Semantic Based Mapping from XML to Relations

Kamsuriah Ahmad
Faculty of Information Science And Technology,
National University of Malaysia,
Bangi, 43600, Malaysia
kam@ftsm.ukm.my

Reduan Samad
School of ICT
Asia e-University, Malaysia
rs@gmail.com

Abstract— **Extensible Markup Language (XML) is still the dominant standard for data interchange and data representation on the web. With large amount of data now being represented in XML in the web, the question raised is how to store, index, access and retrieve the information effectively. Even though XML can exist as a database on its own but the capability is very limited when compared with sophisticated storage and query ability provided by existing relational database. Now, relational databases are the most widely used technology for storing XML data, where they store the data efficiently and with no redundancy because each unit of information is saved in only one place through normalization step. The process of transferring XML to relations is called mapping and these processes occur frequently. There are many approaches available for mapping XML to relations but the focus are mostly on the structure. The semantic constraints for XML as expressed in functional dependencies are being ignored. In this study we proposed an algorithm for the mapping that is based on the normalization steps through the functional dependencies. When compared with the existing mapping approaches, we proved that our proposed approach is more efficient in terms of generating relations that satisfies the Third Normal Form (3NF).***)*

Keywords- XML, relations, semantic, storing, mapping formatting;

I. INTRODUCTION

The eXtensible Markup Language (XML) has since 2009 emerged as a standard for data representation and interchange on the web. As the size of XML data grows exponentially in the web, the pressure is rising on how to manage the data efficiently. XML is very efficient in supporting other applications, such as annotating text by comments on the content or cross- referencing between documents [16]. Thus it is argued that XML can be effectively used as a database language, therefore the need for a more efficient database language for XML arises. On the other hand, relational databases are already famous for data management in terms of storing, updating and searching capabilities through it communication language, SQL. In view of the maturity of this technology, XML data shall adapt to the way how data has been managed in relational, hence, relational database is the best alternatives for managing XML. Therefore, XML needs to be mapped to relations format and this process is expected to occur frequently. There are many different ways to map and many approaches created in the literature [1], [2] especially considering the flexible nesting structures that XML allows. The approaches for storing XML data, ranging from using files to full-fledged database management systems such as relational, object- relational, object-oriented or native XML database systems. However, there is no standard for XML data type and the method to map data from XML to relations is yet to be defined.

To approach the problem of mapping, the classical relational database design through normalization technique that based on known functional dependency concept is referred. This concept is used to specify the constraints that may exist in the relations and guide the design while removing semantic data redundancies. This concept leads to a good normalized relational schema without data redundancy. To achieve a good normalized relational schema for XML, there is a need to extend the concept of functional dependency in relations to XML and use this concept as guidance for the design. Even though there exist functional dependency definitions for XML [3], [4], [5] but these definitions are not yet considered a standard and still having several limitation. Due to the limitations of the existing definitions, constraints in the presence of shared and local elements that exist in XML document cannot be specified.

To discuss the problem, this paper is organized as follows: section II discusses issues in mapping XML to relations; section III explains the proposed algorithm. Section IV provides a motivating example in the mapping process and at the end is the conclusion and future enhancement.

II. MAPPING FROM XML TO RELATIONS

The mapping from XML to relations is not an easy task because the data model of an XML document is fundamentally different from that of a relational database. Especially the structure of an XML document is hierarchy and the XML elements may be nested and repeated, while relational model is a flat representation of data with tables and columns. Traditionally during the mapping all XML data are directly mapped and stored as one universal table in relational database. It might cause a large number of nested tables and redundant data in the relations.

During the data exchange, XML might come with or without a schema (Data Type Definition, DTD or XML Schema). The existence or the absence of a schema greatly influences the mapping procedure. When the schema of XML data is not available, a generic storage mapping to relational databases is used. Normally, XML document can be seen as a tree model and the mapping is based on the relationship between the nodes and edges of XML document model. But when a schema is available, structural constraint information of an XML document from a schema is used to guide the mapping design. Recently, studies in the context of integrity constraint for XML paying particular attention to the class of keys [6] and functional dependencies [3], [4] as renewed interest to adopt these constraints in the mapping framework. The mapping approaches can be divided into three different categories: (i) Model-based approach, (ii) Structural-based approach, (iii) Semantic-based approach. Model-based approach is a mapping that based on path expression in the XML tree in the absence of schema type. Basically this approach will traverse the tree and store the path for every node visited in a table. This approach ignores totally the semantics aspect of XML. The approaches that fall under this category are Edge [9], XRel [8] and XPev [7]. Structural-based approach is a mapping that based on the existence of type definition such as XML DTD or XML Schema, which conform to XML document. By analyzing the structural properties, it then automatically converts a DTD into relational schemas. The approaches that can be classified under this category are Inlining [14], LegoDB [13] and CPI[17]. However these approaches do not take into account the information about semantic dependencies. Semantics-based approach is a mapping that based on the semantics such as in keys, foreign key and functional dependencies both in the absence or presence of the schema. Some of the proposed strategies that capture various kinds of constraints are: X2R [10], ICDE [4], RRXS [11] and Lv&Yan [15]. However these approaches are not sufficient to generate an optimal relational presentation of XML data. Therefore a method to overcome this limitation is proposed.

III. XTOR: A METHOD FOR MAPPING FROM XML TO RELATIONS

In relational, functional dependencies constraint is useful for generating optimal schema decomposition thru its normalization step. Ironically, this constraint is the most neglected aspect by many researchers as the approach in model-based and structural-based. Two evaluations studies [12],[13] on alternative storage strategies indicate that the shared-inlining algorithm [2] outperforms other strategies in data representation and performance across different datasets and different queries when DTD are available. The studies also indicate that the presence of DTD during the mapping is vital to achieve good performance and compact data representation of XML in relational settings across different datasets and different queries. However, most of the approaches ignore the existence of semantics as expressed in functional dependencies; therefore the resulted relational schema may contain redundancy in the relations. It would be helpful if tools exist to facilitate the general problem of mapping between different data formats, taking the semantics of data into account. To facilitate such mapping we need a language in which to express transformations and constraints, and the ability to reason about the correctness of the transformations with respect to the constraints. Therefore, the purpose of this paper is to propose a mapping method from XML to relations that based on given DTD, functional dependencies and inference rules. The challenging issues that need to be considered when mapping XML to relations in the presence of XML functional dependency and DTD are

- How to map all possible structures that may exist in XML documents to relations.
- How to map the elements that are not involved in XFD.
- How XFD will be used to represent the structure of DTD.
- How to design XML in its relational representation when the existing mapping principles cannot be applied?
- How to adjust the mapping design to guarantee a good relational representation of XML documents?
- How to preserve the parent-child relationship between element and its sub-element?

Therefore in this study a transformation language is developed that is able to extract the semantics information in XML and preserve it during the transformation. This language consists of three components: functional dependencies for XML, inference rules for XML and the mapping function.

A. *Functional Dependencies for XML*

In relational, functional dependency is used to define that X determine Y or X->Y. However functional dependencies for XML is more complicated since we need to deal with the hierarchical structure of XML and the path expression that can be used to express XML. Functional dependencies for XML (XFDs) that we adopt is an expression of the form: (C, Q : X -> Y), where C is the downward context path which is defined by an XPath expression from the root of the XML document, Q is a target path, X is an LHS (Left-Hand-Side) and Y is an RHS (Right-Hand-Side).

B. *Inference Rules*

The intention of using inference rules in the mapping is to create a method on how to infer other legal XFDs that hold in XML given a set of XFDs. Since functional dependencies can be used to specify constraints in XML and infer a non-redundant relational schema for XML. To infer other non-redundant relational schema for XML is related to studying implication problem. The result of this study is the ability to construct a minimal set of XFD, which is a set of all legal XFDs that hold in XML. These XFDs are necessary and sufficient condition to consider, if one wants to generate a non-redundant relational schema for XML. The implication problem is to

solve the following: Given a set of XFDs, what others can be inferred and how? First we need to define that other XFDs inferred from a given set of XFDs is legal XFDs. However in the presence of DTD, the implication procedure is much simpler since the path expression will be computed based on the path language in DTD. The definition of XFD implication is defined as follow [18]:

Definition: XFD Implication

An XFD φ: X -> Y is logically implied by a set of XFD ∑, written ∑ |= φ, if and only if φ holds on every instance that satisfies all dependencies in ∑, that is, φ holds whenever all XFDs in ∑ hold.

The problem of XFD implication is typically addressed by a set of inference rules. These rules are derived in the presence of well-structured DTD, keys, and DTD cardinality constraints. The inference rules that are used for this study are reflexivity, augmentation, transitivity, union, left path expansion, right path expansion, downward expansion, target-to-context, and containment [19]. These rules will be used to find all legal set of XFDs (minimum covers) that guarantee to hold in XML documents.

C. Mapping Function

The strategy adopted in this study, is to produce a relational design, which preserves structural and semantic constraints of the XML data while reduced redundancies. However we have to deal with DTDs that may contain arbitrary regular expressions, that can be recursive; also we have to deals with null values and incomplete relations. All kinds of elements structure that exists in DTD need to be handled properly and a method on how to map these structures to relations must be defined. Since the DTD are built based on the relationship between element and its sub-element, therefore designing the mapping method must be based on this relationship. In a DTD declaration, there are four possible cardinality relationships between an element *e* and its sub-elements e_i. The relationships can be described as below:

- " 1: 1" – ("only operator"): one element *e* has one and only one sub-element e_i
- " 1 : N" – one element *e* has one or more sub-elements e_i
- " 0 : 1" – one element *e* has either zero or one sub-elements e_i
- "0 : N" - one element *e* has either zero or more sub-elements e_i

This concept of relationships is very similar with relational cardinality constraints and is important in database design. Mapping rules (MR) are proposed to cover all possibilities in these relationships. The mapping rules that used in this study are:

MR1: Mapping in the presence of shared-element,
MR2: Mapping in the presence of set-element
MR3: Mapping in the presence of local-element
MR4: Mapping in the presence of keys
MR5: Mapping in the presence of extended simple elements.
MR6: Mapping in the presence of 1:N DTD cardinality constraints between element e and its sub-element e_i
MR7: Mapping in the presence of M:N DTD cardinality constraints between element e and its sub-element e_i.
MR8: Mapping in the presence of IDREF.
MR9: Mapping in the presence of recursive element.

Based on these mapping rules, the mapping method called XtoR is proposed that has the following steps:

i). Construct the DTD graph and store to D={E_1, E_2, A, M, N, r} where E_1 is a finite set of complex element types, E_2 is a finite set of simple element types, A is a finite set of attributes, M is a mapping function from E_1 to element type definitions, N is a mapping function from E_1 to a set of attributes and r is the start symbol.

ii). Using depth-first search strategy traverses the graph starting from the root

iii). For each $\boldsymbol{e} \in E_1$ construct schema trees and determine the root using root determinations (RD).

iv). At each ***e*** node visited, by using inference rules and mapping rules infer a new legal XFD that can be added to $\sum_m$, i.e. the list of all legal XFDs.

v). Read a user input set of XFDs F.

vi). For each XFDs in $\sum_m$ and F, reduce all redundant XFDs to F_m, i.e. a minimum set of XFDs.

vii). Map each attribute in F_m to an attribute in FD relational schema

viii). Generate a 3NF relational schema over the attribute set according to F_m by creating a relation for the root of the schema tree and assign the LHS of each FD as the keys to the relations.

This method is coded into XtoR algorithm as shown in Figure 1. This algorithm is explained as follows: First the structural of XML data is captured using DTD graph and generate the DTD schema, which is the formal description of XML. The algorithm traverses DTD D graph top-down using depth-first starting from the root of D (A(e) = r). The output of this process is an array D that will store all the elements, attributes and features of DTD. The information that was stored for each element is

- EName – element name
- ChildElem – a list of child for the element
- NumChild – the number of child for the element
- parent – the parent of the element
- indegree – the number of nodes point to the element
- cardinality – the relationship of the element
- visited – Boolean function to indicate that the element has been visited

Algorithm XtoR
Input: A set of XFDs and DTD D
Output: A target relational schema R with constraints C

1. Begin
2. D = ReadDTD(DTD)
3. $\sum_m$, SchemaTree() = ConstructSchemaTree(D)
4. F = ReadXFD (XFD)
5. F_m = MinimumCover($\sum_m$, F)
6. Map each attribute in F_m to FD relational attribute
7. Output R + C
8. End

Figure 1: The Proposed XtoR algorithm

Based on the information in D, the algorithm generates schema tree by following the rules described in RD and at the same time it will generate trivial XFDs $\sum_m$. Constructing schema tree and generating $\sum_m$ are done while traversing the DTD graph in one parse. The result of this process is a set of generated schema tree and a set of all generated XFDs $\sum_m$ that guarantees to hold in XML. The schema tree is used to store the XFD variables and the relational schema is generated based on this schema tree. The schema trees will posses all the attributes and features from XML DTD. The structures for schema tree and XFD are shown in Figure 2.

```
Struct SchemaTree {          Struct XFD {
    Topnode                      context
    child()                      target
    Key                          LHS
    Keyref                       RHS
}                            }
```

Figure 2. The Structure for Schema Tree and XFD

While traversing the DTD graph, if the node has more than one parent, indicated by checking e.indegree > 1 then no schema tree will be created. Avoiding this checking step will result in a redundant schema being created. An extra schema tree is created for recursive and multiple element (element that has a cardinality of M:N). Function ConstructSchemaTree (a recursive function) will generate all unique childs (element that has a cardinality of 1:1) and unique parent (element that has a cardinality of 1:M). The algorithm then read the given XFD and stored into F. However only a minimal set of XFDs is considered, therefore a mechanism is needed to reduce the number of XFDs. For this reason, finding a minimal set F_m is proposed. That is to reduce a list of XFDs in $\sum_m$ that is redundant. To the best of our knowledge, this is the first algorithm for finding a minimum cover for XFDs by inferring from XML constraints (XFD, DTD, keys and DTD cardinality constraints).

After a set of F_m is constructed, the final step is mapping the paths in XFDs to relational attributes in order to produce a set of relational functional dependencies and a relational storage for the XML data. The resulted relational schema preserves the content and the structure information of the original XML document, removes redundancy as indicated by the XFDs, and enforced efficiently using relational primary key constraints. The generated schema resulted from the XtoR is correct with respect to keys and functional dependencies. In fact the schema produced is in 3NF, as proved by the following proposition:

Proposition: Given a mapping σ, an XML document *T* conforming to DTD *D*, and a set Σ of XML FDs that generated from keys over D, if T |= Σ, then each relation in σ(*T*) is in Third Normal Form(3NF).

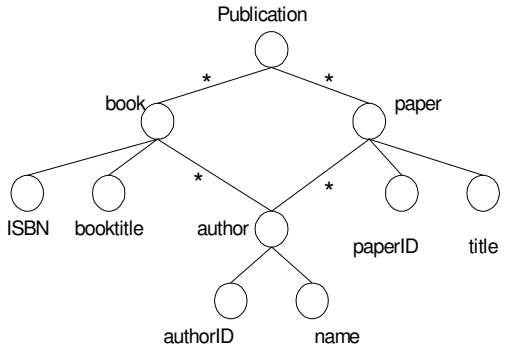
Proof. To satisfy the Third Normal Form, we need to prove that each relation is in First and Second Normal Form. Since attributes of all relations in σ(*T*) are extracted from attributes or text nodes in a document, attributes of all relations are atomic. That is, all relations are in First Normal Form (1NF). Because of XML FDs are all in the set of Σ, the semantics is in Σ. All FDs on relational data are in the correspondence Γ of Σ. We can conclude that each non key attribute in each relation is functionally dependent upon the primary key of the relation. That is, all relations are in Second Normal Form (2NF). According to the process of mapping, a relation is created for each FD. Therefore, the relations created in step 2 are in 3NF. Additionally, the relations created in other steps used FD to describe the property that the values of some attributes of a tuple (keys) uniquely determine the values of other attributes of the tuple and the attributes that are not dependent upon the primary key have been eliminated. That is, these relations are in 3NF. So, all the relations σ(*T*) are in 3NF.

IV. MOTIVATING EXAMPLES

Publication dataset [14] as in Figure 3 is used to illustrate the effectiveness of the proposed method. This dataset describes the publication that has many books and papers. Each books and papers contain information about the authors who published either books or papers. The complexity of this dataset is the existence of shared-element as shown in author nodes. The author nodes is appeared under both book and paper nodes. Based on manual observation to discover the dependency constraints in the document, three XFDs can be derived from the Publication set. The XFD1 and XFD2 are used to express that ISBN and paperID are the keys to book and paper elements respectively. XFD3 is used to express that authorID determines the name of author in the whole document. These set of constraints become an input to the proposed algorithm (XtoR) and other algorithms under discussion (RRXS[11] and Lv&Yan [15]).

To evaluate the effectiveness of the proposed method, an experiment is conducted where the relational schema generated by XtoR, RRXS and Lv&Yan method is compared when using Publication dataset as an input. RRXS and Lv&Yan methods which are under semantic-based

approach are used in the comparison because they have taken into consideration an XFD in the mapping.



<!ELEMENT publication(book*, paper*)>
<!ELEMENT book (ISBN, booktitle, author*)>
<!ELEMENT paper(paperlD, title, author*)>
<!ELEMENT author (name)
<!ATTLIST author authorlD

XFD1 //book: ISBN -> ε
XFD2 //paper: paperID -> ε
XFD3 //author: authorID -> name

Figure 3. DTD graph, its schema and corresponding XFD

Given the DTD graph, its schema and corresponding XFD that may exist in the Publication dataset, the relational schema generated by the three methods are shown in Figure 4. XtoR method generates a good relational schema for Publication dataset when compares with the schema generated by RRXS and Lv&Yan method. The relational schema generated by RRXS as in Figure 4(ii), produce two equivalent tables that belong to the same concept which are Author and Author1 table. The reason for this redundant creation is that the algorithm did not check the existence of already created table for the same concept of object (Author), the algorithm blindly created a new one. These redundant tables may lead to the update anomaly problems. The method by Lv&Yan will create two types of tables: based on structural DTD and based on semantics presented by XFD. Basically three steps involved in this algorithm: i) using structural DTD, a separate relation will be created for each non-leaf vertex and leaf, ii) for each parent-child relation between two vertexes connected by a * operator, a separate relation is created, and ii) for each XFD defined over DTD, a separate relation is created. Based on this algorithm the resulted relational schema is shown as in Figure 4(iii). As observed, the F_3 table and the Author table are redundant. These tables are used to describe the same concept of object (Author), therefore this will lead to update anomaly problems. For instance if a new author is added to the publication, both the relations F_3 and author need to be updated for the database to satisfy the constraints. The relational schema generated by XtoR overcomes the limitation of these methods by producing a good relational schema design for XML with no redundant relations. In fact the schema produced is correct with respect to keys and functional dependencies.

Book (ISBN, booktitle) Author (authorID, name) Paper(paperID, title)
Book-author(ISBN, aID)
Paper-author(paperID, authorID)

i.Schema generated by XtoR

Book (ISBN, booktitle) Author (authorID, name) Author1(ID, authorID, name) Paper (paperID, title)
Book-author(ISBN, aID)
paper-author1(ID, paperID, author1.ID)

ii.Schema generated by RRXS

Publication(ID)
Book(ID, ISBN, booktitle, author.ID) Paper(ID, paperID, title, author.ID) Author(ID, authorID, name)
Publication-book (publication.ID, book.ID)
Publication-paper(publication.ID, paper.ID) Book-author(book.ID, author.ID)
Paper-author(paper.ID, author.ID)
F_1(publication.book.ISBN, publication.book.ID)
F_2(publication.paper.paperID, publication.paper.ID)
F_3(publication.paper.author.authorID, publication.paper.author.name)

iii. Schema generated by Lv&Yan

Figure 4. The comparison of schema generated by XtoR, RRXS and Lv&Yan

V. MOTIVATING EXAMPLES

This paper investigated the problem of how to design a normalized relational schema for XML data and how to automate the instance mapping. A new approach called XtoR is proposed, where when functional dependencies and DTD were given, redundancy in XML document can be detected and used for mapping XML to relational. The effectiveness of XtoR algorithm is evaluated by comparing the resulted relational schema produced by XtoR with RRXS and Lv&Yan[15]. The proposed approach is able to generate a reduced redundancy relational schema when compared with the other two algorithms. XtoR also able to preserve the constraints as expressed in functional dependencies. It can be efficiently operated, automated and eliminates unnecessary ID caused by Hybrid Inlining algorithm[2]. However this study had considered only DTD language for XML mapping. Since XML Schema is widely used as a schema to define the structure of XML, now there is a need to extend the proposed algorithm which will consider this schema. Also as an immediate task, we would like to find efficient algorithm for the implication problem in the functional dependencies defined above. Through this study we hope that it will give some contributions to the database community.

REFERENCES

[1] K. Ahmad, "A comparative analysis of managing XML data in relational database". Lecture Notes in Artificial Intelligence LNCS/LNAI 6591, 2011, pp: 100-108.

[2] M. Atay, C. Chebotko, D. Liu, S. Lu and F. Fotouhi. "Efficient schema-based XML-to-relational data mapping". Inf. Sys., 3, 2007, pp: 458-476.

[3] F. Fassetti and B. Fazzinga, "Approximate functional dependencies for XML data". Local Proceedings of ADBIS, 2007, pp: 86-95.

[4] S. Shahriar and J. Liu, "On defining functional dependency for XML". International Conference on Semantic Computing, 2009, pp: 595-600.

[5] T. Trinh, "Using transversals for discovering XML functional dependencies". Proc. of FoIKS, LNCS4932, 2007, pp:199-218.

[6] P. Buneman, S. Davidson, S., Fan, W., Hara, and C. Tan, "Keys for XML. In Proceeding of the *10th World Wide Web Conference.* 2001, pp. 201-210.

[7] J. Qin, S . Zhao, S . Yang, S., Dou, " XPEV: A storage model for well-formed XML documents". FSKD. LN AI 3613, 2005, pp.360-369.

[8] M. Yoshikawa, T. Amagasa, a n d T. Shimura, "XRel: A p ath-based approach to storage and retrieval of XML documents using relational database". ACM Transactions on Internet Technology, vol.1, 2001, pp.110-141.

[9] P. Bohannon, J. Freire, J., P. Roy, and J. Simeon. "From XML schema to relations: A cost-based approach to XML storage". In Proceedings of the 18th International Conference on Data Engineering, pp.64-74. 2002.

[10] Y. Chen, S . Davidson, a n d Y . Zheng, " Constraint preserving XML storage in relations". In Proceeding of the 9th International Conference of Database Theory, 2002, pp.7-12.

[11] Y. Chen, S., Davidson, C. Hara C., Y. Zheng. "RRXS: Redundancy reducing XML storage in relations". In Proceedings of 29th. International Conference on Very Large Data Base, 2003, pp.189-200..

[12] S. A. Amer-Yahia., F .Du., a n d J. Freire. "A c o m prehensive solution to the XML to relational mapping problem". In Proceedings of the 6th Annual ACM International Workshop on Web Information and Data Management, 2004, pp.31-38..

[13] D. Florescu, and D. Kossman. "A performance evaluation of alternative mapping schemes for storing XML data in a relational database". In Proceedings of the VLDB. 1999.

[14] J.Shanmugasundaram, "Relational databases for querying XML documents: limitations and opportunities". Proceedings of the 25th VLDB Conference, 1999, pp 302-314.

[15] L. Tv. and P. Yan. "Mapping DTDs to relational schemas with semantic constraints". Journal of Information and Software Technology vol. 48(4), 2006, pp 245-252.

[16] K. Schewe, "Redundancy, dependencies and normal forms for XML databases". Sixteenth Australasian Database Conference ADC Vol 39. Dobbies G. and Williams H (eds). 2005.

[17] D. Lee, and W.W. Chu, "CPI: Constraint-preserving inlining algorithms for mapping XML DTD to relational schema". Journal of Data Knowledge and Engeering volume 39(1): 2001, pp 3-25

[18] Vincent, M.W. and Liu, J., "Checking functional dependency satisfaction in XML". XML Symposiums Lecture Notes in Computer Science. 2005, 3671, pp 4-17.

[19] K. Ahmad, H. Ibrahim, "Functional dependencies and inference rules for XML", Proceedings of International Symposium on Information Technology. 2008.