XML 스키마 기반의 데이터베이스 스키마 생성기 설계

임종선* · 김경수**

요 약

현재, 기업간 전자상거래는 XML 문서를 이용하여 기업간 정보유통에 부분적으로 적용되고 있으나 웹서비스가 본격 구현되면 기업 기간시스템도 XML 기반으로 점차 전환될 것으로 보고 많은 업체들이 XML DBMS 개발에 경쟁적으로 나서고 있다. 기존의 XML DBMS 연구들에서는 XML 문서의 구조를 표현하기 위하여 XML DTD를 사용하였다. 이러한 XML DTD는 단순한 형태의 표현을 정의하고 있어서 XML 문서의 구조를 정의하는데 많은 어려움이 있다. 이를 극복하기 위하여, 본 논문에서는 W3C의 표준으로 채택된 XML 스키마를 기반으로 데이터의 컨텐츠 저장에 일반적으로 사용되고 있는 관계형 데이터베이스에 XML 데이터를 저장하기 위하여 XML 스키마를 이용한 관계형 데이터베이스 스키마의 자동 변환 메커니즘을 통한 알고리즘을 제안하였다. 이 알고리즘을 토대로 XML 스키마를 관계형 데이터베이스 스키마로 변환하는 모듈을 개발하면, XML 데이터의 관리를 관계형 데이터베이스를 통하여 효과적으로 할 수 있을 것이다.

Design of Automatic Database Schema Generator Based on XML Schema

Jong-Seon Lim* · Kyung-Soo Kim**

ABSTRACT

B2B e-business is an economic transaction formed between companies through various networks including internet. At present, e-business between companies partly applies information distribution between companies, but many enterprises expect that a corporate basic system will be gradually changed into XML basis if web service is earnestly materialized, so they are competing with each other in developing XML DBMS. Existing XML DBMS studies used XML DTD in order to represent the structure of XML document. Such XML DTD defines the expression of a simple type, so there are many difficulties in defining the structure of XML document. To cope with this, in this paper, the author will develop database schema generator utilizing relational database generally used in storing contents of data, on the basis of XML schema selected as a standard of W3C. Also, to store XML data, the author proposed the automatic conversion method of relational database schema that used XML schema.

Key words : XML Schema, RDB Schema, Schema Conversion

* 순천향대학교 컴퓨터학부
** 백석문화대학 컴퓨터정보학부

1. 서 론

기존의 XML DBMS 연구들에서는 XML 문서의 구조를 표현하기 위하여 XML DTD를 사용하였다. 이러한 XML DTD는 단순한 형태의 표현을 정의하고 있어서 XML 문서의 구조를 정의하는데 많은 어려움이 있다. XML DTD의 문제점을 해결하기 위하여 W3C(World Wide Web Consortium)에서는 2001년 XML 스키마를 표준안으로 채택하여 사용을 권고하고 있다. XML 스키마는 XML DTD에서 지원하지 못하였던 XML 문법의 준수, 네임스페이스의 지원, 다양한 형태의 빌트인 타입이나 사용자 지정 데이터 타입, 반복 횟수의 다양한 지원 등을 지원하도록 하였다[3, 9].

본 논문에서는 기존 XML DTD를 이용한 XML DBMS의 단점을 극복하기 위하여, W3C의 표준으로 채택된 XML 스키마를 기반으로 데이터의 컨텐츠 저장에 일반적으로 사용되고 있는 관계형 데이터베이스 스키마로의 자동 변환하는 모듈을 개발 하였다. 이를 이용하면, XML 데이터를 저장하기 위하여 XML 스키마를 이용한 관계형 데이터베이스 스키마의 자동 변환 방법을 토대로 XML DBMS를 구현할 수 있을 것이다.

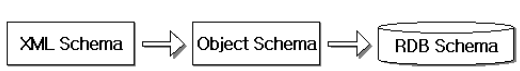
2. 관련 연구 및 국내외 동향

위와 같이 XML DBMS의 수요가 증가하고 있으며, XML DBMS를 제공하는 업체 역시 다양한 솔루션을 제공하는 XML DBMS를 개발 중에 있거나 발표하였다. 기존의 DBMS 시장을 차지하고 있는 관계형 데이터베이스에 XML DTD나 XML 스키마의 변환을 통하여 XML 문서를 저장하는 방법들에 관한 여러 연구들이 있었다. 이러한 XML 문서의 저장 방법은 크게 DTD나 XML 스키마와 같은 스키마 정보가 없는 경우와 있는 경우로 나눌 수가 있다. 스키마 정보가 없는 경우에는 XML 문서로부터 스키마 정보를 추출해 내어 관계형 테이블들을 생성하는 방법과 스키마 정보는 무시하고 XML 문서를 구성하는 요소들인 노드와 연결선들만을 저장하는 방법이 있다. 전자의 경우에는 STORED가 있고, 후자의 경우에는 반 구조적 데이터를 저장하기 위한 방법과 Binary 방식이 있다. 또한, 스키마 정보가 있는 경우에는 DTD나 XML 스키마로부터 관계형 스키마를 생성하는 방법인 X-ray, Inlining, Extended Inlining 등이 있다[1, 6, 7]. Inlining에서는 XML 문서의 DTD를 기반으로 관계형 스키마를 추출하는 기법을 제안하고 있다. 또한 Extended Inlining에서는 DTD 기반의 Inlining을 확장하여 XML 스키마를 기반으로 하는 기법을 제안하였다. Extended Inlining 기법은 테이블의 개수와 조인되는 컬럼의 수가 매우 커지는 단점이 있으며, XML 데이터가 생성된 테이블에 따라 분산 저장되는 것에 따른 조인 횟수의 증가도 해결해야 할 문제이다[7]. 본 논문에서는 이러한 문제점을 최대한 감소시키고자 XML 스키마를 관계형 데이터베이스 스키마로 변환시키는 메커니즘을 생성하고, 이를 토대로 알고리즘을 개발하였다.

3. XML 스키마 변환 메커니즘 및 알고리즘

3.1 XML 스키마 변환 메커니즘

XML 문서의 구조를 정의한 스키마를 관계형 데이터베이스 스키마로 변환하기 위하여 (그림 1)에서 보여주는 것과 같이 변환방법이 필요하다. 이는 XML 문서를 저장하거나 검색할 때 변환해 주는 미들웨어를 통하여 진행된다[8, 9].



(그림 1) 객체 기반 변환방법

본 논문에서 사용하고 있는 변환방법은 객체를 이용한 변환방법으로서 복잡한 XML 문서를 관계형 데이터베이스에 저장하려고 할 때 사용하는 변환방법이며, XML DTD를 기반으로 한 변환방법에 비하여 다양하게 변환이 이루어지며, 스키마를 객체로 변환한 후, 그 변환된 객체를 관계형 데이터베이스 스키마로 변환함으로서 기존의 관계형 데이터베이스 시스템의 활용영역을 확장하는데 기여한다.

XML 스키마를 관계형 스키마로 변환하는 과정은 (그림 1)에서 보였듯이 3단계를 이용하여 변환한다. 우선, XML 스키마를 객체 스키마로 변환한 후, 객체 스키마를 관계형 데이터베이스 스키마로 변환을 하고자 하였다.

3.1.1 XML 스키마를 객체 스키마로의 변환 메커니즘

(1) 스키마 매핑

<표 1>은 자바의 패키지와 같은 관계된 프로그래밍 구조의 집합으로 스키마를 매핑 한다.

〈표 1〉 스키마의 매핑

특 성	객체지향 구조로 매핑
모든 특성들	대응되는 구조는 없다. 스키마 컴포넌트의 특성들은 다른 컴퍼넌트의 리스트나 클래스의 리스트와 관련될 것이다. 이것은 확장 이외에는 다른 의미가 없다.

(2) 어트리뷰트 매핑

어트리뷰트 노드(다중 값이 가능한)를 스칼라 타입으로 매핑하고, 끝지점의 어트리뷰트 노드를 properties로 매핑한다. <표 2>는 테이블이 어떻게 어트리뷰트 노드를 type으로 매핑하느냐를 보여준다.

(3) 엘리먼트 타입의 매핑

간단한 엘리먼트 타입 노드는 일반적으로 scalar data type으로 매핑되고, complex type 노드는 일반적으로 클래스로 매핑된다. 엘리먼트 타입 노드의 edges pointing은 property로 매핑된다. <표 3>은 어떻게 엘리먼트 타입 노드를 scalar type이나 클래스로 매핑하는 가를 보여준다.

〈표 2〉 어트리뷰트의 매핑

특 성	객체지향 구조로 매핑
name, target namespace	타입 name, 다중 어트리뷰트 노드는 동일한 타입으로 매핑된다.
type definition	스칼라 타입의 집합을 각 노드로 매핑되게 결정한다.
scope(global)	scope는 모든 전역 타입을 포함한다.
scope (complex type)	클래스를 각각의 complex type으로 매핑된다. type은 local scope. scope는 모든 global type을 포함한다.
annotation	코드내의 코멘트

<표 3> 엘리먼트 타입의 매핑

특 성	객체지향 구조로의 매핑
name, target namespace	class 이름은 그들의 범위에서 유일해야 한다. 그러나 scalar type 이름은 유일해야 된다는 것이 필요하지 않다. 이것은, 다중 엘리먼트 타입 노드들이 동일한 scalar type으로 매핑 될 수 있기 때문이다.
type definition (simple)	어떤 노드가 scalar type의 집합으로 매핑될 것인지를 결정한다.
type definition (complex)	클래스의 집합에 어떤 노드가 매핑될 것인지를 결정한다. 그리고 자식 노드를 종속되는 클래스내의 property로 매핑한다.
scope(global)	모든 전역 클래스를 포함하는 범위이다.
scope (complex type)	클래스는 각각의 complex type으로 매핑된다. 클래스는 local 범위를 가진다. 모든 전역 클래스를 포함하는 범위를 가진다.
annotation	코드 내의 코멘트이다.

(4) 단일 타입의 매핑

단일 타입의 매핑은 <표 4>에서 어떻게 매핑되는가를 나타낸다. 기본 타입의 정의와 variety의 옵션에 의하여 결정하며, 일반적인 주석문장은 코드내에서 작성된다.

〈표 4〉 단일 타입으로의 매핑

특 성	객체지향 구조로의 매핑
name, target namespace	이름은 unique하지 않아도 된다. 이것은 다중 simple type이 같은 데이터 타입으로 매핑이 가능하다는 뜻이다.
기본 타입 정의	이것은 객체지향언어 데이터 타입으로 직접 매핑 될수 있는 타입으로 도달 할때 까지 타입 정의 계층으로 거꾸로 따라간다.
facets	어떤 simple type으로 매핑될 것인지에 대한 데이터 타입을 결정해 준다. 이 매핑에서 사용되는 type은 property의 집합을 사용하는 mutator code로 매핑되어야 한다.
variety(atomic)	single-valued 타입이다.
variety(list)	multi-valued 타입이다.
variety(union)	union
annotation	코드의 코멘트이다.

〈표 5〉 Complex Type의 매핑 방법

특 성	객체지향 구조로의 매핑
name, target namespace	class name과 target namespace의 조합은 name으로 매핑될 것이다. complex type이 name을 가지지 않는다면, class로 매핑할 경우 이름을 주어야 한다.
base type definition (simple)	complex type이 매핑되는 class의 superclass는 명시된 simple type의 단일 property를 포함하는 클래스이다.
base type definition (complex)	class의 superclass는 complex type으로 매핑된다.
derivation method(extension)	상속
derivation method(restriction)	객체지향 analog에서는 직접적으로 나타나지 않는다. 이것은 역상속의 정렬이다.
final (extension)	Final class.
final (restriction)	객체지향 analog에서 직접적으로 나타나지 않는다.
abstract	어떤 경우에 클래스가 abstract인지 아닌지에 대한 경우는 매핑되지 않는다.
attribute use pairs	edge pointing에서 쌍을 사용하는 어트리뷰트는 complex type의 클래스내의 속성으로 매핑된다. 만약 불린 쌍이 true 이면 속성은 non-nullable이다. 만약 boolean이 false이면 property는 nullable이다.
content type (empty)	어떤 구성도 매핑되지 않는다. empty는 단지 맵에서 엘리먼트 타입 선언을 포함하여 정의하는 complex type에 대한 플래그이다.
content type (simple type definition)	non-nullable, scalar-값을 가진 속성은 클래스로 추가된다.
content type (content model, element-only)	content model의 범위는 element-only 값으로 매핑시 사용하지 않는다.
content type (content model, mixed)	content model의 범위는 추가로 다중값 속성이 Java의 String[ ]처럼 클래스에 추가된다.
annotations	코드 내의 주석

(5) Complex Type의 매핑

complex type, complex type과 엘리먼트 type 사이의 관계를 매핑하는 동안에 일어나는 문제를 고려하였으며, 이를 적용한 <표 5>는 어떻게 complex type 정의가 매핑되는가를 나타낸다. complex type은 스키마 그래프에서 엘리먼트 type node와 어트리뷰트 group과 particle node사이에 나타난다. 이런 경우, 엘리먼트 타입에서의 edge pointing은 엘리먼트 타입과 complex type사이의 관계에서 선택되어 compelx type으로 매핑될 것이다.

3.1.2 객체 스키마에서 관계형 데이터베이스 스키마로의 변환 메커니즘.

객체 스키마에서 관계형 데이터베이스 스키마로의 변환 방법은 <표 6>과 같다. XML 스키마로부터 객체 스키마로 변환될 때는 객체 구조에 따라 변환되고, 객체 스키마에서 관계형 데이터베이스 스키마로 변환될 때는 내용에 따라서 매핑이 이루어진다.

〈표 7〉 XML 스키마의 Attribute 매핑 알고리즘

```
i, j := 0 ;
 for each choosing AS_i in SP_i do
  if AS_i contains name, typedef, parents_id
    then begin
     OAT_i.att_name := AS_i.name
     OAT_i.att_type := AS_i.typedef
     OAT_i.class_id := AS_i.parents_id
  end if
  if OAT_i contains att_name, att_type, class_id
     then begin
     RDBS_temp.table_name := OAT_i.class_id
    for each RDBS_j.name (j be change 0 to i)
     if RDBS_i.table_name == RDBS_k.tabel_name
      then begin add
       RDBS_k.col_name := OAT_i.att_name
       RDBS_k.col_type := OAT_i.att_type
      else begin create RDBS_i.table_name
       RDBS_i.col_name := OAT_i.att_name
       RDBS_i.col_type := OAT_i.att_type
      end if
      j := j + 1
    end for
  end if
  i : = i + 1
 end for
return(RDBS_i)
```

<표 6> 관계형 데이터베이스 스키마로의 매핑

객체지향 구조	관계형 데이터베이스 구조
추상 클래스	추상 클래스는 inheritance 매핑을 제외하고 매핑되지 않는다.
class	class table로 알려져 있으며, object는 클래스 테이블의 행으로 나타난다.
상속성(inheritance)	superclass와 subclass는 unique key/foreign key로 join되어 테이블로 나눈다. unique key는 superclass 테이블이며, object는 각 테이블의 행으로 나타난다. superclass property들은 subclass 테이블에 저장된다.
scalar 데이터타입의 single-valued property	클래스 테이블의 열은 property column이며, 데이터타입은 열의 가능한 데이터타입의 set으로 결정된다(void나 object 포인터의 데이터타입은 BLOB으로 매핑된다. property는 poperty column의 값으로 나타나며, 분리된 테이블의 property column은 property table이다. property talble은 unique key/foreign key의 관계로 class table에 조인된다. unique key는 class talbe에 있다).
scalar 데이터타입의 Multi-valued (collection) property	클래스 테이블의 다중 property column은 collection의 특정 property 열에 매핑된다. 이것은 오직 collection이 제한된 최대 숫자의 값처럼 알려져 있을 때만 가능하다. property 테이블의 property 열은 collection의 값이 하나의 행만이 있음을 뜻한다. property 테이블은 unique key와 foreign key의 관계로 클래스 테이블로 조인된다. unique key는 클래스 테이블에 있다.
Property nullability	열 nullability
Property filnality	열 writeability는 최종 property가 read-only 열로 매핑 되고, non-final property는 read-write 열로 매핑된다.

3.2 XML 스키마 변환 알고리즘

다음은 XML 스키마 변환 메커니즘을 기반으로 하여, XML 스키마를 관계형 데이터베이스 스키마로 변환을 위한 알고리즘을 나타낸 것이다. 모든 알고리즘에서는 XML 스키마를 파싱한 결과인 *SP*를 이용하여 매핑한다. <표 7>은 어트리뷰트를 매핑시키는 알고리즘이다. SP_i는 XML 스키마 파싱 모듈에서 생성된 결과라고 하고, AS_i를 XML 스키마 파싱 결과 내의 어트리뷰트 들의 집합이라고 한다. 또한, OAT_i는 변환된 객체 스키마 어트리뷰트 집합이라고 놓고, $RDBS_i$는 객체 스키마에서 변환된 관계형 데이터베이스 스키마라고 하자. 그럼 <표 7>과 같은 알고리즘을 이용하여, XML 스키마 중 어트리뷰트를 관계형 데이터베이스 스키마로 생성한다.

또한, <표 8>은 엘리먼트를 매핑시는 알고리즘을 보여준다. SP_i는 XML 스키마 파싱 모듈에서 생성된 결과라고 하고, ES_i를 XML 스키마 파싱 결과 내의 엘리먼트 들의 집합이라고 한다. 또한, OET_i는 변환된 객체 스키마 엘리먼트 집합이라고 놓고, $RDBS_i$는 객체 스키마에서 변환된 관계형 데이터베이스 스키마라고 하자. 그럼 다음과 같은 알고리즘을 이용하여, XML 스키마 중 어트리뷰트를 관계형 데이터베이스 스키마로 생성한다.

< 표 8 > XML 스키마의 Element 매핑 알고리즘

```
i, j := 0 ;
 for each choosing ES_i in SP_i do
  if ES_i contains name, exist_CP
   then begin
   OET_i.ele_name := ES_i.name
   OET_i.exist_CP := ES_i.exist_CP
  end if
  if OET_i contains ele_name, exist_CP
   then begin
     RDBS_i.table_name := OET_i.ele_name
      if OET_i.exist_CP has child
       then begin(has child)
         RDBS_i.table_name has Foreign Key
         references RDBS_p.Primary Key
      else begin(has parents)
       RDBS_c.table_name has Foreign Key
       references RDBS_i.Primary Key
      end if
  end if
   i := i + 1
end for
return(RDBS_i)
```

〈표 9〉 XML 스키마의 Complex Type 매핑 알고리즘

```
i, j := 0 ;
  for each choosing CTS_i in SP_i do
    if CTS_i contains name, base_type_def CTS_i
    then begin
      OCTS_i.class_name := CTS_i.name
      if CTS_i.base_type_def is simple ur_type
        then begin
          OCTS_i.class_name :=
                CTS_i.base_type_def
        else if
          OCTS_i.upper_class :=
                CTS_i.base_type_def
        add superclass of OCTS_i.class_name
      end if
    end if
  if OCTS_i contains class_name
   then begin
     RDBS_i.table_name := OCTS_i.class_name
   end if
   i := i + 1
 end for
return(RDBS_i)
```

<표 9>는 Complex Type을 매핑시키는 알고리즘을 보여준다. SP_i는 XML 스키마 파싱 모듈에서 생성된 결과라고 하고, CTS_i를 XML 스키마 파싱 결과 내의 Complex Type들의 집합이라고 한다. 또한, $OCTS_i$는 변환된 객체 스키마 Complex Type 집합이라고 놓고, $RDBS_i$는 객체 스키마에서 변환된 관계형 데이터베이스 스키마라고 하자. 그럼 다음과 같은 알고리즘을 이용하여, XML 스키마 중 Complex Type을 관계형 데이터베이스 스키마로 생성한다. Complex Type은 복잡한 타입인 관계

로 Complex Type의 전체를 수용하지 못하고 그 일부인 Simple Type만 생성하도록 하였다.

4. 결 론

기존의 관계형 데이터베이스에 XML 문서를 저장하기 위한 스키마 생성 기법에 대한 여러 연구들이 있었다. 이러한 방법들은 XML 문서를 저장하기 위하여 XML DTD나 XML 스키마로부터 관계형 데이터베이스 스키마를 생성하는 방법인 X-ray, XML DTD를 기반으로 관계형 스키마를 추출하는 Inlining, 그래프를 생성하여 엘리먼트를 하나의 테이블에 인라인 시키는 Extended Inlining 등이 있다. 위의 방법들은 관계형 데이터베이스 스키마 생성시, 테이블의 개수가 많아진다는 단점이 있다.

본 논문에서는 이러한 단점을 극복하고자 하였으며, W3C의 표준으로 채택된 XML 스키마를 기반으로 데이터의 컨텐츠 저장에 일반적으로 사용되고 있는 관계형 데이터베이스를 이용한 XML DBMS를 개발하기 위하여, XML 스키마를 이용한 관계형 데이터베이스 스키마의 변환 메커니즘을 통한 알고리즘을 제안하였다. 향후, 본 논문에서 제안한 알고리즘을 토대로 관계형 데이터베이스 스키마로의 자동 변환 모듈을 개발할 것이다. 이를 이용하여 XML DBMS를 구축한다면, 기존의 XML DBMS에서 제공하지 못했던 XML 스키마 기반의 DBMS를 이용함으로써 e-business의 활성화, 기업간 또는 기업내 문서전달이 용이해질 것이다. 그리고 효율적이며 안정적으로 XML 컨텐츠를 저장, 관리할 수 있을 것이다.

참 고 문 헌

[1] 김경수, 최문영, 주경수, “객체지향 데이터베이스 기반의 XML 응용을 위한 통합 설계방법론”, 한국컴퓨터정보학회지, 제7권, 제1호, pp. 54-61, 2002.

[2] 박준범, 박경우, 오수열, “ODMG 객체기반의 XML 문서 저장 관리 시스템에 관한 연구”, 한국컴퓨터정보학회지, 제8권, 제2호, pp. 16-23, 2003.

[3] 신일순, 정부연, “XML을 통한 B2B 비즈니스 모델의 변화 및 시사점”, 정보통신정책 ISSUE, 제13권, 제6호, pp. 1-51, 2001.

[4] 오암석, “XML 전용 DBMS인 eXcelon의 소개”, 한국멀티미디어학회지, 제6권, 제4호, pp. 117-120, 2002.

[5] 이상태, 임종선, 주경수, “관계형 DBMS를 이용한 XML 스키마 기반의 XML DBMS 설계”, 한국컴퓨터정보학회논문지, 제9권, 제4호, pp. 19-26, 2004.

[6] 진민, 이종학, 신병주, “관계 데이터베이스를 이용한 XML 문서 저장시스템 설계”, 멀티미디어학회논문지, 제7권, 제1호, pp. 1-12, 2004.

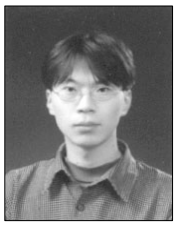
[7] 조정길, “인라이닝에 기반한 XML 스키마의 관계형 스키마 변환 기법”, 정보처리학회논문지D, 제11권, 제5호, pp. 1022-1023, 2004.

[8] 최재혁, 정현주, 김권양, “XML 저장 관리 시스템에서 효율적인 버전 관리 및 문서 저장 방안”, 컴퓨터교육학회논문지, 제6권, 제4호, pp. 11-22.

[9] Mark Graves, Designing XML Databases, Prentice Hall PTR, pp. 150-228, 2002.

[10] MunYoung Choi, JongSeon Lim, KyungSoo Joo, “Developing an XML Schema Generator Based on UML Class”, ICMLC2005, Vol. 4, pp. 2336-2342, 2005.

[11] JongSeon Lim, Kyungsoo Joo, “Developing an XML Repository for Workflow Management”, TENCON2004, Vol. 1, pp. 318-322, 2004.



임 종 선

1997년 순천향대학교 전산학과(공학석사)
2006년 순천향대학교 전산학과(공학박사)
2002년~현재 청운대학교 강사

김 경 수

2001년 순천향대학교 전산학과 (공학박사)
2005년~2006년 Virginia Common-wealth University(객원교수)
1998년~현재 백석문화대학 컴퓨터정보학부 조교수