International Journal of Database Theory and Application
Vol. 3, No. 2, June, 2010

Towards the Preservation of Referential Constraints in XML Data Transformation for Integration*

Md. Sumon Shahriar and Jixue Liu

Data and Web Engineering Lab
School of Computer and Information Science
University of South Australia, Adelaide, SA-5095, Australia
E-mail: shamy022@students.unisa.edu.au, Jixue.Liu@unisa.edu.au

Abstract

We investigate the transformation and the preservation of XML referential constraints in XML data transformation for integration purposes in this paper. In the transformation and the preservation, we consider XML inclusion dependencies and XML foreign keys. We show how XML referential constraints should be transformed and preserved using important transformation operators with sufficient conditions.

1. Introduction

Transformation of schema with its conforming data is an important task in many data intensive activities such as data integration [3, 4] and data warehousing [1]. In recent days, XML [16] is widely used data representation and storage format over the web and hence the task of XML data transformation [2, 15] for integration purposes is getting much importance to the database researchers and developers. In XML data transformation, a source schema with its conforming data is transformed to the target schema. An XML source schema is often defined with referential integrity constraints [5, 6, 7, 8] such as XML inclusion dependency [9, 10] and foreign key [11]. Thus, in transforming schema and its data, referential integrity constraints can also be transformed. There is also a need to see whether constraints are preserved [13, 14] by the transformed data at the target schema.

We illustrate our research problems with following simple motivating examples.

Example 1. Consider the DTD D_1 in Fig.1(a) that describes the enrolled students(sid) of different courses(cid) and the students who has project in the department. Now consider the XML key [12] $\mathbb{k}_1(dept, \{enroll/sid\})$ on D_1. We say $dept$ is the $selector$ and $enroll/sid$ is the $field$ of key. We also say that the key is valid on D_1 because it follows the path of the DTD. The meaning of the key is that under $dept$, all values for the element sid are distinct. Also, consider the XML inclusion dependency (XID) [11] $\Upsilon_1(dept, (\{project/sid\} \subseteq \{enroll/sid\}))$ on D_1 meaning that the students who have project must be enrolled students. We see that both key and inclusion dependency are satisfied by the document T_1 in Fig.1(b). In case of key, the values of the element sid in $dept$ are $(v_3 : sid : 009), (v_5 : sid : 001)$ and $(v_8 : sid : 007)$ are value distinct.

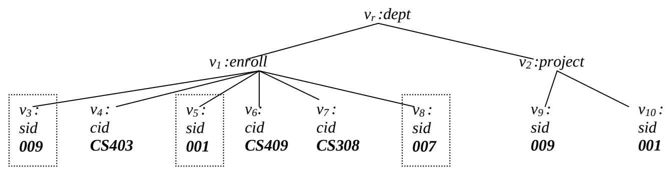
*This research is supported with Australian Research Council(ARC) Discovery(DP) Fund.

For inclusion dependency, for every values of sid in $project$, there is a value for sid in $enroll$. Now consider the XML foreign key (XFK) [11] $\mathcal{F}_1(dept, (\{project/sid\} \subseteq \{enroll/sid\}))$ on D_1. We see that the foreign key is also satisfied because both key and inclusion dependency are satisfied. Note that the DTD D_1 also conforms to the document T_1.

Now we transform the DTD D_1 and its conforming document T_1 to the DTD D_2 in Fig.2(a) and the document T_2 in Fig.2(b). In transformation, we use a new element $stud$ to push down the structure (sid, cid^*). We term this transformation as $expand$. Now we want to know whether the key $\mathbb{k}_1(dept, \{enroll/sid\})$ and XID $\Upsilon_1(dept, (\{project/sid\} \subseteq \{enroll/sid\}))$ need any transformation and thus whether they are valid after transformation. We observe that the paths of both key and XID need transformation as a new element $stud$ is added in the DTD.

$<!ELEMENT\ \ dept(enroll, project) >$
$<!ELEMENT\ \ enroll(sid, cid^*)^+ >$
$<!ELEMENT\ \ project(sid^+) >$

(a) DTD D_1



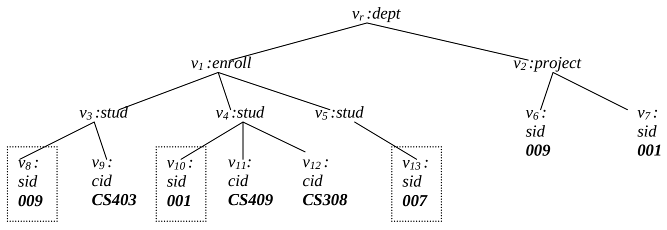
(b) Tree T_1

Figure 1. Source

We transform $\mathbb{k}_1$ to $\mathbb{k}_2(dept, \{enroll/stud/sid\})$ adding new element $stud$ in the middle of $field$. In similar way, we transform Υ_1 to $\Upsilon_2(dept, (\{project/sid\} \subseteq \{enroll/stud/sid\}))$. We see that both $\mathbb{k}_2$ and Υ_2 are valid on the DTD D_2 because they follow the path of D_2. As both key and XID are transformed and valid, we transform the XFK $\mathcal{F}_1(dept, (\{project/sid\} \subseteq \{enroll/sid\}))$ to $\mathcal{F}_2(dept, (\{project/sid\} \subseteq \{enroll/stud/sid\}))$ which is also valid on D_2.

$<!ELEMENT\ \ dept(enroll, project) >$
$<!ELEMENT\ \ enroll(stud^+) >$
$<!ELEMENT\ \ stud(sid, cid^*) >$
$<!ELEMENT\ \ project(sid^+) >$

(a) DTD D_2



(b) Tree T_2

Figure 2. Target

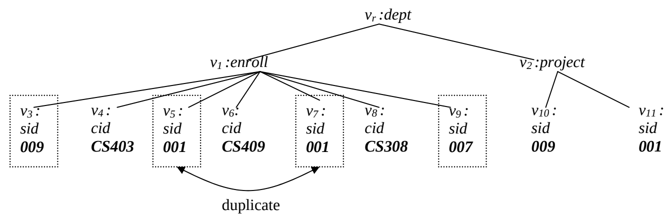
Observation 1.*Transformation of key, XID and XFK needs to be determined when the DTD is transformed.*

Example 2. In the previous example, we showed how to transform key, XID and thus XFK. In this example, we show that though key, XID or XFK are transformed, as key is not preserved so XFK is also not preserved. By key preservation, we mean that if a key is satisfied by the document, after the transformation, the transformed key is also satisfied by the transformed document. Same is with the XID and XFK preservations.

Consider the DTD D_1 in Fig.1(a) and its conforming document T_1 in Fig.1(b). We want to transform D_1 and T_1 to D_3 in Fig.3(a) and T_3 in Fig.3(b). We see that in D_1, the courses are nested with each student. In transformation, we distribute the course id cid with student id sid. We term this transformation as $unnest$. After the transformation, we see that there is no change of paths in the DTD and thus there is no need to change the key $\Bbbk_1(dept, \{enroll/sid\})$, the XID $\Upsilon_1(dept, (\{project/sid\} \subseteq \{enroll/sid\}))$ or the XFK $\digamma_1(dept, (\{project/sid\} \subseteq \{enroll/sid\}))$. Now we want to see whether the key, the XID or the XFK is preserved.

$<!ELEMENT\ \ dept(enroll, project) >$
$<!ELEMENT\ \ enroll(sid, cid^?)^+ >$
$<!ELEMENT\ \ project(sid^+) >$

(a) DTD D_3



(b) Tree T_3

Figure 3. Target

We observe that the key $\Bbbk_1(dept, \{enroll/sid\})$(we can say $\Bbbk_3(dept, \{enroll/sid\})$ because it is now on the DTD D_3) is not satisfied by the transformed document T_3 because there there are two values $(v_5 : sid : 001)$ and $(v_7 : sid : 001)$ for the element sid under node $v_1 : enroll$ which are not distinct. However, the XID Υ_1(we can say Υ_3) is preserved. As the key is not preserved, so the XFK $\digamma_1$(we can say $\digamma_3$) is also not preserved.

Observation 2.*The XFK may not be preserved after the transformation.*

While addressing the problem, we consider the following contributions. *First*, we define the XML foreign key(XFK) and XML Inclusion Dependency (XID) on the XML Document Type Definition (DTD) [16] and their satisfactions using a novel concept called $tuple$. *Second*, we show the transformation on the definition of XFK and XID using basic transformation operations. We also check whether the transformation makes the transformed XFKs and XIDs valid. *Last*, we study whether the transformed XIDs and XFKs are preserved by the transformed XML documents using important operators with sufficient conditions.

2. Transformation on Referential Integrity Constraints

In this section, we study the transformation on XID and XFK. For detailed definitions of XID and XFK, we refer our research in[11, 12]. Given a DTD D and a document T such that T conforms to D as $T \Subset D$, a transformation τ transforms D to $\bar{D}$ and T to $\bar{T}$, denoted by $\tau(D, T) \rightarrow (\bar{D}, \bar{T})$. The problem whether $\bar{T}$ conforms to $\bar{D}$ was investigated in [15]. We investigate how the transformation affects the properties of an XID Υ defined on D. More formally, given D, T, Υ and a transformation τ such that $\Upsilon \sqsubset D \wedge T \Subset D \wedge T \prec \Upsilon$, and $\tau(D, T, \Upsilon) \rightarrow (\bar{D}, \bar{T}, \bar{\Upsilon})$), we would like to know what $\bar{\Upsilon}$ is, whether $\bar{\Upsilon}$ is valid on $\bar{D}$, and whether $\bar{T}$ satisfies $\bar{\Upsilon}$.

2.1. Transformation on XID

We now define the transformation of XID. In defining the transformation, we need to refer to the DTD type structure g and the paths w of an XID. For space reason, we refer to the research in[15] for detailed definitions on DTD structure and transformation operations.

Transformation on XID using $Rename$ **Operator.** The rename operators changes the element name with a new element name. As the path on the DTD is changed using rename operator, the XID defined on the DTD is also transformed. Formally, let $w = e_1/\cdots/e_k/e_{k+1}/\cdots/e_m$. If the transformation $\tau = rename(e_k, \bar{e}_k)$, then $\tau(w) = \bar{w} = e_1/\cdots/\bar{e}_k/e_{k+1}/\cdots/e_m$.

For example, consider the XID $\Upsilon_1(dept, (\{project/sid\} \subseteq \{enroll/sid\}))$ on the DTD D_1 in Fig.1(a). If we use $rename(sid, studentID)$, then the path P and the path R are transformed and the XID is transformed to $\bar{\Upsilon}_1(dept, (\{project/studentID\} \subseteq \{enroll/studentID\}))$.

Transformation on XID using $Expand$ **Operator.** Let $w = e_1/\cdots/e_{k-1}/e_k/e_{k+1}/\cdots/e_m$ be a key path, g be the target of the transformation in $\beta(e_{k-1})$ and $e_k \in g$ (thus $e_k \in \beta(e_{k-1})$ too), and $\tau = expand(g, e_{new})$ be the operator that pushes g one level away from the root. The operator τ transforms $w = e_1/\cdots/e_{k-1}/e_k/e_{k+1}/\cdots/e_m$ to a few cases depending on whether τ adds to the end of Q (i.e., $w = Q$ and $m = k-1$) and whether, in case of appending to the end of Q, all P paths and R paths are pushed down (i.e. $\forall\, w \in P(beg(w) \in g)$ and $\forall\, w \in R(beg(w) \in g)$). A path is said **pushed down** means that a new element is added to the beginning of the path.

(1) If w is a dependent P path or w is a referenced R path or w is the selector Q path and $m > (k-1)$ (τ is not adding to the end of Q), then $\tau(w) \rightarrow \bar{w} = e_1/\cdots/e_{k-1}/e_{new}/e_k/e_{k+1}/\cdots/e_m$.
This case is described in Fig.4 using Case 1. In Case 1(a), $w = Q$ and the new element e_{new} is added between e_{k-1} and e_k in path w. In Case 1(b), $w = P_i$ and e_{new} added between e_{k-1} and e_k in path w. Similarly, In Case 1(c), $w = R_i$ and e_{new} added between e_{k-1} and e_k in path w.

(2) If w is the Q path and $m = (k-1)$ and $\forall\, w \in P(beg(w) \in g) \wedge \forall\, w \in R(beg(w) \in g)$, then
  (1) Option 1: $\bar{Q} = Q/e_{new}$ and $\forall\, w \in P(\bar{w} = w)$ and $\forall\, w \in R(\bar{w} = w)$.
  This is shown in Case 2(option 1) of Fig. 4 where $w = Q$ and e_{new} is added at the end of Q and all P paths and R paths are not changed.
  (2) Option 2: $\bar{Q} = Q$ and $\forall\, w \in P(\bar{w} = e_{new}/w)$ and $\forall\, w \in R(\bar{w} = e_{new}/w)$.
  This case corresponds to Case 2(option 2) of Fig. 4 where $w = Q$ and e_{new} is added at the beginning of all P paths and R paths.
Both options represent reasonable semantics. The choosing from the two options is determined by the user.

(3) (a) If $last(Q) = e_{k-1} \wedge \exists\, w \in P(beg(w) \notin g)$, then $\bar{Q} = Q$ and $\forall\, w \in P$ (if $beg(w) = e_k$ then $\bar{w} = e_{new}/w$ else $\bar{w} = w$). This case is shown in Case 3(a) of Fig. 4 where e_{new} is added at the beginning of some P paths.
  (b) If $last(Q) = e_{k-1} \wedge \exists\, w \in R(beg(w) \notin g)$, then $\bar{Q} = Q$ and $\forall\, w \in R$ (if $beg(w) = e_k$ then $\bar{w} = e_{new}/w$ else $\bar{w} = w$). This case is shown in Case 3(b) of Fig. 4 where e_{new} is added at the beginning of some R paths.

For example, consider the XID $\Upsilon_1(dept, (\{project/sid\} \subseteq \{enroll/sid\}))$ on the DTD D_1 in Fig.1(a). Now if we use $expand([sid{\times}cid^*], stud)$, then the path on R is only transformed and the transformed XID is $\bar{\Upsilon}_1(dept, (\{project/sid\} \subseteq \{enroll/stud/sid\}))$ which is valid on the DTD D_2 in Fig.2(a). In this case, the Case 1(c) in Fig.4 is used in transformation.

Now consider another transformation $expand([enroll{\times}project], courseDB)$, then we have two options to transform the XID $\Upsilon_1(dept, (\{project/sid\} \subseteq \{enroll/sid\}))$.

In first option(using Case 2 and option 1 in Fig.4), we transform the selector by adding the new element $courseDB$ at the end of path $dept$ as $dept/courseDB$ and the transformed XID is $\bar{\Upsilon}_1(dept/courseDB, (\{project/sid\} \subseteq \{enroll/sid\}))$. In second option(using Case 2 and op-

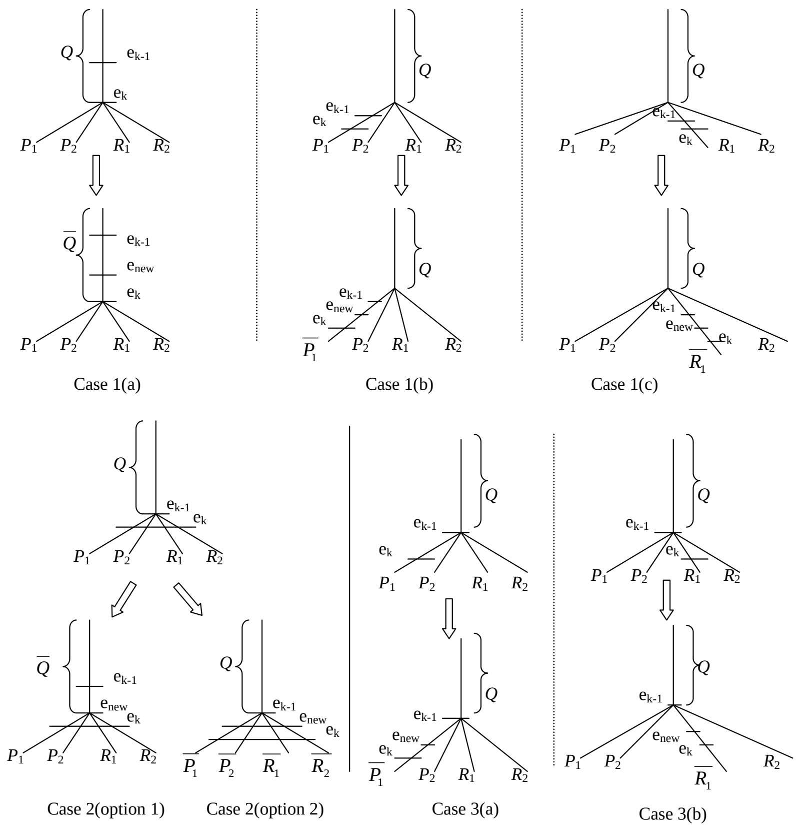


Figure 4. Transformation on XID using $expand$ operation

tion 2 in Fig.4), we transform both P path $project/sid$ and R path $enroll/sid$ by adding the new element in the beginning and the transformed XID is $\bar{\Upsilon}_1(dept, (\{courseDB/project/sid\} \subseteq \{courseDB/enroll/sid\}))$.

Transformation on XID using $Collapse$ Operator. If $e_{k+1} \in \beta(e_k)$ and $\tau = collapse(e_k)$, then any path $w = e_1/\cdots/e_{k-1}/e_k/e_{k+1}/\cdots/e_m$ where $(k-1) \geq 1$ is transformed as $\tau(w) \rightarrow \bar{w} = e_1/\cdots/e_{k-1}/e_{k+1}/\cdots/e_m$. We note that $collapse(e_k)$ is not applicable if $\beta(e_k) = Str$ and $e_k = \rho$ where ρ is the root.

Theorem 2.1 *In XID $\Upsilon(Q, (\{P_1, \cdots, P_n\} \subseteq \{R_1, \cdots, R_n\}))$ transformation, Q, or P, or R can be transformed, but not Q, P or Q, R are not transformed together.*

Theorem 2.2 *Let τ be a transformation defined above such that $\tau(\Upsilon) = \bar{\Upsilon}$. Then $\bar{\Upsilon}$ is valid on $\bar{D}$, denoted as $\bar{\Upsilon} \sqsubset \bar{D}$.*

2.2. Transformation on XFK

As we already mentioned earlier that the XFK $\digamma(Q, (\{P_1, \cdots, P_n\} \subseteq \{R_1, \cdots, R_n\}))$ is defined using XID $\Upsilon(Q, (\{P_1, \cdots, P_n\} \subseteq \{R_1, \cdots, R_n\}))$ and XML key $\Bbbk(Q, \{R_1, \cdots, R_n\})$, thus in transforming the XFK, we use the rules of transformation on XID described in the previous subsection and we need to know whether there is a need to transform the key as result of XID transformation. We see that in both XID $\Upsilon(Q, (\{P_1, \cdots, P_n\} \subseteq \{R_1, \cdots, R_n\}))$ and XML key $\mathbb{k}(Q, \{R_1, \cdots, R_n\})$, the selector path Q and the dependent paths $R_1, \cdots, R_n$ are the same. Thus the rules of transformations on XML key are the same as the rules of transformations on XID except the transformations on the paths of P.

For example, consider the XFK $\digamma_1(dept, (\{project/sid\} \subseteq \{enroll/sid\}))$ on the DTD D_1 in Fig.1(a). If we use the transformation $expand([enroll{\times}project], courseDB)$, then we transform the XID $\Upsilon_1(dept, (\{project/sid\} \subseteq \{enroll/sid\}))$. We also need to transform the key $\mathbb{k}_1(dept, \{enroll/sid\})$. In this case, we have two options like XID transformation. In option 1, we transform as $\bar{\mathbb{k}}_1(dept/courseDB, \{enroll/sid\})$. In option 2, we transform as $\bar{\mathbb{k}}_1(dept, \{courseDB/enroll/sid\})$.

Theorem 2.3 *For the XFK $\digamma(Q, (\{P_1, \cdots, P_n\} \subseteq \{R_1, \cdots, R_n\}))$ where the XID is $\Upsilon(Q, (\{P_1, \cdots, P_n\} \subseteq \{R_1, \cdots, R_n\}))$ and the key is $\mathbb{k}(Q, \{R_1, \cdots, R_n\})$, Q of both key and XID can be transformed or R of both key and XID can be transformed or P of XID can be transformed.*

Theorem 2.4 *Let τ be a transformation defined above such that $\tau(\digamma) = \bar{\digamma}$. Then $\bar{\digamma}$ is valid on $\bar{D}$, denoted as $\bar{\digamma} \sqsubset \bar{D}$.*

3. Preservation of XML Referential Integrity Constraints

First, we investigate how a transformation affects the satisfaction of transformed XID. More specifically, given $\tau(D, T, \Upsilon) \rightarrow (\bar{D}, \bar{T}, \bar{\Upsilon}) \wedge \bar{\Upsilon} \sqsubset \bar{D}$, we investigate whether $\bar{T}$ satisfies $\bar{\Upsilon}$ as $\bar{T} \prec \bar{\Upsilon}$.

Definition 3.1 (XID Preservation) *Given the transformations on D, T, Υ as $\tau(D, T, \Upsilon)$ $\rightarrow (\bar{D}, \bar{T}, \bar{\Upsilon}) \wedge \bar{\Upsilon} \sqsubset \bar{D}$, if $T \prec \mathbb{k}$ and $\bar{T} \prec \bar{\Upsilon}$, we say that Υ is preserved by the transformation τ.* $\square$

3.1. Preservation of XID

We now recall the definition of XID $\Upsilon(Q, (\{P_1, \cdots, P_n\} \subseteq \{R_1, \cdots, R_n\}))$ satisfaction which requires that for every P-tuple, there must have a R-tuple under the selector node Q. Thus when we check the preservation of XID, we need to check whether the condition of the XID satisfaction is violated when the document is transformed.

Preservation of XID using $Rename$ operation. The rename operator only changes the element name with a new element name in DTD. Thus the path involving the element name to be renamed is also transformed and as a result, XID is also transformed. But the P-tuple or R-tuple is not changed. Also there is no deletion of either P-tuple of R-tuple. So the transformed XID is satisfied by the transformed document.

Preservation of XID using $UnNest$ and $Nest$ operation. The $unnest$ operator transforms the nested structure of a document to the flat structure. In this case,we recall the example where the source tree $T = (\rho(A : 1)(B : 2)(B : 3)(A : 4)(B : 5))$ is transformed to $\bar{T} = (\rho(A : 1)(B : 2)(A : 1)(B : 3)(A : 4)(B : 5))$ using $unnest(B)$ operation. We see that the number of A element is increased in the transformed document. However, there is no loss of information in the transformation. As the XID satisfaction requires that for every P-tuples, there must be a R-tuple, there is no violation XID satisfaction for the transformed document.

In case of $nest$ operation, the source tree $T = (\rho(A : 1)(B : 2)(A : 1)(B : 3)(A : 4)(B : 5))$ is transformed to $\bar{T} = (\rho(A : 1)(B : 2)(B : 3)(A : 4)(B : 5))$ using $nest(B)$. We see that the element A with the same values are nested with B elements. Thus there is no violation of XID satisfaction property for the transformed document. We note here that there is no transformation of XID definition using both $unnest$ and $nest$ operations.

Preservation of XID using $Expand$ **and** $Collapse$ **operation.** The $expand$ operator pushes a structure with a new element for semantics. We showed the the transformations on XID using $expand$ operation in the previous section. Regardless of different transformation rules and options in transforming XID, there is no change of either P-tuples or R-tuples in the document. The transformed XID is satisfied by the the transformed document and thus XID is preserved.

In case of $collapse$, an element is deleted from the DTD and the document as well. Note that there $collapse$ operation is not applicable where the element is the root of the document and $\beta(e) = Str$. After $collapse$, there is no change to P-tuples or R-tuples. The transformed XID is satisfied by the transformed document and thus XID is preserved.

3.2. Preservation of XFK

We recall the definition of XFK $F(Q, (\{P_1, \cdots, P_n\} \subseteq \{R_1, \cdots, R_n\}))$ satisfaction which requires two things: (a) for every P-tuple, there must be a R-tuple under the selector node Q node and (a) all the R-tuples are distinct under the selector node Q. The first requirement means the XID satisfaction and the second requirement means the XML key satisfaction. We already showed how XID is preserved for important transformation operations. Now we show XFK preservation using the same transformation operators.

In XFK preservation, we need both the XID and the XML key preserved. In the previous subsection, we showed the XID preservation. The $rename$, $nest$, and $collapse$ are XID preserving. The XML key is also preserved using $nest$, $collapse$ and $rename$ operators because there is no change of R-tuples under the selector nodes for path Q. Although $unnest$ and $expand$ operators are XID preserving, but they are key preserving with some sufficient conditions. We now study the XFK preservations with $unnest$ and $expand$ operations.

Preservation of XFK using $UnNest$ **and** $Expand$ **operations.** We already showed that the $unnest$ operation is XID preserving. In $unnest$ operation, although the XID is preserved, but the XML key is not preserved if some conditions are not satisfied. Thus XFK is also not preserved using $unnest$ operation if some conditions of key preservation are not satisfied. We illustrate the condition using the following example.

The $unnest$ operator spreads the hedge of the structure g_1 to the hedges of the structure g_2. For example, given $\beta(e) = [A{\times}B{\times}[C{\times}D]^+]^*$ and $T = (e(A:1)(B:1)(C:2)(D:3)(C:2)(D:4)(A:1)(B:2)(C:2)(D:4))$, the operator $unnest(C{\times}D)$ spreads the hedge of the structure $A{\times}B$ to the hedges of the structure $C{\times}D$ and produces $\beta_1(e) = [A{\times}B{\times}[C{\times}D]]^*$ and $\bar{T} = (e(A:1)(B:1)(C:2)(D:3)(A:1)(B:2)(C:2)(D:4)(A:1)(B:1)(C:2)(D:4))$.

Consider the XML key $\Bbbk(Q, \{R\})$. The $unnest(g_2) : [g_1{\times}g_2^+]^+ \rightarrow [g_1{\times}g_2]^+$ operator is key preserving if (a) g_1 does not cross the selector Q, and (b) if g_1 crosses R paths, g_2 also crosses some R paths. For example, in the above example, the key is $\Bbbk(e, \{A, B\})$, then it is satisfied by the document T because there are R-tuples $((A:1)(B:1))$ and $((A:1)(B:2))$ which are distinct. But the after the $unnest$ operation, the key is not satisfied by the document $\bar{T}$ because there are two R-tuples as $((A:1)(B:1))$ which are vale equivalent in the document $\bar{T}$. Thus the key is not preserved after $unnest$ operation.

We showed that the $expand$ operator is XID preserving. But $expand$ operations is key preserving if some sufficient conditions are satisfied. Thus the $expand$ operator is XFK preserving if some sufficient conditions are satisfied.

4. Conclusions

We studied the preservations of both XML inclusion dependency and XML foreign key. We found that although inclusion dependency is preserved for transformation operations, but foreign key is not preserved for some important transformation operations. We then identified sufficient conditions for preservations. Our study on preserving referential integrity for XML is towards the data integration in XML with integrity constraints.

References

[1] Fankhouser P., Klement T.: XML for Datawarehousing Chances and Challenges. In: DAWAK,LNCS 2737,pp.1-3(2003)

[2] Zamboulis L., Poulovassilis A.: Using Automed for XML Data Transformation and Integration. In: DIWeb, pp.58-69(2004)

[3] Zamboulis L.: XML Data Integration by Graph Restructuring. In: BNCOD,pp.57-71(2004)

[4] Poggi A., Abiteboul S.: XML Data Integration with Identification. In: DBPL,pp. 106-121(2005)

[5] Buneman P., Fan W., Simeon J., Weinstein S.: Constraints for Semistructured Data and XML. In: SIGMOD Record, pp. 47-54(2001)

[6] Fan W.: XML Constraints: Specification, Analysis, and Applications. In:DEXA,pp.805-809(2005)

[7] Fan W., Simeon J.: Integrity constraints for XML. In: PODS,pp.23-34(2000)

[8] Fan W., Libkin L.: On XML Integrity Constraints in the Presence of DTDs. In: Journal of the ACM, vol.49, pp. 368-406(2002)

[9] Vincent M., Schrefl M., Liu J., Liu C., Dogen S.: Generalized inclusion dependencies in XML. In: APWeb(2004)

[10] Karlinger M., Vincent M., Scherefl M.: On the Definition and Axiomitization of Inclsuion Dependency for XML. In: Tecnical Report, No.07/02,Johanne Kepler University(2007)

[11] Shahriar Md. S., Liu J.: On Defining Referential Integrity for XML. In IEEE International Symposium of Computer Science and Its Applications(CSA),pp.286-291(2008)

[12] Shahriar Md. S., Liu J.: On Defining Keys for XML. In: IEEE CIT2008, Database and Data Mining Workshop, DD'08,pp. 86-91(2008)

[13] Shahriar Md. S., Liu J.: Preserving Functional Dependency in XML Data Transformation. In: ADBIS, LNCS 5207, pp. 262-278(2008)

[14] Shahriar Md. S., Liu J.: Towards the Preservation of Keys in XML Data Transformation for Integration. In: COMAD, pp. 116-126(2008)

[15] Liu J., Park H., Vincent M., Liu C.: A Formalism of XML Restructuring Operations. In: ASWC, LNCS 4185, pp. 126-132(2006)

[16] Tim Bray, Jean Paoli, and C. M. Sperberg-McQueen, Extensible Markup Language (XML) 1.0., World Wide Web Consortium (W3C), Feb 1998. `http://www.w3.org/TR/REC-xml`.

[17] Henry S. Thompson, David Beech, Murray Maloney, and Noah Mendelsohn, XML Schema Part 1:Structures, W3C Working Draft, April 2000. `http://www.w3.org/TR/xmlschema-1/`.

Biography

Md. Sumon Shahriar: Sumon Shahriar is currently PhD researcher in Data and Web Engineering Lab, School of Com puter and Information Science, University of South Australia. He achieved his Bachelor of Science (Honours) and Master of Science (Research) degrees both with first class in Computer Science and Engineering from University of Dhaka, Bangladesh. His research interests include XML database, Data Integration, Data Quality and Data Mining.

Dr. Jixue Liu: Jixue Liu got his bachelor's degree in engineering from Xian University of Architecture and Technology in 1982, his Masters degree (by research) in engineering from Beijing University of Science and Technology in 1987, and his PhD in computer science from the University of South Australia in 2001. His research interests include view maintenance in data warehouses, XML integrity constraints and design, XML and relational data, constraints, and query translation, XML data integration and transformation, XML integrity constraints transformation and transition, and data privacy.