# 메타데이터 레지스트리를 이용한 XML-문서 교환 방법

홍종하[0] , 양유승 , 나홍석 , 백두권
고려대학교 컴퓨터학과 소프트웨어 시스템 연구실
{sam, yys, nhs, baik}@swsys2.korea.ac.kr

An XML-Documents Exchanging Method Using A Metadata Registry

Jong-Ha Hong[0] , Yoo-Seung Yang , Hong-Seok Na , Doo-Kwon Baik
Software System Lab., Dept. of Computer Science & Engineering, Korea University

요 약

웹 기반의 분산 환경에서 데이터를 공유, 사용하려는 노력은 끊임 없이 계속되어 왔다. 기존의 HTML 문서를 이용할 경우에는 그 언어자체가 가지고 있는 한계성 때문에 효과적으로 문서를 공유하기가 어렵다. 이에 대한 대안으로 XML을 이용한 문서 교환 방법이 제시되고 있다. 하지만 서로 다른 DTD를 기반으로 작성된 XML문서를 교환할 경우에는 문제가 발생하게 된다. DTD가 서로 다른 사용자에 의해서 작성되었기 때문에 XML 문서 내의 태그 뿐만 아니라 문서가 가지고 있는 그 구조 또한 서로 상이하게 된다.

본 논문에서는 상이한 DTD를 기반으로 작성된 XML문서를 교환할 경우에 고려 해야 하는 XML 문서의 구조적 상이성의 예를 보여주고 이에 대한 해결 알고리즘을 제시한다. 문서 구조의 상이성은 적절한 매핑 테이블과 트리 구조를 이용한 태그 변환 방법을 이용하여 해결할 수 있다. 데이터 레지스트리와 본 논문에서 제안한 문서의 구조와 태그 변환 방법을 사용하면 XML 문서를 효과적으로 교환 할 수 있다.

## 1. 서론

지식과 정보 교류의 기반이 웹으로 옮겨지면서 현재 거의 모든 정보가 웹을 이용해서 전달되고 있다. 이러한 정보들은 대부분이 웹브라우저를 통해서 HTML의 형태로 보여진다. 이렇게 HTML로 보여지는 정보들은 대부분 데이터베이스 내에 저장되어 있고, ASP나 JSP 등과 같은 적절한 스크립트 기술을 이용하여 HTML문서의 형태로 보여진다. 하지만 HTML이 가진 확장의 한계성, 관계성 지원 불가능 등의 단점으로 인해 서로 다른 어플리케이션 및 호환이 되지 않는 데이터베이스간의 정보 전달이 효율적으로 이루어질 수 없었다. 이러한 단점을 극복하기 위해서 국제 표준 마크업 언어인 SGML의 일부를 계승한 XML이 나오게 되었다[2].

XML은 확장성, 관계성이 좋기 때문에 새로운 정보 공유 환경의 매체로 자리잡아가고 있다. 하지만 문서의 구조를 파악하는 DTD가 서로 다른 도메인에서 작성되기 때문에 문서 유통을 제대로 하기 어렵다. 이러한 문제는 문서 유통시에 데이터레지스트리를 참조 하는 것으로 해결 할 수 있다[2]. 본 논문에서 제안한 XML 문서 교환 방법은 문서 교환 시에 발생하는 문제점을 해결함으로써 문서 유통의 효과를 높인다.

## 2. 관련연구

### 2.1 XML과 DTD

XML(extensible markup language)은 텍스트 기반의 구조화된 언어로서 HTML처럼 역시 마크업 태그를 사용한다. 하지만 HTML과 달리 XML은 저작자 자신이 태그를 정의 할 수 있도록 해준다.

이렇게 함으로써 XML 태그는 그 내용에 대한 표현 보다는 실제 내용을 묘사 할 수 있다[3]. XML은 HTML과 달리 여러 어플리케이션을 지원한다. HTML만으로는 여러 어플리케이션을 지원하기가 힘들다. 따라서, 스크립트 언어의 사용이 늘고 있는 것이다. HTML은 너무 제한적이다. XML은 SGML의 일반적인 성격을 받았지만, 실제로 확장이 가능한 유연성이 추가되었다

DTD는 XML 문서의 논리적 구조를 표현하는 문서이다. DTD는 어떤 요소가 반드시 있어야 하고, 어떤 것이 옵션인지, 요소의 속성 서로의 관계를 구조화 하는 방법 등을 말해준다. HTML에 DTD가 하나인 반면, XML에는 어플리케이션별로 DTD를 별도로 만들 수 있다. 그렇게 해서 어플리케이션을 위해 작성한 XML 문서의 내용과 구조를 검사하는 처리를 제어할 수 있다.

### 2.2 데이터 레지스트리

데이터 레지스트리는 데이터에 대한 유일한 식별 및 등록, 서비스 등의 역할을 하는 기능을 가지고 있는 것으로 데이터의 사용자와 소유자 모두에게 데이터에 대한 의미와, 표현, 식별에 대하여 명확한 이해를 돕는다[2]. 데이터 레지스트리에 저장되는 데이터를 기술하는 단위가 데이터 요소이다. 데이터 요소는 데이터의 기본 단위로 데이터에 대한 식별, 설명, 표현을 명세하는 단위이다[1]. ISO 11179에 정의된 데이터 요소의 구조를 살펴보면 <그림 1>과 같이 객체 클래스(Object Class)와 속성(Property), 표현(Representation)의 3부분으로 구성되어 있다.

| Object class | Property | Representation |
|---|---|---|
| Teacher | Age | Integer |
| Employee | Income | Money $ |
| Manager | Name | Char(10) |
| Etc | Etc | Etc |

<그림 1> 데이터 요소의 구조

객체 클래스(Object Class)는 사람, 자동차 등과 같이 표현하고자 하는 실제 대상을 나타낸다. 속성(Property)은 나이, 이름 등과 같이 객체 클래스의 모든 멤버에게 공통되는 특징이다. 표현(Representation)은 데이터 요소를 표현하는 값의 도메인, 데이터 타입을 기술한다[1].

데이터 레지스트리 내의 각 데이터 요소는 모두 유일한 아이디를 가지고 있기 때문에 데이터 레지스트리를 사용하는 모든 사용자나 프로그램이 이 아이디 값에 의해서 데이터 요소를 구분해 낼 수 있다.

### 3. 레지스트리 기반 XML DTD 변환

서로 다른 도메인에서 DTD를 작성했을 경우 작성된 DTD 내의 태그의 의미가 서로 상이하게 된다. 이때 데이터 레지스트리를 참조하게 되면 레지스트리에 있는 데이터 요소의 유일한 id 값이 DTD의 데이터 요소(태그)의 속성 값으로 포함되게 된다. 그렇기 때문에 레지스트리를 참조한 서로 다른 DTD 간에는 표현은 다르지만 그 의미는 같은 태그들이 존재하게 된다.

서로 다른 DTD를 기반으로 작성된 XML 문서를 교환할 수 있도록 하기 위해서는 먼저 두 DTD를 이용해서 문서의 구조를 알아내야 한다. 구조 분석 과정에서는 2가지 프로세스가 동시에 발생하게 된다. 첫 번째는 DTD에서 id를 가지고 있는 태그들을 매핑테이블로 저장하는 프로세스이다. 매핑테이블에는 두 개의 DTD에서의 태그이름과 id값이 저장된다. 두 번째는 XML 문서의 태그구조를 이용해서 트리를 생성하는 것이다. 이 트리는 링크드리스트를 이용해서 작성되게 된다. 트리를 구성하는 이유는 두 DTD의 태그 구조가 다를 경우 XML 문서를 전달 받은 쪽의 DTD에 맞도록 XML 문서의 구조를 재구성 하기 위함이다. DTD가 서로 다른 환경에서 작성되었기 때문에 의미는 같고 표현만 다른 태그들이 1:1 매핑 관계에 있다는 것을 보장 할 수 없고 또 1:1 매핑 관계에 있다고 하더라도 그 구조가 동일하다는 보장을 할 수 없다. <그림 2>는 서로 다른 구조를 가지는 두 개의 DTD를 보여준다.

```
DTD A
<!Element Name(firstname, lastname)>
<!Attlist Name E_ID CDATA #Fixed "R.001">
<!Element firstname(#PCDATA)>
<!Element lastname(#PCDATA)>
<!Element Job>
<!Attlist Job E_ID CDATA #Fixed "R.002">
<!Element Phone>
<!Attlist Phone E_ID CDATA #Fixed "R.003">
```

```
DTD B
<!Element 이름>
<!Attlist 이름 E_ID CDATA #Fixed "R.001">
<!Element 전화번호>
<!Attlist 전화번호 E_ID CDATA #Fixed "R.003">
<!Element 직업>
<!Attlist 직업 E_ID CDATA #Fixed "R.002">
```

<그림 2> 서로 다른 구조의 DTD

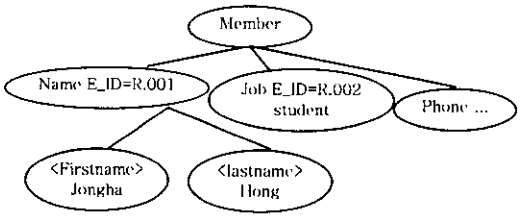
그림에서 보면 DTD A에서는 Name 태그가 firstname과 lastname이라는 서브태그를 가지고 있고 DTD B에서 동일한 id 값을 가지는 이름 태그는 서브 태그를 가지고 있지 않다는 것을 알 수 있다. 이 경우 DTD A를 기반으로 작성된 XML 문서는 서로 다른 태그 구조로 인해 DTD B를 이용해서 인식 될 수 없다.
그러므로 이와 같이 구조가 다른 경우에는 문서를 받는 쪽의 DTD가 인식할 수 있도록 XML 문서를 변환해야 한다.

<테이블 1>은 두 DTD를 분석하는 과정에서 만들어진 매핑 테이블을 보여주고 <그림 3>은 DTD A를 기반으로 작성된 XML 문서를 트리 형태로 변환한 그림을 보여준다

| E_ID | DTD A | DTD B |
|---|---|---|
| R.001 | Name | 이름 |
| R.002 | Job | 직업 |
| R.003 | Phone | 전화번호 |
| ... | ... | ... |

<테이블 1> 데이터 요소에 대한 매핑 테이블

```
<member>
  <name E_ID ="R.001">
    <firstname>Jongha </firstname>
    <lastname> Hong </lastname>
  </name>
  <job E_ID="R.002">student </job>
  <phone E_ID="R.003">011-737-1111</phone>
</member>
```

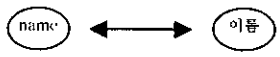



<그림 3> XML 문서와 변환된 트리

문서 변환 시에 고려해야 할 구조적 상이성은 크게 4 가지로 나누어 진다.

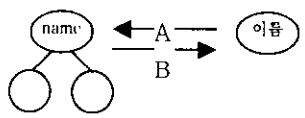
다음은 가장 일반적인 4가지 구조적 상이성을 제시한다.

1> 1:1 관계



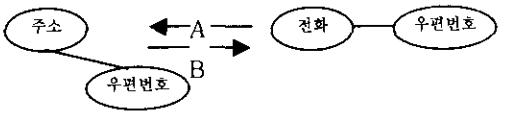

태그의 구조가 서로 1:1 관계로 대응되는 경우. 두 개의 DTD 에 사람의 이름을 나타내는 태그가 <name>과 <이름>으로 표현되는 경우.

2> 한쪽에만 포함관계가 존재



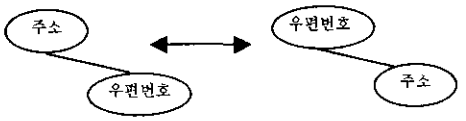
<그림 2>와 같은 경우로 한쪽에만 서브태그가 존재하는 경우. <name> 태그 아래로 <firstname>와 <lastname>태그가 존재하고 다른 쪽에는 <이름>만 존재하는 경우가 그 예이다.

3> 수평 관계와 수직 관계




부모 태그와 서브태그가 다른 DTD에서 같은 레벨의 태그로 표시될 경우.

4> 포함 관계가 반대인 경우




포함관계를 가지는 태그의 포함관계가 다른 DTD에서는 반대인 경우.

5> 예외 상황

태그 구조의 변경이 불가능하거나 위에 명시되지 않은 다른 상황이 발생할 경우.

다음은 위에 나열한 상이성에 대한 케이스별 해결 방법이다.

1> 매핑 테이블을 이용하여 같은 아이디를 가지는 태그를 바꾼다.

2> B의 경우에는 서브 태그에 대한 id가 없을 경우 value를 합치고 태그는 삭제한다. A의 경우에는 예외 상황(Exception)을 발생시킨다.

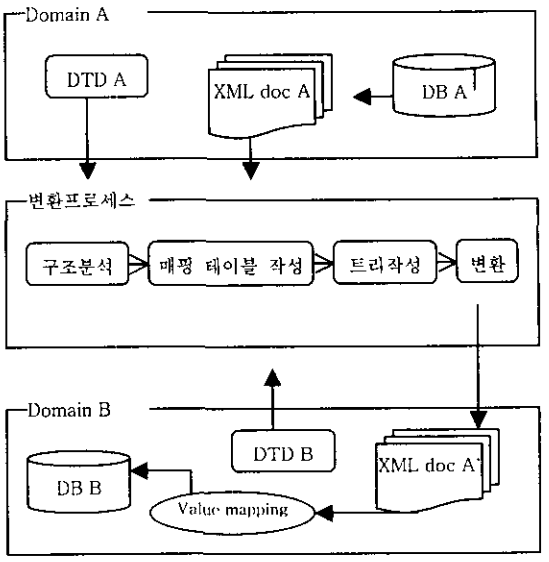
3> A의 경우에는 주소 태그를 상위태그로 첨가하고 우편번호 태그를 그 아래에 위치시킨다. B의 경우에는 상위 태그를 없애고 전화 태그와 동일한 레벨로 태그를 옮긴다.

4> 태그의 포함관계가 서로 반대인 경우에는 태그의 위치를 변경하고 서브태그의 값은 상위 태그의 속성값으로 추가 한다

5> 예외 상황이 발생했을 경우에는 발생한 예외 상황에 대한 설명을 포함하는 메시지를 출력함으로써 사용자가 이에 대한 처리 결정권을 갖도록 한다

변환 과정에서는 위에 나열한 4가지 경우를 반복적으로 (recursive) 적용한다. 문서를 1:1 관계에 대해서 태그 변환을 실시하고, 다음에 한쪽에만 서브태그가 존재하는 경우를 찾아서 변환을 실시한다. 이러한 방법으로 3의 경우, 4의 경우에 해당되는 부분을 찾아서 차례로 변환을 시행한다. 5에 언급한 예외 상황이 발생했을 경우에는 사용자의 입력을 받아서 적절한 처리가 이루어지도록 한다. 4가지 변환이 모두 적용된 최종 결과 문서는 문서를 받는 쪽의 DTD가 인식할 수 있는 형태로 그 구조가 바뀌어져 있게 된다.

<그림 5>는 서로 다른 DTD를 가지는 XML 문서를 익스포트(export) 하고 임포트(import) 하는 전체적인 프로세스를 보여준다. DTD B를 기반으로 작성된 XML 문서 B는 전달 과정에서 변환 과정을 거치게 된다. 변환 과정에서는 매핑 테이블 작성, 문서의 트리 변환, 실제 변환이 수행된다. 변환 과정을 마친 새로운 XML 문서 B'는 DTD A가 인식할 수 있게 된다.




<그림 5> XML 문서의 변환 과정

## 4. 결론

본 논문에서는 메타 데이터 레지스트리를 이용한 XML 문서의 교환 방법에 대해서 제안 하였다. DTD 작성시에 도메인이 서로 상이하기 때문에 상이한 DTD를 기반으로 작성된 문서를 교환하기 위해서는 문서를 전송 받는 쪽의 DTD에 맞게 문서를 변환 해주어야 한다. 본 논문에서 제안한 방법은 태그의 표현만 다르고 그 구조는 같은 1:1 관계의 문서 뿐만 아니라 사용자가 DTD의 구조 또한 다르게 작성할 수 있다는 가정 하에 상이한 구조에 따른 문서 변환 방법까지 제시하고 있다.

실제로 문서의 내용은 데이터베이스로부터 가져오는 경우가 대부분이고, 전달 받은 XML 문서의 데이터 값을 데이터베이스로 저장해야 할 경우도 있다. 이 경우에는 실제 데이터의 타입 불일치에 따른 타입 변환 과정이 필요하게 된다. 이러한 타입 변환은 데이터베이스에 값을 넣을 때 타입 변환과정(value mapping process)을 거쳐서 이루어 지게 된다. 이러한 타입 변환과정은 추후 연구과제로 현재 연구 중이다.

### 참고 문헌

[1] Information technology – "Specification and *standardization of data element*", ISO/IEC 11179 part2

[2] 나홍석,채진석,백두권 "데이터 레지스트리에 기반한 XML 문서 공유 환경", 한국 정보과학회 추계 학술발표논문집

[3] 박혜숙, 나홍석,백두권 "데이터 레지스트리에 기반한 전자상거래 메타데이터 공유 환경", 한국 정보과학회 춘계 학술발표논문집

[4] Christof Bornhovd "Semantic Metadata for the Integration of Web-based Data for Electronic Commerce"

[5] Deil Zhang, Yisheng Dong "An Object Oriented Data Model for Web and Its Algebra"