EER 모델 기반의 XML 통합 설계 방법론

A XML Unified Design Methodology based on Extended Entity-Relationship Model

최문영* 주경수**
순천향대학교 공과대학 컴퓨터학부
griffin@hyejeon.ac.kr* gsoojoo@sch.ac.kr**

요약 : B2B 전자상거래와 같이 XML을 이용한 정보 교환이 확산되고 있으며 이에 따라 상호 교환되는 정보에 대하여 체계적이며 안정적인 정장관리가 요구되고 있다. 이를 위해 XML 응용과 데이터베이스 연계를 위한 다양한 연구가 관계형 데이터베이스를 중심으로 수행되었다.

본 논문에서는 EER 스키마로부터 XML 문서로 자동생성 되는 것을 단순화하기 위하여 알고리즘을 기술하고 관계형 데이터베이스 스키마로 변환을 시도했다. 특히 개념적 데이터 모델에 의해서 기술된 데이터 구조를 DTD로(Document Type Definition) 변환하는 방법을 나타낸다. 이 변환의 중요한 목적은 개념 스키마로부터 구조적인 정보를 가능한 한 많이 보존하는 것이다. 이것은 XML 파서가 가능한 한 일부분의 상태를 확인하고 서로 다른 데이터베이스사이의 데이터 교환을 단순화한다. 특히 데이터베이스 스키마에서 XML 문서로 데이터를 가져오는 것이다. 개념 스키마의 알고리즘을 기준으로 하는 것은 매우 일반적이고 개념적 데이터베이스의 데이터뿐만 아니라 객체-관계 데이터베이스의 데이터에 대해서도 알고리즘이 모델링될 것이다.

1. 서 론

XML는 데이터 교환에 대한 표준 언어로써 부각되고 있다. XML의 이점은 호환성, 확장성이다. 예를 들면 문서 그 자체가 데이터 구조적인 상태이다. 그러므로 중요한 작업중의 하나는 서로 다른 데이터와 정보의 교환이다. 데이터가 기존의 데이터베이스에(관계형, 객체-관계형, 객체지향 또는 다른 형) 존재함으로, 데이터베이스의 정보를 XML 파일로 만드는 자동화 처리는 중요하다. 물론 데이터베이스에서 XML 파일까지 가능한 한 많은 정보를 보존해야 한다.

본 논문에서 정의된 변환은 어떠한 데이터베이스 시스템 적용시킬 것이냐 이다. 가장 간단한 것은 논리 스키마를 개념스키마로의 변환이 잘 정의된 관계형과 객체-관계형 시스템에 적용하는 것이다. 데이터베이스 시스템의 형에 대한 알고리즘을 데이터베이스 디자인에 사용할 수도 있다. XML 문서 구조를 기술하는 것은 표준으로 정의된 DTD를 사용할 것이다. DTD의 선택으로 인한 제한 사항과 솔루션을 언급할 것이다[4].

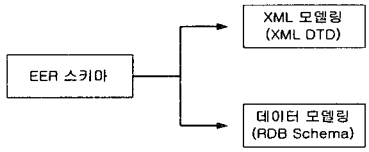

그림 1. EER 스키마를 이용한 XML 모델링과 데이터 모델링

본 논문에서는 그림 1과 같이 EER 스키마에서 XML 모델링과 테이터 모델링으로 변환하는 것을 기술한다. 본 논문의 2장에는 정보의 가장 적은 손실에 촛점을 맞춘 ER모델 구조를 DTD로 변환하는 것에 대하여 기술한다. 3장에서는 주어진 ER 모델을 DTD로의 변환에 대한 알고리즘을 설명한다. 4장에서는 EER 스키마로 변환을 확장한다. 5장에서는 EER 스키마를 데이터베이스로 변환하고, 끝으로 결론과 차후 연구에 대하여 기술한다.

2. ER 스키마로부터 DTD 생성

이 장에서는 보기를 들어 ER스키마에서 DTD로의 변환에 대한 설명을 한다. 그림 2에 보여진 회사 데이터베이스를 ER 스키마에 적용하는 것을 설명한다[1]. 알고리즘에 의해서 만들어진 DTD는 그림 3에 있다.

이 장에서는 개념 스키마부터 XML파일을 만들기 위하여 DTD를 사용한다. 각각의 강한 엔티티 형 E는 ER 스키마로 인해 DTD에 같은 이름 E의 XML 엘리먼트로 변환된다. ER스키마의 단순 데이터형에 관해 보통 취해지는 E의 간단한 속성 Ai는 각각의 XML속성에서 엘리먼트 E에 부속된 Ai로 모델링 된다. 스키마에 키 속성인 REQUIRED 사양에 대해서는 ID형 속성으로 변환되어야 하는 데 반하여, 다른 속성은 IMPLIED 또는 REQUIRED 사양에 대해서 CDATA로써 선언되어야 된다. 각각의 복합속성 A는 ER 스키마에서 E의 서브 엘리먼트 A로 변환된다. 간단한 속성은 복합속성으로 인해 A의 복합 속성이 A를 삽입하는 서브 엘리먼트로 변환하기 때문에 XML 속성으로서 모델링 된다. 이 처리는 일단 모든 속성이 단순하면 된다. 결국 스키마에서 다중-값 속성 M은 카디널리티의 서브 엘리먼트로 변환된다. 서브 엘리먼트 그 자체는 엔티티 E 처럼 내용을 구성할 수 있다 :

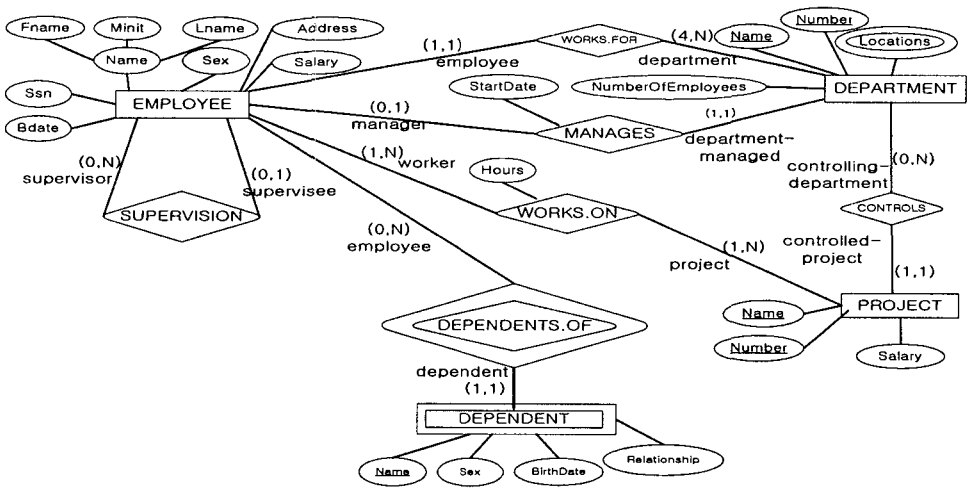

그림 2. 회사 데이터베이스의 ER스키마 예

단순한 ER 속성은 위쪽에 기술된 서브 엘리먼트 만큼 XML속성과 복합속성이 일어난다.

```
<!ELEMENT Company (Department*, WorksOn*)>

<!ELEMENT Department (Location*, Controls*, WorksFor, WorksFor.
WorksFor, WorksFor+) >
    <!ATTLIST Department NameNumber ID #REQUIRED>
    <!ATTLIST Department ManagesStartDate CDATA #IMPLIED>
    <!ATTLIST Department Manager IDREF #REQUIRED>
<!ELEMENT Controls (Project) >
<!ELEMENT WorksFor (Employee) >

<!ELEMENT Location EMPTY>
    <!ATTLIST Location Name CDATA #IMPLIED>

<!ELEMENT Project EMPTY>
    <!ATTLIST Project NameNumber ID #REQUIRED>
    <!ATTLIST Project Location CDATA #IMPLIED>

<!ELEMENT Employee (Name, DependentsOf*, Supervision*) >
    <!ATTLIST Employee Ssn ID #REQUIRED>
    <!ATTLIST Employee Bdate CDATA #IMPLIED>
    <!ATTLIST Employee Adress CDATA #IMPLIED>
    <!ATTLIST Employee Sex CDATA #IMPLIED>
    <!ATTLIST Employee Salary CDATA #IMPLIED>
<!ELEMENT DependentsOf (Dependent) >
<!ELEMENT Supervision EMPTY>
    <!ATTLIST Supervision Supervisee IDREF #REQUIRED>

<!ELEMENT Dependent EMPTY>
    <!ATTLIST Dependent Name CDATA #REQUIRED>
    <!ATTLIST Dependent Sex CDATA #IMPLIED>
    <!ATTLIST Dependent BirthDate CDATA #IMPLIED>
    <!ATTLIST Dependent Relationship CDATA #IMPLIED>

<!ELEMENT WorksOn EMPTY>
    <!ATTLIST WorksOn Project IDREF #REQUIRED>
    <!ATTLIST WorksOn Worker IDREF #REQUIRED>
    <!ATTLIST WorksOn Hours CDATA #IMPLIED>
```

그림 3. 회사 스키마에 대한 DTD

루트 엔티티 타입 E와 ER 스키마의 각기 서브 엔티티 타입 W는 E와 W사이의 확인된 관계타입의 서브 엘리먼트 W에 의해서 모델링 된다. 이 관계타입이 그 자체 카디널리티로 E의 서브 엘리먼트로써 모델되었다. 식별된 관계타입 자체는 기술된 규칙적인 1:n-관계로써 모델링 된다. W의 모든 속성은 루트 엔티티의 속성과 같은 방법으로 모델링 된다. 여기서 W를 E의 서브 엘리먼트로 사용하는 것을 수행한 다음에 W를 위해 E와 W의 복합키를 생성하는 것이 필요하다. 만약 W가 존재하면 일부분인 키는 ID에 대한 그이후의 ID 속성으로서 전체적으로 필수이다. 그러므로 일부분인 키를 강요하지 않고 필수 CDATA 속성으로서 전체적으로 유일 식별자를 얻기 위하여 일부분인 키와 E의 키를 아울러 겸하는 것처럼 모델링 된다.

1:1 관계는 엔티티 타입 S와 T사이에서 1개의 엘리먼트에 대응하는 XML 엘리먼트 S와 T를 병합하여 양쪽 관계들이 완전한 장소에 변환되어야 한다. 1 side에 엔티티 S를 가진 각 정규 2진 1:n-관계 R과 n쪽에 T는 카디널리티로의 S의 서브 엘리먼트 R에 의해서 표현될 수 있다. R과 T는 * 또는 + 관계로 결정된다. 만약 그 관계성 R이 추가적인 속성을 전하면 서브 엘리먼트 R은 위쪽에 기술된 R의 모든 속성으로 부속되어야 한다.

이진 n:m-관계형 R은 DTD의 최상위 엘리먼트 R에 매핑된다. 엘리먼트 R 그 자체는 IDREF로 정의된 두 가지의 속성으로 이루어져 있다. 이들의 참조는 뒤의 XML 문서에서 R의 엘리먼트를 강조한다. 최소 중복과 IDREF XML 문서를 사용하여 각 튜플이 한 번 일어나고 다른 장소에서 참조표가 달라진 다음에 획득된다.

결과적으로 어떠한 카디널리티의 n-ary 관계성은 이진 n:m-관계형과 마찬가지로 모델링 된다 : 최상위 엘리먼트는 각 관계타입에 대하여 도입된다. 그것은

n 과 함께하는 엔티티 타입의 각자에 대하여 n은 IDREF 속성 또는 XPointer 서브 엘리먼트로 이루어져 있다. 관계성의 속성이 더해지는 것은 이전 패러그래프에 기술된다.

3. EER 기반의 XML DTD 설계

이 장에서는 개념 스키마에 적용되는 알고리즘을 이용하여 DTD로 변환할 것이다. 알고리즘은 2장에 ER 스키마로부터 DTD를 만들기 위하여 적용된 예에 의해 확인될 것이다. 그림 4은 완성된 알고리즘이다. 루트 엘리먼트는 스키마 명에 의해서 표현된 모든 데이터베이스에 해당하는 엘리먼트를 사용한다. 상위 계층에서의 엔티티에 대한 엘리먼트는 스키마로부터 가능하면 많은 상태을 보존하기 위하여 객체를 가지고 직접적인 서브 엘리먼트로서 추가된다.

단계 3은 이전의 단계로부터 보조를 맞추어 엘리먼트 상세 내용의 정의에서 시작한다. 처리에 있어서 언제든지 새로운 엘리먼트를 필요로 할 때, 상세 정의를 엘리먼트의 리스트에 추가한다. 각 엘리먼트에 대한 그 처리는 알고리즘 안에 기술된 엔티티 속성으로 인해 배치된 (a), (b) 그리고 (e)를 포함한다. 서브 엘리먼트는 약한 엔티티 타입 (c)와 다른 기능을 가지고 의존하는 엔티티 (d)의 관계타입에 대하여 XML 파서가 확인하는 특별한 문서에 이 상태를 검증할 수 있는 것을 포함한다.

최근 엔티티와 반대되는 측면의 전체 관여를 가진 (f)의 1:1-관계는 반대되는 엔티티 타입에 대한 IDREF 속성뿐만 아니라 최근 엘리먼트에 모든 관계 속성을 더하는 것으로 인지된다. 한편으로는 양쪽 관여가 완전하다(g), XML 속성 또는 이전 장에 기술된 서브 엘리먼트로써 필수 속성을 모두 부착하는 것은 최근 엘리먼트에 반대되는 측면의 엔티티와 관계 속성을 병합한다. 결론적으로 4의 엘리먼트는 n:m- 또는 k-ary 관계형이(k>2) 문서의 루트 엘리먼트의 직접적인 서브 엘리먼트로 정의된다. 관여한 엔티티의 참조는 XML 엘리먼트와 어울리는 존재를 강요할 수 있는 IDREF에 의해서 인지된다. 그러나 원하는 엘리먼트의 정확도는 기술되지 않는다. 관계성의 속성은 엔티티의 속성이 기술된 같은 방법으로 추가된다.

4. ER 데이터 모델의 확장

전통적인 ER 모델의 구조가 진보된 애플리케이션 모델링을 위해 충분하지 않기 때문에 많이 확장을 한다. 본 논문에서 사용된 그 확장된 ER 모형은(EER 모델) [1]에 상세히 기술되었고 다른 것과 유사한 ER 모델의 사용 확장은 [2] 그리고 [3]에 제시되었다. 이용할 수 있는 그 추가적인 구조는 특수화와 일반화이다. 둘 다는 같은 의미를 가지고 있고 경우에 따라 스키마에 처리된 후에 특수화와 일반화사이에 차별화한다. 그 유일한 차이는 모델링 처리에 있다.

풀어서 말하면, 직접적으로 사례를 들어 증명될 DTD에 슈퍼클래스 엘리먼트를 포함하는데 필요한 것

1. 루트 엘리먼트이기 위하여 <SchemaName>를 정의
2. 카디널리티의 서브 엘리먼트 * 루트 엘리먼트<SchemaName>:
(a) 1:n-관계의 n-side에 발생하지 않은 모든 엔티티에 대한 엘리먼트.
(b) k>2로써 모든 이진 n:m-관계와 모든 k-ary 관계에 대한 엘리먼트.
(c) n-side와 관계타입에 부분적으로 일어난 모든 엔티티에 대한 엘리먼트.
3. 모든 엔티티 엘리먼트가 사용되지 않을때 DTD에 정의하는 것은 다음 엘리먼트를 정의한다.(약한 엔티티에 관계를 식별하는 것을 가지고 있는 지원 엔티티)
(a) 각기 복합 속성을 위해 서브 엘리먼트를 삽입.
(b) 서브 엘리먼트의 삽입은 카디널리티 * 또는 +을 각각의 다중-값 속성에 대하여 사용.
(c) 약한 엔티티 타입으로의 각 관계에 대하여 서브 엘리먼트를 규정
(d) 엘리먼트가 카디널리티로의 * 또는 +을 1-side에 있는 장소에 각 1:n-관계를 위해 서브 엘리먼트를 삽입.
(e) 엘리먼트의 모든 simple-type 속성에 대하여 XML 속성을 규정(키 속성은 ID, 다른 속성은 CDATA)
(f) 엘리먼트가 관여의 전체적인 것과 다른면인 일부분은 1:1-관계의 모든 속성에 대하여 XML 속성을 정의 ; 그러한 관계성의 상반되는 쪽을 위해 IDREF 속성을 정의
(g) 양쪽에 전체 관여와 모든 1:1-관계를 위해 엘리먼트에 상반된 측면 엔티티와 모든 관계 타입 속성을 병합한다, 만약 그 상반되는 측면 엔티티가 새로운 엔티티이면
(h) 각 관계타입에 대하여 엘리먼트는 (c) 또는(d)를 적용.
h1. 만약 새로운 엔티티이면 관계성의 상반되는 측면을 위해 서브 엘리먼트를 정의한다.
h2. 관계성의 상반되는 측면을 위해 IDREF 속성을 정의한다. 만약 그것이 이전에 보인 엔티티이면
h3. 엔티티 속성과 같이 위쪽에 모든 관계타입 속성에 대하여 속성을 정의.
4. n:m 또는 k-ary 관계성으로부터 모든 관계타입 엘리먼트에 대한 k>2.
(a) 모든 관계타입에 관여한 엔티티에 대하여 IDREF 속성을 도입.
(b) 엔티티 속성과 같이 위쪽에 모든 관계타입 속성에 대하여 속성을 정의.

그림 4. ER 스키마로부터 DTD 생성 알고리즘 .

은 일부분인 특수화이다. 이처럼, 모든 엔티티에 대하여 엘리먼트를 필요로 한다. 그리고 모든 가능한 서브클래스로부터 선출되기 위하여 선택 서브 엘리먼트로의 DTD에 직접적으로 슈퍼클래스 엘리먼트를 포함한다. 만약 지금 특수화에 가능한 서브 클래스를 가진 슈퍼클래스 G가 S1...Sn 이라는 것을 의미하면 G 설명으로써 조건 <!ELEMENT G(S1|...|SN)?>를 포함한다.

특수화는 전체 또는 부분의 독립적인 방법을 취급할 수 있다. 특수화에 모든 서브 클래스들뿐만 아니라 양쪽 슈퍼클래스 G S1....Sn을 위해 엘리먼트를 전한다. 슈퍼클래스의 엘리먼트는 어떤 가능한 서브 클래스에 대한 선택 서브 엘리먼트로의 DTD에 직접적으로 포함되고 조건 <!ELEMENT G (S1?|...|SN?>는 사용된다. 다중 서브 클래스가 가능함으로, 다중 서브 엘리먼트에 대한 옵션을 포함하는 것이 필요하다. EER 스키마에 가능한 전체 제한은 이 매핑에 강요되지 않는다. 비록 특수화가 완전하기 위하여 정의되었더라도 주어진 XML 엘리먼트는 오직 슈퍼클래스의 엘리먼트를 포함할 수 있다. 적어도 가능한 한 서브 엘리먼트 중의 하나가 여기 있어야만 한다. 이 추가는 특정한 애플리케이션에 유용할 것이다 그러나 단순한 이유 때문에 배치된 것은 포함되지 않는다. 실행할 수 있는 경우에는 두 가지 서브 클래스에 있다. 왜냐하면 그 매핑이 양방향인 특수화의 종류를 지정할 독특한

방법이기 때문이다.

5. EER 데이터 모델 기반의 관계형 데이터베이스 설계

EER 모델로부터 실제 데이터베이스 시스템에 사용될 수 있는 관계형 테이블들이 설계될 수 있다. EER 모델의 각 엔티티 타입과 관계 타입은 독자적인 테이블로 만들어질 수 있다. 만약 2차 관계 타입의 매핑수가 1:1 이거나 1:m의 경우 관계 타입을 독자적인 테이블로 존재시키는 대신 엔티티 타입에 외래키(foreign key)를 사용하여 그 연관성을 표시할 수 있다. 그리고 관계 타입에 독자적인 속성이 존재하거나 n:m 경우의 관계 타입은 독자적인 테이블을 가지는 것이 일반적이다. 일반화의 경우 제약조건에 따라 여러가지 선택이 가능하다. 그림 5는 EER 스키마 예에 대하여 관계형 데이터베이스를 작성한 것이다.

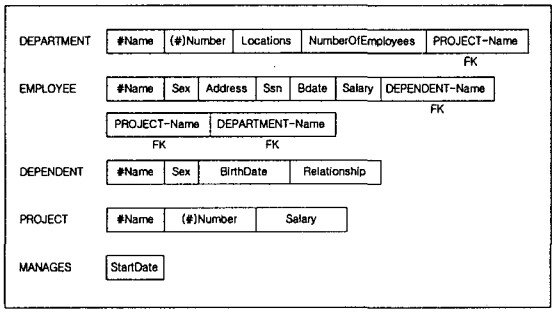

그림 5 EER 스키마 예의 관계형 데이터베이스

6. 결 론

본 논문에서는 ER 모형으로부터 XML DTD 구조로 변환하는 방법과 관계형 데이터베이스 스키마로의 변환을 제안하였다. 부분적인 상태 검사에 대하여 XML 파서를 제대로 사용할 수 있는지 확인하여 많은 정보를 스키마에서 DTD까지 보존하려고 시도하였다. 변환은 역으로도 가능할 것이다. 확장된 ER 모델의(EER 모형) 구조 변환 보여졌고, 이것은 현재 데이터베이스와 장래 애플리케이션에 저장된 가장 많은 데이터의 교환에 대하여 DTD의 발전을 가져올 것이다.

위쪽에 주어진 알고리즘을 사용하여 관계형 데이터베이스 스키마에서 자동으로 DTD 변환에 대한 구현이 현재 행해지고 있고 중요한 상태를 포함한 관계 스키마로부터 ER 스키마를 만드는 역 공학을 연구하고 있다. 그 다음 단계는 객체-관계형 데이터베이스와 EER 모델의 확장일 것이다. 최근에는 UML 도표를 사용한 객체지향 기법도 모델링 될 것이다. 그러므로 EER 스키마에서 UML 클래스 도표까지 변환의 확장일 것이다. 끝으로 객체지향 또는 native XML과 같은 다른 예로 데이터베이스 기법의 사용에 대하여 연구할 것이다.

참고문헌

[1] R. Elmasri, S. B. Navathe: Fundamentals of Database Systems (Third Edition), 3rd edition. World Student Series, Addison-Wesley, Reading, MA, 2000.

[2] T. J. Teorey, D. Yang, J. P. Fry: A Logical Design Methodology for Relational Databases Using the Extended Entity-Relationship Model. ACM Computing Surveys 18:2 (1986), 197-22.

[3] M. Gogolla, U. Hohenstein: Towards a Semantic View of an Extended Entity-Relationship Model. ACM Transactions on Database Systems 16:3 (1991), 369-416.

[4] 김채미, 최학열, 김심석 공저, 전문가와 함께 가는 XML Camp, 마이트 Press, 2001,7,31.