Mapping Relational Schemas to XML DTDs with Constraints

Teng Lv\textsuperscript{1,2} and Ping Yan\textsuperscript{3}

\textsuperscript{1}College of Mathematics \\& System Science, Xinjiang University, Urumqi, 830046, China
\textsuperscript{2}Teaching \\& Research Section of Computer, Artillery Academy, Hefei, 230031, China
\textsuperscript{3}College of Mathematics \\& System Science, Xinjiang University, Urumqi, 830046, China
Lt0410@163.com; yanping@xju.edu.cn

Abstract

XML is becoming a prevalent format and de facto standard for data exchange in many applications. While traditionally, lots of data are stored and managed in relational databases. There is an urgent need to research some efficient methods to convert these data stored in relational databases to XML format when integrating and exchanging these data in XML format. The semantics of XML schemas are crucial to design, query, and store XML documents and functional dependencies are very important representations of semantic information of XML schemas. As DTDs are one of the most frequently used schemas for XML documents in these days, we will use DTDs as schemas of XML documents here. This paper studies the problem of schema conversion from relational schemas to XML DTDs. As functional dependencies play an important role in the schema conversion process, the concept of functional dependency for XML DTDs is used to preserve the semantics implied by functional dependencies and keys of relational schemas. A conversion method is proposed to convert relational schemas to XML DTDs in the presence of functional dependencies, keys and foreign keys. The methods presented here can preserve the semantics implied by functional dependencies, keys and foreign keys of relational schemas and can convert multiple relational tables to XML DTDs at the same time.

1. Introduction

XML (eXtensible Markup Language) \cite{1} has become one of the primary standards for data exchange and representation on the World Wide Web and is widely used in many fields. Historically, lots of data and information are still stored in and managed by relational database management systems, such as Oracle, Sybase, SQL Server, etc. So it is necessary and urgent to develop some efficient methods to convert relational data to XML data in order to take advantage of all the merits of XML. As DTDs (Document Type Definitions) \cite{2} are still the most frequently used schemas for XML documents in these days \cite{3}, we will use DTDs as schemas of XML documents. It is widely accepted that schema semantics plays a very important role in the schema conversion, and functional dependencies, keys, and foreign keys of relational schemas are very important representations of semantic information. So it is significant that the conversion method from relational schemas to XML DTDs must consider the semantics implied by functional dependencies, keys, and foreign keys, and the obtained XML DTDs can represent such semantics in some way.

For the problem of converting relational schemas to XML schemas, an intuitive method called FT (Flat Translation)\cite{4} converts tables and attributes of relational schemas to elements and attributes of XML DTDs, respectively. Another two methods called NeT(Nesting-based Translation Algorithm) and CoT(Constraints-based Translation Algorithm)\cite{5,6} can treats more complicated relational schemas than FT, but all of them omit the semantics implied by functional dependencies of relational schemas. Ref.\cite{7} proposes an automatic generator for translating ER models to XML DTDs, which mainly considers cardinality constraints, composite attribute, multi-valued attributes, weak entity, and strong entity. But all of them omit the semantics implied by functional dependencies of relational schemas, which are very important representation of semantic information. Ref.\cite{8} converts relational schemas to XML schemas through denormalization by joining the normalized relations into tables which are mapped into DOMs. Then DOMs are integrated into a user specified XML document trees, which is finally mapped into an XML schema in the form of DTD. Ref.\cite{9} translates relational schemas into an Extended Entity Relationship (EER) Model, which is then mapped to an XML Schema Definition Language (XSD) Graph as an XML conceptual schema. The XSD Graph will be finally mapped into the XSD as an XML logical schema. We do not apply DOM,
ER model, or EER model in our work, so these approaches are orthogonal to our work presented in this paper. Furthermore, we mainly focus on functional dependencies, keys, and foreign keys in relational schemas when converting relational schemas to XML DTDs. The final converted XML DTDs can conserve these semantic constraints in XML DTDs’ functional dependencies, which are major contributions of our work. Another related work is the problem of converting relational schemas to XML DTDs with functional dependencies, keys, and foreign keys. The method presented in the paper has a significant improvement over Net and CoT in the aspect of preserving the semantics implied by functional dependencies of relational schemas.

Organization. The rest of the paper is organized as follows. Section 2 gives some notations and the definition of functional dependencies for XML as a preliminary work. Section 3 presents the conversion methods to convert relational schemas to XML DTDs with functional dependencies, keys, and foreign keys. A case study is given in Section 4 as an application of the conversion method. Section 5 concludes the paper and points out the direction of future work.

2. Notations

We give the definition of relation schema based on the counterpart of Ref.[4].

Definition 1. A relational schema is defined to be =(T, C, M, N, Δ), where (1) T is a finite set of table names; (2) C is a finite set of column names; (3) M is a mapping from a table name to a finite set of column names where each column name is a finite set of column names; (4) N is a mapping from a column name to column type definition = (baseType, unique, null, domain, default), where baseType is atomic base type defined by DBMS, unique is true or false determined by the value of c is unique or not, null is true or false determined by the value of c can be null or not, domain is the domain type of c if known, default is the default value of c or ε if not known; (4) Δ is a finite set of functional dependencies, keys, and foreign keys over T.

We do not give definitions of FDs, keys, and foreign keys in a relational schema as they are trivial and obvious. We give definitions of DTD, path, and XML tree [16].

Definition 2. A DTD (Document Type Definition) is defined to be D=(E, A, P, R, r), where (1) E is a finite set of element types; (2) A is a finite set of attributes; (3) P is a mapping from E to element type definitions. For each ∈ E, P(τ) is a regular expression α defined as α := S | | α | α | α | α | , where S denotes string types, ε is the empty sequence, τ ∈ E, “,” “,” “,” and “ * ” denote union (or choice), concatenation and Kleene closure, respectively; (4) R is a mapping from E to the power set (A); (5) r ∈ E is called the element type of the root.

A path p in D=(E, A, P, R, r) is defined to be p=ω0,...,ωn, where (1) ω0=r; (2) ωi∈P(ωi−1), i∈[2,...,n−1]; (3) ωn∈P(ωn−1) if ωn∈E and P(ωn)≠Φ, or ωn=S if ωn∈E and P(ωn)=Φ, or ωn∈R(ωn−1) if ωn∈A. Let paths(D)=p | p is a path in D}. last(p) denotes the last symbol of path p, and p−last(p) denotes the remaining path excluding last(p) of p. For two paths p,q∈paths(D), p q (or q q p) denotes path p is a prefix of path q or they are the same path; p q (or q q p) denotes path p is a prefix of path q but they are not the same path.

Example 1. Consider the following DTD D1, which describes the information of course, student, and teacher:

```xml
<!ELEMENT courses (course*)>
<!ELEMENT course (title, takenby)>
<!ATTLIST course cno CDATA #REQUIRED>
<!ELEMENT title (#PCDATA)>
<!ELEMENT takenby (student*)>
<!ELEMENT student (sname, teacher)>
<!ATTLIST student sno CDATA #REQUIRED>
<!ELEMENT surname (#PCDATA)>
<!ELEMENT teacher (tname)>
<!ATTLIST teacher tno CDATA #REQUIRED>
<!ELEMENT tname (#PCDATA)>
```

According to Definition 2, D1 is defined as D1=(E1, A1, P1, R1, r1), where

E1={courses, course, title, takenby, student, surname, teacher, tname}
A1={cno, sno, tno}.
P1(courses)=course
P1(course)=title, takenby
P1(title)=P1(surname)=P1(tname)=S
P1(takenby)=student
P1(student)=surname, teacher
P1(teacher)=tname
R1(courses)={cno}
R1(student)={sno}
R1(teacher)={tno}
R1(courses)=R1(title)=R1(takenby)=R1(surname)=R1(tname)=Φ
r1=courses,

Paths courses, courses.course, courses.course@cno, courses.course.title, courses.course.title.s, and courses.course.takenby are some paths in paths(D1).

Definition 3. Let D=(E, A, P, R, r). An XML tree T conforming to D (denoted by T=D) is defined to be T=(V, lab, ele, att, val, root), where (1) V is a finite set of
vertexes; (2) \( lab \) is a mapping from \( V \) to \( E \cup A \); (3) \( ele \) is a partial function from \( V \) to \( V' \) such that for any \( v \in V \), \( ele(v) = [v_1, \ldots, v_n] \) if \( lab(v_1), \ldots, lab(v_n) \) is defined in \( P(lab(v)) \); (4) \( att \) is a partial function from \( V \) to \( A \) such that for any \( v \in V \), \( att(v) = R(lab(v)) \) if \( lab(v) \in E \) and \( R(lab(v)) \) is defined in \( D \); (5) \( val \) is a partial function from \( V \) to \( S \) such that for any \( v \in V \), \( val(v) \) is defined if \( P(lab(v)) = S \) or \( lab(v) \in A \); (6) \( lab(root) = r \) is called the root of \( T \).

Given a DTD \( D \) and an XML tree \( T \models D \) , a path \( p \) in \( T \) is defined to be \( p = v_1, \ldots, v_m \) where (1) \( v_1 = root \); (2) \( v_i \in ele(v_{i-1}) \), \( i \in [2, \ldots, n] \); (3) \( v_i \in att(v_{i-1}) \) if \( lab(v_{i-1}) \in E \), or \( v_i \in att(v_{i-1}) \) if \( lab(v_{i-1}) \in A \), or \( v_i = S \) if \( P(lab(v_{i-1})) = S \). Let \( paths(T) = \{ p \mid p \text{ is a path in } T \} \).

**Example 2.** Fig. 1 is an XML tree \( T \) conforming to \( D_1 \) in Example 1 (Note: Each node is marked by its lab mapping value for clarity). According to Definition 2, \( T \) is defined as \( T = (V, lab, ele, att, val, root_1) \), where \( V \) is the finite set of nodes of Fig. 1, \( lab(root) = \text{courses, ele(course)} = \text{title, takenby} \) and for the leftmost title node, \( val(title) = \text{"db"} \). \( courses \), \( courses.course \), \( courses.course.title \), \( courses.course.title.S \) , and \( courses.course.title.S.\text{title} \) if \( S \text{title} \) is some paths in \( T_1 \), and \( courses.course\text{.course} = \text{cno, courses.course.title} \), \( courses\text{.course}\text{.title}.S = \text{courses} \), \( courses\text{.course}\text{.title}.S\text{.\text{course}\text{.title}} \) is the three course nodes in Fig. 1. For the first course node, \( p(course) = \text{courses.course} \), which is the leftmost path node.

If \( n \) and \( p \) are a node and a path in XML tree \( T \), respectively, then the node set started from node \( n \) via the path \( p \) is \( n[p] \). Specifically, the node set started from the root node via the path \( p \) is \( root([p]) \), which can be simplified as \( [p] \).

We give the definition of value equality of two nodes. Intuitively, two nodes are value equal iff the two sub-trees rooted on the two nodes are identical.

**Definition 4.** Two nodes \( x \) and \( y \) are value equal denoted as \( x =_v y \) iff (1) \( lab(x) = lab(y) \); (2) \( val(x) = val(y) \) if \( x, y \in A \) or \( x = y = S \); (3) if \( x, y \in E \), then (a) for any attribute \( a \in att(x) \), there exists \( b \in att(y) \) and satisfies \( a =_v b \), and vice versa; (b) If \( ele(x) = v_1, \ldots, v_k \), then \( ele(y) = w_1, \ldots, w_k \). And for any \( i \in [1, k] \), there exists \( v_j =_v w_i \), and vice versa.

A functional dependency over DTDs is defined as follows[15].

**Definition 5.** Given a DTD \( D \) , a functional dependency (FD) \( f \) over \( D \) has the form \( \{ S_h \subseteq \{S_{x_1}, \ldots, S_{x_n} \} \rightarrow \{S_{y_1}, \ldots, S_{y_m} \} \} \) , where \( 1 \leq h \leq \text{last(S)} \) is an element name, i.e., \( \text{last(S)} \in E \). If \( S_h \neq \emptyset \) and \( S_h \neq r \), then \( f \) is called a relative FD which means the scope of \( f \) is the sub-tree rooted on \( \text{last(S)} \); otherwise, \( f \) is called an absolute FD which means the scope of \( f \) is overall \( D \). (2) \( \{S_{x_1}, \ldots, S_{x_n} \} \) is called the left path of \( f \). For each \( i \in [1, n] \), it is the case that \( S_{x_i} \subseteq \text{paths(D)} \), \( S_{x_i} \supseteq \text{path(S}_{h} \) (\( S_h \) is a sub-path of \( S_{x_i} \), but not necessarily a proper sub-path), \( S_{y_j} \neq \emptyset \) , and \( \text{last(S)} \subseteq E \cup A \cup S \). (3) \( \{S_{y_1}, \ldots, S_{y_m} \} \) is called the right path of \( f \). For each \( j \in [1, m] \), it is the case that \( S_{y_j} \subseteq \text{paths(D)} \), \( S_{y_j} \supseteq \text{path}(S_h \) , \( S_{y_j} \neq \emptyset \) , and \( \text{last(S)} \subseteq E \cup A \cup S \).

For an XML tree \( T = D \) , we call \( T \) satisfies FD \( f \) (denoted as \( T \models f \) iff for any nodes \( H \subseteq \{S_i\} \) let \( H = \text{root} \) if \( S_i \neq \emptyset \) and \( X_i, X_j \in H \{S_i \cap \ldots \cap S_n \} \) in \( T \), if there exist
nodes $X_{i[[S_{i}]]}=X_{i[[S_{j}]]}$, ..., $X_{i[[S_{m}]]}=X_{i[[S_{m}]]}$, and it is the case that for any nodes $Y_{i}, Y_{j} \in H([S_{i}],[S_{j}])$ and $H(p(X_{i}) \cup p(Y_{i}))$, $H(p(X_{j}) \cup p(Y_{j})) \in H([S_{i}],[S_{j}] \cdots [S_{m}])$ such that $Y_{i}[[S_{i}]] = Y_{i}[[S_{j}]], ... , Y_{i}[[S_{m}]] = Y_{i}[[S_{m}]]$.

**Example 3.** In Fig. 1, we have the following FD:

\[ \text{\{\text{courses.course}, [\text{courses.course}.\text{takenby.student}@\text{sno}] \rightarrow [\text{courses.course}.\text{takenby.student}].\text{student}\}} \]

which implies that within the sub-tree rooted on a course node, a student’s number (@sno) can uniquely determines a student node.

For the problems of soundness and completeness of FDs over DTDs defined here, they are similar as those proposed in our previous work [17]. We do not elaborate them here considering the space.

3. Converting Relational Schemas to XML DTDs

Given a relation schema $(T, C, M, N, \Delta)$, it can be converted to a DTD $D=(E, A, P, R, r)$ by the following conversion method based on the CoT method [7].

1. If there are no foreign keys in $\Delta$, convert by method NeT_FD [18]. The method terminates.
2. Construct the IND-Graph [5] $G$ for by foreign keys in $\Delta$ and let $N_{top}$ is the set of top nodes in $G$.
3. Build a root element type $r$.
4. Each top node $n_{i} \in N_{top}$ is converted to an element type $e_{i}$ in $D$, and $P(r) = \{P(r), e_{i}\}$. Suppose FD $f \in \Delta$ over $n_{i}$ has the form $(n_{j}, c_{j}, ..., n_{k}, c_{k}) \rightarrow (n_{j}, c_{j}, ..., n_{k}, c_{k})$, then $\sum \bigcup \{ [r.e_{j}.c_{j}] \rightarrow [r.e_{j}.c_{j}] \}$. Moreover, suppose the key of $n_{i}$ is $(c_{i}, ... , c_{i})$, then $\sum \bigcup \{ [r.e_{i}.c_{i}] \rightarrow [r.e_{i}] \}$. Do Breath-First Search (BFS) to find the next node to be processed. For each such node $n_{j}$(i.e., $n_{j}(X)$ $n(Y)$, where $X$ and $Y$ are attributes of tables $n_{j}$ and $n_{i}$, respectively, and $Y$ is the key of table $n_{j}$):

   (4.1) If $n_{j}$ is not yet a sub-element of some other node, convert $n_{j}$ to an element type $e_{j}$, and $P(e) = \{P(e), e_{j}\}$ if $X$ is unique, otherwise $P(e) = \{P(e), e_{j}\}$. Suppose the key of $n_{j}$ is $(c_{k_{1}}, ..., c_{k_{l}})$, and $X$ is in $(c_{k_{1}}, ..., c_{k_{l}})$, then $\sum \bigcup \{ [r.e_{i}.c_{k_{1}}, ..., r.e_{i}.c_{k_{l}}] \rightarrow [r.e_{i}.c_{k_{1}}, ..., r.e_{i}.c_{k_{l}}] \}$. For $n_{j}$’s FD $f \in \Delta$:

   (a) $\sum \bigcup \{ [r.e_{j}.e_{j}] \rightarrow [r.e_{j}.e_{j}] \} \text{ if } f \text{ has the form } (n_{j}, c_{j}, ..., n_{j}, c_{k}) \rightarrow (n_{j}, c_{j}, ..., n_{j}, c_{k}) \text{ and attribute } X \text{ does not appear in left or right part of } f$;

   (b) $\sum \bigcup \{ [r.e_{j}.e_{j}] \rightarrow [r.e_{j}.e_{j}] \} \text{ if } f \text{ has the form } (n_{j}, c_{j}, ..., n_{j}, c_{k}) \rightarrow (n_{j}, c_{j}, ..., n_{j}, c_{k})$;

   (c) $\sum \bigcup \{ [r.e_{j}.e_{j}] \rightarrow [r.e_{j}.e_{j}] \} \text{ if } f \text{ has the form } (n_{j}, c_{j}, ..., n_{j}, c_{k}) \rightarrow (n_{j}, c_{j}, ..., n_{j}, c_{k}) \text{ if } f \text{ has the form } (n_{j}, c_{j}, ..., n_{j}, c_{k}) \rightarrow (n_{j}, c_{j}, ..., n_{j}, c_{k})$.

(4.2) Otherwise $R(n_{i}) = \{R(n_{i}).ID_\text{_}\_n_{i}\}$, $R(n_{j}) = \{R(n_{j}).Ref_\text{_}\text{_}_n_{j}\}$, where $ID_\text{_}\_n_{i}$ is the ID attribute of $n_{i}$, and $Ref_\text{_}\text{_}_n_{j}$ is the IDREF attribute of $n_{j}$ referencing to $ID_\text{_}\_n_{i}$. \sum $\bigcup \{ [r.e_{j}.e_{j}] \rightarrow [r.e_{j}.e_{j}] \}$. Suppose the parent element of node $n_{j}$ is $e_{j}$, the key of $n_{j}$ is $(c_{j}, ... , c_{j})$, and $X$ is in $(c_{j}, ... , c_{j})$, then $\sum \bigcup \{ [r.e_{j}.e_{j}] \rightarrow [r.e_{j}.e_{j}] \} \text{ if } f \text{ has the form } (n_{j}, c_{j}, ..., n_{j}, c_{k}) \rightarrow (n_{j}, c_{j}, ..., n_{j}, c_{k})\text{ and attribute } X \text{ does not appear in left or right part of } f$;

(4.3) Do NeT for other FDs, keys, and columns of table $n_{i}$ and other columns of table $n_{j}$ excluding $X$, and convert FDs and keys in $\Delta$ to FDs over $D$.

Some explanations: (1) Step 1 deals with the case when there are no foreign keys at all. In such case, we just use NeT_FD method. (2) In step 4, we first construct individual element type for each top node and converts the integrity constraints (FDs, keys, and foreign keys). Then in sub-step 4.1, we deal with each non-top node which is not yet a sub-element of some other element. This kind of non-top nodes is converted to sub-elements of the top node. Then we rewrite the FDs and keys of them to conform to the XML FDs. Similarly, in sub-step 4.2, we deal with each non-top node which is already a sub-element of some other element. We just add ID and IDREF attributes to corresponding elements to capture the semantic of foreign keys. We also rewrite the FDs and keys of them to conform to the XML FDs. And finally, sub-step 4.3 deals with the remaining FDs, keys, and attributes of each table. (3) As all keys, foreign keys, and FDs of a relational schema are translated to their corresponding representations in the final DTD, we can see that the method can definitively preserve those semantics of the original relational schema. (4) As the nodes in IND-Graph are limited, and the number of keys, foreign keys, and FDs of relational schema are also limited, the time complexity of the method is PTIME. (5) The DTD obtained here is better than the one of Refs. [5]
4. A case study

In this section, we give an example to illustrate the application of the conversion method in Section 4. Consider the following three tables:

Student(sno, sname, course, department),
Course(cname),
and
Department(dname),

where a student can choose more than one course and only belongs to one department.

The relational schema for above tables is \( (T, C, M, N, \Delta) \), where

\[ T = \{ \text{Student}, \text{Course}, \text{Department} \}, \]
\[ C = \{ \text{sno, sname, course, department, cname, dname} \}, \]
\[ M(\text{Student}) = \{ \text{sno, sname, course, department} \}, \]
\[ M(\text{Course}) = \{ \text{cname} \}, \]
\[ M(\text{Department}) = \{ \text{dname} \}, \]

and suppose

\[ N(\text{sno}) = (\text{integer, true, false, } e, e), \]
\[ N(\text{sname}) = (\text{string, false, true, } e, e), \]
\[ N(\text{course}) = (\text{string, false, false, } e, e), \]
\[ N(\text{department}) = (\text{string, false, false, } e, e), \]
\[ N(\text{cname}) = (\text{string, true, false, } e, e), \]
\[ N(\text{dname}) = (\text{string, true, false, } e, e), \]

\[ \Delta = \{ \text{sno, course} \} \text{ is the key of Student, } \{ \text{cname} \} \text{ is the key of Course, } \{ \text{dname} \} \text{ is the key of Department, } \]

\[ \text{sno} \rightarrow \text{sname, sno} \rightarrow \text{department, } \]

\[ \text{Student(course)} \subseteq \text{Course(cname), and } \]
\[ \text{Student(department)} \subseteq \text{Department(dname)}. \]

There is foreign keys in \( \Delta. \) First construct IND-Graph (Fig. 2, where top nodes are Course and Department.

The final DTD for is:

\[
\text{<!ELEMENT r(Course*,Department*)>}
\text{<!ELEMENT Course(Student*)>}
\text{<!ATTLIST course cname CDATA #REQUIRED>}
\text{<!ELEMENT Student EMPTY>}
\text{<!ATTLIST Student sno CDATA #REQUIRED>}
\text{<!ATTLIST Student surname CDATA #IMPLIED>}
\text{<!ATTLIST Student Ref_Department IDREF>}
\text{<!ELEMENT Department EMPTY>}
\text{<!ATTLIST Department dname CDATA #REQUIRED>}
\text{<!ATTLIST Department ID_Department ID>}
\]

The set of FDs \( \Sigma \) over \( D \) includes:

\[ \text{FD}_1: \{ \text{r.Course.cname} \rightarrow \{ \text{r.Course} \}, \]
\[ \text{FD}_2: \{ \text{r.Course.Student.sno} \rightarrow \{ \text{r.Course.Student} \}, \]
\[ \text{FD}_3: \{ \text{r.Course.Student.sname} \rightarrow \{ \text{r.Course.Student} \}, \]
\[ \text{FD}_4: \{ \text{r.Department.dname} \rightarrow \{ \text{r.Department} \}, \]
\[ \text{FD}_5: \{ \text{r.Department.ID_Department} \rightarrow \{ \text{r.Department} \}, \]

and

\[ \text{FD}_6: \{ \text{r.Course.Student.sno} \rightarrow \{ \text{r.Course.Student.ID_Department} \}. \]

5. Conclusions and future work

We give a method to convert relational schemas to XML DTDs in the presence of keys, foreign keys, and functional dependencies in the paper, which can preserve the semantics implied by functional dependencies, keys, and foreign keys of relational schemas and can convert multiple tables to XML DTDs. Although the method presented here can preserve the semantics implied by functional dependencies, keys, and foreign keys of relational schemas in the result DTDs, there are more semantics (e.g. multi-valued dependencies \[19\], etc.) that should be preserved. We plan to do this work in our future work.

Acknowledgements

This work is supported by National Natural Science Foundation of China (NSFC) under grant No. 60563001 and Science Research Foundation for Young Teachers of Xinjiang University under grant No.QN040101.

References

translation,” in Proc. of the 4th International Workshop on the
methods between XML and relational models. Knowledge
Transformation for the Semantic Web,” Frontiers in Artificial
Intelligence and Applications, Vol. 95, Amsterdam, IOS Press,
relational schemas to XML schemas using semantic constraint”,
in Proc. of the 2002 ACM CIKM International Conference on
Information and Knowledge Management, ACM Press,
DTDs from conceptual database schemas,” GI Jahrestagung (1)
database into XML documents with DOM,” in Information and
into XML schema definition with data semantic preservation and
XSD graph,” in Information and Software Technology, 47(7), pp.
schema with constraints,” Journal of Computer Science,
relational schemas in the presence of functional dependencies
over DTDs,” in Proc. of the 10th Joint International Computer
querying XML documents: limitations and opportunities,” in
Proc. of the 25th VLDB Conference, Edinburgh, Scotland:
DTDs to relational schemas,” in Proc. of ER workshops 2003,
Spinger, Lecture Notes in Computer Science, Vol. 2814,
preserving XML storage in relations,” Technical Report,
MS-CIS-02-04, University of Pennsylvania, 2002.
[15] T. Lv and P. Yan, “Mapping DTDs to relational schemas
with semantic constraints,” Information and Software
[17] T. Lv and P. Yan, “Functional dependencies of XML and
their inference rules,” Journal of Computer Research and
[18] T. Lv, P. Yan, and Z. Wang, “Converting relational schemas
to XML schemas with functional dependencies and keys,” in
Proc. Of 22nd National Database Conference of China