achieving efficient mapping approach. Some approaches are well suited for storing purposed but consume large amount of time when it comes to data retrieval process. Contradict, approaches like SS approach and SMX/R may take more time in storing process but returning query result in much shorter time.

For future works, we would like to design a mapping approach that able to give maximum performance with minimum joins operation and memory consumption. Create a new labelling scheme to annotate node position and utilize recent XML technologies to help boost the performance of mapping approach. In addition, performance evaluation will be performed against more recent approaches, such as XAncestor [14], XMap [15], and Mini-XML [10].

REFERENCES

1. S.C. Haw, E. Soong, N.A. Amirah, and A. Amin, "XMapDB-Sim: Performance Evalaution on Model-based XML to Relational Database Mapping Choices", *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 7(2), 2017, pp. 551-566.
2. F. Abduljwad, N. Wang, and D. Xu, "SMX/R: Efficient way of storing and managing XML documents using RDBMSs based on paths", *International Conference on Computer Engineering and Technology*,1, 2010, pp. 143-147.
3. M. Yoshikawa, T. Amagasa, S. Shimura, and S. Uemura, "XRel: A Path-Based Approach to Storage and Retrieval of XML Documents Using Relational Databases", *ACM Transactions on Internet Technology*, vol. 1(1), 2001, pp. 110-114.
4. M.A.I. Fakharaldien, J.M. Zain, and N. Sulaiman, "XRecursive: An efficient method tostore and query XML documents", *Australian Journal of Basic and Applied Sciences*, vol. 5(12), 2011, pp. 2910-2916.
5. S. Prakas, S.S. Bhowmick, and S. Madria, "SUCXENT database mapping techniques", *Advances in Computer Science and Information Technology*, vol. 2(2), 2015, pp. 162–166.
6. P. Suri, and D. Sharma, "A Model Mapping Approach for storing XML documents in Relational databases", *International Journal of Computer Science Issues*, vol. 9(3), 2012, pp. 495-498.

7. J. Qin, S.M. Zhao, S.Q. Yang, and W.H. Dou, "XPEV: A Storage Model for Well-Formed XML Documents", *Lecture Notes in Computer Science*, 3613, 2015.
8. S. Subramaniam, S.C. Haw, and KH Poo, "s-XML: an efficient mapping scheme to bridge XML and relational database", *Knowledge-Based Systems*, 27, 2012, pp. 369–380.
9. D. Florescu, and D. Kossmann, "A performance evaluation of alternative mapping schemes for storing xml data in a relational database," 1999.
10. 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.
11. UW, University of Washington Repository. Available: http://aiweb.cs.washington.edu/research/projects/xmltk/xmldata/www/repository.html
12. 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*, 114, 2016, pp. 167-192.
13. I. Bousalem, and I. Cherti, "XMap: a novel approach to store and retrieve xml document in relational databases", *Journal of Software*, vol. 10(12), 2015, pp. 1389-1401.
14. P. Chikhale, S. Harihar, V. Adhude, and P.Raut, "Mapping of XML Document and Relational Database (using Structural Queries), *International Journal for Research in Applied Science & Engineeering Technology*, 6, 2018, pp. 2425- 2430.
15. A.V. Lyamin, and E.N. Cherepovskaya, "XML-Relational Mapping using Production System", *Intelligent Systems Conference*, 2017, pp. 422- 429.

AUTHORS PROFILE

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.

Su-Cheng Haw is Associate Professor at Faculty of Computing and Informatics, Multimedia University, where she leads several funded researches 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.

Fang-Fang Chua is a senior lecturer in the Faculty of Computing and Informatics, Multimedia University, Cyberjaya, Malaysia. Her research includes requirements engineering, collaborative learning, adaptive coaching and learning analytics. She has published several research papers and has also been invited as a reviewer and program committee in various reputed international journals and conferences. She is involved in multiple funded international and national research projects.