XML 문서의 빠른 변환을 위한 XSLT 스크립트

(XSLT Scripts for Fast XML Document Transformation)

신 동 훈 † 이 경 호 ††
(Dong-Hoon Shin) (Kyong-Ho Lee)

요 약 본 논문에서는 스키마를 구성하는 단말 노드 간의 일대일 매칭으로부터 XML 문서의 빠른 변환을 지원하는 XSLT 스크립트를 생성하는 방법을 제안한다. 제안된 방법은 XML 스키마를 구성하는 빈도 지시자 간의 대응관계 생성과 XSLT 스크립트 생성의 두 단계로 이루어진다. 어휘 및 구조 유사도를 이용하여 빈도 지시자 간의 대응관계를 생성하며 이를 바탕으로 적은 수의 템플릿을 포함하는 XSLT 스크립트를 생성한다. 성능을 평가하기 위하여 다양한 크기의 문서와 다수의 XSLT 처리기에 대해 실험한 결과, 제안된 방법은 기존 연구보다 XML 문서를 보다 빠르게 변환하는 XSLT 스크립트를 생성하였다.

키워드 : 변환, XML 문서, XSLT 스크립트

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 consists of two steps: computing matchings between cardinality nodes and generating XSLT scripts. Matching relationships between cardinality nodes are computed by using proposed lexical and structural similarities. Based on the cardinality node matching relationships, an XSLT script is generated. Experimental results show that the proposed method has generated XSLT scripts that support the faster transformation of XML documents, compared with previous works.

Key words : Transformation, XML document, XSLT scripts

1. 서 론

XML(eXtensible Markup Language) [1] 문서는 논리적 구조 정보를 표현할 수 있으며 플랫폼에 독립적이라는 장점 때문에 다양한 분야에서 정보의 공유 및 교환을 위한 표준으로 널리 사용되고 있다. 또한 XML 문서가 보편적으로 사용됨에 따라 XML 문서의 구조 정보를 정의한 XML 스키마 역시 급증하고 있다. 한편 서로 다른 스키마를 이용하여 작성된 XML 문서를 통해 정보를 공유하거나 교환하기 위해서는 두 스키마 사이의 의미적 관계를 찾고 이를 바탕으로 XML 문서를 변환하는 작업이 필요하다.

일반적으로 XML 문서의 변환 과정은 그림 1과 같이 스키마 매칭과 XSLT(eXtensible Stylesheet Language Transformations) [2] 스크립트 생성의 두 단계로 구성된다. 스키마 매칭 단계에서는 소스 스키마와 타겟 스키마를 입력으로 받아 의미적인 대응관계를 생성하고, 변환 스크립트 생성 단계에서는 전 단계에서 계산된 대응관계를 이용하여 XML 문서를 변환하는 스크립트를 생성한다.

본 논문에서는 XML 스키마[1]를 구성하는 단말 노드(leaf-node) 간의 일대일(one-to-one) 대응관계 [3]으로부터 XSLT 스크립트를 생성하는 방법을 제안한다. 특히 본 논문에서는 수행 속도가 빠른 XSLT 스크립트를 생성하는 방법에 중점을 둔다. 여기서 XSLT 스크립트의 수행 속도는 해당 XSLT 스크립트를 적용하여 XML 문서를 변환하는데 걸리는 시간을 의미한다. 일반적으로 한번 생성된 XSLT 스크립트는 대용량의 문서를 대상으로 반복적으로 적용되기 때문에 XML 문서의 변환에 있어서 XSLT 스크립트의 수행 속도는 매우 중요하다.

예를 들어, 바이오 인포매틱스(Bioinformatics) 분야

· 이 논문은 2004년도 한국학술진흥재단의 지원에 의하여 연구되었음 (2004-041-D00613)
† 학생회원 : 연세대학교 컴퓨터과학과
dhshin@icl.yonsei.ac.kr
†† 종신회원 : 연세대학교 컴퓨터과학과 교수
khlee@cs.yonsei.ac.kr
논문접수 : 2005년 4월 8일
심사완료 : 2005년 9월 30일

1) 본 논문에서 XML 스키마와 XML DTD(Document Type Definition) [1]은 동일한 의미로 사용된다.

에서는 많은 바이오 데이타들이 특정 스키마를 따르는 XML 형태로 표현되어 저장된다. 이렇게 생성되는 데이타들은 그 양이 방대할 뿐만 아니라 개별 데이타의 크기도 매우 크다. 한편 바이오 인포매틱스 분야는 연구의 특성상 다른 연구자들에 의해 생성된 다양한 데이타들을 비교 및 분석할 필요가 있고, 이를 위해 서로 다른 스키마로부터 생성된 다수의 바이오 데이타들이 교환 및 공유되어야 한다. 이러한 경우, XSLT 스크립트는 대용량의 바이오 데이타들에 대해 반복적인 변환 작업을 수행해야 하므로 변환 수행 속도가 매우 중요해진다. 또한 대용량 문서에 대한 변환뿐만 아니라 B2B 분야에서와 같이 다른 스키마를 따르는 XML 문서를 빈번하게 교환하게 되는 경우, XSLT 스크립트의 빠른 변환 수행 속도는 작업의 효율과 능률을 향상시킬 수 있다.

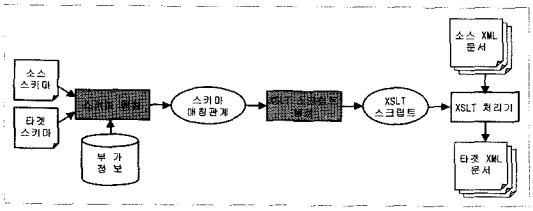


그림 1 XML 문서의 변환 과정

일반적으로 반복적인 구조를 갖는 XML 문서를 변환할 때 XSLT 템플릿(template)을 사용하면 스크립트의 양을 줄일 수 있다. 그러나 템플릿 사이의 호출이 증가하면 XSLT 스크립트의 변환을 위한 처리시간은 증가한다. 또한 같은 변환을 수행하는 XSLT 스크립트라도 어떤 명령어를 사용했는지에 따라 변환 속도에 많은 차이가 생길 수 있다. 본 논문에서는 이러한 점에 착안하여 수행 속도가 빠른 XSLT를 생성하기 위해서 기존 방법보다 적은 수의 템플릿과 효율적인 명령어를 사용하여 XSLT 스크립트를 생성하는 방법을 제안한다.

제안된 방법은 빈도 지시자 노드(cardinality node) 간의 대응관계 생성과 XSLT 스크립트 생성의 두 단계로 이루어진다. 입력으로 주어진 단말 노드간의 일대일 대응관계를 기반으로 제안된 어휘 유사도와 구조 유사도를 적용하여 빈도 지시자 노드간의 대응관계를 생성한다. 두 번째 단계에서는 타겟 스키마를 깊이 우선(depth first) 탐색하면서 이전 단계에서 계산된 대응관계에 기반하여 각각의 노드에 대해 적절한 XSLT 스크립트를 생성한다.

제안된 방법은 빈도 지시자 노드 간의 대응관계 생성 단계에서 다대일 매칭을 허용한다. 또한 XSLT 스크립트 생성 단계에서 임의의 타겟 노드에 매칭된 여러 개의 소스 노드들 중에서 실제 문서에서 빈도수가 가장 높은 노드를 선택하는 XSLT 스크립트를 생성한다. 이는 기존의 XSLT 스크립트 생성에 관한 연구들[5,6,10]이 중간 노드 간의 일대일 혹은 일대다 매칭만을 허용함으로써 발생할 수 있는 정보 손실을 줄이는데 기여한다. 또한 제안된 방법은 재귀적인 구조를 포함하는 스키마들을 표현할 수 있는 문서 모델을 제안하고 이를 해결할 수 있는 방법을 제시한다.

제안된 방법의 성능을 평가하기 위하여 다양한 크기의 XML 문서와 다수의 XSLT 처리기를 대상으로 실험한 결과, 제안된 방법은 기존 방법과 비교하여 XML 문서를 보다 빠르게 변환하는 XSLT 스크립트를 생성하였다.

본 논문의 구성은 다음과 같다. 2절에서 XML 스키마를 효과적으로 표현할 수 있는 문서 모델을 제안한다. 또한 XSLT 스크립트 생성에 관한 기존 연구를 간략히 기술하고, 생성된 스크립트의 변환 수행속도 측면에서 문제점을 기술한다. 3절에서는 제안된 방법을 빈도 지시자 노드간의 대응관계 생성과 XSLT 스크립트 생성의 두 단계로 구분하고, 각 단계에 대한 자세한 설명을 기술한다. 4절에서는 실험 결과를 통하여 제안된 방법을 기존 연구와 비교 및 분석한다. 마지막으로 5절에서는 결론 및 향후 연구 방향을 기술한다.

2. 문서 모델 및 관련 연구

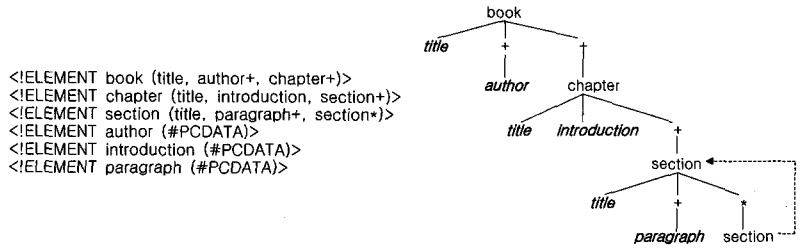
본 절에서는 XML 스키마를 효과적으로 표현할 수 있는 문서 모델을 제안한다. 또한 XSLT 스크립트 생성에 관한 기존 연구 결과를 간략히 소개하고 이의 문제점을 기술한다.

2.1 문서 모델

일반적으로 XML 스키마는 문서의 논리적인 구성 요소에 해당하는 엘리먼트(element)를 정의하는 엘리먼트 선언문(element declaration) 의 집합으로 구성된다. 또한 엘리먼트 선언문에 포함되는 내용 모델(content model)은 특정 엘리먼트가 포함할 수 있는 하위 엘리먼트의 종류와 문자열의 포함 여부를 명시한다.

특히 특정 엘리먼트에 포함되는 하위 엘리먼트와 문자열 사이에는 순서가 존재한다. 따라서 본 논문에서는 XML 스키마를 표현하기 위하여 루트 노드(root node)를 포함하며 형제 노드(sibling node) 간에 순서가 존재하는 순서 트리(ordered tree)에 기반한 문서 모델을 사용한다. 이 문서 모델에 따라 XML 스키마의 엘리먼트는 트리 구조의 노드를 구성한다.

문서 모델을 구성하는 노드는 엘리먼트와 애트리뷰트(attribute)를 표현하는 일반 노드, 내용 모델을 표현하는 내용 모델 노드, 그리고 노드의 반복 회수를 나타내는 빈도 지시자 노드로 나누어진다.



(a) XML 스키마의 예 (b) (a)를 제안된 문서 모델에 따라 표현한 결과

그림 2 XML 스키마의 예

일반 노드는 레이블을 가지며 엘리먼트와 애트리뷰트를 구분하기 위해 애트리뷰트의 레이블은 '@'을 접두어로 가진다. 내용 모델 노드는 선택 연산자(choice operator)인 '|' 또는 순차 연산자(sequence operator)인 ',' 을 레이블로 가진다. 순차 내용 모델을 나타내는 순차 연산자 ','는 선택 연산자 '|'의 자식 노드인 경우에만 문서 모델에 나타난다. 빈도 지시자 노드는 레이블로서 '?', '*', 또는 '+'를 포함한다. 그림 2(b)는 그림 2(a)의 XML 스키마를 제안된 문서 모델로 표현한 결과이다. 그림 2(b)의 section 노드처럼 하나의 경로에 동일한 노드가 두 번 이상 반복적으로 나타나는 재귀적인 구조는 자손 노드로부터 조상 노드로의 참조로써 표현된다. 단말 노드 x에 대해, 경로(x)는 x의 루트 노드로부터 부모 노드까지의 노드들의 순차적인 집합이다. 또한 단말 노드 x와 y가 대응관계를 갖는 경우, 경로(x)와 경로(y)는 서로 대응 한다고 정의한다.

2.2 관련 연구

본 절에서는 표 1과 같이 XSLT 생성을 위한 기존의 연구결과를 간략히 소개하고 생성되는 XSLT 스크립트의 변환 수행속도 측면에서의 문제점을 기술한다. 기존 연구 각각에 대한 기술은 다음과 같다.

Leinonen과 Kuikka 등 [4,5]은 XML 문서의 변환을 위한 XSLT 스크립트 생성 알고리즘을 제안한다. 소스 스키마와 타겟 스키마에 대해 단말 노드간의 레이블(label) 대응관계를 입력으로 받아들여 중간 노드간의 대응관계를 생성하고 소스 스키마와 타겟 스키마 를 서로 대응하는 여러 개의 부 구조(sub structure)로 나눈다. 생성된 중간 노드 대응관계와 부 구조들을 바탕으로 XSLT 스크립트를 생성한다. 특히 중간 노드 대응관계를 생성하기 위해서 TAN(Terminating Associated Nonterminal) 집합을 사용한다. 여기서 TAN(n)은 중간 노드 n을 루트 노드로 갖는 서브 트리(sub tree)에서 대응관계를 갖는 단말 노드들의 집합으로 정의한다. 두 스키마를 구성하는 노드간의 TAN 집합이 서로 일대일 대응관계를 형성한다면 두 노드를 의미상 동일한 노드로 간주하여 대응관계를 생성한다.

표 1 XSLT 스크립트 생성에 관한 연구

저자	연도	특징
Leinonen [4]	2003	사용자로부터 단말 노드에 대한 레이블 대응관계를 입력받아 중간 노드 대응관계를 생성하고 이를 바탕으로 XSLT 스크립트를 자동으로 생성한다.
Kuikka 등 [5]	2002	
Pankowski [6][7]	2004	Tang과 Tompa가 제안한 방법에 착안하여 개념 트리를 사용하는 변환 기술 언어인 XDTrans를 제안한다. 사용자가 제안된 언어를 사용하여 기술한 변환 정보를 XSLT 스크립트로 변환한다.
	2003	
Tang과 Tompa [8]	2001	변환 기술 언어인 Paired SynTrees를 사용하여 사용자가 기술한 변환 정보를 XSLT 스크립트로 변환한다.
Su 등 [9]	2001	스키마 변환 연산과 비용 모델을 사용하여 두 XML 스키마 간의 대응관계를 찾고 이를 바탕으로 XSLT 스크립트를 자동으로 생성한다.

Tang과 Tompa [8]는 구조화된 문서의 변환을 기술하는 고수준 언어인 Paired Syntrees를 제안한다. 사용자는 제안된 언어를 사용하여 두 스키마 사이의 필요한 대응관계, 대응규칙, 그리고 제약조건을 기술할 수 있다. 기술된 변환정보가 XSLT 스크립트로 변환되는 과정에서 각각의 대응규칙은 XSLT 템플릿으로 치환된다.

Pankowski [6,7]는 구조화된 문서의 변환을 기술하는 언어인 XDTrans를 제안한다. 사용자는 소스 스키마와 타겟 스키마를 표현할 수 있는 개념 트리(concept tree)를 정의하고 소스 스키마와 개념 트리간 그리고 타겟 스키마와 개념 트리 간의 매칭 관계를 기술한다. 기술된 변환 정보를 XSLT 스크립트로 변환하는 과정에서 각각의 매칭 관계는 XSLT 템플릿으로 치환된다.

Su 등 [9]은 두 스키마 간의 대응관계를 찾고 XSLT 스크립트를 생성하는 방법을 제안한다. 이를 위해서 변환 연산 및 비용 모델을 정의한다. 변환 연산을 적용하여 가능한 대응관계를 계산하며 찾아진 대응관계들 중에서 비용이 가장 적게 드는 대응관계를 선택한다. 이때

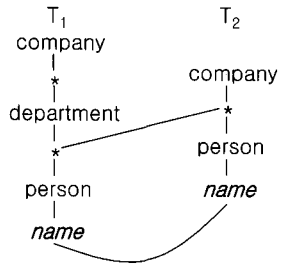


그림 3 두 스키마 간의 대응관계 생성의 예

```
<xsl:template match="company">
  <company>
    <xsl:for-each select="department/person">
      <xsl:call-template select="person_trans"/>
  </xsl:for-each>
 </company>
</xsl:template>
<xsl:template name="person_trans">
  <person>
    <xsl:apply-templates select="name">
  </department>
</xsl:template>
<xsl:template match="name">
  <name>
    <xsl:apply-templates/>
  </name>
</xsl:template>
```

(a) Kuikka 등과 Leinonen의 방법이 생성한 XSLT 스크립트의 예

```
<xsl:template match="company">
  <company>
    <xsl:for-each select="department/person">
      <person>
        <name>
          <xsl:value-of select="name"/>
        </name>
      </person>
    </xsl:for-each>
  </company>
</xsl:template>
```

(b) 수행 속도가 빠른 XSLT 스크립트의 예

그림 4 그림 3의 대응관계에 해당하는 XSLT 스크립트의 예

두 스키마는 여러 개의 서로 대응하는 부 구조로 나누어진다. Kuikka 등과 Leinonen의 방법과 같이 생성된 XSLT 스크립트는 부 구조의 수만큼의 템플릿을 포함한다.

전술한 바와 같이 기존의 연구들은 모두 템플릿에 기반하여 XSLT 스크립트를 생성한다. 수동 또는 자동으로 계산된 대응관계들이 XSLT 스크립트를 구성하는 템플릿으로 변환된다. 이러한 스크립트는 매우 직관적이고 코드의 양을 줄일 수 있다는 장점을 갖지만 잦은 템플릿 호출로 인하여 수행 속도가 느려지는 단점을 갖는다.

예를 들어, Kuikka 등과 Leinonen은 두 스키마 T_1과 T_2에 대해서 그림 3과 같은 대응관계를 생성하며, 그림 4(a)와 같은 XSLT 스크립트를 생성한다. 한편 그림 4(b)는 이와 동일한 변환을 수행하며 보다 간결한 XSLT 스크립트이다. 그림 4(b)의 경우 템플릿 호출 회수의 감소로 인해 수행 속도의 향상을 기대할 수 있다. 이와 같이 본 논문에서는 같은 변환을 수행하면서도 적은 수의 템플릿과 템플릿 호출 회수를 갖는 XSLT 스크립트를 생성하고자 한다.

3. XSLT 스크립트 생성 알고리즘

제안된 알고리즘은 그림 5와 같이 빈도 지시자 노드 매칭과 XSLT 스크립트 생성의 두 단계로 구성된다. 첫 번째 단계에서는 서로 대응하는 모든 단말 노드 쌍에 대해 루트 노드로부터의 경로 쌍을 구하고 각각의 경로 쌍에 대해 빈도 지시자 노드간의 대응관계를 생성한다. 두 번째 단계에서는 이전 단계에서 생성된 대응관계를 기반으로 타겟 스키마를 하향식 깊이 우선 탐색 하면서 XSLT 스크립트를 생성한다. 각 단계에 대한 자세한 설명은 다음과 같다.

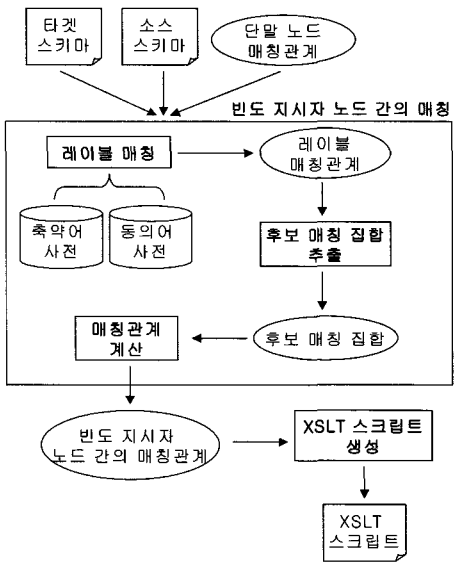
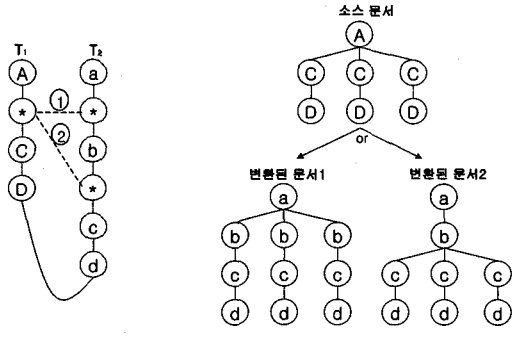


그림 5 XSLT 스크립트의 생성 과정

3.1 빈도 지시자 노드 간의 매칭

일반적으로 두 스키마 간의 단말 노드 매칭만으로는 소스 스키마로부터 생성되는 모든 XML 문서들에 대해 올바른 변환을 수행하는 XSLT 스크립트를 생성할 수 없다. 이것은 스키마에 포함된 빈도 지시자와 선택 연산자에 의해 동일한 스키마로부터 다양한 구조를 가진 XML 문서들이 생성될 수 있기 때문이다. 예를 들어, 그림 6(a)에서와 같이 두 스키마 T_1, T_2와 단말 노드 매칭만이 주어진 경우, T_1으로부터 생성된 그림 6(b)의 소스 문서는 두 개의 서로 다른 구조를 갖는 문서로 변

환될 수 있다. 따라서 단말 노드 매칭만으로는 이러한 변환 결과 중에서 어떤 것이 올바른 것인지 판단할 수가 없다. 생성되는 XML 문서의 구조를 유일하게 결정하기 위해서는 빈도 지시자 노드 또는 선택 연산자 간의 대응관계와 같은 부가적인 정보가 필요하다.



(a) 스키마 T1, T2 (b) T1으로부터 생성된 소스 문서와 변환된 문서들의 예

그림 6 단말 노드 매칭만 주어진 경우 가능한 변환의 예

그림 6(a)에서 ① 또는 ②와 같은 빈도 지시자 노드 간의 대응관계가 주어졌다면 소스 문서는 대응관계 ①을 통해 변환된 문서1로, 대응관계 ②를 통해 변환된 문서2로 유일하게 변환될 수 있다.

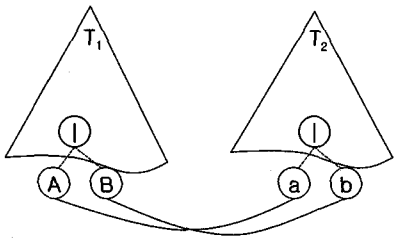


그림 7 선택 연산자 노드가 포함된 스키마의 예

한편 선택 연산자의 경우, 대응관계를 계산하지 않아도 주어진 단말 노드 매칭 정보를 활용하여 선택 연산자에 의해 생성되는 선택적인 구조를 지원하는 XSLT 스크립트를 생성할 수 있다. 예를 들어, 그림 7과 같이 단말 노드 간의 매칭이 주어진 경우, 소스 XML 문서에서 노드 A가 존재하면 노드 a를, 노드 B가 존재하면 노드 b를 타겟 XML 문서에 생성하면 된다. 따라서 제안된 방법은 소스와 타겟 스키마에서 빈도 지시자 간의 대응관계만 계산하고, 계산된 대응관계와 단말 노드 매칭 정보를 활용하여 소스 스키마로부터 생성되는 다양한 구조의 XML 문서들을 변환할 수 있는 XSLT 스크립트를 생성한다.

제안된 방법은 대응 경로 간의 비교를 통해 빈도 지시자 노드 간의 매칭을 생성한다. 먼저 두 스키마에서 주어진 각각의 단말 노드 매칭에 대해 대응 경로를 구하고 대응 경로 간의 레이블 매칭을 찾는다. 이 때 조상 순서[2]를 위배하는 매칭은 생성하지 않는다. 두 스키마는 같은 도메인(domain)에 속하고 루트 노드는 서로 매칭된다고 가정한다. 따라서 대응 경로 상에 존재하는 노드들 중에 레이블이 같거나 유사한 노드는 의미상 서로 대응하는 노드로 간주될 수 있다.

한편 두 스키마가 모두 재귀적인 구조를 포함하고 있는 경우 예외적인 처리 과정이 필요하다. 제안된 방법은 기본적으로 단말 노드 간의 일대일 매칭을 입력으로 받아들인다. 하지만 재귀적인 구조들을 포함하는 스키마의 경우 단말 노드 간의 다대일 또는 일대다 매칭을 허용한다. 예를 들어 제안된 방법은 그림 8의 subpart와 part처럼 재귀 단말 노드 간의 매칭을 입력으로 받는다. 또한 주어진 재귀 노드 간의 매칭을 통해 참조되는 노드 간의 매칭이 자동으로 계산된다. 결과적으로 T_2의 노드 A와 노드 B는 각각 T_1의 노드 a와 a′ 그리고 노드 b와 b′과 매칭된다.

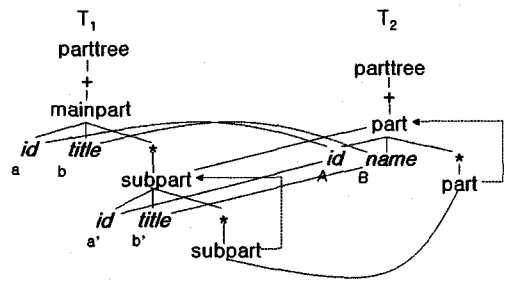


그림 8 재귀적인 구조를 포함하는 스키마의 예

그림 8의 T_2와 같이 단말 노드가 두 개 이상의 노드와 대응하는 경우 XSLT 스크립트 생성 시에 어떤 노드로부터 값을 복사해야 하는지 모호한 경우가 발생한다. 이러한 문제를 해결하기 위해 제안된 방법은 두 개 이상의 노드와 대응하는 단말 노드를 갖는 스키마 트리를 모든 노드가 일대일 대응관계를 갖도록 확장한다. 트리의 확장은 모든 단말 노드가 일대일 대응관계를 가질 때까지 반복된다. 예를 들어, 그림 8에서 T_2는 노드 A와 노드 B가 각각 두 노드와 대응관계를 갖는다. 이러한 경우 그림 9와 같이 참조하는 단말 노드를 한 단계 확장하여 노드 a는 노드 A와, 노드 b는 노드 B와, 노드 a′은 노드 A′와 그리고 노드 b′은 노드 B′과 대응하도

2) 임의의 두 매칭 $[i_1, j_1]$와 $[i_2, j_2]$에 대해, 노드 j_1이 노드 j_2의 조상인 경우 노드 i_1은 반드시 노드 i_2의 조상이 된다.

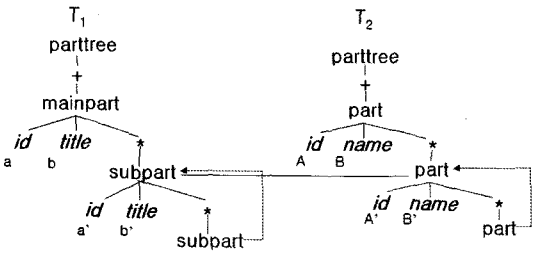


그림 9 재귀적인 구조의 지원을 위한 스키마 트리 확장의 예

록 만들고 참조 관계를 수정한다. 이후 확장된 스키마 쌍에 대해 일반적인 빈도 지시자 노드 매칭과 XSLT 스크립트 생성 과정이 적용된다. XSLT 스크립트 생성 단계에서는 'for-each'와 부가적인 템플릿을 통해 스키마에 포함된 재귀적인 구조의 변환을 지원한다.

레이블 매칭 과정에서 워드넷(WordNet)과 같은 동의어 및 유사어 사전을 사용하여 레이블이 완전히 일치하는 노드뿐만 아니라 유사한 의미를 가지는 노드들 간에도 레이블 매칭을 생성한다. 레이블 매칭을 위한 레이블 간의 어휘 유사도는 수식(1)에 의해 계산된다. 각 레이블을 대문자나 특수문자를 기준으로 토큰화하여 각 토큰 사이의 유사도를 계산한다. 토큰 간의 유사도는 두 토큰이 동일한 경우 1, 유사한 경우 0.8, 그 외의 경우는 0으로 계산한다. 레이블 사이의 유사도는 토큰들 사이의 유사도를 이용해 토큰 사이의 유사도의 합을 전체 토큰수로 나눈 값이 된다.

$$\text{어휘 유사도}(N_s, N_t) = \frac{2\times\sum\text{유사도}(N_{si}, N_{tj})}{|N_s| + |N_t|} \quad (1)$$

N_{si} : 소스 레이블을 구성하는 토큰, $1\le i\le n$ N_{tj} : 타겟 레이블을 구성하는 토큰, $1\le j\le m$

$|N_s|$: 소스 레이블을 구성하는 토큰의 수, $|N_t|$: 타겟 레이블을 구성하는 토큰의 수

각각의 대응 경로는 레이블 매칭을 통하여 그림 10과 같이 쌍을 이루는 여러 개의 부분으로 분할된다. 매칭된 노드 사이의 대응관계를 갖지 않는 부분 경로(sub path)는 서로 쌍을 형성한다. 부분 경로는 임의의 경로에서 대응관계를 갖는 두 노드 사이에 존재하는 대응관계를 갖지 않는 노드들의 순차적인 집합이다.

레이블 매칭에 의해 분할된 각각의 부분 경로에 대해 후보 매칭 집합(Candidate Matching Set)이 계산된다. 후보 매칭 집합은 그림 10과 같이 레이블 매칭을 통해 계산된 부분 경로에서 빈도 지시자 노드만을 포함한다.

각각의 경로에서 레이블 매칭을 통해 부분 경로를 계산하고 이로부터 후보 매칭 집합을 추출하는 것은 중간 노드 매칭의 정확성을 높이고 불필요한 계산을 줄이는

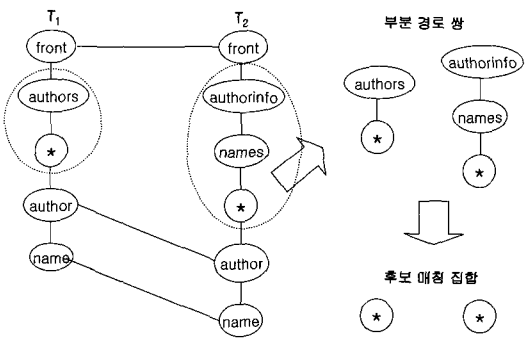


그림 10 레이블 매칭에 의한 경로 분할과 후보 매칭 집합 생성의 예

데 기여한다. 빈도 지시자 간의 매칭을 위해서 매번 전체 경로를 비교하는 비용을 줄일 수 있게 해주며 대응관계 계산이 필요한 구간을 줄임으로서 좀 더 정확한 매칭을 찾을 수 있게 된다.

각각의 부분 경로 쌍으로부터 후보 매칭 집합이 추출되면 두 집합 간의 대응관계를 계산한다. 주어진 두 집합의 원소들 중에서 제안된 구조 유사도가 가장 큰 두 노드를 선택하고 매칭을 생성한다. 이때 구조 유사도는 수식(2)를 통해 계산된다. 유사도가 같은 경우에는 단말 노드의 개수가 많은 매칭을 선택한다. 구조 유사도 계산에서는 Kuikka 등의 방법에서 정의한 TAN 집합 개념을 사용한다. TAN(n)은 중간 노드 n을 루트 노드로 갖는 서브 트리에서 대응관계를 갖는 단말 노드들의 집합을 의미한다.

$$\text{구조 유사도}(n, n') = \frac{\text{TAN}(n)\text{과 TAN}(n')\text{간의 대응되는 원소의 개수}}{(n\text{의 단말 노드의 개수} + n'\text{의 단말 노드의 개수})/2} \quad (2)$$

가장 적합한 하나의 매칭이 선택되면 선택된 매칭을 기준으로 후보 매칭 집합을 두 개의 후보 매칭 집합으로 분할한 후, 분할된 각각의 후보 매칭 집합에 대하여 매칭을 계산한다. 이러한 과정은 모든 매칭을 찾을 때까지 재귀적으로 반복된다. 여기서 후보 매칭 집합을 상위와 하위 두개로 나누는 것은 조상 순서를 위배하는 대응관계를 생성하지 않도록 한다.

그림 11은 두 스키마에 대하여 제안된 알고리즘을 적용하여 중간 노드 매칭을 생성한 예이다. 각각의 단말 노드 매칭에 대해 대응경로를 계산하고 대응경로 간의 빈도 지시자 노드 매칭 과정을 거치면 총 세 개의 빈도 지시자 간의 매칭이 생성된다. 빈도 지시자 노드 간의 매칭을 생성하는 대응경로는 {경로(***name***), 경로(**name**)}, {경로(***email***), 경로(**email**)}, {경로(***univ***), 경로(**univ**)}이다.

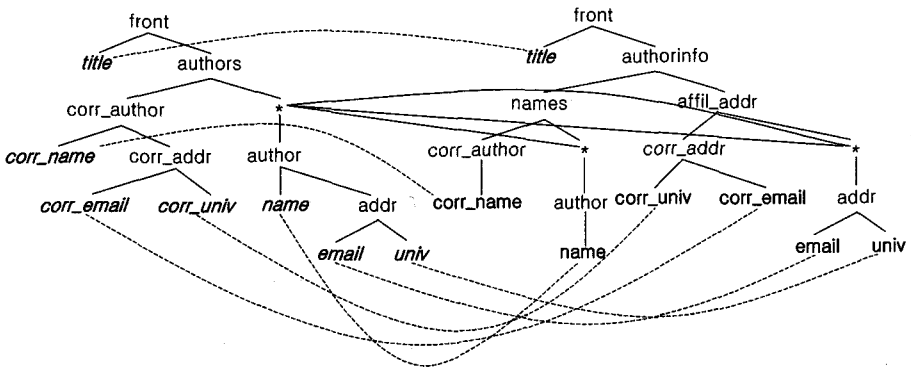


그림 11 빈도 지시자 노드 간의 매칭의 예

3.2 XSLT 스크립트 생성

제안된 방법은 소스 스키마의 루트 노드에 대한 템플릿을 생성하고, 타겟 스키마를 깊이 우선 탐색하면서 각각의 노드 타입에 따라 적절한 XSLT 스크립트를 생성한다. 생성되는 XSLT 스크립트는 두 스키마가 재귀적인 구조를 포함하지 않는 경우에는 소스 스키마 의 루트 노드에 대한 템플릿 하나로 구성된다. 한편 두 스키마가 재귀적인 구조를 포함하고 두 스키마에서 참조를 포함하는 재귀 단말 노드가 서로 매칭되는 경우, 참조를 포함하는 노드에 대해 새로운 템플릿을 생성하고 이를 통해 재귀적인 구조의 변환을 지원한다.

본 단계를 통해 생성되는 XSLT 스크립트는 이전 단계에서 계산된 매칭의 수와 관계없이 서로 매칭되는 재귀적인 구조의 수에만 비례하는 템플릿을 포함한다. 따라서 생성되는 매칭이나 부 구조의 수만큼의 템플릿을 포함하는 기존 방법들에 비해 적은 수의 템플릿을 가지며 감소된 템플릿 호출로 인해 변환 속도의 향상을 기대할 수 있다. 각각의 노드 타입에 따라 XSLT를 생성하는 방법은 다음과 같다.

선택 연산자 노드의 경우, 자식 노드들 중 오직 한 노드만 문서에서 나타날 수 있다. 선택 연산자 노드의 자식 노드 n′에 대해 TAN(n′)의 임의의 원소와 매칭되는 소스 스키마의 단말 노드가 소스 문서에서 나타난다면 노드 n′은 타겟 문서에서 나타날 수 있다.

예를 들어, 그림 12의 T_2에서 노드 a와 노드 b는 선택 연산자 노드의 자식 노드이고 TAN(a)는 {e, f}, TAN(b)는 {g, h}이다. 이 때 소스 문서에서 E 혹은 F가 나타나고 G와 H 노드가 나타나지 않는다면, 노드 a는 타겟 문서에서 반드시 나타나야 한다. 한편 소스 문서에서 E, F 그리고 G가 나타나는 경우나 E, G, 그리고 H가 나타나는 경우처럼 선택 연산자 노드의 자식 노드들 중 두 개 이상의 노드에 대해 어떤 노드를 선택해야 할지 모호한 경우가 발생할 수 있다. 이러한 경우, 어떤 노드를 선택하는 것이 타당한 것인지 자동으로 판단하기는 어렵다. 따라서 제안된 방법에서는 이러한 경우가 발생하면 문서에서 나타나는 순서에 따라 먼저 나타나는 노드를 선택한다. 전술한 조건들은 XSLT 명령어 'choose'와 'when'을 사용하여 기술된다.

빈도 지시자 노드의 경우, '?'인 경우와 그 밖의 경우로 나뉜다. 빈도 지시자가 '?'인 경우, 매칭되는 빈도 지시자 노드의 자식 노드가 소스 문서에 존재한다면 타겟 문서로 변환될 것이다. XSLT 스크립트에서 이러한 조건은 'if'를 사용하여 기술된다.

그 밖의 경우, for-each 태그를 사용하여 문서에서 반복되는 노드들을 변환할 수 있다. 단, 타겟 스키마에서 하나의 빈도 지시자 노드가 두 개 이상의 소스 노드와 매칭되는 경우, 주의가 필요하다. 예를 들어, 그림 13에서 T_2에 의하면 두 노드 a, b의 수는 동일하다. 그런데 T_1에 의하면 두 노드A와 B의 개수는 소스 문서에서

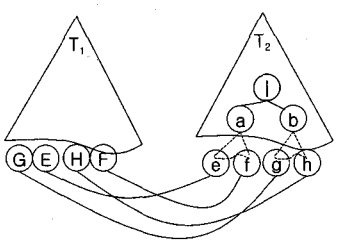


그림 12 선택 연산자 노드의 예

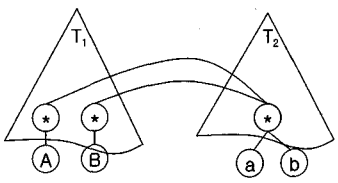


그림 13 빈도 지시자 노드가 두 개 이상의 노드와 매칭되는 경우의 예

```
<xsl:transform version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="xml" indent="yes" encoding="UTF-8"/>
<xsl:template match="front">
<front>
  <title>
    <xsl:value-of select = "title"/>
  </title>
  <authorinfo>
    <names>
      <corr_author>
        <corr_name>
          <xsl:value-of select = "authors/corr_author/corr_name"/>
        </corr_name>
      </corr_author>
      <xsl:for-each select = "authors/author">
        <author>
          <name>
            <xsl:value-of select = "name"/>
          </name>
        </author>
      </xsl:for-each>
    </names>
    <affil_addr>
      <corr_addr>
        <corr_univ>
          <xsl:value-of select = "authors/corr_author/corr_addr/corr_univ"/>
        </corr_univ>
        <corr_email>
          <xsl:value-of select = "authors/corr_author/corr_addr/corr_email"/>
        </corr_email>
      </corr_addr>
      <xsl:for-each select = "authors/author/addr">
        <addr>
          <univ>
            <xsl:value-of select = "univ"/>
          </univ>
          <email>
            <xsl:value-of select = "email"/>
          </email>
        </addr>
      </xsl:for-each>
    </affil_addr>
  </authorinfo>
</front>
</xsl:template>
</xsl:transform>
```

그림 14 그림 11의 스키마 간의 변환을 지원하는 XSLT 스크립트의 예

서로 다를 수 있다. 이러한 경우 제안된 방법은 정보의 손실을 줄이기 위해 A와 B 중에서 더 많이 반복되는 노드를 택하여 그 수만큼 a와 b를 생성한다. 기존의 방법들은 이와 같은 경우에 유사도 비교를 통하여 더 유사한 하나의 노드만을 선택함으로서 선택되지 않는 노드의 정보가 손실되었다.

제안된 알고리즘을 적용하여 그림 11의 스키마 간의 변환을 지원하는 XSLT 스크립트를 생성하면 그림 14와 같다. 소스 스키마와 타겟 스키마가 재귀적인 구조를 포함하지 않기 때문에 생성된 XSLT 스크립트는 소스 스키마의 루트 엘리먼트에 대한 단일 템플릿으로 구성된다.

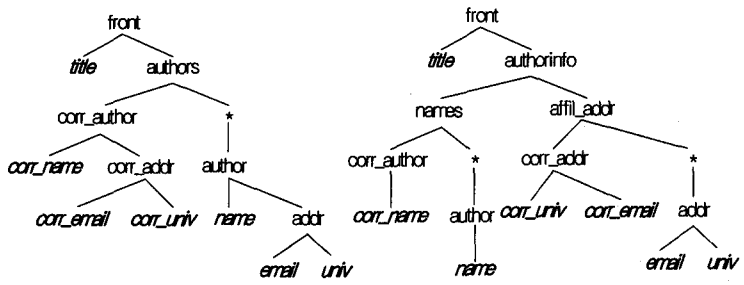
4. 실험결과

본 장에서는 제안된 방법의 성능을 평가하기 위한 실험 환경과 성능 분석에 대해 기술한다. 실험 환경에서는 실험에 사용된 스키마와 XSLT 처리기에 대해 간략히 소개하고, 성능 분석에서는 기존 방법들에 의해 생성된 XSLT 스크립트와 제안된 방법에 의해 생성된 XSLT 스크립트의 변환 수행 속도를 비교 및 분석한다.

4.1 실험 환경

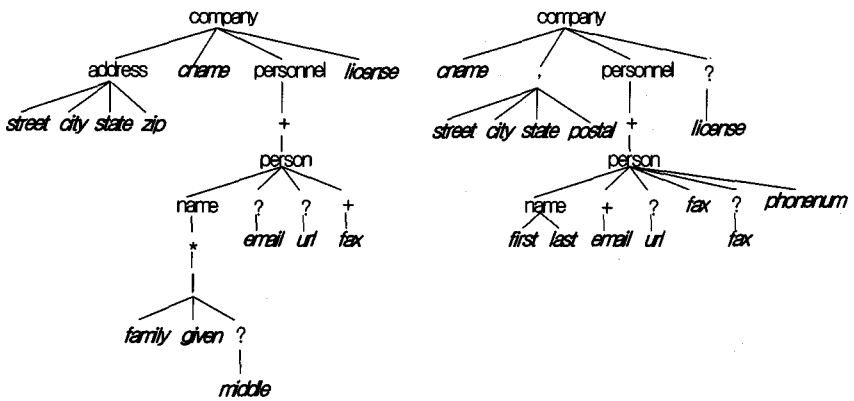
제안된 방법의 성능을 평가하기 위하여 Leinonen의 연구에서 예제로 사용한 스키마와 Su 등의 연구에서 예제로 사용한 스키마를 각각 스키마쌍1과 스키마쌍2로 하여 실험하였다. 스키마쌍1과 스키마쌍2를 제안된 문서 모델로 표현한 것은 각각 그림 15와 그림 16과 같다. 레이블이 같은 단말 노드는 서로 매칭된다고 가정하고 제안된 방법과 기존 방법을 적용하여 각각의 XSLT 스크립트를 생성하였다.

제안된 방법은 템플릿 수와 템플릿 호출의 수를 줄임으로서 생성되는 XSLT 스크립트의 수행 속도를 향상시키는데 목적을 두고 있다. 따라서 다양한 방법들에 의해 생성된 XSLT 스크립트의 변환 수행시간이 문서의



(a) 소스 스키마 (b) 타겟 스키마

그림 15 실험 데이타 스키마쌍1



(a) 소스 스키마 (b) 타겟 스키마

그림 16 실험 데이타: 스키마쌍2

표 2 실험에 사용된 XSLT 처리기

XSLT 처리기	특징	출처
Xalan-Java 2.6.0	JAXP(Java API for XML Processing) 1.2의 XSLT 처리기이다. XSLT 1.0과 XPath 1.0을 지원하며 SAX 2와 DOM level 2를 기반으로 한다.	http://xml.apache.org/xalan-j/
Saxon-B 8.1.1	Michael Kay에 의해 개발된 open-source XSLT 처리기이다. XSLT 2.0, XPath 2.0 그리고 XQuery 1.0을 반영한다.	http://saxon.sourceforge.net/
XSLTCommand	Microsoft의 .NET 프레임웍을 위한 XSLT 처리기로서 XSLT 1.0을 지원한다.	http://msdn2.microsoft.com/library/9ewd6765.aspx

크기에 따라 어떻게 변화하는지를 측정하였다. 이를 위해서 주어진 소스 스키마를 따르는 서로 다른 크기의 문서 6개를 임의로 작성하였고, 각각의 방법을 적용하여 생성된 XSLT 스크립트를 개별 문서마다 100번 실행한 후 수행 속도를 계산하였다.

실험은 펜티엄IV 2.4GHz와 512M 메인 메모리를 가진 Windows 2000 환경에서 표 2와 같이 다수의 대표적인 XSLT 처리기를 사용하여 수행하였다.

4.2 성능 분석

스키마쌍1에 대한 실험 결과는 그림 17, 그림 19 그리고 그림 21과 같고 스키마쌍2에 대한 실험 결과는 그림 18, 그림 20 그리고 그림 22와 같다.

제안된 방법이 생성한 XSLT 스크립트는 모든 크기의 문서와 XSLT 엔진에서 기존의 방법과 동일한 변환 결과를 보이면서 가장 빠른 수행 속도를 보였다. 특히 기존 방법들에 비해 평균적으로 20% 정도의 속도 향상이 있었다. 하지만 기존의 방법에 의해 생성된 XSLT 스크립트들은 XSLT 엔진에 따라 수행속도의 편차가 심하였다. 표 3은 스키마쌍1에서 각각의 방법들에 의해 생성된 XSLT 스크립트의 특징, 템플릿 수, 그리고 템플릿 호출 회수를 비교한 것이다.

본 논문은 템플릿 호출 회수가 줄어들면 XSLT 스크립트의 수행 속도가 향상된다는 점에 착안하고 있다. 따라서 표 3에서 템플릿 호출 회수에 의하면 변환 수행

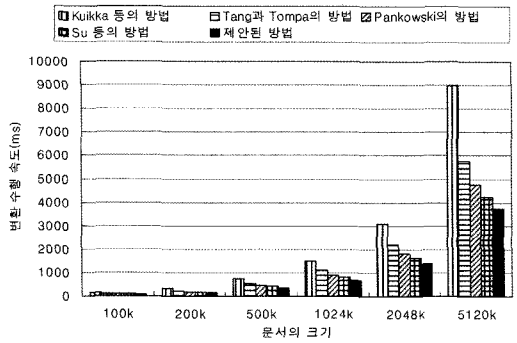


그림 17 스키마씽1과 Xalan 처리기를 사용한 실험 결과

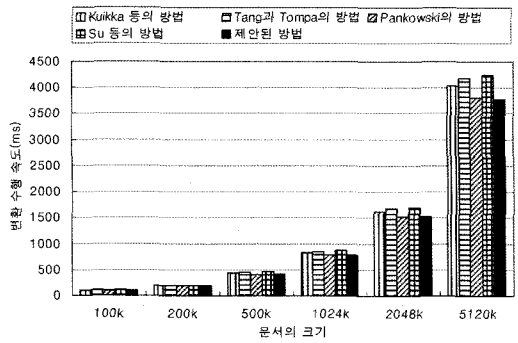


그림 18 스키마씽2와 Xalan 처리기를 사용한 실험 결과

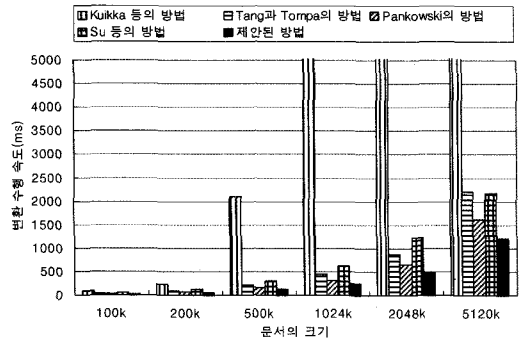


그림 19 스키마씽1과 Saxon 처리기를 사용한 실험 결과

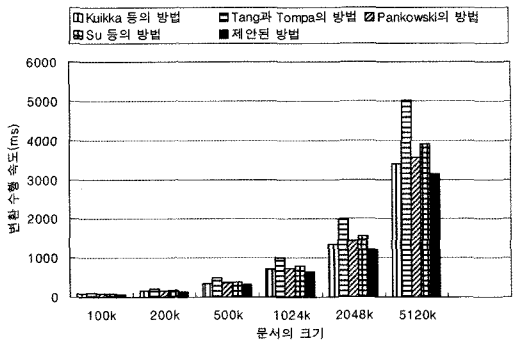


그림 20 스키마씽2와 Saxon 처리기를 사용한 실험 결과

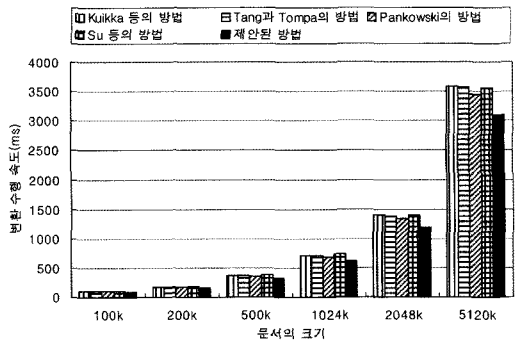


그림 21 스키마씽1과 XSLTCommand 처리기를 사용한 실험 결과

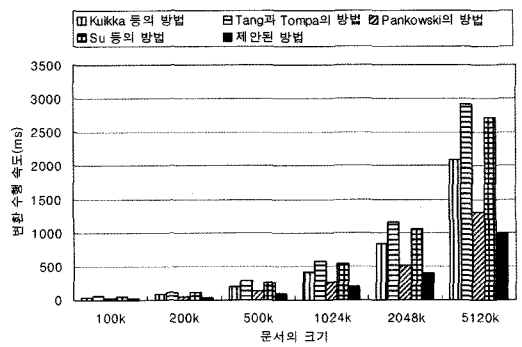


그림 22 스키마씽2와 XSLTCommand 처리기를 사용한 실험 결과

표 3 생성된 XSLT 템플릿 비교

방법	특징	템플릿 수	템플릿 호출 회수
Kuikka 등의 방법	· XPath 표현식에서 descendant 축(axis) 사용 · 단말 노드 값 복사를 위해 apply-template 사용	11	17+8n+
Tang과 Tompa의 방법	· 구조의 복사를 위해 copy-of 태그 사용	18	8+3n
Pankowski의 방법	· 구조의 복사를 위해 copy-of 태그 사용 · element 태그를 사용하여 엘리먼트 생성	12	10+2n
Su의 방법	· 단말 노드 값 복사를 위해 apply-template 사용	13	18+7n
제안된 방법	· 템플릿을 호출하지 않음	1	0

+n은 소스 XML 문서에서 엘리먼트 AUTHOR의 빈도수

속도는 제안된 방법, Pankowski의 방법, Tang과 Tompa의 방법, Su의 방법 그리고 Kuikka 등의 방법의 순서로 나타나야 한다. 하지만 실험 결과에서는 이러한 순서가 지켜지지 않는데 이는 템플릿 호출 회수 이외에 각각의 방법에 의해 생성되는 XSLT 스크립트가 가지는 특징과 각 XSLT 엔진의 최적화(optimization) 기술의 차이 때문이다.

Kuikka 등의 방법은 다른 방법들이 XSLT 스크립트에서 특정 노드를 지정하는데 XPath[11]의 자식 축(child axis)만을 사용하는 것과 달리 자손 축(descendant axis)을 사용한다. 자손 축을 사용하는 경우 XPath 표현식에서 지정하는 노드를 찾기 위해서는 현재 노드를 루트 노드로 하는 서브 트리의 모든 노드를 검색해야만 한다. 특히 자손 축을 사용하는 XPath 표현식이 반복적으로 계속 수행되어야 하는 경우나 현재 노드와 찾고자 하는 노드셋(node set)과의 깊이 차이가 크면 클수록 수행 속도는 저하될 수밖에 없다. 자손 축의 사용으로 인해 Kuikka 등의 방법에 의해 생성되는 XSLT 스크립트는 다른 방법들에 의해 생성된 XSLT 스크립트보다 대체적으로 느린 수행 속도를 보였으며 특히 스키마 쌍1과 XSLTCommand 처리기를 사용한 실험에서 가장 느린 수행 속도를 보였다.

Tang과 Tompa의 방법은 소스 스키마와 타겟 스키마에서 동일한 부 구조에 대해 copy-of 태그를 사용하여 소스 문서의 정보를 타겟 문서로 복사한다. 그런데 copy-of의 사용은 부 구조들에 포함된 각각의 노드에 대한 태그를 XSLT 스크립트에서 직접 출력하고 단말 노드 값을 value-of나 apply-templates를 사용하여 복사하는 것보다 수행 속도가 느리다. 따라서 copy-of를 사용하여 부 구조를 복사하기 때문에 템플릿의 수와 호출 회수는 적지만 결과적으로 Su 등의 방법에 의해 생성된 XSLT 스크립트보다 수행 속도가 느리다.

Pankowski의 방법은 Tang과 Tompa의 방법과 마찬가지로 부 구조를 복사하는데 copy-of를 사용한다. 또한 노드의 태그를 XSLT 스크립트에서 직접 출력하지 않고 연산자 element를 사용하여 태그를 생성한다. Element는 동적으로 노드의 태그를 사용할 수 있지만 직접 출력하는 것에 비해 속도가 느리다. Copy-of와 element의 사용으로 인해 템플릿 호출 회수가 적음에도 불구하고 수행 속도가 느려지는 단점을 갖는다.

각각의 방법에 의해 생성된 XSLT 스크립트에서 자손 축과 element를 제거하고, copy-of를 value-of로 대체한 후 같은 방법으로 실험한 결과, 예상대로 템플릿 호출 회수가 적은 XSLT 스크립트가 실험에 사용된 문서에서 빠른 수행 속도를 보였다.

5. 결론 및 향후연구 방향

본 논문에서는 단말 노드 간의 일대일 대응관계가 주어져 있다고 가정하고 변환 수행 속도가 빠른 XSLT 스크립트를 생성하는 알고리즘을 제안하였다. 주어진 단말 노드 대응관계를 사용하여 중간 노드 간의 대응관계를 생성한 후 이를 바탕으로 XSLT 스크립트를 생성한다. 이때, 템플릿의 호출 회수를 줄여 변환 수행속도를 향상시키기 위하여 적은 수의 템플릿을 사용하여 XSLT 스크립트를 생성한다.

기존의 방법들은 중간 노드 간의 대응관계나 대응규칙의 수와 비례하는 템플릿을 갖는 XSLT 스크립트를 생성한다. 그러나 제안된 방법은 대응관계의 수와 관계없이 매칭된 재귀 노드의 개수에만 비례하는 적은 수의 템플릿을 사용하여 XSLT 스크립트를 생성한다. 같은 스키마를 따르는 서로 다른 크기의 문서들을 대상으로 한 실험에서 제안된 방법에 의해 생성된 XSLT 스크립트는 기존의 방법에 의한 XSLT 스크립트보다 평균적으로 20% 정도 빠른 변환 수행속도를 보였다. 따라서 제안된 방법은 XML 문서의 빠른 변환을 지원하는 XSLT 스크립트를 생성한다고 할 수 있다.

한편, 제안된 방법은 단말 노드 간의 일대일 대응관계를 가정하고 있다. 일반적인 문서에서는 병합(merge) 또는 분할(split)과 같은 다대일(many-to-one)이나 일대다(one-to-many)의 대응관계가 존재할 수 있는데 현재는 이러한 문서들에 대해 올바른 변환을 수행하지 못한다. 본 논문은 제한적인 문서들에 한해서 기존의 방법들보다 더욱 빠른 수행속도를 갖는 XSLT 스크립트를 생성하는데 초점을 맞추었다.

본 논문에서는 제안된 방법의 성능을 평가하기 위해서 기존 연구에서 사용한 스키마를 대상으로 실험을 수행하였다. 제안된 방법의 성능을 정확하게 검증하기 위해서는 B2B나 바이오 인포매틱스 분야에서 실제로 사용되고 있는 복잡하고 다양한 스키마를 대상으로 한 실험이 필요하다. 그러나 이러한 스키마들은 일반적으로 전술한 다대일 혹은 일대다 대응관계를 포함하고 있기 때문에 현재의 방법을 그대로 적용하여 실험을 수행하기 어렵다. 향후 다대일이나 일대다의 대응관계에 대해 효율적인 XSLT 스크립트를 생성할 수 있는 방법을 연구할 것이며 이를 바탕으로 복잡하고 다양한 스키마에 대한 실험을 통해 제안된 방법의 성능을 보다 정확하게 검증할 것이다.

참 고 문 헌

[1] World Wide Web Consortium, Extensible Markup Language (XML) 1.0(Third Edition), W3C Recommen-

dation, http://www.w3c.org/TR/REC-xml, 2000.
[2] World Wide Web Consortium, XSL Transformations (XSLT) 1.0, W3C Recommendation, http://www.w3c.org/TR/1999/REC-xslt-19991116, 1999.
[3] Jun-Seung Lee and Kyong-Ho Lee, "XML Schema Matching Based on Incremental Ontology Update," Proc. Int'l Conf. Web Information Systems Engineering, Vol. 3306, pp. 608-618, 2004.
[4] Paula Leinonen, "Automating XML Document Structure Transformations," Proc. ACM Symposium Document Engineering, pp. 26-28, 2003.
[5] Eila Kuikka, Paula Leinonen, and Martti Penttonen, "Towards Automating of Document Structure Transformations," Proc. ACM Symposium Document Engineering, pp. 103-110, 2002.
[6] Tadeusz Pankowski, "A high-level language for specifying XML data transformations," Proc. Conf. Advances in Databases and Information Systems, pp. 22-25, 2004.
[7] Tadeusz Pankowski, "Specifying transformations for XML data," Workshop. Emerging Database Research in Europe. Int'l Conf. Very Large Data Bases, 2003.
[8] Xuerong Tang and Frank Wm. Tompa, "Specifying Transformations for Structured Documents," Proc. Int'l Workshop the Web and Databases, pp. 67-72, 2001.
[9] 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.
[10] World Wide Web Consortium, XML Path Language(XPath) 1.0, W3C Recommendation, http://www.w3.org/TR/xpath.

신 동 훈

2003년 연세대학교 컴퓨터과학과 졸업(학사). 2005년 연세대학교 컴퓨터과학과 졸업(석사). 2005년~현재 연세대학교 컴퓨터과학과 박사과정. 관심분야는 Internet Computing, Service-Oriented Computing, Multimedia Document Engineering

이 경 호

1995년 연세대학교 전산과학과 졸업(학사). 1997년 연세대학교 컴퓨터과학과 졸업(석사). 2001년 연세대학교 컴퓨터과학과 졸업(박사). 2001년 National Institute of Standard and Technology(NIST) 객원연구원. 2002년~현재 연세대학교 컴퓨터과학과 조교수. 관심분야는 Internet Computing, Service-Oriented Computing, Multimedia Document Engineering