객체 식별자를 이용한 객체지향 데이타베이스의 XML 문서로의 변환

(Transformation of Object-Oriented Databases into XML Documents using Object Identifiers)

윤 정 희 † 박 창 원 † 정 진 완 ††
(JungHee Yun) (Chang-Won Park) (Chin-Wan Chung)

요 약 데이타 변환은 데이타 재사용, 데이타 교환 및 통합 등에 사용되는 중요한 기술이다. 본 논문에서는 객체지향 데이타베이스를 유효한 XML 문서들로 변환시키는 두 가지 알고리즘을 제시한다. 먼저 객체지향 데이타베이스의 스키마, 객체지향 데이타베이스, DTD 그리고 XML 문서를 정의한 뒤 두 가지 알고리즘, 즉 객체지향 데이타베이스의 스키마를 DTD로 변환시키는 알고리즘과 객체지향 데이타베이스를 XML 문서들로 변환시키는 알고리즘을 제시한다.

그리고 제시한 두 가지 알고리즘의 결과는 항상 잘 구성된 XML 문서들이고 유효한 XML 문서들임을 증명한다. 잘 구성된 XML 문서는 XML 문서가 갖추어야 하는 필수 조건이므로 반드시 필요하다. 또한 유효성은 유효한 XML 문서들을 필요로 하는 XML 응용에 유효한 XML 문서를 제공할 수 있도록 한다.

Abstract Data transformation is an important technique for data reuse, data exchange and data integration, etc. In this paper, we present two algorithms to transform object-oriented databases(OODBs) into valid XML(eXtensible Markup Language) documents. First, we formally define the OODB schema, OODB, DTD(Document Type Definition) and XML document, and then present the two algorithms : the first algorithm is to transform an OODB schema into a DTD, and the second is to transform OODBs into XML documents.

In addition, we show that the two algorithms produce well-formed and valid XML documents. We need to prove the well-formedness because the well-formedness is a mandatory requirement of all XML documents. Furthermore, ensuring validity allows us to offer valid XML documents to XML applications that can process only the valid XML documents.

1. 서 론

최근 웹(Web)은 많은 데이타를 얻을 수 있는 곳으로, 여러 분야의 다양한 사용자들에게 점점 친숙해지고 있다. 이러한 웹을 이용하면 원하는 데이타를 쉽게 얻을 수 있을 뿐만 아니라 자신의 데이타를 표현하고 널리 알릴 수 있다. 그런데, 이를 위해서는 특정 데이타 소스(source)에 보유하고 있는 데이타를 웹 상에서 볼 수 있는 데이타로 표현할 수 있는 기술이 필요하다. 이와 관련하여 여러 데이타 소스들에 대한 래퍼(wrapper)와 중재기(mediator), 그리고 질의 언어를 제공하여 여러 데이타들을 HTML 형식으로 제공하는 기술을 제시한 연구 결과가 있었다[1].

그런데, 현재는 웹 데이타들이 대부분 HTML 형식으로 표현되고 있다. 하지만 이 형식은 데이타 교환 및 통합을 쉽게 할 수 없다는 단점 때문에 XML(eXtensible Markup Language)[2]이 새롭게 등장하였다. 그리고 최근 많은 전문가들이 XML을 차세대 인터넷 문서 표준으로 생각하고 있다.

따라서 웹 데이타라고 하면 지금까지 주로 HTML 형식을 따른다고 생각했던 것이 이제는 XML이 웹 문

† 비 회 원 : 한국과학기술원 전산학과
cotoy@islab.kaist.ac.kr
cwpark@islab.kaist.ac.kr
†† 종신회원 : 한국과학기술원 전산학과 교수
chungcw@islab.kaist.ac.kr
논문접수 : 2000년 7월 1일
심사완료 : 2001년 3월 30일

서를 표현하는 데이타 형식으로 여겨지게 되었다. 이에 따라 HTML 형식의 데이타들을 XML 형식으로 변환시키는 것과 관련된 연구도 이루어져 왔다[3].

기존의 HTML에서는 태그가 단순히 데이타를 어떻게 표현할 것인가를 나타내기 위한 것이라면 XML에서의 태그는 데이타 자체를 표현하는 것이다. 이러한 XML의 특성은 XML 문서를 얻어와서 그 문서에 포함된 데이타를 끌어내는 질의어가 가능하게 한다. 따라서 웹 상에서 제공되는 XML 문서에 대해 데이타베이스 기술을 적용시키려는 노력이 있어 왔다[4][5].

그런데, 이와는 반대로 데이타베이스 시스템에 저장된 방대한 데이타를 어떻게 웹 상의 데이타로 표현할 것인가에 대한 연구도 필요하다. 즉, 기존의 데이타베이스 시스템에 저장된 데이타를 끌어내어 XML 형식으로 변환시켜 웹 상에서 제공한다면 현재 존재하는 정보를 보다 효율적으로 사용할 수 있는 기반을 마련하게 되는 것이다.

그런데 현재 관계형 데이타베이스에 저장된 데이타를 XML 문서로 변환하는 것과 관련된 연구는 이미 나와 있으며[9], 객체지향 데이타베이스에서는 질의 결과를 XML 데이타로 변환하는 연구도 나와있지만[12], 객체지향 데이타베이스에 저장된 데이타 전체를 XML 문서로 직접 변환하는 것과 관련된 연구는 나와 있지 않다. 이러한 배경에서 본 논문에서는 데이타 변환 기술의 하나로 객체지향 데이타베이스 시스템에 저장된 데이타를 XML 문서로 변환시키는 기술을 제안하고자 한다. 즉, 객체지향 데이타베이스 시스템에 저장된 데이타를 웹 데이타 포맷인 XML 문서로 제공하여 많은 사용자들이 접근할 수 있고 또 다른 목적에서 재사용할 수 있도록 하는 기반을 마련하고자 하는 것이다.

그리고 결과 XML 문서가 잘 구성된(well-formed) XML 문서이고 데이타베이스 스키마로부터 얻은 결과 DTD(Document Type Definition)에 대해 유효한(valid) XML 문서임을 보임으로서 제시한 데이타 변환 기술의 유용성을 보인다. 먼저 잘 구성된 XML 문서라는 것은 제시한 기술이 구조상으로 옳은 XML 문서를 만든다는 사실을 나타낸다. 그리고 유효한 XML 문서라는 것은 제시한 기술에 의한 결과 DTD에 대해 XML 문서가 유효함을 나타내는 것으로, 이는 유효한 XML 문서를 필요로 하는 응용에 결과 XML 문서를 제공할 수 있게 한다.

본 논문의 구성은 다음과 같다. 2장에서는 기존의 데이타 변환 기술에 관련된 연구 결과들에 관해 설명한다. 3장에서는 객체지향 데이타베이스의 스키마, 객체지향 데이타베이스, DTD 그리고 XML 문서를 정의한 뒤 이 정의를 바탕으로 객체지향 데이타베이스를 XML문서로 변환하는 알고리즘을 제시한다. 그리고 제시한 알고리즘의 유용성을 확인하기 위해서 변환에 의한 결과 DTD와 XML 문서가 잘 구성된 유효한 XML 문서임을 보인다. 4장에서는 제시한 알고리즘을 구현하고 실험한 결과에 대해서 보이고, 마지막으로 5장에서는 결론을 논의한다.

2. 관련 연구

대부분의 중재기 시스템에서 이질적인 데이타들간의 교환과 통합 기술을 요구하는데 이때 데이타 변환 기술이 필요하다. 이러한 중재기 시스템으로 STRUDEL 시스템[1]과 MIX(Mediation of Information using XML) 중재기 시스템[6, 7]을 들 수 있다. MIX 중재기 시스템의 경우 데이타 모델로 XML을 사용하여 웹 전체를 커다란 분산 데이타베이스 시스템으로 보기 때문에 여러 데이타 소스들과 XML 간의 래퍼가 필요하다. 즉, 여러 가지 데이타 포맷들과 XML 간의 데이타 변환 기술이 사용되는 것이다.

임의의 데이타 포맷간의 데이타 변환이 가능하도록 하는 목적에서 만들어진 시스템으로 YAT 시스템[8]이 있는데, 트리 형태의 YAT 모델을 정의하고 변환 언어 YATL(YAT Language)를 사용한다. 이질적인 데이타들의 통합을 필요로 하는 응용이 증가하면서 데이타 변환 기술의 필요성이 높아졌는데 기존의 데이타 변환 기술이 재사용 불가능한 경우가 대부분이라는 것에 착안하여 YAT 시스템의 경우는 YAT 데이타 모델에 대한 래퍼가 구현된 모든 이질적인 데이타 포맷간의 데이타 변환이 가능하도록 하였다.

본 논문에서의 연구와 가장 유사한 연구로 관계형 데이타베이스에 저장된 데이타를 XML 문서로 변환하는 기술에 대한 연구를 들 수 있는데, 이 분야에 대한 연구 결과들은 많다. 먼저 관계형 데이타베이스에 저장된 데이타를 XML 문서로 변환하는 연구가 있었는데[13], 여기서는 관계형 데이타베이스에 저장된 데이타 전체를 XML 데이타로 변환한다는 점에서 본 연구와 유사한 부분이 있지만, 구현상의 여러 가지 가능한 방법들로 구분하여 가장 효율적으로 변환하는 것이 어떤 것인지를 실험을 통하여 찾아내는 방법을 사용하였다는 점에서 차이가 있다. 또 다른 한가지로 DB2XML[9]이라는 툴이 있다. 현재 웹 페이지들 중 많은 부분이 기존의 데이타베이스에 저장된 데이타로부터 생성되었다는 점에 착안하여 관계형 데이타베이스에 저장된 데이타를 XML 문서로 자동 변환시켜주는 툴을 만든 것이다. DB2XML

은 관계형 데이타베이스에 저장된 데이타에 대해서 XML 문서와 DTD를 제공하는데 관계형 모델의 각 구성 요소에 대한 사상(mapping) 규칙을 제시하여 그 규칙에 의한 데이타 변환을 제시하였다.

3. 객체지향 데이타베이스의 XML로의 변환 기법

객체지향 데이타베이스에 저장된 데이타의 XML로의 변환은 데이타베이스 스키마의 DTD로의 변환과 실제 저장된 데이타의 XML 문서로의 변환 두 가지 부분으로 나누어 볼 수 있다. 먼저 객체지향 데이타베이스의 스키마와 데이타, 데이타 변환의 결과가 될 DTD와 XML 문서를 정의하고 이 정의에 따라서 두 가지 변환 알고리즘을 제시한다.

3.1 객체지향 데이타베이스와 XML

3.1.1 객체지향 데이타베이스의 스키마와 데이타 정의

객체지향 데이타베이스의 스키마 정보는 데이타베이스 이름과 클래스(class)들의 스키마 정보로 이루어진다. 그리고 각 클래스는 클래스 이름과 애트리뷰트(attribute)들의 정보로 구성되며 각 애트리뷰트는 이름과 자신의 타입 정보를 가진다. 여기서 각 애트리뷰트의 타입은 정수, 스트링 등의 기본 타입이거나 다른 클래스 타입, 또는 set, bag, list와 같은 collection 타입일 수 있다. 여기서 collection 타입인 경우는 collection의 원소 타입이 다시 기본 타입, 다른 클래스 타입 또는 collection 타입이 된다[10, 11]. Relationship의 경우는 애트리뷰트와 같이 생각할 수 있고 단지 역경로(inverse path) 정보가 추가되면 된다. 따라서 알고리즘 기술의 간결성을 위해서 relationship은 고려하지 않겠다. 이러한 특징을 가지는 객체지향 데이타베이스 스키마 정보를 <그림 1>과 같이 나타낼 수 있다.

객체지향 데이타베이스의 스키마와 데이타를 <표 1>과 <표 2>로 정의하였다. <표 1>의 *database_def* 타입이 스키마를 표현하고 <표 2>의 *database* 타입이 데이타를 표현한다.

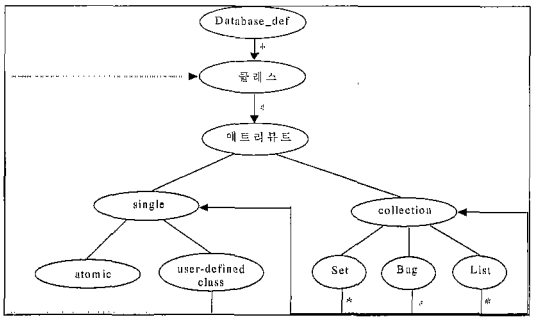

그림 1 객체지향 데이타베이스의 스키마 구조

표 1 객체지향 데이타베이스의 스키마 정의

database_def := (*name*, *class*; *class*; ...; *class*)
class := (*name* : *sname*, *attribute*; *attribute*; ...; *attribute*)
attribute := (*name*, *single*) | (*name*, *collection*)
single := *atomic* | *user_defined*
collection := **set**(*single*) | **set**(*collection*)
| **bag**(*single*) | **bag**(*collection*)
| **list**(*single*) | **list**(*collection*)
user_defined := *name*
atomic ∈ **BASIC_TYPE**
name ∈ **NAME**
sname ∈ **NAME** ∪ {**empty**}
BASIC_TYPE = {**int**, **char**, **string**, ..., **point**, **line**, **polygon**, ... }

표 2 객체지향 데이타베이스의 데이타 정의

database := (*class_extent*; ... ; *class_extent*)
class_extent := (*object*; ... ; *object*)
object := (*oid*, *attr_value*; ... ; *attr_value*)
attr_value := *single_value* | *collection_value*
collection_value := **set**(*single_value*; ... ; *single_value*)
| **set**(*collection_value*; ... ; *collection_value*)
| **bag**(*single_value*; ... ; *single_value*)
| **bag**(*collection_value*; ... ; *collection_value*)
| **list**(*single_value*; ... ; *single_value*)
| **list**(*collection_value*; ... ; *collection_value*)
single_value ∈ **BASIC_TYPE_VALUE** ∪ **OID_VALUE**
oid ∈ **OID_VALUE**

3.1.2 DTD와 XML 문서 정의

본 논문에서 제시하는 데이타 변환 기법에 의한 결과 XML 문서는 변환 알고리즘에서 제시하는 규칙에 따라서 다음과 같은 조건을 만족하는 문서들이 된다.

- 속성은 ID, HREF 속성만 포함한다. 그리고 한 엘리먼트는 하나의 속성만을 가진다. 즉 속성을 가지지 않거나 ID 또는 HREF 속성을 가진다.
- 엘리먼트의 내용으로 데이타와 다른 엘리먼트의 혼합이 되는 경우는 없다.
- 엔터티(entity)는 포함하지 않는다.

<표 3>과 <표 4>가 각각 DTD와 XML 문서를 정의한 것인데 <표 3>의 타입 *DTD*가 XML의 DTD를 표현하는 타입이고 <표 4>의 타입 *XML_DOC*가 실제의 XML 문서를 정의한 타입이다. 여기서 DTD의 정의는 [6]에서 정의한 DTD를 확장한 것이다.

표 3 DTD 정의

DTD := {<*n* : *attr(n)* : type(*n*)>}
n ∈ **NAME**
attr(n) ∈ {ID, HREF, *null*}
type(*n*) : regular expression over **NAME** or PCDATA or EMPTY

· regular expression
r_1, r_2 : concatenation of r_1 and r_2
$r_1 | r_2$: union of r_1 and r_2
$r*$: Kleene closure of r
$r+$: $r, r*$
$r?$: $r \mid \varepsilon$

표 4 XML 문서 정의

XML_DOC := *element*
element := **start_tag**(*name*) : *attribute* < *content* > **end_tag**
attribute := **null** | (*attr_name* : *data*)
content := **empty** | *pcdata* | *element*; ... ; *element*
name ∈ **NAME**

attr_name ∈ {ID, HREF}

data ∈ **ID_VALUE** ∪ **HREF_VALUE**

pcdata ∈ **STRING_VALUE**

3.2 데이타 변환 알고리즘

3.2.1 결과 XML 문서의 구조

먼저 객체지향 데이타베이스를 어떤 구조를 가지는 XML 문서들로 표현할 것인지를 설명하겠다. 여기서는 각 클래스별로 XML 문서를 만드는 방법을 선택하였다. 데이타 베이스 전체를 하나의 XML 문서로 만들거나 객체별로 XML 문서를 만드는 방법은 XML 문서가 커지거나 XML 문서의 개수가 너무 많아진다는 단점이 있기 때문이다. <그림 2>에서 결과 XML 문서의 구조를 보여준다.

각 클래스별로 XML 문서를 만드는 방법은 여러 가지가 가능하다. 첫 번째로 각 클래스에 속한 객체들을 모두 XML 문서에서의 엘리먼트로 표현하여 전체 객체들을 하위 엘리먼트들의 리스트로 표현하는 방법이다. 이 방법은 객체지향 데이타베이스의 구조와 가장 유사하게 XML 문서를 만드는 방법이다. 두 번째로 각 클래스에 속한 객체들을 모두 애트리뷰트 리스트로 표현할 수 있다. 이 방법은 구조적으로는 하나의 클래스에 속한 전체 객체들을 XML에서의 한 엘리먼트로 표현하는 방법이지만 객체를 애트리뷰트로 표현하기 위해서는 IDREF 애트리뷰트를 이용해야 하므로 각 객체에 대한 XML 엘리먼트가 존재해야만 한다. 각 객체가 결국 엘리먼트로 표현되는 것은 첫 번째 방법과 동일하지만 각 엘리먼트가 클래스에 해당하는 엘리먼트와는 별도로 존재하는 것에서 차이가 있다. 세 번째로 생각할 수 있는 방법은 앞의 두 가지 방법을 혼합한 것이다. 한 클래스에 속한 객체들에 대해서 일부는 엘리먼트로 나머지는 애트리뷰트 리스트로 표현하는 방법으로 클래스에 속한 객체들 중 일부는 하위 엘리먼트로 표현되는 것이고 나머지는 애트리뷰트로 표현되는 것이다. 이 방법은 한 클래스에 속한 객체들에 대해서 서로 다른 규칙을 적용하는 면에서 일관성이 적어지는 단점이 있지만 이러한 구조를 필요로 하는 환경에서는 유용할 것이다. 이처럼 여러 가지 방법을 고려할 수 있는데 이 방법들간의 대체는 쉽게 이루어 질 수 있다. 따라서 환경과 목적을 고려하여 가장 적절한 방법을 선택하여 사용하는 것이 고려되어야 할 것이다. 여기서는 이러한 방법들 중 객체지향 데이타베이스의 구조를 가장 유사하게 유지하는 방법을 선택하였고 그 방법에 대한 변환 알고리즘을 고려하였다.

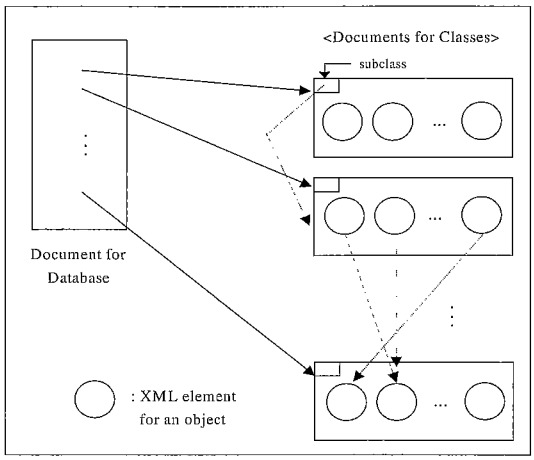

그림 2 결과 XML 문서들간의 관계

3.2.2 스키마 정보의 DTD로의 변환 기법

<표 5>는 객체지향 데이타베이스의 스키마 정보를 DTD로 변환하는 알고리즘인 **DTD_make**를 제시한 것으로 입력과 결과는 각각 3.1에서 정의한 *database_def* 타입과 *DTD* 타입이다.

표 5 **DTD_make** 알고리즘(스키마 정보를 DTD로 변환)

```
procedure DTD_make((u, c1; ... ; cn) : database_def)
input : u는 name 타입이고 ci (i=1, ... ,n)는 class 타입
output : DTD
begin
        X = {<u : null : (make_name(c1), ... , make_name(cn))|
        (SUBCLASS*, (get_name(c1)*| ... | get_name(cn)*))>};
        X0 = {<SUBCLASS : HREF : EMPTY>};
        X1 = {<make_name(c1) : HREF : EMPTY>};
        . . . ;
        Xn = {<make_name(cn) : HREF : EMPTY>};
        Y1 = class_DTD_make(c1);
        . . . ;
        Yn = class_DTD_make(cn);
        return union(X, X0, X1, ..., Xn, Y1, ..., Yn);
end

procedure class_DTD_make((u : s, a1; ... ; an) : class)
input : u는 name 타입, s는 sname 타입, ai (i=1, ... ,n)는 attribute 타입
output : DTD
begin
        b1; ... ; bm = get_attribute(s);
        return
        union
        ({<u : ID : get_name(b1), ..., get_name(bm),
         get_name(a1), ..., get_name(an)>},
         attr_DTD_make(a1), ... , attr_DTD_make(an));
end

procedure attr_DTD_make((u, v) : attribute)
input : u는 name 타입, v는 single 또는 collection 타입
output : DTD
begin
        if(v : single) then
                if(s ∈atomic) then
                        atomic_DTD_make((u, s));
                else if(s ∈user_defined) then
                        udefine_DTD_make((u, s));
        else if(v : collection) then
                if(v == set(s) or bag(s) or list(s)) then
                        return
                        union
                        ({<u:null:make_name(v)*>},
                         attr_DTD_make((make_name(u, v), s)));
end

procedure atomic_DTD_make((u, a))
input : u는 name 타입, a는 atomic 타입
output : DTD
begin
        return
                {<u : null : PCDATA>};
end

procedure udefine_DTD_make((u, a))
input : u는 name 타입, a는 user_defined 타입
output : DTD
begin
        return
                {<u : HREF : EMPTY>};
end
```

3.2.3 객체지향 데이타의 XML 문서로의 변환 기법

객체지향 데이타베이스에 저장된 데이타를 XML 문서로 변환하는 알고리즘인 **XML_make**의 입력과 결과는 각각 3.1에서 정의한 *database* 타입과 *XML_DOC* 타입의 집합으로 알고리즘의 전체 구조는 스키마 정보를 DTD로 변환하는 것과 유사하다[14].

3.3 잘 구성된 XML 문서와 유효한 XML 문서

본 절에서는 데이타 변환의 결과로 얻은 XML 문서가 DTD에 대해서 잘 구성된 XML 문서이고 유효한 XML 문서임을 보임으로서 주어진 알고리즘의 유용성을 알아본다.

3.3.1 잘 구성된 XML 문서

다음과 같은 조건을 만족하는 XML 문서는 잘 구성된 XML 문서이다.

- 태그가 완전히 중첩(nest) 되어야 한다.
- 한 엘리먼트내의 한 애트리뷰트는 유일해야 한다.

3.1에서 정의한 XML 문서는 위의 조건을 모두 만족하므로 잘 구성된 XML 문서를 표현하는 것이다. 따라서 알고리즘 **XML_make**의 결과가 *XML_DOC* 타입이 된다면 결과로 얻어지는 XML 문서는 잘 구성된 XML 문서가 된다. 먼저 *attr_value* 타입값에 대한 깊이(depth)를 다음과 같이 정의한다.

<정의 1>

$$\text{depth}(v : \textit{attr\_value}) := \begin{cases} 1 & \textbf{if } v \in \textit{single\_value} \\ 1 + \text{depth}(v_i) & \textbf{if } v = \textbf{set}|\ \textbf{bag}|\ \textbf{list}(v_1;...;\ v_n) \end{cases}$$

(정의 끝)

위의 정의를 이용하여 다음에 주어진 정리를 증명할 수 있다.

<정리 1>

XML_make는 항상 잘 구성된(well-formed) XML 문서만을 만든다.

(증명) [14] 참조

3.3.2 유효한 XML 문서

본 장에서는 객체지향 데이타베이스의 스키마 정보가 주어지고 그 스키마 정보를 만족하는 객체지향 데이타가 주어졌을 때 앞에서 제시한 알고리즘에 의한 결과 DTD에 대해서 결과 XML 문서는 유효한 XML 문서가 된다는 것을 보인다.

다음에 주어진 정의는 객체지향 데이타가 스키마를 만족하는 것은 어떤 경우인가를 정의한 것이다.

<정의 2>

$v \vdash s$: v가 s를 만족한다.

type(v) ∈{*database*, *class_extent*, *attr_value*.

single_value, collection_value}

type(s) ∈{*database_def, class, attribute, single, collection*}

1) v : *single_value* ⊢ s : *single*

s ∈*atomic* 일 때, v ∈BASIC_TYPE_VALUE(s)

s ∈*user_defined* 일 때, v ∈OID_VALUE

2) v : *collection_value* ⊢ s : *collection*

s = **collection**$_1$(*e*), v

= **collection**$_2$(v_1 ; ... ; v_n) 일 때,

i) **collection**$_1$ = **collection**$_2$,

collection$_1$, **collection**$_2$ ∈ {**set, bag, list**}

ii) v_1 ⊢ *e* , ... , v_n ⊢ *e*

3) v : *attr_value* ⊢ (n, s) : *attribute*

i) v ⊢ s

4) (o_1; ... ; o_n) : *class_extent* ⊢ (n : sn, a_1; ... ; a_m) : *class*

sn = **empty** 이고 o_i

= (id_i , b_{i1}; ... ; b_{ik_i}) (i = 1, ..., n) 일 때,

i) k_1 = ... = k_n = m

ii) b_{i1} ⊢ a_1, ... , b_{ik_i} ⊢ a_m (i = 1, ..., n)

sn ≠ **empty** 이고, g_1; ... ; g_z

= **inherited**(sn) 이고,

o_i = (id_i , b_{i1}; ... ; b_{ik_i}) (i = 1, ..., n) 일 때,

i) k_1 = ... = k_n = m + z

ii) b_{i1} ⊢ g_1, ... , b_{iz} ⊢ g_z, b_{iz+1} ⊢ a_1,

... , b_{ik_i} ⊢ a_m

(i = 1, ..., n)

5) (c_1; ... ; c_m) : *database* ⊢ (n, d_1; ... ; d_k) : *database_def*

i) m = k

ii) c_i ⊢ d_i , i = 1, ... , m

(정의 끝)

다음은 XML의 엘리먼트가 어떤 경우에 주어진 DTD를 만족한다고 표현하는지에 대해서 *element* 타입과 *DTD* 타입을 기반으로 정의한 것으로 [7]에서 사용한 정의를 확장시킨 것이다.

<정의 3>

다음이 성립하는 경우 element e (*element* 타입)가 DTD D(*DTD* 타입)를 만족한다고 정의한다.

e = **start_tag**(n) : a < c > **end_tag**

1) n ∈**NAME**, **NAME** : a set of element names

2) i) a = **null** 일 때 *attr*(n) = *null*

ii) a ≠ **null** 일 때 *attr*(n) = attr_name(a)

3) i) c = **empty** 일 때 type(n) = EMPTY

ii) c ∈ *pcdata* 일 때 type(n) = PCDATA

iii) c = e_1;...; e_m 일 때 name(e_1) ... name(e_m) ∈ L(type(n)) 이고, e_i (i=1, ..., m)는 D를 만족한다.

(단, L(r) : regular language described by r)

(정의 끝)

다음은 <정의 3>을 기반으로 유효한 XML 문서가 되는 경우의 조건을 제시한다.

<정의 4>

임의의 XML 문서 d와 DTD D, 문서 타입 t가 주어졌을 때 다음의 조건을 만족하는 경우 d는 D에 대해서 타입 t의 유효한 XML 문서라고 정의한다.

1) 임의의 엘리먼트 e에 대해서 e가 d의 루트 엘리먼트(root element)일 때 t는 엘리먼트 e의 이름과 같고,

2) e는 D를 만족한다.

(정의 끝)

이렇게 주어진 정의를 바탕으로 앞에서 제시한 알고리즘의 결과 DTD에 대해서 결과 XML 문서는 유효한 XML 문서임을 <정리 2>로 보인다.

<정리 2>

임의의 OODB 스키마 정보 schema(*database_def* 타입)와 OODB 데이타 oodata(*database* 타입)에 대해서

oodata ⊢ schema 이고,

DTD_make(schema) = D : *DTD*

XML_make(schema, oodata)

= {d_1, ... , d_m} d_i : *XML_DOC* (i = 1, ... , m)

t = **name**(schema) ∈**NAME**일 때,

각각의 d_i (i = 1, ..., m)는 D에 대해서 타입 t의 유효한 XML 문서이다.

(증명) [14] 참조

4. 데이타 변환 알고리즘의 구현

4.1 구현 환경

구현에 사용한 객체지향 데이타베이스 시스템은 ODMG 2.0 표준을 지원하는 OMEGA(Object Management systEm for Geo-spatial Applications)이다. 알고리즘의 구현을 위해서는 OML(Object Management Language)과 GNU C++을 사용하였다.

결과 DTD와 XML 문서는 XML 1.0 사양에 따르는

것으로 결과 XML 문서가 잘 구성된 문서이고 결과 DTD에 대해서 유효한 XML 문서임을 확인하기 위해서 마이크로 소프트 인터넷 익스플로러(Microsoft Internet Explorer) 5.0을 파서로 사용하였다.

4.2 실험 데이타

실험에 사용하기 위해서 university라는 데이타베이스를 만들었는데 <그림 3>과 같은 스키마를 가진다. 그림에서 직사각형은 각각의 클래스를 표현한 것이고 직사각형의 윗부분에는 클래스 이름, 아랫부분에는 클래스에 속하는 애트리뷰트들의 이름과 타입 정보를 표현하였다. 굵은 화살표는 상속을 표현하기 위한 것이고 점선으로 표현된 화살표는 애트리뷰트의 타입이 다른 클래스 타입이 되는 경우 타입이 되는 클래스로의 참조를 표현한 것이다.

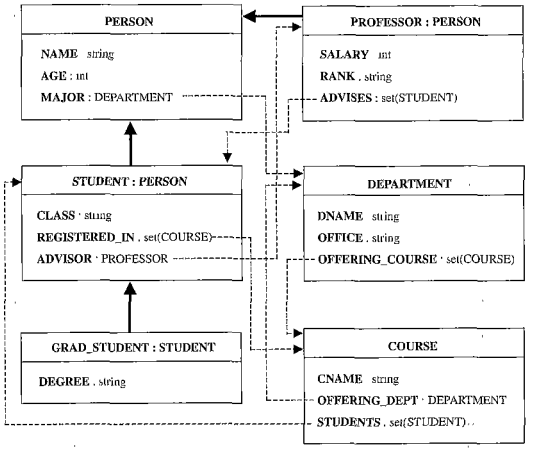

그림 3 예제 데이타베이스의 스키마 구조도

주어진 그림과 같은 스키마를 가지는 데이타베이스를 객체수를 달리하여 <표 6>과 같이 3가지 경우에 대해서 고려한다.

표 6 실험 데이타베이스의 객체수

	경우 1	경우 2	경우 3
PERSON	5	10	500
DEPARTMENT	8	8	8
STUDENT	80	800	8000
PROFESSOR	8	40	400
COURSE	8	40	400
GRAD_STUDENT	5	10	500
합계	114	908	9808

그리고 point, line, polygon 등의 공간 속성을 포함하는 데이타베이스에 대해서도 실험을 하였는데 jamsil이라는 데이타베이스로 jamsil의 지도 데이타를 포함하고 있다. jamsil 데이타베이스의 객체수는 69345개이다.

4.3 실험 결과

위에서 설명한 3가지 경우와 공간 속성을 포함하는 jamsil 데이타베이스에 대해서 구현한 알고리즘을 적용시켜 보았다. 먼저 university에서 고려한 세 가지 경우와 jamsil 데이타베이스에 대해서 각 알고리즘의 반응시간을 측정한 결과를 <표 7>에 표현하였다.

표 7 알고리즘 반응 시간 측정 결과(단위 : 초)

	경우 1	경우 2	경우 3	jamsil
전체 객체수	114	908	9808	69345
DTD_make	0.0006			0.0002
XML_make	0.0051	1.0012	35.9998	201.9999

결과 DTD와 XML 문서에 대해서 잘 구성된 문서이고 유효한 문서임을 확인하기 위한 XML 파서로 인터넷 익스플로러 5.0을 이용하였다. <그림 4>는 결과 문서를 인터넷 익스플로러로 확인한 화면을 보여준다. 인터넷 익스플로러 5.0을 이용하여 아무런 에러 없이 결과 XML 문서를 볼 수 있다는 점에서 결과로 얻은 XML 문서들이 잘 구성된 문서라는 점을 알 수 있다. 그리고 XML 문서의 앞부분에 DTD 문서를 포함시켰다는 점에서 결과로 얻은 DTD에 대해서 XML 문서가 유효한 문서임을 확인할 수 있다.

그림 4 인터넷 익스플로러로 본 결과 XML 문서 (student.xml)

5. 결 론

본 논문에서는 데이타 변환 기법의 하나로 객체지향 데이타베이스에 저장된 데이타를 XML 데이타로 변환하는 기법을 제안하였다. 먼저 객체지향 데이타베이스의 스키마와 데이타, 그리고 결과가 되는 DTD와 XML 문서를 정의하였고 그 정의에 의한 두 가지 알고리즘을 제안하였다. 첫 번째 알고리즘은 객체지향 데이타베이스의 스키마 정보를 DTD로 변환하는 알고리즘이고 두 번째 알고리즘은 실제의 객체지향 데이타베이스에 저장된 데이타를 XML 문서들로 변환하는 알고리즘이다. 객체지향 데이타베이스를 특징짓는 여러 가지 개념들에 대해서 어떠한 규칙을 적용시켜서 어떤 구조의 XML 데이타로 만들 것인지 변환 규칙을 제안하였다.

두 알고리즘의 결과로 얻는 DTD와 XML 문서들이 옳은 것임을 보이기 위해서는 잘 구성된 XML 문서이고 유효한 XML 문서가 됨을 보여야 한다. 따라서 객체지향 데이타베이스의 스키마가 있고 그 스키마에 맞는 데이타가 주어졌을 때 제안한 두 가지 알고리즘을 각각 적용시켜서 얻은 DTD와 XML 문서들이 잘 구성된 XML 문서이고 유효한 XML 문서임을 보였다. 이에 의해서 제안한 알고리즘의 유용성을 보임과 동시에 결과 XML 문서에 대한 애플리케이션의 개발에 있어서 좀 더 빠른 파서를 사용하면서도 DTD에 대해서 유효한 XML 문서라는 점을 얻을 수 있는 것이다.

이렇게 제안한 두 가지 알고리즘을 실제로 구현하였고 실제의 객체지향 데이타베이스에 저장된 데이타에 대해서 알고리즘을 적용시켜 보았다. 그리고 그 결과 DTD와 XML 문서들에 대해서 XML 파서를 이용하여 잘 구성된 문서이고 유효한 문서임을 확인하였다.

더 나아가 객체지향 데이타베이스를 특징짓는 요소 중의 하나로 method를 들 수 있는데 본 논문의 내용에서는 데이타간의 변환 알고리즘을 고려하면서 단순히 데이타라고만 보기 힘든 method에 대한 고려가 이루어지지 않았다. XML에서 제공하는 PI(Processing Instructions)를 이용한 방법을 고려하여 method를 제공하는 방법을 연구하여 더 나아가 객체지향 데이타베이스를 특징짓는 모든 요소들을 XML 데이타로 제공할 수 있는 기반을 마련하도록 할 것이다.

참 고 문 헌

[1] Mary Fernandez, Daniela Florescu, Alon Levy, and Dan Suciu. A Query Language for a Web-Site Management System. In SIGMOD Record, September 1997.

[2] World Wide Web Consortium(W3C). Extensible Markup Language(XML) 1.0, 1998. http://www.w3.org/TR/1998/REC-xml-19980210.

[3] Arnaud Sahuguet and Fabien Azavant. Looking at the Web through XML glasses. In International Conference on Cooperative Information Systems, 1999.

[4] Jayavel Shanmugasundaram, Kristin Tufte, Gang He, Chun Zhang, David DeWitt, and Jffrey Naughton. Relational Databases for Querying XML Documents : Limitations and Opportunities. In Proceedings of the 25th VLDB Conference, 1999.

[5] Alin Deutsch, Mary Fernandez, and Dan Suciu. Storing Semistructured Data with STORED. In Proceedings of ACM SIGMOD Conference, 1999.

[6] Yannis Papakonstantinou and Pavel Velikhov. Enhancing Semistructured Data Mediators with Document Type Definitions. In IEEE International Conference on Data Engineering, 1999.

[7] Chaitanya Baru, Amarnath Gupta, Bertram Ludascher, Richard Marciano, Yannis Papakonstantinou and Pavel Velikhov. XML-Based Information Mediation with MIX. In ACM SIGMOD, 1999.

[8] S. Cluet, C. Delobel, J. Simeon, and K. Smaga. Your mediators need data conversion!. In Proceedings of ACM SIGMOD Conference, 1998.

[9] DB2XML home page : http://www.informatik.fh-wiesbaden.de/~turau/DB2XML/index.html.

[10] R.G.G. Catteli and Douglas K. Barry. The Object Database Standard : ODMG 2.0.

[11] E. Bertino, L. Martino. Object-Oriendted Data Models, Object-Oriented Database Systems : Concepts and Architectures, pp. 12-80. Addison-Wesley publishing company, 1993.

[12] 김태현, 김경일, 이강찬, 이규철. XML 기반의 정보 통합을 위한 OODB2XML 래퍼의 설계 및 구현. 정보과학회 가을 학술발표논문집 Vol. 26. No. 2, 1999.

[13] J. Shanmugasundaram, E. Shekita, R. Barr, M. Carey, B. Lindsay, H. Pirahesh and B. Reinwald. Efficiently Publishing Relational Data as XML Documents. In Proceedings of VLDB Conference, 2000.

[14] 윤정희, 객체지향 데이타베이스의 유효한 XML 문서로의 변환, 석사학위논문, 한국과학기술원, 2000.

윤 정 희

1998년 한국과학기술원 전산학과(학사). 2000년 한국과학기술원 전산학과(석사). 2000년 ~ 현재 한국과학기술원 전산학과 박사과정. 관심분야는 XML 질의 처리, XML 저장시스템

박 창 원

1992년 인하대학교 전자공학과(학사). 1992년 ~ 1993년 LG전자 VIDEO 연구소 연구원. 1995년 서강대학교 전자계산학과(학사). 1997년 한국과학기술원 전산학과(석사). 현재 한국과학기술원 전산학과 박사과정. 관심분야는 Optimizing and processing XML queries, Managing XML data, Managing Web Data

정 진 완

1973년 서울대학교 공과대학 전기공학과(학사). 1983년 University of Michigan 컴퓨터공학과(박사). 1983년 ~ 1993년 미국 GM 연구소 전산과학과 선임연구원 및 책임연구원. 1793년 ~ 현재 한국과학기술원 전산학과 부교수 및 교수. 관심분야는 XML, 멀티미디어 데이타베이스, GIS, 웹 정보검색, 객체지향 데이타베이스