Proceedings of KFIS Spring Conference 2004 Volume 14, Number 1

개체-관계 모델에서 XML Schema로 변환

A Transformation from Entity-Relationship Model to XML Schema Model

김창석*, 김형석*, 김의정*, 김대수**
공주대학교 컴퓨터교육과*, 한신대학교 컴퓨터학과**

Chang Suk Kim*, Hyeong-Seok Kim*, Eui Jeong Kim*, Dae Su Kim**
Dept. of Computer Education, Kongju National University*
Dept. of Computer Science, Hanshin University**
E-mail : csk@kongju.ac.kr

요 약

XML이 웹 상에서 문서 교환의 표준으로 자리잡고 있으며 그 수요가 나날이 증가하고 있다. 그에 따라 XML 데이터나 문서 구조를 모델링하는 XML Schema(W3C XML Schema Spec) 또한 수요가 증가하고 있다. 그러나 XML Schema는 다양한 자료형과 풍부한 표현력을 제공하지만 그 복잡성으로 인해 모델링하기가 어려운 단점이 있다. 본 논문에서는 관계형 데이터베이스 설계의 기본적인 도구인 개체-관계 모델을 이용하여 XML Schema를 간단하게 생성하는 방법을 제시한다. 개체-관계 모델과 변환될 XML Schema의 구조는 서로 일대일로 매핑되지 않아 직접 변환할 수는 없다. 그래서 몇 가지 알고리즘을 이용하여 개체-관계 모델을 계층적 구조모델로 변환을 한다. 이렇게 변환된 계층적 구조 모델을 이용하여 최종적으로 XML Schema를 생성한다. 여기서 제시한 알고리즘의 특징은 XML Schema의 중요한 특성들인 재사용성, 전역 및 로컬 기능 등을 가진 문서를 생성한다는 것이다.

1. 서론

XML(eXtensible Markup Language) 문서는 인터넷상에서 데이터를 표현하고 교환하는 새로운 표준으로 등장하고 있다. XML로 문서를 작성할 때 문서 구조를 기술하는 모델 언어 또는 스키마라 할 수 있는 DTD(Document Type Definition)를 이용하여 문서나 데이터를 모델링한다. 그러나 DTD는 XML 문서와는 그 문법이 다르고 XML 네임스페이스를 제대로 지원하지 못하며, 데이터 형식에 대한 지원이 미흡하고 내용 모델을 기술하는데 한계를 가진다. 그래서 최근에는 DTD 보다 강력하고 융통성 있는 스키마 언어인 XML Schema(W3C XML Schema Spec)를 사용하는 추세이며, 표준으로도 확정되어 가고 있다.

XML Schema는 풍부한 표현의 문서를 좀더 쉽게 처리할 수 있도록 여러 가지 자료형을 제공한다. XML Schema 자체가 XML로 되어 있어서 DTD의 EBNF 형태의 다른 구문을 배울 필요가 없고, DOM, XSLT 등 XML 도구들을 그대로 사용할 수 있다. 또한 XML Schema 네임스페이스 권고안을 완전히 지원하고, 복잡한 내용모델과 재사용 가능한 내용모델을 쉽게 만들 수 있으며 객체 상속 및 형식 대체와 같은 개념들을 모델링 할 수 있게 해주는 장점들을 가진다[1].

그러나 XML Schema는 다양한 자료형과 풍부한 표현력을 제공하지만 그 복잡성으로 인해 설계가 어렵고 복잡한 단점이 있다. 즉 어떤 복잡한 데이터나 문서를 XML Schema의 특성인 전역 및 로컬 기능, 확장성과 다양한 자료형 등을 활용하여 설계하기는 쉽지 않다. 본 논문에서는 XML Schema로 XML 문서나 데이터를 쉽게 모델링할 수 있는 방법은 제안하고 구현한다. 관계형 데이터 모델이 제안된 이후 지금까지도 관계형 데이터베이스의 논리적 구조를 설계하는 기본적인 도구로 개체-관계 모델(Entity-Relationship model)이 널리

본 연구는 한국과학재단 목적기초연구 R01-2002-000-00068-0의 지원을 받았음.

이용되고 있다. 개체-관계 모델은 단순성과 범용성으로 데이터베이스 세계에서 가장 많이 이용되고 있으며, 상용 도구들도 흔하다. 대부분의 사람들이 쉽게 접근할 수 있는 개체-관계 모델을 이용하여 XML Schema로 표현하고자 하는 문서나 데이터 모델을 설계하면 자동으로 복잡한 XML Schema를 생성해 내는 것이 본 논문의 목적이다.

개체-관계 모델과 변환될 XML Schema의 구조는 서로 일대일로 매핑되지 않아 직접 변환할 수는 없다. 개체-관계 모델에서 XML Schema 문서를 자동 생성하기 위해서 개체-관계 모델을 XML 문서의 특징인 계층형으로 변환해야 한다. 개체-관계 모델을 XML Schema 생성을 용이하게 하기 위해서 의미 변화를 최소화 하는 범위 내에서 계층적 구조 모델로 변환한다. 계층형으로 변환된 개체-관계 모델을 변환 규칙과 제약사항을 이용하여 XML Schema 문서(xsd)로 생성한다.

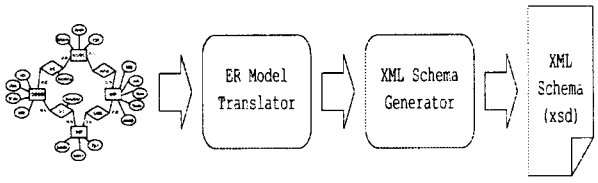

그림 1 개체-관계 모델에서 XML Schema로의 변환 과정

본 변환과정에서의 중요한 사항은 개체-관계 모델을 그대로 XML Schema 로 생성하는 것이 아니라 중복된 참조가 있을 때 중복된 엔티티가 없도록 XML Schema의 중요한 특징 중의 하나인 재사용성을 사용하여 생성한다는 것이다. 이전에 발표되었던 생성방법은 엔티티간의 병합 및 마이그레이션을 하므로 XML Schema의 계층 구조가 단순해지는 이점은 있지만 중복된 관계가 있을 경우 하나의 XML Schema 문서 내에서 같은 내용을 두 번 이상 정의해야 하는 경우가 생길 수 있다[2]. 그러나 본 논문에서 제시하는 생성과정은 XML Schema의 특성 중의 하나인 재사용성을 이용하여 중복 참조 엔티티가 있을 경우에 지명 모델 그룹을 사용하여 XML Schema를 생성해 낸다. 그래서 XML Schema의 중요한 특성들인 재사용성, 전역 및 로컬 기능 등을 가진 문서를 생성한다.

2. 관련 연구

2.1 XML Schema 변환에 관한 연구

관계형 데이터베이스 스키마를 DTD로 변환하는 연구는 UCLA 대학의 EXPRESS, Univ of Applied Sciences의 DB2XML, Stanford 대학의 Lore 프로젝트, SilkRoute, XPERANTO 등으로 현재 활발히 진행되고 있다. 이 후 DTD를 대체하기 위해 개발된 XML Schema가 개발됨으로써 관계형 데이터베이스를 XML Schema로 변환하는 연구가 진행되고 있다.

대표적으로 Elmasri의 연구가 있다[2]. Emasri의 연구에는 개체-관계 모델 구조가 엔티티들간의 마이그레이션으로 인해 XML Schema 문서가 간략해지는 장점이 있지만 참조해야할 뚜렷한 엘리먼트가 사라지므로, 하나의 엔티티에 중복된 관계가 있을 경우 같은 내용을 두 번 이상 정의해야 하는 경우가 발생하며 XML Schema의 큰 특징 중의 하나인 확장성 및 재사용성을 사용하는 데에 제약을 받게 된다.

본 논문에서는 Elmasri 연구의 문제점을 보안하여 XML Schema의 전역 및 로컬 기능 및 재사용성을 활용하여 개체-관계 모델에서 XML Schema 생성하는 방법을 제시한다.

3. 개체-관계 모델에서 계층적 구조 모델로 변환

3.1 변환의 개요

본 절에서는 개체-관계 모델을 XML Schema 문서로 생성해내는 알고리즘에 대해 기술한다. 3장에서는 개체-관계 모델을 XML Schema로 쉽게 생성해 내기 위해 개체-관계 모델을 XML 문서의 특징인 계층형으로 변환하기 위한 과정을 기술하며 4장에서는 계층형으로 변환된 개체-관계 모델을 XML Schema 문서(xsd)로 생성해 내는 과정을 기술한다.

- 개체-관계 모델 변환기: 설계된 개체-관계 모델을 XML Schema 생성을 용이하게 하기 위해서 의미 변화를 최소화 하는 범위 내에서 개체-관계 모델을 계층적 구조 모델로 변환하는 모듈
- XML Schema 생성기: 변환된 계층적 구조 모델을 이용하여 XML Schema로 생성해내는 모듈

본 변환과정에서의 중요한 사항은 개체-관계 모델을 그대로 XML Schema 로 생성하는 것이 아니라 중복된 참조가 있을 때 중복된 엔티티가 없도록 XML Schema의 중요한 특징 중의 하나인 재사용성을 사용하여 생성한다는 것이다. 이전에 발표되었던 생성방법[2]은 엔티티간의 병합 및 마이그레이션을 통해 XML Schema의 계층 구조가 단순해지는 이점은 있지만 중복된 관계가 있을 경우 하나의 XML Schema 문서 내에서 같은 내용을 두 번 이상 정의해야 하는 경우가 생길 수 있다. 본 생성과정은 XML Schema의 특성 중의 하나인 재사용성을 사용하여 중복 참조 엔티티가 있을 경우에 지

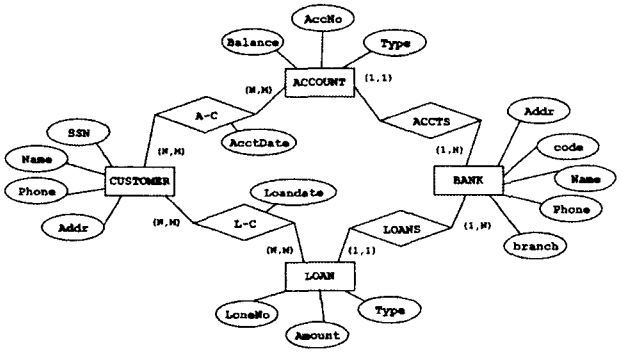

그림 2 개체-관계 모델

명 모델 그룹을 사용하여 XML Schema를 생성해 낸다.

일반적으로 관계형 데이터베이스 설계 시 사용되는 개체-관계 모델은 그래프 구조로 되어 있다. 관계형 데이터베이스는 그 구조가 플랫한 구조이므로 개체-관계 모델을 관계형 데이터베이스 스키마로 설계할 때에는 매우 용이하다. 그러나 XML Schema는 계층 구조로 되어 있어 개체-관계 모델과는 그 형태는 유사하지만 일치하지는 않기 때문에 XML Schema 생성 이전에 개체-관계 모델을 계층적 구조 모델로 변환하는 단계를 거쳐야 한다.

또한 계층 구조의 특성은 하나의 항목은 서로 관련 있는 것들을 내포한다. 다시 말해 부모/자식 관계를 맺고 있다. 그러므로 개체-관계 모델에서 계층적 구조 모델을 생성해 내기 위하여 플랫한 개체-관계 모델에서 계층적 구조의 특성을 찾아내야 한다. 이 계층적 구조의 특성을 찾아내기 위해 각 엔티티 간의 대응 제약 조건을 이용한다.

개체-관계 모델을 XML Schema로 표현하기 위해 그래프 형태로 되어있는 개체-관계 모델구조를 계층형 구조로의 변환이 필요하다. 이 계층형 구조로의 변환을 위해 미리 결정된 최초 탐색 엘리먼트로 부터 너비우선탐색(BFS)과 대응 제약 조건을 이용한다. 너비우선탐색방법을 사용하면 중복되어 있는 그래프의 연결고리를 끊어 낼 수 있고 중복 엔티티를 찾아 낼 수 있으며 엔티티의 탐색 순서를 결정할 수 있다. 그리고 대응제약조건을 이용하여 상위 엔티티와 하위 엔티티 간의 출현회수를 결정할 수 있다. 다시 말해 XML Schema의 요소출현지시자(minOccurs, maxOccurs)를 결정할 수 있다.

XML Schema는 같은 내용을 표현하더라도 여러 가지 표현방법이 존재한다. 다시 말해서 XML 문서에 데이터를 넣을 때 요소로 넣을 수도 있고 속성으로 넣을 수도 있다. 그 이외에도 같은 내용이지만 많은 데이터의 표현을 할 수가 있다. 그러므로 본 변환방법에서 사용된 몇 가지 일반적인 제약조건을 제시한다.

- 개체-관계 모델에서의 엔티티는 지명 복잡 형식으로 전역으로 생성되며 개체-관계 모델에서의 애트리뷰트는 단순 형식로 생성된다.
- 개체-관계 모델에서 관계는 각 엔티티 사이의 요소출현지시자를 결정하는데 사용되며 릴레이션쉽에 애트리뷰트가 있다면 엔티티 사이의 대응제약조건에 의해 상위 엔티티 혹은 하위 엔티티에 병합된다.
- 중복된 엔티티는 중복을 피하기 위해 XML Schema의 지명모델그룹을 이용하여 생성하여 관계가 있는 엔티티에서 참조하도록 한다.

3.2 BFS를 이용한 탐색

최상위 루트엔티티가 결정되었다면 결정된 루트 엔티티를 중심으로 그래프 형태인 개체-관계 모델을 XML Schema의 특징인 계층형 구조로의 생성을 용이하게 하기 위해 계층형 구조로 변환을 해야 한다. 이를 위해 우선적으로 너비우선탐색방법을 사용한다.

BFS의 탐색 범위는 엔티티만을 탐색한다. 위에서 결정된 루트 엔티티로 부터 BFS 탐색방법을 사용하여 스캔은 시작되고 모든 엔티티를 스캔한다. 이때 만일 중복된 엔티티가 발생을 하면 BFS 실행 이전 지정해 놓은 어떠한 특정 큐에 중복 엔티티를 등록한다.

3.3 계층형 구조로 변환

BFS의 탐색과정을 거치면 그림 3과 같이 된다. 현재 엔티티가 ACCOUNT 일때 처음으로 중복 엔티티인 BANK 엔티티를 스캔하게 된다. 그러면 BANK 엔티티는 위의 탐색과정중의 조건에 따라서 새로운 그룹요소가 'BANKGroup' 의 이름으로 생성이 된다. 그리고 BANK 엔티티의 애트리뷰트인 branch, phone, name, code, addr 애트리뷰트는 생성된 BANKGroup의 애트리뷰트로 이동하게 되고 BANK 엔티티는 생성된 BANKGroup 그룹요소를 참조하게 된다.

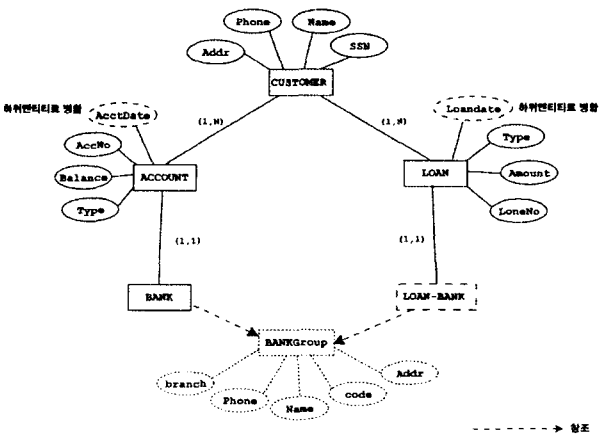

그림 3 계층적 구조모델로의 변환

```
algorithm GenerateHierView
{
 for(엔티티의 총 개수){
   if(현재 엔티티의 자식 엔티티가가 중복 엔티티이면){
      if(처음 중복이면){
         새로운 group 요소 생성;
         현재 엔티티의 자식 엔티티의 애트리뷰트를
            생성된 group 요소로 이동;
         자식엔티티는 생성된 group 요소 참조;
      }else{
         중복된 엔티티 복사;
         중복된 엔티티는 이전에 생성된 group 요소
            참조;
      }
   }
 }
```

그림 4 Algorithm GenerateHierView

3.4 개체-관계 모델 관계 정리

계층적 구조로 변환된 개체-관계 모델의 엔티티 변화를 최소화하는 범위 내에서 릴레이션쉽 관계를 정리한다.

■ 만일 상위 엔티티와 하위 엔티티 사이의 대응제약조건이 M:N 혹은 1:N 일때 만일 M:N 이라면 1:N으로 변환되며 만일 릴레이션쉽에 애트리뷰트가 있을 경우에는 하위엔티티에 병합된다. 그리고 만일 상위 엔티티와 하위 엔티티 사이의 대응제약 조건이 N:1 혹은 1:1 일때 만일 N:1 이라면 1:1로 변환되며 만일 릴레이션쉽에 애트리뷰트가 있을 경우에는 상위 엔티티에 병합된다.

```
Algorithm RelationshipMerge
for(엔티티의 총 개수){
   if(현재 엔티티와 하위 엔티티의 관계가 M:N 혹은
1:N 이라면){
      if(M:N 관계라면){
    의미의 변화가 없는 1:N 관계로 변환한다.
      }
   만일 개체-관계 모델에서 관계 의 Attribute 가
있다면 하위 엔티티에 병합된다.
   }else{        // N:1 or 1:1 관계라면
      if(N:1 관계라면){
    의미의 변화가 없는 1:1 관계로 변환한다.
      }
      만일 개체-관계 모델에서 관계에 Attribute 가
있다면 현재 엔티티에 병합된다.
   }
}
```

그림 5 Algorithm RelationshipMerge

불필요한 엔티티가 제거되고 최종적으로 변환된 과정은 그림 6과 같다.

4. 계층형 구조 모델을 XML Schema로 변환

본 장에서는 계층적 구조 모델에서 XML Schema를 변환하는 방법을 기술한다. 동일한 데이터 요소로 여러 가지 형태의 XML Schema를 생성할 수 있기 때문에 다음과 같은 제약사항을 가정한다.

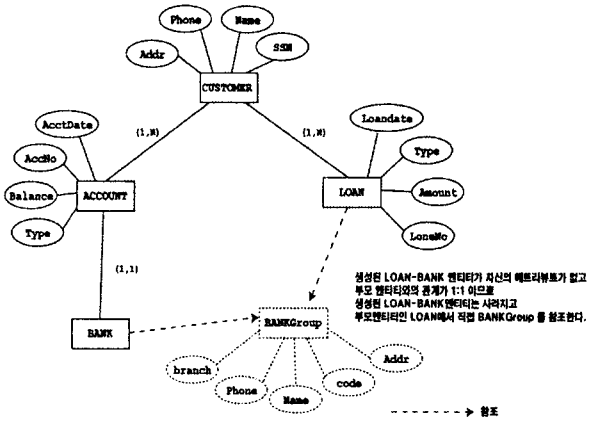

그림 6 계층적 구조 모델로의 변환

■ 개체-관계 모델에서 변환된 계층적 구조 모델에서 엔티티는 지명복잡형식으로 전역으로 생성한다. 이때 타입은 '엔티티명Type' 으로 정의한다.

■ 개체-관계 모델에서 변환된 계층적 구조 모델에서 애트리뷰트는 단순 형식로 생성된다. 만일 현재 엔티티에 자식 엔티티가 존재한다면 전역으로 선언될 하위 엔티티를 포함시킨다.

만일 참조할 그룹요소가 있다면 참조 그룹요소를 참조한다.

■ 엔티티간의 대응제약조건은 각 생성될 엘리먼트의 요소출현지시자를 결정하는데 사용된다. 만일 1:1 관계이면 minOccurs = "1" maxOccurs = "1" 로 생성되고 1:N 관계이면 minOccurs = "1" maxOccurs="unbounded" 으로 생성한다.

■ 그룹요소가 존재한다면 그룹요소를 전역으로 선언한다.

위의 제약사항을 계층적 구조 모델에 적용하여 XML 스키마를 아래와 같이 생성한다.

4.1 XML Schema 파일 생성 및 Schema 선언

XML Schema를 생성할 때 다른 URI와 결합된 동일한 이름의 다른 요소와 속성과는 구별될 수 있게 하기 위하여 접두사를 사용한다. 앞으로 생성될 XML Schema 에서는 모든 요소에 접두사를 사용하며 본 예제에서는 'xs:' 라는 접두사를 사용한다.

모든 XML Schema문서의 루트 요소는 schema 요소가 선언되어야 하므로 처음 여는 schema 태그에는 XML Schema 권고안에 대한 네임스페이스를 아래와 같이 선언해 준다.

개략적인 알고리즘은 다음과 같다.

Algorithm createXmlSchemaDoc
XML Schema 파일생성;

```
write("<?xml version="1.0" encording="UTF-8"?>");
write("<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema
">");
```

아래는 위에서 변환한 예제를 XML Schema로 생성한 일부이다.

```
<?xml version="1.0" encording="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
.....
</xs:schema>
```

4.2 최상위 루트타입 생성

개체-관계 모델에서 계층적 구조 모델로 변환된 모델을 이용하여 XML Schema로 생성해 내기 위해 처음으로 필요한 요소는 XML Schema의 최 상위 루트 엘리먼트를 정의하는 것이다. 위의 예제에서는 최 상위 루트 엘리먼트는 편의상 엔티티 이름에 "Doc"를 붙여 "CUSTOMERDoc"라 칭하기로 한다.

4.3 루트타입 생성

XML Schema 문서를 위한 루트엘리먼트를 생성할 때에는 지명복잡형식으로 생성하며 생성할 지명복잡 형식의 엘리먼트의 타입은 'rootType' 로 하고 complexType 으로 정의한다. 컴퍼지터는 <sequence>로 정의하고 엘리먼트를 루트 엔티티명으로 정의한다. 아래과정에서 전역으로 선언될 루트 엔티티인 '루트엔티티명Type' 을 포함하게 되며 출현회수는 minOccurs= "1" maxOccurs= "unbound" 로 선언한다. 본 예제에서는 루트타입의 이름을 CUSTOMER 로 정의한다.

4.4 엔티티를 지명 복잡 형식으로 생성

개체-관계 모델에서 계층적 구조모델로 변환된 모델에서 엔티티의 개수만큼 위에서 정의한 순서대로 loop 문을 돌면서 각각의 엔티티를 XML Schema의 엘리먼트로 선언하며 엔티티는 지명복잡형식으로 전역으로 선언한다. 이때 타입은 '엔티티명Type' 으로 정의한다.

```
algorithm createEntity
for(엔티티의 총 개수){
   write(현재의 엔티티를 지명복잡형식으로 생성);
   write(현재 엔티티의 애트리뷰트들을 단순형식 생성);
   if(하위 엔티티가 존재하면){
      write(하위 엔티티 이름으로 요소 생성);
   }
   if(참조할 그룹이 있다면){
      write( '참조할 그룹요소Group' 의 이름으로 그룹
      요소 생성 );
   }
}
```

그림 7 Algorithm createEntity

- 컴퍼지터는 <sequence>로 선언한다.
- 계층적 구조모델에서의 애트리뷰트는 단순형식으로 생성한다. name은 애트리뷰트의 이름으로 정의하고 타입은 "xs:string" 으로 정의한다.
- 계층적 구조모델에서 만일 하위 엔티티가 존재한다면 엔티티 이름으로 요소를 생성하고 type 은 "엔티티명Type"으로 하며 요소 출현회수는 위의 변환과정에서 정의한 것에 따라 만일 1:1 관계이면 minOccurs = "1" maxOccurs = "1"로 생성되고 1:N 관계이면 minOccurs = "1" maxOccurs="unbounded" 으로 생성한다.
- 계층적 구조모델에서 만일 참조할 그룹요소가 있다면 그룹요소를 생성하고 ref는 '참조할 그룹요소Group' 로 정의하고 생성한다.

개략적인 알고리즘은 그림 9와 같다.

4.5 그룹요소 생성

만일 계층적 구조모델에서 그룹요소가 존재하면 그룹요소의 개수만큼 그룹요소를 생성하고 이름은 '그룹요소명Group'로 정의한다. 컴퍼지터는 <sequence> 로 선언한다. 그룹요소의 애트리뷰트는 단순 형식으로 생성한다. name은 애트리뷰트의 이름으로 정의하고 타입은 "xs:string" 으로 정의한다.

'개략적인 알고리즘은 다음과 같다.

```
algorithm createGroup
for(그룹의 총 개수){
  write('그룹요소명Group'의 이름값으로 그룹요소생성);
}
```

그림 8 Algorithm createGroup

아래는 위에서 변환한 예제를 XML Schema로 생성한 일부이다.

```
<xs:group name="BANKGroup">
   <xs:sequence>
      <xs:element name="Addr" type="xs:string"/>
      <xs:element name="code" type="xs:string"/>
      <xs:element name="Name" type="xs:string"/>
      <xs:element name="Phone" type="xs:string"/>
      <xs:element name="Branch" type="xs:string"/>
   </xs:sequence>
</xs:group>
```

개체-관계 모델에서 계층적 구조모델로 변환된 모델을 이용하여 위에서 제시한 방법으로 XML Schema를 생성해 낼 수 있다. 아래는 위에서 제시한 방법으로 위에서의 예제를 모두 생성한 결과물이다.

```
<?xml version="1.0" encording="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:element name="CUSTOMERDoc" type="rootType"/>
   <xs:complexType name="rootType">
      <xs:sequence>
         <xs:element name="CUSTOMER"
```

```
type="CUSTOMERType" minOccurs="0"
maxOccurs="unbounded">
      </xs:sequence>
   </xs:complexType>

   <xs:complexType name="CUSTOMERType">
      <xs:sequence>
         <xs:element name="Ssn" type="xs:string"/>
         <xs:element name="Name" type="xs:string"/>
         <xs:element name="Phone" type="xs:string"/>
         <xs:element name="Addr" type="xs:string"/>
         <xs:element name="ACCOUNT" type="ACCOUNTType"
minOccurs="1"                maxOccurs="unbounded"/>
         <xs:element       name="LOAN"       type="LOANType"
minOccurs="1"
maxOccurs="unbounded"/>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="ACCOUNTType">
      <xs:sequence>
         <xs:element name="Balance" type="xs:string"/>
         <xs:element name="AccNo" type="xs:string"/>
         <xs:element name="Type" type="xs:string"/>
         <xs:element name="AcctDate" type="xs:string"/>
         <xs:element       name="BANK"       type="BANKType"
minOccurs="1"                          maxOccurs="1"/>
      </xs:sequence>
   </xs:complexType>

   <xs:complexType name="LOANType">
      <xs:sequence>
         <xs:element name="LoanNo" type="xs:string"/>
         <xs:element name="Amount" type="xs:string"/>
         <xs:element name="Type" type="xs:string"/>
         <xs:element name="LoanDate" type="xs:string"/>
         <xs:group ref="BANKGroup"/>
      </xs:sequence>
   </xs:complexType>

   <xs:complexType name="BANKType">
      <xs:sequence>
         <xs:group ref="BANKGroup"/>
      </xs:sequence>
   </xs:complexType>

   <xs:group name="BANKGroup">
      <xs:sequence>
         <xs:element name="Addr" type="xs:string"/>
         <xs:element name="code" type="xs:string"/>
         <xs:element name="Name" type="xs:string"/>
         <xs:element name="Phone" type="xs:string"/>
         <xs:element name="Branch" type="xs:string"/>
      </xs:sequence>
   </xs:group>

</xs:schema>
```

5. 구현 및 평가

5.1 구현 결과

본 논문에서 제안한 개체-관계 모델을 계층형 구조 모델로 변환과 계층형 구조 모델을 XML Schema로 생성하는 알고리즘은 자바 언어로 구현하였다. 자바는 버전 JDK 1.4.2를 사용하였고 구현 및 테스트 환경은 아래와 같다.

구현은 GenerateXmlSchema 클래스를 사용한다. GenerateXmlSchema 클래스는 엔티티를 표현해 주는 MakeE 클래스, 관계를 표현하는 MakeR 클래스를 포함 하여 사용한다.

5.2 평가

여기서는 본 논문에서 제시한 변환방법과 기존에 발표되었던 Elmasri[5]가 제시한 개체-관계 모델에서 XML Schema 로의 변환방법에 대해 비교하고 본 논문의 독창성을 보인다. Elmasri의 연구에서 가장 주목할 부분은 개체-관계 모델에서 XML Schema로의 변환시 엔티티 사이의 대응제약조건을 이용하여 그 조건에 따른 애트리뷰트의 병합 및 이동이다. 본 논문에서 제시한 개체-관계 모델을 Elmasri의 변환방법에 따라 변환하면 다음과 같다.

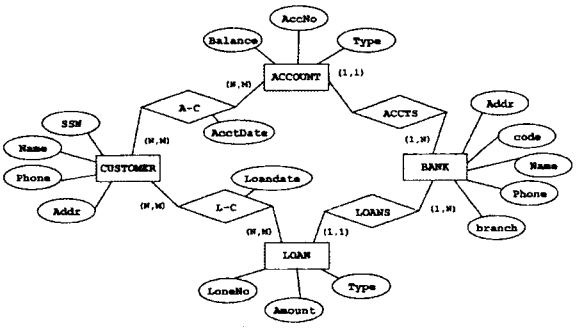

그림 9 변환할 개체-관계 모델

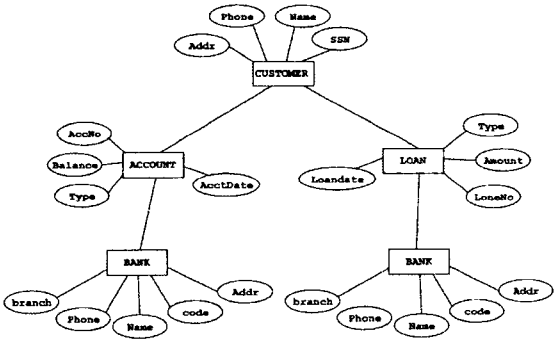

그림 10 Elmasri 의 계층적 모델구조로의 변환과정

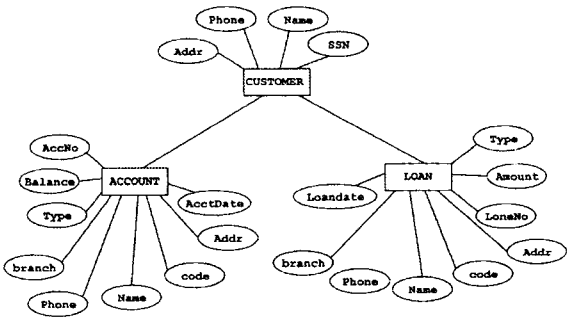

그림 11 Elmasri 의 최종 계층적 모델구조 변환

다음은 Elmasri가 제시한 방법으로 위의 개체-관계 모델을 변환하여 생성되는 최종적인 XML Schema 문서이다.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="customerDoc" type="rootType"/>

 <xs:complexType name="rootType">
    <xs:sequence>
       <xs:element name="CUSTOMER"
type="CUSTOMERType"
       minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
 </xs:complexType>
 <xs:complexType name="CUSTOMERType">
    <xs:sequence>
       <xs:element name="Addr" type="xs:string"/>
       <xs:element name="Phone" type="xs:string"/>
       <xs:element name="Name" type="xs:string"/>
       <xs:element name="SSN" type="xs:string"/>
       <xs:element name="ACCOUNT"
type="ACCOUNTType"
       minOccurs="0" maxOccurs="unbounded"/>
       <xs:element name="LOAN"  type="LOANType"
       minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
 </xs:complexType>
 <xs:complexType name="ACCOUNTType">
    <xs:sequence>
       <xs:element name="Type" type="xs:string"/>
       <xs:element name="Balance" type="xs:string"/>

       <xs:element name="AccNo" type="xs:string"/>

       <xs:element name="AcctDate" type="xs:string"/>

       <xs:element name="BankAddr" type="xs:string"/>
<!-엘리먼트가 중복됨->
       <xs:element name="Bankcode" type="xs:string"/>
       <xs:element name="BankName" type="xs:string"/>
       <xs:element name="BankPhone" type="xs:string"/>
       <xs:element name="Bankbranch" type="xs:string"/>
    </xs:sequence>
 </xs:complexType>
 <xs:complexType name="LOANType">
    <xs:sequence>
       <xs:element name="Loandate" type="xs:string"/>
       <xs:element name="Amount" type="xs:string"/>
       <xs:element name="LoneNo" type="xs:string"/>
       <xs:element name="Type" type="xs:string"/>
       <xs:element name="BankAddr" type="xs:string"/>
<!-엘리먼트가 중복됨->
       <xs:element name="Bankcode" type="xs:string"/>
       <xs:element name="BankName" type="xs:string"/>
       <xs:element name="BankPhone" type="xs:string"/>
       <xs:element name="Bankbranch" type="xs:string"/>
    </xs:sequence>
 </xs:complexType>
</xs:schema>
```

Elmasri의 연구에서 제시한 변환방법은 개체-관계 모델을 XML Schema 로 변환하고 생성할 때에 변환의 조건에 따라 엔티티의 수가 줄어들 수 있어 문서가 간결해지고 그 구조가 단순해지는 장점을 가진다. 그러나 Elmasri가 제시한 방법은 엔티티 간의 애트리뷰트가 상위 혹은 하위 엔티티로 이동하면서 최초 설계시의 엔티티 형태가 사라지게 되어 XML Schema의 특징인 재사용을 사용 할 수 없는 계층적 구조모델이 된다. 또한 탐색 중에 중복참조 된 엔티티가 나타나게 되면 복사되기 때문에 하나의 XML Schema 문서에서 같은 내용으로 구성된 엔티티이지만 중복되는 엔티티를 두 번 이상 정의하는 경우가 발생한다.

즉, Elmasri가 제시한 변환방법은 XML Schema 문서가 DTD와 구별되는 가장 기본적인 특징인 전역 및 로컬 기능 및 재사용성을 전혀 고려하지 않고 XML Schema 문서의 단순화에만 치중하여 제시한 변환방법을 제시하고 있다.

6. 결론

지금까지 본 논문에서는 XML Schema의 여러 장점에도 불구하고 그 문법이 복잡하고 표현방식이 다양하여 설계 시 어려움이 있었던 XML Schema 설계에 대한 문제를 기존의 데이터베이스 설계의 기본 도구인 개체-관계 모델을 이용하여 설계하는 방법을 제시하였다. 그리고 복잡하고 많은 표현양식을 가지고 있는 XML Schema 문법에 대하여 몇 가지 제약사항을 제시하여 XML Schema 문서를 생성해 내었고 XML Schema 의 장점인 재사용성을 활용하고 전역 및 로컬 기능을 활용하여 설계하는 방법을 제안하고 서술하였다. 또한 설계한 개체-관계 모델을 입력받아 자동으로 조건에 따라 XML Schema 문서를 생성해 내는 변환 시스템을 자바로 구현하였다.

7. 참고문헌

[1] Jon Duckett 외 8인 공저, "Professional XML Schemas", wrox

[2] Ramez Elmasri, "Conceptual Modeling for Customized XML Schema", Proceedings of the 21st International Conference on Conceptual Modeling 2002, page 429-443

[3] Dongwon Lee, "Schema Conversion Methods between XML and Relational Models", Knowledge Transformation for the semantic Web, 2003

[4] 허보진, 김형석, 김창석 , "XML 스키마 변환 방법에 관한 비교론적 고찰" 한국정보처리학회 춘계학술발표대회, 2003년 5월

[5] 김형석, 허보진, 김창석, "EER 다이어그램을 이용한 XML 스키마 설계방법" , 한국정보처리학회 추계학술발표대회, 2003년 11월