Generating XSLT Scripts for The Fast Transformation of XML Documents

Dong-Hoon Shin
Dept. of Computer Science, Yonsei Univ
134 Shinchon-dong, Sudaemoon-ku,
Seoul 120-749, Korea
+82-2-2123-3878
dhshin@icl.yonsei.ac.kr

Kyong-Ho Lee
Dept. of Computer Science, Yonsei Univ
134 Shinchon-dong, Sudaemoon-ku,
Seoul 120-749, Korea
+82-2-2123-5712
khlee@cs.yonsei.ac.kr

ABSTRACT
This paper proposes a method of generating XSLT scripts, which support the fast transformation of XML documents, given one-to-one matching relationships between leaf nodes of XML schemas. The proposed method enhances the transformation speed of generated XSLT scripts through reducing template calls. Experimental results show that the proposed method has generated XSLT scripts that support the faster transformation of XML documents, compared with previous works.

Categories and Subject Descriptors
I.7 [**Computing Methodologies**]: Document and Text Processing

General Terms: Documentation, Languages

Keywords: Document transformation, XML, XSLT

1. Introduction
To share and exchange XML documents that conform to different schemas, finding semantic relationships between two schemas and transforming XML documents based on the relationships are needed. Moreover, since an XSLT script is repeatedly applied to a large volume of XML documents, its transformation speed is important. This paper proposes a method of generating XSLT scripts, which support the fast transformation of XML documents, given one-to-one matching relationships [1] between leaf nodes of source and target schemas.

2. Related Work
There has been some related works [2, 3, 4, 5] that generate XSLT scripts to support the transformation of XML documents. Previous works translate each matching relationship between nodes of XML schemas into a template in an XSLT script. These scripts are intuitive but have the defect that the transformation speed slows down by frequent template calls. Moreover, since some previous works [3, 5] only consider one-to-one matching relation-ships between internal nodes of schemas, the information of non-selected nodes may be lost. Previous works lack the appropriate selection of XSLT commands. To support the fast transformation of XML documents, the proposed method uses fewer templates than previous works and selects effective XSLT commands.

3. Document Model
Since there is a hierarchical order among elements in XML documents, a proposed document model is based on an ordered tree that has a root node. Specifically, each element constructs a node of a tree structure. Nodes in the proposed document model are divided into three classes: A general node that represents an element or an attribute, a content model node that represents a sequence or choice content model, and a cardinality node that represents the frequency of a node. A recursive is represented as a reference from a leaf node to its corresponding ancestor node.

WWW 2005, May 10-14, 2005, Chiba, Japan.
ACM 1-59593-051-5/05/0005.

4. XSLT Scripts Generation Algorithm
The proposed algorithm consists of two steps : cardinality node matching and generating XSLT scripts as shown in <Figure 1>.

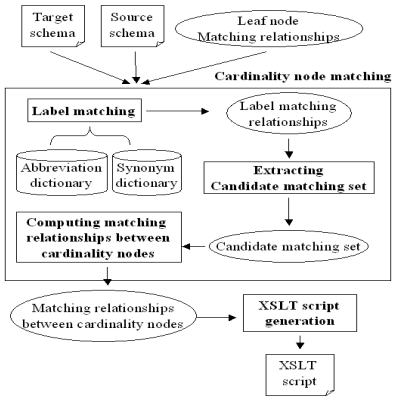

Figure 1. The XSLT script generation process.

4.1 Cardinality Node Matching
The proposed method creates internal node matchings through comparing the corresponding paths. Two paths, Path(x) and Path(y), are called corresponding where there is a matching relationship between two leaf nodes x and y. For a leaf node x, Path(x) is a sequence of nodes from the parent node of x to the root node.

Cardinality and choice operators make it possible to create virtually any number of XML documents with different structures. In order to support repeating structures which is created by cardinality nodes, matching relationships between cardinality nodes have to be computed by considering capacities of source and target schemas. On the other hand, in case of a choice operator, appropriate XSLT scripts can be generated from given leaf node matching relationships and structural information of a schema without having to compute matchings between choice operators.

For each given leaf node matching relationship, the proposed method extracts paths from the root node to leaf nodes, and then finds matchings between labels of the internal nodes on the two paths. We assume that two schemas belong to the same domain

and the root nodes are associated with each other. The lexical similarity for label matching is computed by Equation (1).

$$\text{Lexical Similarity}(N_s, N_t) = \frac{2\times\sum\text{TokenSimilarity}(Ns_i, Nt_j)}{|N_s| + |N_t|} \quad (1)$$

Ns_i: a token of a source label, $1\le i\le n$, Nt_j: a token of a target label, $1\le j\le m$

Through the label matching, each path is divided into sub-pahts. For a path, a sub-path is defined as a consecutive list of nodes that do not have matching relationships. For each pair of sub-paths computed after the label matching, a candidate matching set is extracted. A candidate matching set includes only cardinality nodes in the corresponding sub-path. Computing a candidate matching set from the sub-path improves a precision of internal node matching and saves the computing power.

Between nodes of the given two candidate matching sets, two nodes that have a maximal similarity are selected and a matching relationship between them is created based on Equation (2).

$$\text{Structural Similarity}(n_s, n_t) = \frac{|\text{Matchings between TAN}(n_s) \text{ and TAN}(n_t)|}{(|\text{leaf nodes of } n_s| + |\text{leaf nodes of } n_t|)/2} \quad (2)$$

For an internal node n, TAN(n) is the set of associated leaf nodes of the subtree starting from n.

Once a matching relation ship with the maximal similarity is selected, the candidate matching set is divided into two candidate matching sets and the procedure is applied recursively.

4.2 XSLT Script Generation

In this step, XSLT scripts are generated from the cardinality node matching relationships, which were computed in the previous step.

First, the proposed method generates a template for the root node of a source DTD tree, and a target DTD tree is traversed in depth-first order and the corresponding part of an XSLT script for each node type is generated. Nodes are divided into four types: A choice operator node, a cardinality node and a leaf node.

In case of a choice operator node, only one of its child nodes can appear in a document. For a child node n_t, if a source leaf node, which is associated with any element of TAN(n_t), appears in a source document, node n_t can appear in a target document. These conditions are represented in an XSLT script by using the '*xsl:choose*' and '*xsl:when*'.

Cardinality nodes are divided into two types: the ? node and others. In case of a ? node, its child nodes can appear in a target document if a source document contains at least one child node of a cardinality node that is associated with the ? node. This condition is represented in an XSLT script by using the '*xsl:if*'. For other types of cardinality nodes, the proposed method uses the '*xsl:for-each*' element of XSLT. If a target cardinality node is associated with more than one source cardinality node, the proposed method generate XSLT script code which select the node with more repeating child node in a source document. Through this code, all children of the matched source cardinality nodes are copied into a target document as many as the repeating count of chosen node. In this process, implicit information loss which may occur in previous works can be reduced.

In case of a leaf node, text data of each leaf node is copied by the 'xsl:value-of'.

5. Experimental Results

To evaluate the proposed method, we experimented with two DTDs which were used in the work of Su et al. (test data 1) and Kuikka et al. (test data 2). The transformation speed of the XSLT scripts generated by the proposed method and the previous methods are measured while varying the size of XML documents. Particularly, experiments have been performed by using three representative XSLT processors, Xalan-Java 2.6.0, Saxon-B 8.1.1, and XSLTCommand. Experimental results with test data 1 on XSLTCommand is shown in <Figure 2>.

In the experimental results for test data 1 and 2, the XSLT script generated by the proposed method shows the fastest transformation speed. Average improvement rate of transformation speed was 20% over previous works.

Also, in additional experiments for effectiveness of XSLT commands, it is found that using a descendant axis in an XPTH, '<xsl:copy-of>', or <xsl:element> slows down overall transformation speed of generated XSLT scripts. In the proposed method, the improvement rate of transformation speed by using effective XSLT commands was about 8%.

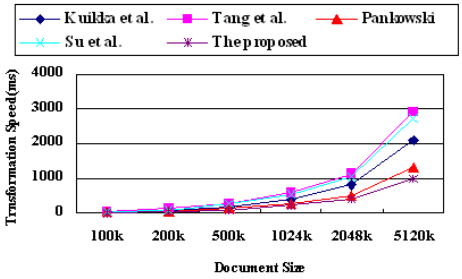

Figure 2. Result for test data 1 on XSLTCommand.

This paper proposed the method of generating XSLT script, which supports the fast transformation of XML documents. However, the proposed method assumes that one-to-one matching relationships between leaf nodes. In general, many-to-one or one-to-many relationships that need more complex operations such as merge or split can exist. In the future, we study a method that supports those matching relationships.

6. Acknowledgments

This work was supported by the Korea Research Foundation Grant (KRF-2004-041-D00613).

7. References

[1] Jun-Seung Lee and Kyong-Ho Lee, "XML Schema Matching Based on Incremental Ontology Update," Lecture Notes in Computer Scien-ce, Vol. 3306, pp. 608-618, 2004.

[2] Tadeusz Pankowski, "A high-level Language for Specifying XML Data Transformations," Conf. Advances in Databases and information Systems, pp. 22-25, 2004.

[3] Eila Kuikka, Paula Leinonen, and Martti Penttonen, "Towards Automating of Document Structure Transformations," Proc. ACM Symposium Document Engineering, pp. 103-110. 2002.

[4] Xuerong Tang and Frank Wm. Tompa, "Specifying Transformations for Structured Documents," Proc. Int'l Workshop the Web and Databases, pp. 67-72, 2001.

[5] Hong Su, Harumi Kuno, and Elke A Rundensteiner, "Automating The Translation of XML Documents, "Proc. Int'l Workshop Web Information and Data Management, pp. 68-75, 2001.