Jurnal Teknologi Maklumat & Multimedia **9**(2010): 31 – 40

Comparative Analysis of XML Functional Dependencies

KAMSURIAH AHMAD & HAMIDAH IBRAHIM

ABSTRAK

The concept of functional dependency plays a fundamental role in relational databases where they are used to specify constraints and to detect redundancies. Similarly, these constraints will play fundamental role in XML as well. Recently XML functional dependencies (XFDs) have been intensively studied than other constraints for XML data. Since there is no standard definition of XFD for XML, many attempts were made to define XFD. However these definitions capture different types of constraints. Therefore, the objective of this paper is to compare the existing definitions of XFD and propose a new definition of XFD which unifies and generalizes the previous ones. We show that the proposed definition can express more constraints and hence can be used to detect more XML data redundancies than those in previous works.

Keywords: comparative analysis, XML functional dependencies, relational databases, semantic constraints.

ABSTRAK

Konsep sandaran fungsian memainkan peranan yang penting di dalam pangkalan data hubungan. Konsep ini digunakan untuk menyatakan kekangan dan juga mengesan pengulangan data. Konsep ini juga sepatutnya memainkan peranan yang sama penting di dalam XML. Kebelakangan ini konsep sandaran fungsian bagi XML (XFD) telah mula diberi perhatian jika dibandingkan dengan konsep kekangan lain dalam XML. Disebabkan masih tiada definisi piawai bagi XFD, banyak usaha telah dilakukan oleh ramai penyelidik untuk mencadangkan definisi mereka. Walaupun kini terdapat pelbagai definisi yang dicadangkan tetapi ia merangkumi kekangan yang berlainan. Oleh itu, tujuan kertas kerja ini adalah untuk membuat perbandingan ke atas pelbagai definisi XFD, dan kemudian mencadangkan satu definisi baru yang bersifat lebih umum. Kertas kerja ini juga membuktikan bahawa definisi baru ini mampu menyatakan kekangan dan dengan itu dapat mengesan pengulangan data dengan lebih baik daripada definisi XFD yang wujud.

Kata kunci: Analisis perbandingan, sandaran fungsian XML. pangkalan data hubungan, kekangan semantik.

BACKGROUND

The notion of functional dependency (FD) plays a central role in specifying constraints and discovering redundancies in relational databases (Abiteboul et al. 1995), and should play a central role in XML as well (Arenas and Libkin 2004). However, it is not immediately obvious how to extend the definitions of redundancies from relations to XML because of the flexible tree structure of XML. Also the concept of functional dependency in relations is not immediately applies to XML. The difference is on the structure: relational model is flat whereas the structure of XML model is in tree form (Hartmann and Link 2003). Now, the theory of functional dependencies in relational database context has matured. If we are to achieve

the same functionality for XML as in relations, it is essential to adapt the study of functional dependency in the context of XML. Such attempts on proposing new definitions of XML functional dependency (XFD) have been made by several groups of researchers (Vincent et al. 2004; Arenas and Libkin 2004; Lee and Topor 2005; Lv and Yan 2006; Fassetti and Fazzinga 2007; Trinh 2008). However, these definitions have different expressive power and capture different types of constraints. In this paper, the existing definitions of XFDs were studied and compared in terms of the constraints they expressed. The definitions of existing XFD differ primarily on two aspects: the definition of path expression, and the equality testing between nodes in the tree. These two aspects are important and can determine the expressive power of the XFD definition in terms of its ability to specify the existence of data redundancies in XML, and able to express the semantics that functional dependency hold in a sub-tree rather than in the whole tree. It is believed that if the definition these two aspects is more general then it will increased the expressive powers of XFDs and more data redundancies in the XML data can be captured. In this paper, an analysis of existing XFDs is conducted and a table of comparison in terms of the two aspects is presented. A new definition of XFD which generalized the existing XFD is proposed and proved that this definition can be used to express some natural constraints that cannot be captured by existing XFDs, and hence can detect more redundancies in XML data.

MOTIVATING EXAMPLES

Unlike flat representation of relational model, the model for an XML is like a tree form representing the underlying DOM (Document Object Model) structure and are assumed to be unordered trees, i.e. the position of an element does not matter, which is similar approach with other studies that concerning XFDs (Arenas and Libkin 2004; Buneman et al. 2001; Buneman et al. 2001a). This is acceptable because XFDs are not influenced by document order. To illustrate the examples, let's look at Figure 1.

Figure 1 shows an XML documents about a faculty. The faculty offers many courses, information about students who took particular course and a list of students in the faculty. Understanding the semantic constraints of this document, the following constraints can be stated:

C_1: In the whole document, student number uniquely determines a student name.
C_2: In the whole document, course number uniquely identifies a course and no courses share the same course number.
C_3: For every course, a grade will be given to a student for their assessments.

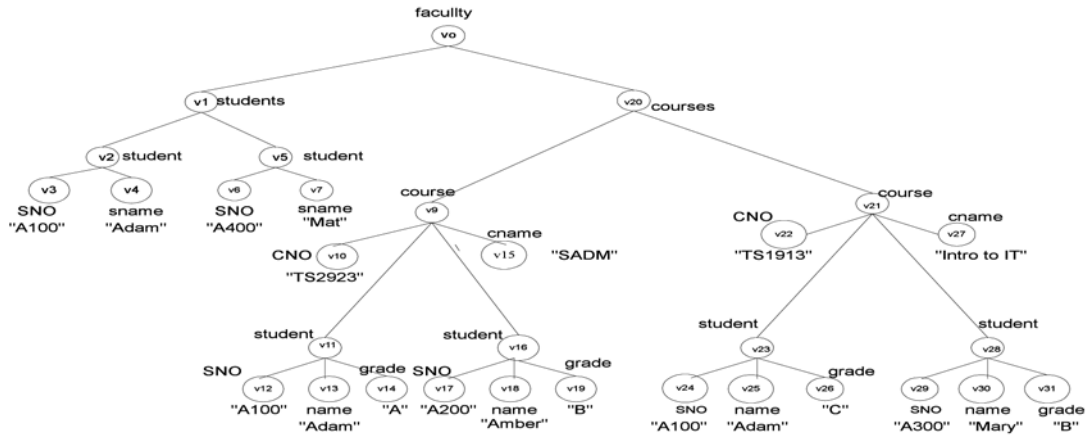

FIGURE 1. An XML document about Faculty

As in Figure 1, *student name* "Adam" holds the same *student number* ("A100") globally in the whole tree. Note that *student* nodes appear under both *students* and *course* nodes. The *student* node is an example of shared element. And for every *course*, there is a list of *students* taking that particular *course,* where every *student* gets his or her *grad*e after sitting an exam. For example "Adam" gets an "A" for "Networking" *course* and "C for "Intro to IT" *course*, which means the student taking particular *course* determines a *grade* locally. This is a clear instance of a need to express the constraints not only globally in the whole tree but also locally in the sub-tree. The *grade* node is an example of local element. Stated also in Figure 1, information about "Adam" is repeated three times under v_2, v_{11} and v_{23} nodes. This is an example of data duplication or data redundancy, which allowed by the data model. A clear method is needed to express the constraints listed above that may exist in the XML documents. The challenges are how (i) to express the constraints that the same elements (shared element) may appear under different parent nodes and (ii) to express the constraints not only globally in the whole tree but also locally in the sub-tree. In this study the concept of functional dependencies is used to express these constraints. Ironically the existing definition of XFDs cannot express the constraints in the presence of shared and local elements. Therefore, a more general definition of XFD is needed.

TERMINOLOGY AND NOTATIONS

First, we should emphasize that semantic constraints are not a part of XML specifications (Clark & Derose 1999). They can be regarded as the extension of XML schema to make XML documents more significant. However, the theory of functional dependencies for relational databases cannot be directly applied in XML documents, as there are significant differences in their structures. As XML model is nested and structured, there is a need to consider defining functional dependency in XML as not too strict to adopt its hierarchical nature. The notion of functional dependencies is based on an unordered tree model of XML data, as illustrated in Figure 1. Although the model is quite simple, two things are needed prior to define functional dependencies. The first is to give a precise definition of value and node equality for XML functional dependencies. The second is to describe the path language that will be used to locate sets of nodes in an XML document (Clark & Derose 1999).

VALUE AND NODE EQUALITY

In relational model, no duplicate tuples are allowed, and so value equality is sufficient in FDs (Arbiteboul et al. 1995). However, in XML document there may exist two different sub-trees that are isomorphic and have exactly the same values. As in Figure 1 sub-tree rooted at node v_2, v_{11} and v_{23} have the same values wrt to C_1. Therefore, if XFDs are to be used for keys or normalizing database to reduce data redundancy, they must be able to detect equality between nodes in the document thus relations produced are redundancy reducing and anomaly problems can be avoided. The definition of value and node is defined below which is adapted from Arenas and Libkin (2004).

Definition 1: Value and Node Equality

Two nodes n_1 and n_2 in XML tree T are said to be value equal, denoted $n_1 =_v n_2$,

- if n_1 and n_2 are of the same label in the trees, an if n_1 and n_2 are both attribute nodes or simple element nodes, then the two nodes have the same value,
- if n_1 and n_2 are both complex elements, and 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.

PATH EXPRESSION

Since XML is hierarchical structured, the definition of XFD must be based on the path expression. Therefore a class of regular expression is introduced, and defines XFDs in terms of this path language. In a uniform syntax it is capable of expressing both global and local XFD, which are important to the hierarchically structured data. In order to define and infer XFD, the following basic notations are needed.

Definition 2. Simple Path.
Given a DTD (Arenas and Libkin, 2004) $D = (E_1, E_2, A, M, N, r)$, a simple path is defined as $l_1/l_2/../l_n$, where $l_i \in E_1$ (i = 1, .., n-1) and $l_n \in E_1 \cup E_2 \cup A$ is called as a sequence of path. $l_1 \in M(r)$, $l_i \in M(l_{i-1})$ for $i = [2, n-1]$ and $l_n \in M(l_{n-1}) \cup N(l_{n-1})$.

Definition 3. Complex Path.
A path is called a complex path if it is in the form of $l_1/l_2/../l_n$, where $l_i \in E_1 \cup \{//\}$ (i = 1, .., n-1) and $l_n \in E_1 \cup E_2 \cup A$. Symbol "//" represent the Kleene closure of the wildcard and can match any label l.

Definition 4. Path Instance.
A path is called a path instance if v_o *be a node in XML tree T, P* a path in DTD D denoted as $l_1/ .. /l_n$, and there is a sequence of nodes $v_0/v_1/.../v_n$ in T such that for $i = 1..n$

- if l_i is in $E_1 \cup E_2 \cup A$, then v_i is a child of v_{i-1} and the label of v_i matches l_i
- if l_i is "//", p represents any path instance on the subtree determined by v_0. If a path instance $v_1/../v_{i-1}/p/v_{i+1}/../v_n$ exists, where $v_1/../v_{i-1}$ matches $l_1/ .. /l_{i-1}$ and $v_{i+1}/../v_n$ matches $l_{i-1}/ .. /l_n$. Then $v_1/.../v_n$ is called a path instance of P wrt v_0.

The sequence of nodes describe above is called an instance of path p wrt v_0. Generally there may be many instance and refer to the set as $\{ v_n \mid v_n$ is the last node of an instance of P wrt $v_0\}$ as the target set of p wrt v_0 , denoted $v_0[p]$. To define XFD, the target set on the left and the right path is defined as determine factor.

COMPARISON ON EXISTING DEFINITIONS OF XFDS

The interaction of previous definition of XFDs by distinguishing the similarities and the differences with respect to these constraints are analyzed and discussed. However, only six of them are selected and analyzed in this study. The selections are based on the published work at the distinguished conference or journal and also based on its popularity where their works are being used as a reference by others.

XFDI: University of Toronto ()

The author introduced the notion of a "tree tuple" in their definition of XFD, where a tree tuple is a finite partial evaluation of the simple paths in the underlying DTD, which extend the total unnesting operator in nested relations. The definition of paths in the DTDs is of the form r.p (r is the root and p is the path) where it can end up with the text label S (S is a string type). The symbol "@" is used for the attribute. XFD was defined on a set of paths on the LHS and a single path on the RHS. The XFDs is said to be satisfied if whenever two tree tuples agree on the paths on the LHS of an XFD, they must also agree on the path on the RHS of the XFD. As an example the XML tree may satisfy the following XFD,

Faculty.students.student.SNO -> Faculty.students.student.SNO.S, but not
Faculty.students.student.SNO -> Faculty.students.student.SNO

The reason for the unsatisfactory condition is because the comparison of value equality is done with the text string S and not with the simple element node. The distinguished features in this definition are:

i. No explicit distinction between simple and complex elements.
ii. The path is based on a sequence of absolute simple paths; therefore to express the constraints in the presence of shared element is not possible. The reason for this inability is the comparison of value equality of complex nodes is impossible.
iii. Don't have the ability to define scope within the sub-document; therefore to express the constraints in the presence of local elements is not possible.

As a result of this definition, it cannot compare the nodes v_2 and v_3, v_{11} and v_{16}, and v_{12} and v_{14}. In other words this definition cannot express the constraints as in C_1 and C_3.

XFD2: University of Australia (Vincent et al. 2004)

The authors define XFDs (Vincent et al., 2004), called strong XFDs does not assume any type specification. A major distinguishing feature of the approach is the concept of strong satisfaction. That is, an XML data tree strongly satisfies an XFD if both LHS and RHS path of XFD is not null. The features used in this definition are:

- Only simple path expression is used.
- No explicit distinction between simple and complex elements
- Does not allow XFD to scope within the sub-document.

This definition is improved in terms of defining on local constraints. Since no complex path expression allowed in the definition therefore redundancy cause by shared element cannot be discovered. Hence this definition cannot compare the nodes v_{11} and v_{16}, and v_{12} and v_{14}. In other words this definition cannot express the constraints as in C_3.

The features used in this definition are:

i. Only simple path expression is used.
ii. No explicit distinction between simple and complex elements.
iii. Does not allow XFD to scope within the sub-document.

XFD3: University of Pennsylvania (Chen et al. 2003)

The author proposed the syntax of XFDs is associated with variable bindings. The definition of XFDs is in the following form:

$\$v_1$ in P_1 {, $\$v_2$ in $\$v_1$ / P_2(optional)}
$\$v_{f(1)}/Q_1$, … , $\$v_{f(n)}/Q_n$ -> $\$v_{f(n+1)}/Q_{n+1}$, … , $\$v_{f(n+m)}/Q_{n+m}$

where $f(k) \in \{1, 2\}$, $1 \le k \le n+m$. P_1, P_2 and Q_i are expressions in the path language and v_1 , v_2 are called valid variable bindings. As an example, according to the definition of XFD, the semantic constraints for XML document about publication as in XML Figure 1 are expressed as follows:

P ∈ faculty.students, $x in P, $y in x/student
$y.SNO/value() -> $y.name/value()

where the path expression bound to a valid variable binding and the path expressions X -> Y are simple paths. There are several limitations to this definition of XFDs:

i. Since the variable bindings are bound to the sequence of path expression starting from the root, hence by the definition of XFDs, they cannot express the constraints: //student/SNO -> name.
ii. The XFD definitionWith this definition, the cannot detect redundancy cause by shared-element, where the student nodes appear under both path faculty.students and faculty.courses. Structurally the XFDs can express this type of constraints because they allows complex path "//" in the path expression, which can go any path in the tree, but by the definition of XFD that required sequence of path, therefore semantically the constraint in the presence of shared-element cannot be expressed.
iii. No appearance of scoping within the sub-document to reflect the hierarchical structure of XML. This can be seen by the flat presentation of XFDs.

Hence this definition cannot compare the nodes v_{11} and v_{16}, and v_{12} and v_{14}. In other words this definition cannot express the constraints as in C_3.

XFD4: University of Queensland (Wang and Topor 2005)

The author introduces more expressive power of XFDs to detect XML data redundancies. Path expression is fully used, where upward and downward path expressions are used in the XFD. Besides, three types of agreement (set, node and intersect) are used to compare value equality of nodes. As a result, more XML data redundancies can be discovered. However there is no concept of local and global XFD is used; therefore this definition cannot specify the semantics in the presence of local element. In other words it cannot compare the nodes v_{11} and v_{16}, and v_{12} and v_{14}, hence this definition cannot express the constraints as in $C_{3.}$

XFD5: Xinjiang Xinjiaang University (Lv and Yan 2006)

The XFD are based on a sequence of simple path, and allow the constraints to scope within the sub-document. The definition of XFD supports the concepts of local and global XFD. However, the simple and complex elements do not distinguished in the XFD definition. Since the path expression used The definition of XFD supports the concept of local and global not support complex path, therefore, dependency constraints in the presence of shared-element cannot be specified.

XFD. However, the simple and complex elements are not distinguished in the XFD definition. Since the path expression used did not support complex path, therefore, dependency constraints in the presence of shared element cannot be specified. Hence it cannot compare the node v_2 and v_3, meaning this definition cannot express the constraints as in C_1.

XFD6: Massey University (Hartman and Trinh, 2005)Trinh 2008)

The definition of XFD is based on sub--trees and the equality testing on the nodes and values of the sub-tree is based on the concept of homomorphism, root-subtrees and isomorphism, where the concept is quite different from the previous ones. The equality testing is based on the structure of the sub-trees. Two sub-trees are said to satisfy homomorphism if they have the same name, the same root, and the same structure. Two sub-trees (T_1, T_2) are said to satisfy isomorphism (a copy) if they are bijectives, that is $T_1 \equiv T_2$. However, no notion of complex path been defined. Hence, this definition of XFD cannot satisfy constraints in the presence of shared element or scopes within the document structure. Meaning this definition cannot compare the nodes v_2 and v_3, hence it also cannot express the constraints as in C_1.

As a comparison on the existing definition of XFDs, Table 1 is drawn to compare in terms of their features used for the definition of path expression and the equality testing.

TABLE 1. Table of Comparison

	XFD1	XFD2	XFD3	XFD4	XFD5	XFD6
1. Path Expression Define XFD on complex path	N	N	Y	Y	N	N
2. Equality Testing Allows XFD to scope within the sub-document	N	Y	Y	N	Y	N
s	Y	N	Y	N	Y	N
	N	N	N	Y	N	Y

Legend: The symbol “N” means the feature is not in the definition of XFD and symbol “Y” is otherwise. N means the definition of XFD does not contain the features and Y means the opposite.

As shown in Table 1, none of the existing definition of XFD consistss of of all the four features listed above. If these features presence in the definition of XFD then the advantages are as follows:

i. The ability to define XFD on the complex path, will allow XFD to specify constraints in the presence of shared elements.
ii. Distinguishing between simple and complex element in the XML tree will reduce the equality testing process.
iii. The ability to scope within the sub-document will allow XFD to specify local and global constraints to adopt the hierarchical structure of XML.
iv. If testing can be done on internal nodes then XFD can be used to express DTD structures.

In order tIn order to specify semantic redundancies that exists in the hierarchical structure of XML document and redundancies that cause by shared and local elements, all these features must adaapt in the definition of XFD. Therefore a new definition is proposed in the next section.

A NEW DEFINITION OF XFD

Based on the limitations of existing XFDs that are not able to express constraints in the presence of shared and local element, therefore a new definition is proposed as follows which will generalize the existing ones.

Definition 5. XFDs constraint language

Given DTD *D*, functional dependency for XML (XFD) is φ over *D* has the form P: Q: $X_1, \ldots, X_n \rightarrow Y_1, \ldots, Y_m$ where

P ∈ paths(*D*) is downward context path starting from the root, which identifies the scope of φ over *D*. If P ≠ r and P ≠ ε (where ε means empty path), then φ is called a local XFD, which means that the scope of φ is the sub-tree rooted at P and last(P) ∈ E_1. Otherwise φ is called a global XFD, which means the scope of φ is the whole *D* or at the root and simplified as Q: $X_1, \ldots, X_n \rightarrow Y_1, \ldots, Y_m$.
Q is called downward target path, where Q ∈ paths(*D*) and P ⊆ Q.
$X_1, \ldots, X_n$ is the left path of φ (Left Hand Side, LHS) and $Y_1, \ldots, Y_m$ is the right path of φ (Right Hand Side, RHS), which is a non-empty subsets of paths(*D*) rooted at [[Q]].

Figure 2 illustrate the way that the XFD been defined in the XML tree and specifies the following:

i. The context path P, starting from the root of an XML tree T, identifies a set of nodes in [[P]];
ii. For each $v_1 \in [[P]]$, φ defines a global XFD on the subtree rooted at v_1; specifically,
iii. The target path Q identifies a set of nodes $v_1[[Q]]$ in the subtree, referred as the target set,
iv. The node in X_i path (i = 1 … n) uniquely identify the node in the Y_j path (j = 1 … m) in the target set. That is, for each $v_2 \in v_1[[Q]]$ the values of the nodes reached by following X_i path from v_2 uniquely identify the values of nodes reached by following Y_j path from v_2.

Drawing an analogy with FD in relational databases, the target set Q corresponds to a relation name in the relational model, and the X_i and Y_j path corresponds to the attributes of FD in the relation. There is no relational counterpart to the context path P since relations are flat (not hierarchical). Given the basic notation of XFD, next is the definition of what it means for an XML document to satisfy a functional dependency constraint.

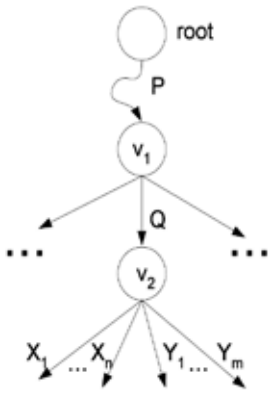

FIGURE 2. Illustration of XFD $\varphi = P: Q : X_1, \ldots, X_n \rightarrow Y_1, \ldots, Y_m$

Definition 6. XFD satisfaction

An XML tree *T* conforming to DTD *D*, $T \models D$, is said to satisfy an XFDs φ (P: Q: $X_1, \ldots, X_n \rightarrow Y_1, \ldots, Y_m$) denote $T \models \varphi$, if and only if for any nodes $v \in [[P]]$ (let v = root if P = ε) and for any two nodes $v_1, v_2 \in v[[Q]]$, if they agree on X_i (i = 1 … n) i.e. for non-empty $v_1[[X_i]]$ and $v_2[[X_i]$ such that $v_1[[X_i]] =_v v_2[[X_i]]$, then they must agree on Yj (j = 1 … m) i.e. for non-empty $v_1[[Y_j]]$ and $v_2[[Y_j]]$ such that $v_1[[Y_j]] =_v v_2[[Y_j]]$. That is, $\forall v \in [[P]]$, $\forall v_1, v_2 \in v[[Q]]$, $(v_1[[X_i]] =_v v_2[[X_i]]) \rightarrow (v_1[[Y_j]] =_v v_2[[Y_j]])$.

Observe that when P=ε, i.e., φ is a local XFDs, where the set [[P]] consists of a unique node, namely, the root of the tree. In this case $T \models \varphi$ if and only if $\forall v_1, v_2 \in [[Q]]$, $(v_1[[X_i]] =_v v_2[[X_i]]) \rightarrow (v_1[[Y_j]] =_{v2} [[Y_j]])$. The definition claimed that for any two instances of sub-trees identified by the XFD header, P, if all LHS entities agree on the values, then they must also agree on the value of the RHS entities if it exists and the LHS entities must exist. If a LHS entity (or RHS entity) is a simple element type (E_2), its value is equivalent to that of the text child (PCDATA) of the element, and to the value of the attribute otherwise.

In relational databases, a table is in BCNF with respect to a set of functional dependencies if the left hand side (LHS) of every functional dependency is a super key. The proposed definition of XFDs allows one to define keys in XML. Like in relational databases, a key is a special case of FDs and defined as follows.

Definition 7. Keys for XML

If there is an XFDs of the form Q: X -> ε over DTD D, then we call X is a key of Q where X ∈ paths(D) and path(X) ∈ R(A) ∪ P(E_2).

Intuitively, X is a key of Q means that, for two nodes n_1 and n_2 if they are value-equal on X, then they must be the same node. Formally, it says for every XML tree *T* conforming to *D*, (*T* |= *D*) for any two nodes n_1 and n_2∈ n[[Q]] (of *T*), if n_1[[X]] =$_v$ n_2[[X]] then n_1 =$_n$ n_2. Note that C_2 express the semantic of keys. Element *cno* is a key of *course*, which ensures entity integrity in document, and is defined as absolute key (Buneman et al. 2001; Buneman et al. 2001a). C_3 is an example of local XFD, since the value of *grade* is determined locally on *student* for particular *course* they took.

As stated in the proposed definition of XFD, the path expression and the equality testing are more general. With this capability, it improved the existing definition of XFDs and is able to express all the constraints as in Example 1. According to the proposed definition the constraint can be represented as the following XFDs:

C_1: //student:sno -> name

C2: //course: cno -> ε

C3: //course: /student: grade

This definition of XFD can be used to detect more data redundancies and express more constraints especially in the presence of shared and local elements.

CONCLUSION

Since there is no standard definition of XML functional dependencies yet, a lot of efforts have been done in this area. This paper studies the existing definition that exists in the literature and makes a comparison on the expressive powers of each definition. This study discovers that the definitions of existing XFD differ primarily on two aspects: how the path expression is being defined, and the equality testing between nodes in the tree. The existing existing definition of functional dependency that used general path expression and the definition of value and node equality, cannot discover the redundancy of data that cause by shared and local-element. Hence the existing approaches cannot detect the redundancies that may occur in the XML document. Furthermore they do not remove the unnecessary element that caused by set-elements. Even though they allow complex path "//" in the path definition but they cannot detect the constraint in the presence of shared elements because it was based on sequence of path rather than a set of path expression. As the result, redundant relational schema will be produced. To overcome the limitation of existing works, this study proposed a new definition that generalized the existing ones. The study done in this paper is based on the presence of schema. However the XML documents may come without a schema, the other study that might be interesting to explore is how to define functional dependencies without the presence of schema for XML. This exploration is the intention of our future works.

REFERENCES

Abiteboul, S. Hull, S. and Vianu, V.1995. *Foundations of Database*, Addison-Wesley, Reading, MA.

Arenas, M. and Libkin, L. 2004. A Normal Form for XML Documents. *ACM Transaction on Database Systems*. 195-232.

Buneman, P. Davidson, S. Fan, W. Hara, C. and Tan, T. 2001. Keys for XML. In *Proceedings of World Wide Web Conference*.

Buneman, P. Davidson, S. Fan, W. Hara, C. and Tan, T. 2001a. Reasoning about Keys for XML. In*: Ghelli G, GrahneG, eds. Proceedings of the 8th Int'l Workshop*. Springer-Verlag. 133-148.

Chen, Y., Davidson, S., Hara C. and Zheng Y. 2003. RRXS: Redundancy Reducing XML Storage in Relations. In *Proceedings of 29th. International Conference on Very Large Data Base:* 189-200.

Clark, J. and Derose, S. *XML: Path Language (XPath)*. W3C Working Draft, Nov.1999. *http://www.w3.org/TR/xpath*.

Fassetti, F. and Fazzinga, B. 2007. Approximate Functional Dependencies for XML Data. *Local Proceedings of ADBIS 2007,* pp. 86-95.

Hartman, S. and Link, S. 2003. More Functional Dependencies for XML. In *Proceedings of ADBI*. LNCS 2798. 355-369.

Lee, M.L. Ling, T.W. and Low, W.L. Designing Functional Dependencies for XML. *EDBT Lecture Notes in Computer Science*, 2287. 124-141.

Lv, T. and Yan, P. 2006. Mapping DTDs to Relational Schemas with Semantic Constraints. *Journal of Information and Software Technology* Vol. 48(4): 245-252.

Trinh, T. 2008. Using Transversals for Discovering XML Functional Dependencies. *Proceedings of FoIKS, LNCS 4932. pp199-218.Springer Verlag*.

Vincent, M.W. Liu, J., and Liu, C. 2004. Strong Functional Dependencies And Their Application to Normal Forms in XML. *ACM Transactions on Database Systems*, 29(3): 445-462.

Wang, J. and Topor, T. 2005. Removing XML Data Redundancies Using Functional and Equality-Generating Dependencies. *The 16th Australasian Database Conference*.

Kamsuriah Ahmad
Faculty of Information Science and Technology
Universiti Kebangsaan Malaysia
kam@ftsm.ukm.my

Hamidah Ibrahim
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia
hamidah@fsktm.upm.edu.my