컨텐츠 통합 관리 시스템에서의 스키마 충돌 해결 방안

이중화* · 권오준*

Schema Conflict Resolution Method on Content Integration Management System

Jung-hwa Lee* · Oh-jun Kwon*

요 약

컨텐츠 통합 관리 시스템에서 지역 스키마를 통합하기 위해서는 스키마 구조 파악이 쉽고 충돌 및 해결 방법에 대한 표현이 자유로워야 된다. 기존 연구들은 통합 방법이나, 충돌 유형, 충돌 해결책이 일관되지 않고 시스템 의존적이었다. 본 논문에서는 이러한 문제점을 해결하기 위하여, 구조적인 문서 표현이 가능하고 검색 및 상호간 교환이 용이한 XML을 이용하여 스키마를 통합 시 발생하는 충돌을 해결하는 방안을 제시한다.

ABSTRACT

We must have a schema conflict resolution method for integrating local schema on content integration and management system. But The past method is inconsistent and depends on local and global databases. In this paper, we propose the management method of schema confilct using XML that can represent and interchange the structual information of schema.

키워드

Content management system, Multi database, Database

Ⅰ. 서 론

사회가 산업사회에서 정보사회로 전환됨에 따라 디지털화 된 컨텐츠가 차지하는 비중이 날로 증가하고 있으며 인터넷 통신망의 지속적인 성장으로 인해 각종 컨텐츠에 대한 디지털화가 가속되면서 세계시장에서 디지털 컨텐츠의 중요성이 부각되고 있다[1,2].

디지털 컨텐츠 산업은 지식 집약적 산업으로 중소기업 위주의 미래산업구조에 적합하고, 관련분야에의 동반성장을 촉진하는 파급효과를 가지고 있다는 점과, 컴퓨터의 발달과 초고속 정보통신망의 구축 등 인프라 구축과 새로운 형태의 서비스제공에 힘입어 인터넷 컨텐츠 산업은 고도 성장이 기대된다[1,2,3].

그러나, 컨텐츠에 대해 국내에서는 컨텐츠의 원자재에 해당하는 제품에 대한 영상자료, 기록 등에 대한 체계적 보존과 정리가 미흡한 상황이며 디지털 컨텐츠의 제작 및 운영, 관리에 대한 방법론 및 도구들 역시 일관성 있는 방법이 제시되지 못하고 있는 실정이다[3]. 여기에 무엇보다도 컨텐츠 수요자와 제공자의 측면에서 효과적인 수요, 공급 방법이 제공되지 못하여 수요, 공급의 불균형을 초래하고 있다. 이러한 문제들은 컨텐츠를 효율적으로 통합 관리할 수 있는 방안의 부족에 기인한다고 할 수 있다.

따라서 웹사이트 구축 시의 컨텐츠의 부족으로 인

* 동의대학교 컴퓨터-소프트웨어공학부

접수일자 : 2005. 7. 1

한 어려움을 해소하고 각 시스템에 고립적으로 사용되고 있는 각종 디지털 컨텐츠의 재 사용성을 높이기 위해 디지털 컨텐츠를 통합 관리할 수 있는 컨텐츠 통합 관리 시스템이 필수적이다.

서로 다르게 서비스되고 있는 컨텐츠들을 공유하고 통합 관리하는데 가장 큰 걸림돌은 같은 내용의 정보일지라도 서로 다른 데이터베이스 스키마를 가지고 있다는 점이다. 따라서 다른 형태의 스키마 정보를 가지는 데이터베이스를 통합하기 위해서는 서로 다른 방식으로 표현되어 있는 데이터베이스 스키마들을 표준화된 방식으로 표현하고 이를 통해 서로 다른 스키마 모델과 개념 모델을 가지고 있기 때문에 각 지역 스키마 통합 시 발생하는 스키마 충돌을 해결하는 것이 필수적이다.[4,5,6,7,8].

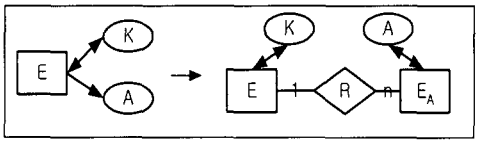
따라서, 본 연구에서는 최근 정보 교환의 표준으로 자리잡고 있는 XML을 이용하여 서로 다른 표현을 가지는 데이터베이스 스키마들을 표현하고, 이를 통해 스키마를 통합할 때 발생하는 충돌의 유형을 파악하고 해결 방안을 제시하여 하나의 통합 스키마를 생성함으로써 실제 저장되어 있는 컨텐츠를 통합할 수 있는 기반을 제공한다[9].

Ⅱ. 관련연구

다중데이터베이스는 여러 데이터 모델을 가지는 지역 스키마를 하나의 공통 모델을 가지는 전역 스키마로 통합하고 사용자는 통합된 전역 스키마를 통해 접근할 수 있도록 한다. 그러나 각 지역 시스템은 서로 다른 데이터 모델을 사용하고 같은 데이터에 대해 서로 다른 표현을 사용하고 있어, 스키마를 통합할 때 스키마 충돌이 일어난다. 따라서 같은 정보에 대한 서로 다른 표현 때문에 발생하는 충돌을 해결하여 단일 스키마를 만드는 스키마 통합 과정이 필요하다. 스키마를 통합할 때 발생하는 스키마 충돌을 해결하는 방법은 시스템이나 스키마 모델에 따라 많은 연구 결과가 발표되고 있다[5,6,7,8,10,11]

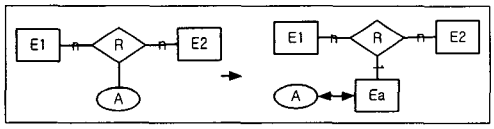
[10]에서는 스키마를 통합할 때 발생하는 충돌 유형을 개체 간 충돌, 애트리뷰트 간 충돌, 개체와 애트리뷰트 간 충돌, 같은 데이터에 대한 서로 다른 표현에 의한 충돌 등으로 분류하고 해결방법을 제시한다.

[11]은 두 개의 개체-관계(entity-relation) 모델을 통합할 때 발생하는 구조적 충돌을 해결하는 방법을 제시한다. 충돌의 종류는 크게 두 가지로 나누어지는데 첫째는 한 스키마에서는 개체로 선언되었지만 다른 스키마에서는 애트리뷰트로 선언된 경우와, 둘째는 관계 집합(relation set)의 애트리뷰트와 개체의 애트리뷰트간 충돌이 발생하는 경우이다. 모든 충돌은 새로운 개체의 애트리뷰트로 확장하는 방법을 이용하여 충돌을 해결한다.



[그림 2.1] 충돌 발생 시 스키마 변환 과정 I

[그림 2.1]은 애트리뷰트와 개체 사이에 충돌이 생겼을 때 해결하는 방법으로 애트리뷰트를 새로운 EA 개체의 애트리뷰트 A로 확장하고 관계 R로 연결하여 해결한다. 이렇게 분리된 EA 개체는 새로운 스키마와 통합된다.



[그림 2.2] 충돌 발생 시 스키마 변환 과정 II

[그림 2.2]는 충돌이 생긴 애트리뷰트가 개체가 아닌 관계에 속할 때 충돌을 해결하는 방법이다. 관계 R의 애트리뷰트 A는 새로운 개체 EA의 애트리뷰트로 변환된다.

충돌이 생긴 스키마를 위의 변환 과정을 거쳐 변환한 다음, 두 스키마를 병합 (merge)하여 새로운 스키마를 만든다. [10]은 스키마 통합 시 스키마가 내포하는 의미 손실이 발생하지 않는 장점이 있지만, 데이터 모델 단계에서의 충돌해결 방법만을 제시하고 있다.

이상에서 살펴본 스키마 통합을 위한 기존 연구들은 시스템 의존적인 방법을 제시하고 있기 때문에 시스템 독립적인 통합 방법 및 스키마 표현 방법에 대한 연구가 필요하다. 그리고 실제 시스템을 구현하기 위해서는 충돌을 해결하기 위해 필요한 스키마 구조 정보의 표현과 검색이 쉽도록 해야 할 필요가 있다.

따라서 본 논문에서는 시스템 독립적이고, 구조적

표현이 용이한 웹 문서 표준인 XML을 이용하여 지역 스키마와 전역 스키마를 표현하고, 스키마를 통합할 때 발생하는 스키마 충돌을 XML로 표현하여 충돌을 해결하는 방법을 제안한다.

Ⅲ. XML을 이용한 스키마 표현

XML은 특정 시스템에 의존적이지 않으며, 구조와 관련 정보의 의미적 표현이 자유롭고, 웹 에이전트를 통한 자동화 처리가 가능하다는 장점이 있다[9].

위와 같은 XML의 특징 중 특히, 구조 표현이 쉽고, 데이터 교환을 용이하게 할 수 있는 특징은 다중 데이터베이스에서 서로 다른 지역 시스템 간 데이터 교환을 쉽게 하고, 지역 스키마 및 전역 스키마의 구조를 효과적으로 표현할 수 있게 한다.

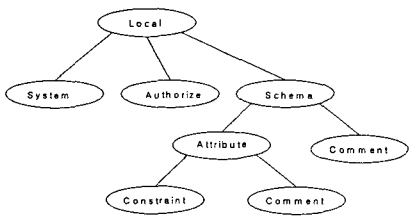
따라서 본 연구에서는 각 지역 컨텐츠 데이터베이스에서의 지역 스키마와 전역 스키마를 XML로 표현하고, 지역 스키마 통합 시 발생하는 스키마 충돌을 XML로 표현하여 충돌을 해결하고자 한다.

3.1 XML을 이용한 지역 스키마 표현

각 사이트에 저장되어 있는 지역 데이터베이스 컨텐츠를 공유 및 통합을 위해 지역 데이터베이스 XML 문서로 변환할 필요가 있다.

본 연구에서는 지역 스키마의 XML 변환을 위해 변환된 XML 문서의 구조를 정의한 DTD(Document Type Declaration)를 작성하고 이를 통해 변환된 XML 문서의 유효성(validation)을 검사한다.

각 지역 사이트에 있는 지역 스키마의 구조를 XML 문서로 변환하기 위한 XML 문서 구조를 엘리먼트 트리로 나타낸 것이 [그림 3.1] 이다.



[그림 3.1] 지역 스키마 엘리먼트 트리

위의 [그림 3.1]의 엘리먼트 트리에 대한 DTD는 [그림 3.2]와 같이 표현되며 각 지역 스키마는 지역 스키마 DTD를 통해 XML형식으로 작성된다. 작성된 XML 문서는 전역 데이터베이스에 전송되어 지역 데이터베이스 스키마 카탈로그에 저장된다. 전역 시스템으로 보내어지는 지역 시스템 정보는 스키마 구조뿐만 아니라, 지역 데이터베이스 시스템이 사용하고 있는 데이터베이스의 종류, 지역 시스템에 대한 정보, 데이터베이스에 접근하기 위한 허가, 스키마의 구조, 스키마 내에 존재하는 제약조건 등을 표현할 수 있어야 한다.

```
<!ELEMENT local (system, authorize, schema+) >
<!ELEMENT system (database) >
<!ATTLIST system
          ip          CDATA   #REQUIRED
          port        CDATA   #REQUIRED >
<!ELEMENT database (#PCDATA) >
<!ATTLIST database
          version     CDATA       #REQUIRED >
<!ELEMENT authorize EMPTY >
<!ATTLIST authorize
          user        CDATA   #REQUIRED
          passwd      CDATA   #REQUIRED >
<!ELEMENT schema (attribute+, comment) >
<!ATTLIST schema
          name        CDATA   #REQUIRED
          extend      CDATA   #IMPLIED >
<!ELEMENT attribute (#PCDATA, constraint*,
                     comment) >
<!ATTLIST attribute
          type        CDATA   #REQUIRED
          length      CDATA   #REQUIRED >
<!ELEMENT constraint EMPTY >
<!ATTLIST constraint
  value   (PK|FK|UK|NN|MN) #IMPLIED
          table       CDATA       #IMPLIED
          name        CDATA       #IMPLIED >
<!ELEMENT comment (#PCDATA) >
```

[그림 3.2] 지역 스키마(local schema) DTD

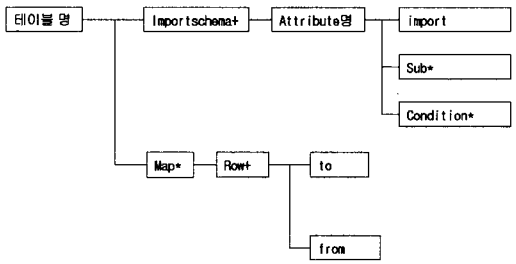
지역 스키마를 나타내는 XML 문서는 최상위 요소로 <local> 요소를 가진다. <local> 요소는 시스템 정보, 사용자 정보, 스키마 정보를 나타내는 구조로 되어 있다. 시스템 정보를 나타내는 <system> 요소는 시스템에 접근하기 위한 접근 포트(port) 와 시스템의 IP 주소를 속성으로 가지며, 내부 요소인 <database> 요소는 지역 데이터베이스의 제품 정보를 가진다. <authorize> 요소는 지역 스키마의 접근 권한을 확인하기 위해 <user>와 <passwd> 속성을 이용한다. <schema> 요소는 지역 스키마 구조를 나타내는 요소로서, 스키마 내에 여러 테이블이 있다면 <schema> 요

소를 테이블 수만큼 정의한다. <schema> 요소 내의 <extend> 속성은 객체지향 데이터베이스에서 상속의 개념을 지원하기 위해 사용한다. <attribute> 요소는 테이블내의 속성을 표현하며, <type> 과 <length> 속성을 이용하여 속성의 데이터형을 나타낸다. <attribute> 요소는 테이블 속성의 제약 조건을 나타내기 위해 하나 이상의 <constraint> 요소를 사용한다. 제약 조건은 기본 키 (PK), 외래 키(FK), 유일한 값 (NN), 널 (null)이 아닌 값 (NN) 에 대해서만 다룬다. 마지막으로 <comment> 요소는 해당 테이블이나 속성에 주석을 달고자 할 때 사용한다.

3.2 XML을 이용한 전역 스키마 표현

전역 스키마는 서로 다른 지역 스키마를 하나의 스키마로 통합하여 사용자로 하여금 단일 인터페이스를 제공하게 한다. 본 논문에서 제안하는 전역 스키마 DTD는 통합된 스키마 구조뿐만 아니라, 스키마를 통합할 때 발생하는 충돌을 해결하기 위한 방법을 표현할 수 있도록 설계한다.

전역 스키마 DTD 내에는 [그림 3.3]과 같은 DTD 구조가 전역 스키마에 있는 테이블 수만큼 존재한다.



[그림 3.3] 전역 테이블에 대한 DTD 구조

전역 테이블에 대한 DTD 구조에서 <table 명> 요소는 전역 스키마의 테이블 이름을 나타낸다. 실제, 전역 스키마 문서에는 <table 명> 요소는 테이블 이름으로 치환된다. 각 테이블은 통합되는 지역 스키마의 수만큼 <importschema> 요소를 가지며, <importschema> 요소는 지역 시스템으로부터 받는 스키마(import schema)에 대한 정보를 가진다.

만약 충돌이 발생했을 때 해결방법 중 변환표(mapping table)가 필요하면 <map> 요소를 사용한다. <map> 요소는 <to> 요소와 <from> 요소의 값이 일대일 대응관계를 갖도록 되어 있다. 충돌이 생긴 속성을 참조하기 위해서는 <from> 요소의 값과 일대일 대응되는 <to> 요소의 값으로 변환된 후 참조하게 된다. <attribute명> 요소는 테이블의 속성을 나타내며 실제 속성 이름으로 치환된다. 각 속성은 지역 스키마의 어느 테이블과 속성으로부터 통합되는지, 충돌 종류는 무엇인지, 그리고 그 충돌을 해결하기 위해서 어떤 정보가 필요하고 어떻게 해결해야하는지를 <import>, <sub>, <condition> 요소에서 나타낸다.

이 세 가지 요소에 대한 상세 DTD는 [그림 3.4]와 같다.

```
<!ELEMENT import  EMPTY >
<!ATTLIST import
          id            ID          #REQUIRED
          sub_refs      IDREFS      #IMPLIED
          con_refs      IDREFS      #IMPLIED
          table         CDATA       #REQUIRED
          name          CDATA       #REQUIRED
          type          CDATA       #REQUIRED
          length        CDATA       #REQUIRED
          category      CDATA       #IMPLIED
          operator      CDATA       #IMPLIED
          operand       CDATA       #IMPLIED
          reference     CDATA       #IMPLIED  >
<!ELEMENT sub  EMPTY>
<!ATTLIST  sub
          id            ID          #REQUIRED
          ref           IDREF       #REQUIRED
          category      CDATA       #IMPLIED
          operator      CDATA       #IMPLIED
          operand       CDATA       #IMPLIED
          reference     CDATA       #IMPLIED >
<!ELEMENT condition EMPTY>
<!ATTLIST condition
          id            ID          #REQUIRED
          ref           IDREF       #REQUIRED
          table         CDATA       #REQUIRED
          name          CDATA       #REQUIRED
          operator (%comparison;)   #REQUIRED
          action        (AND|OR)    #IMPLIED
          value         CDATA       #IMPLIED >
```

[그림 3.4] import,sub,condition 요소 DTD

<import> 요소의 속성 <table>, <name>, <type>, <length>는 통합되는 지역 스키마와 데이터형의 종류를 나타낸다. 이 속성에 의해 이름, 데이터형 충돌을 해결할 수 있다. <sub_refs>와 <con_refs> 속성은 <sub>와 <condition> 요소가 있을 때 서로 참조하기 위해 사용한다. <import> 요소에서 가장 중요한 속성들은 충돌의 유형을 결정짓는 <category> 속성과, 충돌의 유형에 따라 사용용도가 결정되는 <operator>, <operand>,

<reference> 속성이 있다.

<sub> 요소는 하나 이상의 충돌이 생겼을 경우 <import> 요소에 추가로 선언한다. <condition> 요소는 <import> 요소의 지정된 속성을 추출하기 위한 조건을 표현하기 위해 사용한다.

Ⅳ. 스키마 충돌 해결 방안

4.1 XML을 이용한 스키마 충돌 표현 방법

충돌을 해결하기 위해서는 충돌의 유형을 정의하고, 충돌 유형에 따라 각각 충돌을 해결하기 위해 필요한 정보가 무엇이며, 충돌이 생겼을 때 어떻게 해결해야 하는지에 대한 정책이 있어야 한다. 본 연구에서는 충돌의 종류를 이름, 데이터형의 충돌, 동일한 데이터에 대한 다른 표현, 구조적 충돌의 세 가지 큰 범주로 나누고 각 범주에 맞는 충돌 해결 방법을 제안한다.

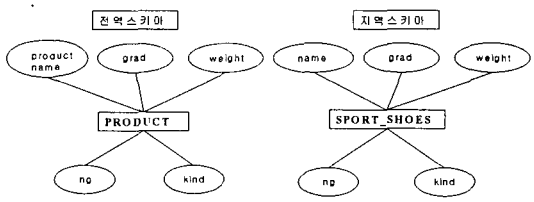
충돌을 표현할 때 앞에서 제시한 [그림 3.5]의 전역 스키마 DTD에서 <import> 요소의 <category> 속성 값에 의해 충돌 유형과 해결방법이 결정된다.

충돌의 종류와 각 충돌 유형에 대한 <category> 속성이 가질 수 있는 속성 값은 [표 4.]과 같다.

[표 4.1] 충돌의 종류와 category 속성 값

충돌의 종류	충돌 유형	영역	관련 속성
이름, 데이터 형의 충돌	다른 이름	NN	name
	다른 데이터 형	NN	type, length
동일한 데이터에 대한 다른 표현	서로 다른 표현	DE	reference
	서로 다른 단위	DU	operator, operand
	서로 다른 정밀도	DP	reference
구조적 충돌	합쳐야 하는 속성	CC	reference
	복합 속성	CA	reference
	존재하지 않지만 뜻을 내포한 속성	MI	reference

충돌 유형, 즉 <category> 값에 따라 충돌 해결을 위한 정보가 어떻게 표현되는지 설명하기 위해 [그림 4.1]의 스키마를 보기로 들어 설명한다. [그림 4.1]의 지역 스키마가 또 다른 지역 스키마들과 통합되어 전역 스키마가 될 때 지역 스키마의 충돌 표현 방법을 설명한다.



[그림 4.1] 지역 스키마와 통합된 전역 스키마

1) 이름 , 데이터 형 충돌 (NN)

이름과 데이터 형 충돌은 가장 기본적인 충돌로서, <import> 요소의 <table>, <name>, <type>, <length> 속성에 의해 해결한다. 충돌 표현 방법은 다음과 같다.

```
<product id="T001">
 <importschema rownum="1" db="LDB1">
   <prodict_name id="A01" type="string"
                 length="20">
       <import id="imp11" table="sports_shoes"
                  name="name" type="varchar"
                 length="10" category="NN" />
    </product_name>
  </importschema>
</product>
```

2) 같은 값에 대한 서로 다른 표현 (DE)

지역 스키마의 등급은 "1등급", "2등급", "3등급" 의 값을 가지는데, 전역 스키마의 등급은 "A등급", "B등급", "C등급" 이라는 값을 가질 때 지역 스키마의 값을 전역 스키마의 값에 맞게 변환해야 하는데 이와 같이 같은 값에 대해 서로 다른 표현을 가질 때 충돌 표현 방법은 다음과 같다.

```
<product id="T001">
  <importschema rownum="2" db="LDB1">
     <product_grade id="A02" type="string"
                                length="20">
        <import id="imp12"
             table="sports_shoes" name="grade"
           type="varchar" length="20"
            category="DE" reference="m01" />
     </product_name>
  </importschema>
  <map id="m01">
     <row id="r21"> <to>A</to>
                <from>1</from> </row>
     <row id="r22"> <to>B</to>
                <from>2/from> </row>
     <row id="r23"> <to>C</to>
                <from>3/from> </row>
  </map>
</product>
```

<category> 값이 "DE"일 때는 <import> 요소의 <reference> 속성 값에 의해 <map> 요소를 참조하게 된다. <map> 요소에는 일대일 대응관계가 있는 <to> 요소와 <from> 요소가 있어 충돌이 생긴 값을 변환하게 된다. 만약, 지역 스키마로부터 "1"의 값을 가지는 grade 속성을 추출했다면, <from> 요소의 "1"과 일대일 대응관계에 있는 <to> 요소의 "A"의 값으로 변환된다.

3) 같은 값에 대한 서로 다른 단위 (DU)

지역 스키마 SPORTS_SHOES의 무게(weight)는 단위로 파운드(lb)를 사용하지만 전역 스키마 PRODUCT의 무게(weight)는 kg을 사용한다고 할 때, 충돌 표현 방법은 다음과 같다.

```
<product id="t001">
  <importschema rownum="3" db="LDB1">
    <product_weight id="A03" type="float"
                    length="3">
      <import id="imp13"
        table="sports_shoes" name="weight"
              type="number" length="3"
              category="DU" operator="M"
             operand="2.2" />
    </weight>
  </importschema>
</product>
```

'1 kg = 2.2 lb' 라고 하면, <import> 요소의 <operator> 속성의 "M" 과 <operand> 속성의 "2.2"를 이용하여 단위를 변환한다. <operator> 속성의 값은 곱하기(M), 나누기(D), 더하기(A), 빼기(S) 중에서 하나의 값을 가진다. 단위 변환에 하나 이상의 연산자가 필요하면 <sub> 요소를 추가하여 해결한다.

4) 같은 값에 대한 서로 다른 정밀도 (DP)

지역 스키마(sports_shoes)의 등급(grade)은 "1" 에서 "4" 의 값을 가지는데 전역 스키마 (product)의 등급(grade)은 "A" 와 "B" 와 같이 서로 다른 정밀도를 가질 때 의 충돌 표현 방법은 다음과 같다.

```
<product id="T001">
  <importschema rownum="2" db="LDB1">
    <product_grade id="A02" type="string"
                   length="20">
      <import id="imp12"
        table="sports_shoes" name="grade"
        type="varchar" length="20"
        category="DE" reference="m02" />
    </product_name>
  </importschema>
  <map id="m02">
    <row id="r21"> <to>A</to> <from>
    <ub>1</ub><lb>2</lb> </from> </row>
    <row id="r22"> <to>A</to> <from>
    <ub>3</ub><lb>4</lb> </from> </row>
  </map>
</product>
```

<category> 속성의 값이 "DE"인 것과 같이 <reference> 속성 값 "m02"에 의해 <map> 요소를 참조하게 된다. 단지, <map> 요소의 <from> 요소에 범위를 지정하기 위한 <ub>와 <lb> 요소가 추가되었다. 지역 스키마(sport_shoes)의 등급(grade) 이 <from> 요소 안의 <ub> 요소의 값과 <lb> 요소 값 사이에 있다면, 일대일 대응 관계에 있는 <to> 요소의 값으로 변환된다.

5) 합쳐야할 속성 (CC)

지역 스키마(sports_shoes)는 제품이름이 만약 lname, fname으로 구별되어 있다고 하자. 이 때 전역 스키마(product)는 합쳐진 이름(name)을 가질 때 다음과 같이 충돌을 표현할 수 있다.

```
<product id="T001">
  <importschema rownum="1" db="LDB1">
    <product_name id="A01" type="string"
                  length="15">
      <import id="imp12"
        table="sports_shoes" name="lname"
        type="varchar" length="15"
        category="CC" reference="fname" />
    </product_name>
  </importschema>
</product>
```

<import> 요소의 <name> 속성과 <reference> 속성 값을 합쳐서 전역 스키마의 product_name 속성을 구성하게 된다. 즉, 두 개의 속성 (lname, fname)를 추출한 다음, 두 문자열을 합치는 문자열 결합 연산을 이용하여 속성 각을 합친다.

6) 복합 속성 (CA)

지역 스키마(sport_shoes)의 재질(ng)이 객체 지향 데이터베이스에서 내부 재질 (inner_ng)과 외부 재질을 (outer_ng)를 인스턴스로 가지는 클래스이고, 전역 스키마 (product)의 재질(ns)은 단지 통합된 재질(ng) 만을 가지는 속성일 때, 복합 속성을 단순화 시켜 표현해야 한다. 충돌 표현 방법은 다음과 같다.

```
<product id="T001">
  <importschema rownum="1" db="LDB1">
    <product_ng id="A04" type="string"
                length="20">
      <import id="imp13"
        table="sports_shoes" name="ng"
        type="object" length="1"
      category="CA" reference="inner_ng" />
      <import id="imp14"
        table="sports_shoes" name="ng"
      type="object" length="1"
    category="CA" reference="outter_ng" />
    </product_ng>
  </importschema>
</product>
```

<import> 요소의 <type> 속성은 "object" 값을 가지며 <name> 속성에 의해 속성 값을 구하는 일반적인 경우와 달리 최종적인 속성은 <reference> 속성 값에 즉, sport_shoes.ng.inner_ng에 의해 product_ng의 값을 구한다.

7) 속성은 없지만 뜻은 내포 (MI)

지역 스키마(short_shoes)는 운동화만을 저장하기 위한 스키마이고, 전역 스키마(product)는 모든 제품을 포함한 스키마라 할 때, 전역 스키마의 제품종류(kind)는 제품의 종류를 구분하는 값을 가진다. 지역 스키마에는 제품의 종류를 나타내는 kind 속성은 없지만 스키마에 운동화라는 뜻을 내포하고 있기 때문에 다음과 같이 충돌을 표현할 수 있다.

```
<product id="T001">
  <importschema rownum="1" db="LDB1">
    <product_kind id="A05" type="string"
length="20">
      <import id="imp14" table="sports_shoes"
name="kind"
        type="varchar" length="20"
category="MI" reference="운동화" />
    </product_kind>
  </importschema>
</product>
```

전역 스키마의 kind는 기본 값으로 <reference> 속성 값인 "운동화"를 가진다.

8) 충돌 해결 불가능 (NA)

충돌을 해결할 방법이 없는 경우로써, 해당 속성에 대한 전역 질의가 있을 경우 질의 결과는 "NA"의 값을 반환하게 된다.

이상에서 지역 스키마와 전역 스키마를 구조적 표현이 용이한 XML을 이용하여 표현하고 , 스키마를 통합할 때 발생하는 스키마 충돌 정보를 XML로 표현하여 전역 스키마 내에 포함하는 방법을 살펴보았다.

4.2 스키마 충돌 해결 기법

본 연구에서는 각 사이트에 존재하는 지역 스키마를 전역 스키마로 통합할 때 발생하는 충돌의 종류를 분석하여 위와 같이 이름, 데이터형의 충돌, 동일한 데이터에 대한 다른 표현, 구조적 충돌의 세 가지 큰 범주로 나누어 설명하였다.

스키마를 통합할 때 충돌이 발생하면 발생하는 충돌이 어떤 충돌 유형에 속하는 지를 파악하고 각 유형에 맞게 충돌을 해결하여야 한다.

아래에서는 앞에서 분류한 스키마 충돌 유형에 따른 해결하는 방법을 제시한다.

1) 이름, 데이터형 충돌

■ 이름 충돌(name conflict)

이름 충돌이 발생할 경우 다중데이터베이스는 전역 데이터베이스 스키마에서 사용할 이름을 정하고 각 지역 스키마의 속성 이름과 대응(mapping)되는 이름의 대응관계를 목록(catalog) 에 기록함으로써 해결한다. 앞에서 살펴본 게시판 테이블에서 사용자 이름을 예로 살펴보면, 두 지역 사이트에 저장된 테이블의 속성이 name과 username으로 서로 다른데, 이 경우 전역 스키마에서는 위의 두 이름 중에서 USERNAME을 사용하도록 한다. 전역 스키마의 이름을 정할 때는 전역 스키마 설계자가 이름을 정하거나, 이름이 외부적으로 이름이 정해지지 않았을 경우는 통합될 지역 사이트에서 인스턴스(instance)가 많은 사이트의 이름을 채택한다. [표 4.2]는 이름 대응 관계 목록의 예이다.

[표 4.2] 이름 대응 관계 목록(Catalog)

전역 스키마 속성 이름	지역 사이트 이름	지역 테이블 이름	지역 테이블 속성 이름
ID	potal_1	freeboard	id
	poral_2	board	no
USERID	potal_1	freeboard	userID
	potal_2	board	ID
USERNAME	potal_1	freeboard	name
	potal_2	board	username

■ 데이터형 충돌(data type conflict)

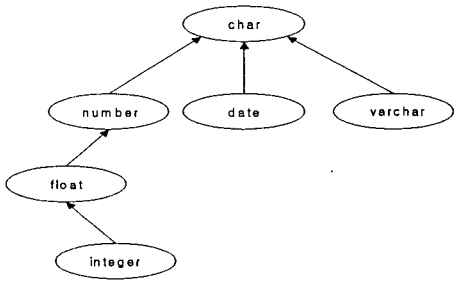
데이터형 충돌을 해결하기 위해서는 데이터 형 변환 규칙이 있어야 한다. 변환 규칙은 시스템 설계자가 정보 손실을 최소로 하는 방법을 선택하여 결정하게 된다.

DBMS마다 다양한 데이터형을 제공하고 있지만 실제로 데이터베이스 설계 시에는 char, varchar, number, int, float 등의 6가지 데이터형의 사용이 거의 대부분이다.

[그림 4.2]는 데이터베이스 설계 시 가장 많이 사용하는 5가지의 데이터 형에 대해 형 변환 규칙을 나타낸 트리이다.

데이터형의 변환은 아래 그림과 같은 형 변환 트리를 이용하여 수행할 수 있는데, 형 변환 규칙은 루트노드에 가까울수록 우선순위가 높으므로 단말노드에서

루트노드로 증가하면서 통합이 이루어진다. 즉 int형과 float형의 통합은 float로 float와 number형의 통 합은 number로 변환된다. 형 변환 트리에서 이웃하는 노드들의 통합은 부모노드의 데이터형으로 이루어진다. 즉, int형과 data형은 date형의 부모 노드인 char형으로 변환된다.



[그림 4.2] 데이터형 변환 트리

데이터형의 길이가 서로 다른 경우는 손실을 최소화하는 방향으로 통합하기 위해 두 데이터형의 길이 중에서 긴 쪽을 선택하게 된다. 예를 들면, char(5)와 char(10)이 통합되면 char(10)이 되고, integer(15)과 float(10.2)는 float(15.2) 로 변환되어야 한다는 규칙을 가질 수 있다.

그 외의 데이터형은 모두 형 변환 트리의 가장 상위 노드인 char형으로 변환되도록 한다.

2) 동일한 데이터에 대한 서로 다른 표현 충돌

동일한 데이터에 대해 발생하는 충돌은 같은 속성 값에 대해 서로 다르게 표현되는 경우와 같은 값에 대해 서로 다른 단위를 사용하는 경우, 그리고 같은 값에 대해 서로 다른 정밀도를 사용하는 경우이다. 두 가지의 서로 다른 형태의 제품 테이블을 충돌을 해결하지 않고 그대로 통합한 [그림 4.3]을 예로 들어 살펴보겠다.

상품명	등급	크기	무게	...
Circle1	A	155	450	
Trek	C	165	300	
Novel	B	145	560	
NNN	1	14.5	1.2	
Cirus	3	16.5	3.3	
Olympic	2	17	2.5	
:	:	:	:	

[그림 4.3] 두 가지 서로 다른 제품테이블을 통합한 예

■ 같은 값에 대한 서로 다른 표현

같은 값에 대한 서로 다른 표현을 가지는 경우는 동일한 속성에 대해 같은 의미를 가지지만 서로 다르게 표현되어 있어서 통합이 이루어지면 서로 의미가 통하지 않게 되는 경우이다. 이 경우는 [그림 4.4]와 같이 같은 의미를 가지는 값에 대한 변환 표(mapping table)를 이용하여 해당 값을 대응시키도록 함으로써 해결할 수 있다.

전역 스키마 속성	지역 사이트	지역 테이블	속성 이름
상품명	potal_1	제품1	상품명
	poral_2	제품2	상품명
등급	potal_1	제품1	등급
	potal_2	제품2	등급
:			

이전값	새로운값
1	A
2	B
3	c

[그림 4.4] 변환 테이블을 이용한 속성 값 변환

■ 같은 값에 대한 서로 다른 단위

같은 값에 대한 서로 다른 단위를 가지는 경우는 서로 다른 단위를 가지는 크기와 무게의 경우인데, 이 경우는 변환 공식을 이용하여 해결하면 된다. 이 경우는 스키마 통합 후 실제로 데이터베이스가 통합될 때 주어진 변환공식에 의해 값이 변화된다. 예를 들어 1 kg = 2.2 lb 의 도량형 환산표를 이용하면 파운드(lb)와 kg을 통합할 수 있다.

■ 값은 값에 대한 서로 다른 정밀도

같은 값에 대한 서로 다른 정밀도의 경우는 같은 값에 대한 서로 다른 표현의 충돌과 비슷하지만 일대일 대응이 되지 않기 때문에 구분된다. 따라서 일대일 대응 값이 아니라 범위 값을 주어서 해결한다. 즉, 'A' 등급은 A+, A등급을 포함하고 B 등급은 B+, B등급을 포함한다는 것을 나타내는 변환 표가 필요하다. 이 경우 충돌해결 방법은 정밀도가 높은 것에서 낮은 것으로만 가능하고, 그 반대의 경우 정보의 손실이 발생한다.

3) 구조적 충돌

구조적 충돌은 서로 다른 데이터 모델을 사용하여

스키마 구조의 표현력 차이에서 발생하거나 같은 데이터 모델일 경우에서도 스키마 구조를 서로 다르게 설계하였을 때 발생하는 충돌이다. 앞에서 살펴본 바와 같이 구조적 충돌에는 합쳐야할 속성, 복합 속성, 속성의 뜻을 내포한 구조 등이 이에 해당한다.

■ 합쳐야 하는 속성

합쳐야 하는 속성에 의한 충돌은 해당하는 합쳐야 하는 속성에 대한 정보를 미리 파악하여 통합이 이루어지기 전에 선 처리(pre-processing)를 통해 두 속성을 하나의 값으로 합친다. 이때 속성의 내용은 문자열을 합치는 연산을 이용하여 여러 개의 속성 값을 하나의 속성 값으로 만들어야 한다. 즉, 지역 스키마의 {성,이름} 이 전역 스키마에서 {이름}으로 합쳐질 때, 전역 스키마로의 통합이 이루어지기 전에 지역 스키마 내에서 두 속성을 하나의 {성-이름} 속성으로 변환하고 해당 속성이 어떤 속성의 결합인지를 기록해 둔 다음 실제 데이터베이스가 통합될 때 기록해 둔 정보를 이용하여 하나의 속성으로 통합하게 된다.

■ 복합 속성

복합 속성의 경우는 통합 데이터베이스의 데이터 모델에 따라 달라지게 되는데, 통합 데이터베이스의 데이터 모델이 객체-관계형이거나 객체지향 데이터 모델일 경우는 그대로 두면 되고 관계형 데이터 모델을 사용하는 경우에는 관계형 데이터 모델에 맞게 변환이 필요하다. 이 때 변환은 [그림 4.5]와 같이 복합 속성을 단순속성으로 변환하는 과정과 변환된 단순 속성을 하나의 속성으로 합치는 과정의 두 단계로 나누어 이루어지게 된다.

[1단계 변환]

주소			
시	구	동	번지
부산시	부산진구	가야동	123-45

=>

주소.시	주소.구	주소.동	주소.번지
부산시	부산진구	가야동	123-45

[2단계 변환]

주소.시	주소.구	주소.동	주소.번지
부산시	부산진구	가야동	123-45

=>

주소
부산시 부산진구가야동123-45

[그림 4.5] 복합 속성의 변환 과정

■ 속성은 존재하지 않지만 뜻을 내포

이와 같은 경우는 전역 스키마 설계자가 임의로 기본 값을 지정해야 한다. 만약 내포하는 뜻이 없어 지정할 수 없으면 통합 불가능으로 처리하고 값으로 널(null)을 넣을 수 있다.

V. 결론

현재 웹사이트에서 운영되고 있는 컨텐츠들은 여러 종류의 DBMS를 사용하여 저장되고 있으며 사용하는 DBMS 및 운영체제에 따라 서로 다른 형식을 가진다. 이 경우 서로 통합 관리를 위해 각 지역에 저장되어 있는 지역 스키마를 통합할 때 어려움이 따르게 된다.

본 연구에서는 각 지역 스키마를 통합할 때 발생하는 충돌의 유형을 분류, 정의하고 각 충돌에서 발생하는 문제점을 분석하여 통합 스키마를 생성하기 위한 충돌 해결 기법을 제시하였다.

본 연구에서는 스키마를 통합할 때 발생하는 충돌을 해결하기 위한 정보를 전역 스키마 내에 XML로 표현함으로써, 스키마의 구조 표현과 검색이 쉬울 뿐 아니라, 시스템 독립적이기 때문에 스키마 구조를 XML로 변환하는 변환기만 있으면 어떤 시스템과도 정보 교환이 가능하며 다중데이터베이스로 통합이 가능하다.

향후 연구과제로는 전역 스키마에 대한 제약 조건을 표현하는 방법이 필요하며 지역 스키마가 변경되었을 때 전역 스키마를 효율적으로 변경할 수 있는 알고리즘을 개발해야 한다.

참고문헌

[1] 박추환, 신현문, 국내외 디지털 컨텐츠 시장의 유료화 현황 및 활성화 방안, http://new.itfind.or.kr/KIC/etlas/industry/990/99002.htm, 2003

[2] 정보통신정책연구원, 디지털 컨텐츠 중장기 육성 전략 수립사업 연구보고서, 2002

[3] 김병곤, 이동만, 박순창, 국내 멀티미디어 컨텐츠 산업의 육성방안에 관한 탐색적 연구

[4] Bright, M.W., Hurson, A.R., and Pakzad, S., Automated Resolution of Semantic Heterogeneity in

Multidatabases, ACM Trans. on Database systems, vol 19(2), 1998,p212-253

[5] M.L. and Ling, T.W., Resolving Structural Conflicts in the Integration of Entity-Relationship Schemas, Proc. of the 14th Int. Conf. on OOER'95, Gold Coast, Australia, pages 422-433.

[6] Pyeong S. Mah, Soon M. Chung, Schema integration and transaction management for multidatabases, Information Sciences 111(1998) pp.153-188

[7] M.L. and Ling, T.W., Resolving Structural Conflicts in the Integration of Entity-Relationship Schemas, Proc. of the 14th Int. Conf. on OOER'95, Gold Coast, Australia, pp. 422-433.

[8] Pyeong S. Mah, Soon M. Chung, Schema integration and transaction management for multidatabases, Information Sciences ,1998, pp.153-188

[9] W3C XML specification, http://www.w3.org/TR/1998/REC-xml-19980210/

[10] Won Kim, Modern Database Systems: The Object Model, Interoperability and Beyond, ACM Press,1995 http://www.w3.org/XML/Activity.html

[11] M.L. and Ling, T.W., Resolving Structural Conflicts in the Integration of Entity-Relationship Schemas, Proc. of the 14th Int. Conf. on OOER'95, Gold Coast, Australia, 1995, pp 422-433.

저자소개

이중화(Jung-hwa Lee)

1992 부산대학교 전자계산학과 (이학사)
1995 부산대학교 전자계산학과 (이학석사)
2001 부산대학교 전자계산학과 (이학박사)
2002.3 - 현재 동의대학교 소프트웨어공학과 조교수
※관심분야 : 데이터베이스, XML, 시맨틱 웹

권오준(Oh-jun Kwon)

1986.2 경북대학교 전자공학과 (전산 전공) (공학사)
1992.2 충남대학교 전산학과 (이학석사)
1998.2 포항공대 전자계산학과 (공학박사)
1986.1-2000.2 한국전자통신연구원 선임연구원
2002.3 - 현재 동의대학교 소프트웨어공학과 조교수
※관심분야 : 데이터베이스, XML