컨텐츠 통합 관리 시스템에서의 스키마 충돌 처리

이중화* · 문경희** · 윤홍원**
*동의대학교 컴퓨터 · 소프트웨어공학부, **신라대학교 컴퓨터정보공학부

Management of Schema Conflict on Content Integration and Mangement System

Jung-hwa Lee* · Kyong-hi Moon**, Hong-won Yun**
*Dong-eui Univ., Division of Computer · Software Engineering,
**Silla Univ., Division of Computer & Information Engineering.
E-mail : junghwa@deu.ac.kr, {khmun,hwyun}@silla.ac.kr

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 통합시 발생하는 충돌을 해결하는 방안을 제시한다.

ABSTRACT

We must have management method of schema conflict 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
Multi database, Database, Content management system

I. 서 론

웹 상에서 서비스되고 있는 컨텐츠들은 각각의 특징적인 스키마 구조를 가지고 있기 때문에 컨텐츠들을 통합 관리하는데 많은 어려움이 따른다.

따라서 다른 형태의 스키마 정보를 가지는 데이터베이스를 통합하기 위해서는 데이터의 저장 구조인 스키마를 통합하여 전역 스키마를 생성하는 것이 가장 먼저 해야할 일이다[1,2]. 이때, 각 지역 스키마는 서로 다른 스키마 모델과 개념 모델을 가지고 있기 때문에 스키마 통합 시 스키마 충돌이 발생할 수 있다[3,4]

본 연구에서는 최근 정보 교환의 표준으로 자리 잡고 있는 XML을 이용하여 서로 다른 표현을 가지는 데이터베이스 스키마들을 통합하여 하나의 통합 스키마를 생성함으로써 실제 저장되어 있는 컨텐츠를 통합할 수 있는 기반을 제공한다.

XML은 특정 시스템에 의존적이지 않으며, 구조와 관련 정보의 의미적 표현이 자유롭고, 웹 에이전트를 통한 자동화 처리가 가능하다는 장점이 있다. 특히 태그의 확장성, 중첩된 구조를 이용한 데이터베이스 정보의 표현 가능성을 이용해 이기종 데이터베이스간의 데이터 교환에 사용할 수 있다[5, 6].

II. XML을 이용한 스키마 표현

2.1 XML을 이용한 지역 스키마 표현

각 사이트에 저장되어 있는 지역 데이터베이스 컨텐츠의 공유 및 통합을 위해 지역 데이터베이스

XML 문서로 변환할 필요가 있다.

본 연구에서는 지역 스키마의 XML 변환을 위해 XML 문서의 구조를 정의한 DTD(Document Type Declaration)를 작성하고 이를 통해 변환된 XML 문서의 유효성(validation)을 검사한다[그림 1].

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 등이 포함된다.

```
<!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 1] 지역 스키마(local schema) DTD

2.2 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 설계하였다.

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 해결해야 하는지를 <import>, <sub>, <condition> 요소에서 나타낸다.

이 세 가지 요소에 대한 상세 DTD는 [그림 2]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 2] 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.

III. 스키마 충돌 해결 기법

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

아래에서는 각 스키마 충돌 유형에 따른 해결하

는 방법을 제시한다.

3.1 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. 전역 스키마의 이름을 정할 때는 전역 스키마 설계자가 이름을 정하거나, 이름이 외부적으로 정해지지 않았을 경우는 통합될 지역 사이트에서 인스턴스(instance)가 많은 사이트의 이름을 채택한다. [그림 3]은 이름 대응 관계 목록의 예이다.

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

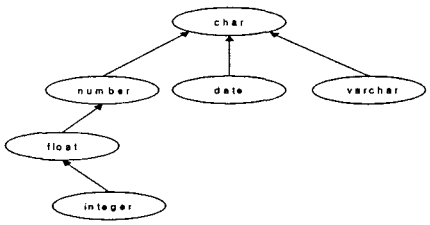
[그림 3] 이름 대응 관계 목록(Catalog)

■ 데이터형 충돌(data type confilict)

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

[그림 4]는 데이터베이스 설계 시 가장 많이 사용하는 5가지의 데이터형에 대해 형 변환 규칙을 나타낸 트리이다.

형 변환 규칙은 루트노드에 가까울수록 우선순위가 높으므로 단말노드에서 루트노드로 증가하면서 통합이 이루어진다. 즉 int형과 float형의 통합은 float로 float와 number형의 통합은 number로 변환된다. 형 변환 트리에서 이웃하는 노드들의 통합은 부모노드의 데이터형으로 이루어진다. 즉, int형과 data형은 date형의 부모 노드인 char형으로 변환된다.



[그림 4] 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 상위노드인 char형으로 변환되도록 한다.

3.2 동일한 데이터에 대한 서로 다른 표현

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.

■ 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 같은 의미를 가지는 값에 대한 변환 표(mapping table)를 이용하여 해당 값을 대응시키도록 함으로써 해결할 수 있다.

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.

■ 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.

3.3 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 합쳐야 할 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 복합 속성을 단순 속성으로 변환하는 과정과 변환된 단순 속성을 하나의 속성으로 합치는 과정의 두 단계로 나누어 이루어지게 된다.

■ 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.

IV. 결 론

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, XML은 시스템 독립적이기 때문에 스키마 구조를 XML로 변환하는 변환기만 있으면 어떤 시스템과도 정보 교환이 가능하며 다중 데이터베이스로 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] 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, pp212-253

[2] 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.

[3] 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.

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

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

[6] Jon Bosak, XML, Java, and the future of the Web , Sun Microsystems 1997