Journal of Computer Science 7 (11): 1659-1666, 2011
ISSN 1549-3636

Normalized Relational Storage for Extensible Markup Language (XML) Schema

[1]Kamsuriah Ahmad and [2]Reduan Samad
[1]Strategic Information System Research Group
Faculty of Information Science and Technology,
University Kebangsaan Malaysia
[2]School of ICT, Asia e-University, Malaysia

Abstract: Problem statement: The use of XML as the common formats for representing, exchanging, storing, integrating and accessing data posses many new challenges to database systems. Most of application data are stored in relational databases due to its popularity and rich development experiences over it. Therefore, how to provide a proper mapping approach from XML model to relational model become the major research problems. Current techniques for managing XML in relational technology consider only the structure of an XML document and ignore its semantics as expressed by keys and functional dependencies. **Approach:** In this study we present an algorithm for generating an optimal design for XML in relational setting. The algorithm is based on computing a set of minimum covers for all functional dependencies on a universal relation when given XML Functional Dependencies (XFDs) and the schema information. However we need to deal with the hierarchical nature of XML and to define XFDs in this structure. **Results:** We show that our algorithm is efficient in terms of reducing data redundancy and preserving semantic expression. **Conclusion/Recommendations:** Being able to infer XML functional dependencies constraints to relational views of XML data is a first step towards establishing a connection between XML and its relational representation at the semantic level.

Key words: XML Functional Dependencies (XFDs), schema mapping, semantic constraints, functional dependencies, relational databases, decision problems

INTRODUCTION

Extensible Markup Language (XML) is fast emerging as the dominant standard for data interchange and data representation on the web (Amirian and Alesheikh, 2008; Ahmad, 2011). It's nested; self-describing structure provides a simple yet flexible means for application to model and exchange data. Data exchange involves transformations of data and therefore the "transformed" data can be seen as a view of its source. Thus, the problem we investigate is how constraints are propagated to views. Even though XML can exist as a database but the capability is very limited when compared with sophisticated relational database storage (Fadda *et al*., 2008; Alfred *et al.*, 2010). We expect that the needs to convert data formats between XML and relational models will grow substantially (Ahmad, 2011). But the problem with XML is that it is only syntax and does not carry the semantics of the data. Recently, keys (Hartmann *et al*., 2008), foreign keys (Hartmann *et al*., 2010) and functional dependencies (Shahriar and Liu, 2009) have been proposed to capture semantic constraints and various aspects of these proposals have found their way into XML-Data and XML Schema. Among these proposals, functional dependencies for XML are important to capture the semantics of XML data. However, in relational databases, the semantic constraints have been proved useful in recognizing keys, normalizing to make a good design, preventing update anomaly, reduced redundancy and etc. Functional Dependencies (FDs) are critical part of its semantics and FDs for XML, called XFDs are the counterpart of those for relational data. They must be taken advantage of in the process of mapping. A natural question to ask, therefore, is how information about constraints in FDs can be used to generate a good database schema. In this study, we analyze constraints for XML as expressed in functional dependenciesand proposed an algorithm on how to preserve these constraints in relational setting.

Corresponding Author: Kamsuriah Ahmad, Strategic Information System Research Group, Faculty of Information Science and Technology, University Kebangsaan Malaysia, Malaysia

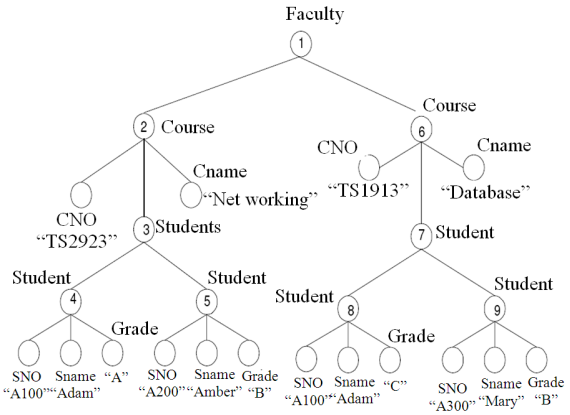

Fig. 1: An XML document about faculty

Our motivations of mapping XML to relational are based on two aspects: (i) Data redundancies and (ii) Preserving semantic constraints. Data redundancies are usually due to some form of dependencies among the data, such as functional dependencies and multi-valued dependencies as expressed in relational databases. Traditional functional dependencies are not suited for XML data because of the structural difference between the two types of database. On the other hand, dependencies naturally exist among data no matter what format the data is in. An array of researches has addressed the issues on storage strategy (Chen *et al*., 2003; Lv and Yan, 2006; Xing *et al*., 2007; Patel and Atay, 2007; Ferraggine *et al*., 2009; Kim and Peng, 2011; Huiling and Feng, 2010; Feng and Jingsheng, 2009) unfortunately the resulting relational applications do not offer the required guarantee for the preservation of data integrity and reduced data redundancy.

In this study we illustrate how in the presence of functional dependencies, data redundancy can be detected in XML documentsand how to produce redundancy free relational schema, based on the information given. At the same time, the semantic constraints, as well as the content and the structure of XML will be preserved. As an example, consider the XML tree representation of a faculty document shown in Fig. 1. Given this document and our understanding of its semantics, we may wish to state the following constraints:

C1: In the context of the whole document, each course is uniquely identified by CNO
C2: If two students with the same SNO, then they must have the same SNAME
C3: Each student will get their GRADE for every course they enrolled

The first constraint is an example of an absolute key, where the key is defined over the whole document and the third is an example of relative key, where the key is defined within the same context, in the terminology used by Hartmann *et al*. (2008). The second constraint is an example of a functional dependency (Hartmann *et al*., 2010; Shahriar and Liu, 2009) and cannot be expressed as a key constraint. Even though attempts to define functional dependencies have been made by several groups of researcher, they have different expressive power and some cannot express the constraint that the student number determines the student name for all Student nodes in the entire document as shown in Fig. 1. This is because Student nodes are located in different paths (under both Course and Faculty nodes). Even though the algorithm proposed in Chen *et al*. (2003); Lv and Yan (2006); Xing *et al*. (2007) and Kim and Peng (2011) tried to map XML to relational schema in the presence of functional dependencies by using redundancy reducing strategies, but they ignored the redundancy that cause by the redundant nodes as in Courses and Students. Also, the Student nodes by the value of "Adam" were stored twice in the trees; inability to detect these redundancies will caused redundancies to occur in the relational views. It is important to check for these redundancies before the mapping process to take place and without the semantic knowledge the same data will be stored multiple times. With the information about constraint in the schema, we can generate an optimal relational database and this is the basis of our study. So the objectives of the study are as follows:

- To propose a more general definition of functional dependencies that will detect data
- redundancy in XML documents efficiently
- To propose an algorithm for generating a good relational view of an XML data
- To prove that the algorithm is able to reduce data redundancy and preserve semantic constraints.

MATERIALS AND METHODS

The methodologies used in this study are as the following:

- Capture the structure of XML data by reading the DTD file, which is the formal description of XML and then generate the DTD schema
- By using the reduced-redundancy and constraint-preserving algorithm (i) remove redundant node

that caused by the redundant elements, (ii) remove system generated IDs if there exist value-based keysand preserved the Parent-child relationship

- By mapping paths in XFDs to relational attributes, we will get a set of minimum covers and produce a relational storage for the XML data which preserves the content and the information structure of the original XML document, removes redundancy as indicated by the XFDsand enforced efficiency by using relational primary key and foreign key constraints

In relational databases, the normalization process reduces or eliminates data redundancies in generating a good relational database designs. Similar to relational databases, updates in an XML structure may cause anomalies if the XML data is redundant. A schema (or data definition) language is used to specify structures and constraints for a model. We study the publication of relational data in XML documents, the propagation of its constraints and the associated decision problems. Constraints are fundamental importance in databases and is also important to many forms of hierarchically structured data including XML documents, particularly in the data mapping. In our algorithm, semantic information in keys and functional dependencies were used to guide the schema design. We ignored the ordered features provided by XML in our mapping algorithm. If the features are so important, we can simply add another parameter to our schema to capture the ordered structures. We omit them, as it is not the focus of our study. Before the mapping algorithm is proposed, the notations used in this study will be defined, which are similar with the one in Kim and Peng (2011); Lv and Yan (2006) and Shahriar and Liu (2009) but with minor modification to suit our mapping strategies. The definition are as follows.

XML Tree: As well known, an XML document can be represented by a tree. We call elements that have sub-elements and/or attribute as a complex element and denote it as E_1. And element that only have a single value as a simple element and denote as E_2. Let E_1 and E_2 be disjoint sets of element names, A be a set of attribute names, E = E_1 union E_2, and E and A be disjoint. Element names and attribute names are called labels. An XML tree is defined to be T = (V, lab, ele, att, val, root), where (1)V is a set of nodes; (2) Lab is a mapping V -> E uA which assigns a label to each node in V; a node v in V is called a complex element node if lab(v)$\in E_1$, a simple element node if lab(v)$\in$E 2 and an attribute node of lab(v)$\in$ A. (3) Ele and att are functions from the set of complex elements in V: for every v $\in$ V, if lab(v)$\in_1$, then ele(v) is a set of element nodesand att(v)is a set of attribute nodes with distinct labels. (4) val is a function that assigns a values to each attribute or simple element. (5) Root is the unique root node labeled with complex element name r. (6) if v'$\in$ ele(v) union att(v), then we call v' a child of v. The parent-child relationships defined by ele and att will form a tree rooted at root.

XML DTD: DTD describes the structure of XML documents and are considered as the schemata for XML documents. A DTD schema is denoted by 6 tuple = (E_1, E_2, A, P, R, r) where (1) $E_1 \subseteq E_1$ is a finite set of complex element names, (2) $E^2 \subseteq E_2$ Is a finite set of simple element names, (3) A$\subseteq$A is a finite set of attribute, disjoint from E, (4) P is a mapping function from E_1 to element type definitions: $\forall \tau \in E_1$, P(τ) is a regular expression, α ::= ε | τ' | α |α | α,α | α* | where ε is the empty word,' τ, E_1 union E_2 and "|", ",", "*", denote union, concatenation and the Kleene closure, respectively; (5) R is a mapping function from E_1 to sets of attributes in A (6) r is the element type of the root, which is distinct from all other symbols. A path in D is a string l_1,…, l_m, where $l_1$is in the alphabet of P(r), li is in the alphabet of P(l_{i-1}) for i$\in$[2, m-1], l_m is in the alphabet of P(l_{m-1}) or in R(l_{m-1}).

Paths in XML trees: The path language we adopt is a common fragment of XPath: Q::= ε | l | Q/Q | // where ε is the empty path, l is a node label, "/" denotes concatenation of two path expressions (child in XPath)and "//" means descendant-or-self in XPath. A path P is a sequence of labels l_1/…/l_n. A path expression Q defines a set of paths, while "//" can match any path. We use p$\in$Q to denote that p is in the set of paths defined by Q. For example, //course/students/student/name $\in$ //name.

Value equality and node identity: To reduce data redundancy, all the nodes in a tree need to be compared. When comparing two nodes n_1 and n_2 in an XML tree T, we need to define the equality between them. Obviously, if $n_1$and n_2 are the same node (denoted n_1=n_2), they should be considered equal, but this kind of node equality is not sufficient because there are cases where two distinct nodes have equal values. So we need to define value equality between nodes. Since we consider the ordering of child elements insignificant, our definition of value equality is different from those published previously. The value of equality is defined as follows:

- Let $n_1$and n_2 be two nodes in T. We say $n_1$and n_2 are value equal, denoted $n_1=_v n_2$, if $n_1$and n_2 areof the same label
- $N_1$and n_2 are both attribute nodes or simple element nodes and the two nodes have the same value
- $N_1$and n_2 are both complex elementsand for every child node $m_1$of n_1, there is a child node m_2 of n_2 such that $m_1=_v m_2$ and vice versa

Functional dependencies for XML: First, we should emphasize that semantic constraints (Shahriar and Liu, 2009; Hartmann *et al*., 2010) are not part of XML specifications. They can be regarded as the extension of XML schema to make XML documents more significant. In this study, we mainly discuss functional dependencies constraints. Functional Dependencies (FDs) were introduced in the context of the relational data model by Codd in 1972. As in relational databases, functional dependencies for XML (XFDs) are used to describe the property that the values of some attributes of a tuple uniquely determine the values of other attributes of the tuple. The difference lies in that attributes and tuples are basic units in relational databases, whereas in XML data, they must be defined using path expressions. We also show how to use this constraint to detect data redundancies in XML documents before mapping to relational. So the resulted relational schema is redundancy free and update anomaly can be avoided. Functional dependency that we adopt is an expression of the form:

$(Q, [P_{x1}, P_{x2,}\ldots, P_{xn}\text{-}> P_y])$

where, Q is the FD header path which is defined by an XPath expression from the root of the XML document. P_{xi} $(1 \leq i \leq n)$ is an LHS (Left-Hand- Side) entity type which consists of an element name with optional attributes(s) and P_y is an RHS (Right- Hand-Side) entity type which consists of an element name with an optional attribute name. An XML FD (Q, $[P_{x1}, P_{x2,}\ldots, P_{xn}\text{-}> P_y]$) specifies as follows: For any two subtrees identified by Q, if they agree on P_{x1}, $P_{x2,}\ldots$, P_{xn}, they must agree on P_y, if it exists. From Fig. 1, the constraints exist in the XML document can be expressed as:

FD1://course (CNO->course)
FD2://student (SNO->student)
FD3://course (/CNO,/students/student/SNO->grade)

The constraint path can be achieved through definition of value equality and node equality. If the value of the path is equal then violation occurred.

RESULTS

Since XML functional dependencies are used to guide the relational design, we now turn to the implication problem: Given a set of XFDs, what others can infer and how?

Implication approach: Implication is defined as follows: An XFD φ: X -> Y is logically implied by a set of functional dependencies F, written F |= φ, if and only if j holds on every instance that satisfies all dependencies in F, that is, φ hold whenever all XFDs in F hold.

This problem is typically addressed by finding a set of inference rules, e.g. Armstrong's Axioms for functional dependencies in relational databasesand proved that they are sound and complete (Yan and Ma, 2011; Shahriar and Liu, 2009; Hartmann *et al*., 2010). Compared to the relational counterpart, however, the task of finding such a set of inference rules for XFDs is much more difficult. This is because XFDs are based on path expressions while relational FDs are defined on attribute names.

Constraint preserving mapping algorithm: In this approach, we extend Armstrong's Axioms (reflexivity, augmentation and transitivity) to use path expressions instead of simple attributes. DTDs and XML Schema documents can be used to restrict the structure of XML documents (Siau, 2011; Feng and Jingsheng, 2009). For simplicity DTD is used in this study, but the ideas presented here also apply to any schema file including XML Schemas documents. In Fig. 2 is the DTD schema that conforms to the diagram in Fig 1.

The generated DTD schema will be:

- E_1= {faculty, courses, course, students, student}
- E_2= {cname, sname, address, sname, address, grade}
- A = {SNO, CNO}
- P(faculty) = {courses}
- P(courses) = {course*}
- P(course) = {cname, 0073tudents}
- P(students) = {student*}

```
<!ELEMENT faculty (courses)
<!ELEMENT course (course*)
<!ELEMENT course (cname, students)
<!ATTLIST course CNO ID >
<!ELEMENT students (student*)
<!ELEMENT student (sname, grade)
<!ATTLIST student SNO ID >
```

Fig. 2: DTD file for faculty

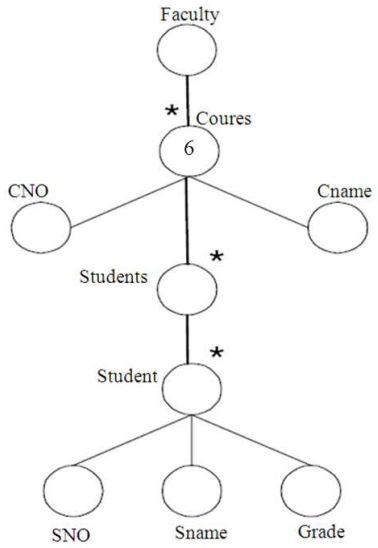

Fig. 3: DTD structure

- P(student) = {sname,grade}
- P(sname)=P(grade)=P(grade)=P(cname)=S
- R(course) = {CNO}
- R(student) = {SNO}
- R(courses)=R(students)=R(CNO)=R(SNO)=_
- r = {faculty}

From the contents above we can see there are some repeated elements because they are repeated in DTD. These are redundant data and should be avoided during the mapping. While traversing the DTD structure, the information that was stored for each of the nodes are:

- EName-element name
- ChildElem-a list of child for the element
- NoChild-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

The DTD structure that conforms to the XML document in Fig. 1 is at Fig. 3, where the symbol "*" denotes zero or many occurrence.

To achieve optimization, a redundant node, which satisfies the following condition, needs to be removed:

- Indegree = 1
- Node cardinality = 1, which is a singleton element
- Node child cardinality = 1 and
- A complex element

Through this step, nodes Courses under node Faculty and node Students under node COURSE will be removed. The mapping algorithm relies on an input set of XFDs and DTD file. An inferring function, which given an XDF $\phi : X \rightarrow Y$ and a schema D, determines whether or not ϕ can be inferred from D. The algorithm studys as follows: Traverses D top-down starting from the root of D, P(e) = rand generates a set F of FDs that is a cover of F^+, i.e., a superset of F_m. More specifically, at each $e \in P(e)$ encountered, it expands F by including certain FDs propagated from Σ. It then removes redundant FDs from F to produce a minimum cover F_m. But in the presence of DTD information, finding minimum covers should be much easier. First, we need to consider the relationship between elements. The relationships that may appear between one elemente and its sub-element e_i in DTD are:

1: 1 = One element e has one and only one subelement e_i
1: N = Meaning that one element e has one or more sub-elements e_i "N:M" – meaning that one element e can have at least one sub-element e_i and these sub-elements are to belong to one or more different parent elements
1: 0 = Meaning that the sub-element possess an optional operator
1:0 = N-Meaning that the sub-element is an element with star operator

The XFDs will be read according to the syntax, the header, the determinant (LHS) and determine (RHS). The rules of implication applied here, i.e., given certain XFD what other XFDs can be implied. We extend the standard Armstrong rules (Reflexivity, Augmentation and Transitivity). But in the existence of DTD, the process can be simplified. For every singleton element (we treat simple elements and attributes are the same), it is true to say that:

- Student/sno -> student/sno/S
- Student/sname -> student/sname/S
- Student/grade -> student/grade/S
- Course/cno -> course/cno/S
- Course/cname ->course/cname/S

The element S is to indicate the values that contain in every element. If two elements have the same values they are considered as an identical element and the concept of value equality applied here. We are not using the forms of key provided in DTD, because the known limitation, the key in the form of XFDs will be input instead. The following proposition is proposed.

Proposition: Every element has at least one key. We assume that in every relation there exists a unique key, so that every relation is unique. If two elements have exactly the same DTD expression and values, then they are considered as identical and denote a key for on element e as e. key. Then the basic functional dependencies exist.

If the XFD is in the form of *e*.key -> *e*, where key is a unique sub-element of *e* then e.key is a key for the element.

Normally the LHS of the XFDs will become the key for the relation. Referring to this form, CNO and SNO are keyed for course and student nodes respectively. When considered e. key is a key for the relation, then this rule can be deduced.

Proposition: If e_i. key -> e_i then e_i. Key will determine every $e_i \in P(E_1) \cup R(E_1)$ by using procedure. The constraint preserving mapping algorithm studys as follows:

- Every complex element in E_1 will be the root of the relations
- Map every e.key to the attribute of the elements
- Consider the relationship between the elements, if exist M:N relationship then create a new elements
- To maintain the parent-child relationship, every child element node needs to refer to the parent node

Example: If given/student/sno/S -> student, can we implied that student/sno/S -> student/sname/S? Since each STUDENT has exactly one SNAME element as a child (1:1 relationship) and nodes have unique identifiers, then it is true to say that /student -> student/sname then using singleton element, this XFDs is trivially satisfied /student/sname->student/sname/S Finally based on transitivity rule, the following can be derived:

- /student/sno/S -> student
- /student -> student/sname
- /student/sno/S -> student/sname
- student/sname -> student/sname/S

Therefore by transitivity rule, /student/sno/S-> student/sname/S.

This will generate the minimum covers for the relation rules. The elements that are in the same set or rules will group into the same classes; we called this as Equivalence class. At the end of this step we will get (Rule (R_1),…, Rule (R_n)) that will form the schema relation. From the example above we called E_{new} as course-student node. Since there exist other element (course-student node) associate with Grade therefore this node is dropped from Student. Then we have /course/cno/S, /student/sno/S -> course-student course-student -> course-student/grade/S by transitivity rule /course/cno/S, /student/sno/S -> coursestudent/ grade/S

Based on the basic Armstrong inference rules, the following can be deduced.

- Two complex elements E_i, E_j where E_i, $E_j \in E_1$, E_i, E_j has a 1:N relationship and E_i is a prefix of E_j, then Ei. key -> E_i, then Ei. key -> E_j
- Two complex elements E_i, E_j where E_i, $E_j \in E_1$, E_i, E_j has a M:N relationship and E I is a prefix of E_j, then E_i.key, E_j. Key ->Enew,where Enewis a new node

A transformation from the above XML data to R can be specified as:

σ = (Rule (student), Rule (course), Rule (CS))

The set of equivalence classes according to the rules are:

- Rule(student) = {SNO, sname, grade}
- Rule(course) = {CNO, cname}
- Rule(CS) = {CNO,SNO}

Where, the minimum covers for the relations are:

- /student/sno/S -> student
- /student/sno/S -> /student/sname/S
- /course/cno/S -> course
- /course/cno/S -> /course/cnameS
- CS/cno/S,course-student/sno/S -> CS
- CS/cno/S,course-student/sno/S->CS/grade/S

At the end, the following schema will be generated:

- Student(sno, sname) PRIMARY KEY SNO
- Course(cno, cname) PRIMARY KEY CNO
- CS(cno, sno, grade)
- Primary Key SNO, SNO
- Reference Key SNO Refer to student(SNO)
- Reference Key CNO Refer to course(CNO)

DISCUSSION

To evaluate the algorithm, the resulted schema is compared with the one that has been generated using Inlining (Patel and Atay, 2011; Huiling and Feng, 2010; Feng and Jingsheng, 2009) because this technique also considers set-value nodes.

Table 1: Student

SNO	Name
A100	Siti
A200	Amin
A300	Mary

Table 2: Course

CNO	Cname
TS2923	Networking
TS1913	Database

Table 3: Course-student

CNO	SNO	Grade
TS2923	A100	A
TS2923	A200	B
TS1913	A100	C
TS1913	A300	D

Table 4: Courses

CID	CNO	Cname
1	TS2923	Networking
2	TS1913	Database

Table 5: Student

SID	Parent ID	Parent code	SNO	Name	Grade
1	TS2923	course	A100	Adam	A
2	TS2923	course	A200	Amber	B
3	TS1913	course	A100	Adam	C
4	TS1913	course	A300	Adam	B

We do not compare our technique with that of Xing *et al*. (2007); Kim and Peng (2011) since our definition of functional dependencies is not expressible in their technique. Our resulted relational schema is appeared as in Table 1-3. Using Patel and Atay (2011); Huiling and Feng (2010); Feng and Jingsheng (2009) the resulted schema is appeared as in Table 4 and 5.

Even though our algorithm will produce more tables if compared with the algorithm proposed by other researchers (Patel and Atay, 2011; Kim and Peng, 2011; Feng and Jingsheng 2009) but we reduced the number of attributes in the relation. Some node ids (ID, parentID and parentCODE) are removed; this is possible as each instance node can be uniquely identified using key-based value information. In our method, which is based on traditional database theory, records can be extracted efficiently by using keys to join relations between parent and child. As the result, our method is able to produce resulting tables with less data redundancies. The generated schema in our algorithm is correct with respect to keys and functional dependencies. In fact, our schema is in 3NF (Xing *et al*., 2007), as proved by the following proposition.

Proposition: Given a mapping, an XML document T conforming to DTD D and a set $\sum$ of XML FDs that generated from keys over D, if $T|= \sum$, then each relation in (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 $\sum$, the semantics is in Σ. All FDs on relational data are in the correspondence $\Gamma$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.

CONCLUSION

We have investigated the problem of how to design a normalized relational schema for XML data and how to automate the instance mapping. We have developed a new approach where, with given functional dependencies and DTD, we can detect redundancy in XML document. This approach able to improve the mapping of XML to relational by reducing data redundancy and at the same time preserve the constraints as expressed in functional dependencies. It can be efficiently operated, automated and eliminates unnecessary ID. As an immediate task, we would like to address implication problem in functional dependencies as defined above. We hope this study will able to give some contributions to the database community.

REFERENCES

Ahmad, K., 2011. A comparative analysis of managing XML data in relational database. Intell. Inform. Database Syst., 6591: 100-108. DOI: 10.1007/978-3-642-20039-7_10

Alfred, R., K.P. Hue, L. S. Khee and R. Alfred, 2010. The importance of maintaining a proper database on forest restoration program for orangutans in Borneo. Am. J. Environ. Sci., 6: 137-151. DOI: 10.3844/ajessp.2010.137.151

Amirian, P. and A.A. Alesheikh, 2008. publishing geospatial data through geospatial web service and XML database system. Am. J. Applied Sci., 5: 1358-1368. DOI: 10.3844/ajassp.2008.1358.1368

Chen, Y., S. Davidson, C.S. Hara and Y. Zheng, 2003. RRXS: Redundancy reducing XML storage in relations. Proceedings of 29th International Conference on Very Large Data Base (ICVLDB'03), ACM, pp: 189-200. portal.acm.org/ft_gateway.cfm?id=1315469&type=pdf

Fadda, E.H.R., M. Kakish and E.J. Akawwib, 2008. Relational GIS and remote sensing database system for al-salt area, Jordan. Am. J. Eng. Applied Sci., 1: 241-247. DOI: 10.3844/ajeassp.2008.241.247

Feng, Y. and X. Jingsheng, 2009. Mapping XML DTD to relational schema. Proceedings of the 1st International Workshop on Database Technology and Applications, Apr. 25-26, IEEE Xplore Press, Wuhan, Hubei, pp: 557-560. DOI: 10.1109/DBTA.2009.85

Ferraggine, V.E., J.H. Doorn and L.C. Rivero, 2009. Handbook of Research on Innovations in Database Technologies and Applications: Current and Future Trends. 1st Edn., IGI Global snippet, Hershey, Pa., ISBN: 1605662429, pp: 1124.

Hartmann, S, S. Link and T. Trinh, 2010. Solving the Implication Problem for XML Functional Dependencies with Properties. Proceedings of the 17th International Workshop, Logic, Language, Information and Computation (WoLLIC'10), Springer-Verlag Berlin, Heidelberg, pp: 161-175. http://dl.acm.org/citation.cfm?id=1886804

Hartmann, S., S. Link, H. Köhler, T. Trinh and J. Wang, 2008. On the notion of an XML key. Semantics Data Knowl. Bases Lecture Notes Comput. Sci., 4925: 103-122. DOI: 10.1007/978-3-540-88594-8_5

Huiling, L. and Y. Feng, 2010. Research of Mapping XML DTD to Relational Schema. Proceedings of the 2nd Pacific-Asia Conference on Circuits, Communications and System, Aug. 1-2, IEEE Xplore Press, Beijing, pp: 129-132. DOI: 10.1109/PACCS.2010.5627006

Kim, J. and Y. Peng, 2011. A semantic similarity analysis for data mappings between heterogeneous XML schemas. IGI Global, pp: 37-52. DOI: 10.4018/978-1-60960-485-1.ch003 http://www.igi-global.com/viewtitlesample.aspx?id=52148

Lv, T. and P. Yan, 2006. Mapping DTDs to relational schemas with semantic constraints. Inform. Software Technol., 48: 245-252. DOI: 10.1016/j.infsof.2005.05.001

Patel, J. and M. Atay. 2011. An efficient access control model for schema-based relational storage of XML documents. Proceedings of the 49th Annual Southeast Regional Conference, Mar. 24-26, Kennesaw, GA, USA., pp: 97-102. DOI: 10.1145/2016039.2016070

Shahriar, S. and J. Liu, 2009. On defining functional dependency for XML. Proceedings of the International Conference on Semantic Computing, Sept. 14-16, IEEE Xplore Press, Berkeley, CA., pp: 595-600. DOI: 10.1109/ICSC.2009.20

Siau, K., 2011. Theoretical and Practical Advances in Information Systems Development: Emerging Trends and Approaches. 1st Edn., Information Science Pub., Hershey, ISBN: 1609605217, pp: 350.

Xing, G., X. Zhonghang and A. Douglas, 2007. X2R: A system for managing XML documents and key constraints using RDBMS. Proceedings of ACM Southeast Regional Conference, Mar. 23-24, Winston-Salem, USA., pp: 215-220. DOI: 10.1145/1233341.1233380

Yan, L. and Z. Ma, 2011. Advanced Database Query Systems: Techniques, Applications and Technologies. 1st Edn., IGI Global snippet, Hershey, PA, ISBN: 160960475X, pp: 392.