오디세우스 객체관계형 DBMS를 위한 XML 스키마 생성자의 설계 및 구현

백승현[0] 김민수 한욱신 황규영
한국과학기술원 전자전산학과 전산학 전공/첨단정보기술연구센터
{shpaek, mskim, wshan, kywhang}@mozart.kaist.ac.kr

Design and Implementation of an XML Schema Generator for the ODYSSEUS Object-Relational DBMS

Seung-Hyun Paek[0] Min-Soo Kim Wook-Shin Han Kyu-Young Whang
Department of Electrical Engineering & Computer Science
Division of Computer Science and
Advanced Information Technology Research Center
Korea Advanced Institute of Science and Technology

요 약

최근 웹에서 XML(extensible markup language) 문서들이 보편화됨에 따라, 기존의 DBMS 를 이용하여 XML 문서를 효율적으로 저장하고 관리하는 방법에 대한 연구가 활발히 진행중이다. 이를 위하여 DTD(document type definition)로부터 데이타베이스 스키마를 생성하는 효율적인 방법이 필요하다. 그러나 지금까지는 대부분의 연구가 풍부한 모델링 기능을 제공하는 객체관계형 DBMS 대신에 관계형 DBMS 관점에서 진행되어왔다. 본 논문에서는 객체관계형 DBMS 인 오디세우스를 위한 XML 스키마 생성자를 설계하고 구현한다. 첫째, DTD 로부터 관계형 데이타베이스 스키마를 생성하는 기존의 방법을 객체관계형 DBMS 의 중요한 모델링 기능인 집합 타입과 참조 타입을 이용하도록 수정한다. 둘째, 수정된 방법을 오디세우스 객체관계형 DBMS 에 구현한다. 마지막으로, 구현을 용이하게 할 수 있도록 기존의 방법에서 예로서만 기술된 DTD 단순화 방법에 대한 구체적인 알고리즘을 제시한다.

1. 서 론

XML(extensible markup language) 문서는 구조 정보를 추가할 수 있는 문서이며, 웹의 새로운 표준 문서로서 많이 사용되고 있다[1]. 이에 따라, 대량의 XML 문서들을 효율적으로 저장하고 관리하는 시스템의 필요성이 증가하고 있다[2]. 이러한 시스템을 구현하는 여러 가지 방법들 중에서 기존 DBMS 를 이용하는 방법에 대한 연구가 활발히 진행중이다[3, 4, 5, 6].

XML 문서를 데이타베이스에 저장하기 위해서는, 우선 XML 문서의 구조로부터 데이타베이스 스키마를 생성하여야 한다. 즉, XML 문서의 구조를 표현하는 DTD(document type definition)[7]로부터 데이타베이스 스키마를 생성하는 일이 필요하다. 데이타베이스 스키마를 어떻게 생성하는가에 따라 전체 시스템의 성능에 많은 영향을 주므로 저장과 관리를 효율적으로 할 수 있는 스키마 생성 방법이 요구된다[6].

상용 데이타베이스 관리 시스템에서는 DTD 로부터 데이타베이스 스키마를 사용자가 직접 생성하도록 하는 방식을 사용한다[3, 4, 5]. 즉, DTD 의 엘리먼트나 애트리뷰트들이 데이타베이스의 테이블이나 애트리뷰트로 어떻게 대응하는지에 대한 정보를 사용자가 직접 기술하도록 한다. 그러나 이와 같은 방식은 사용자가 스키마를 제시해야 하는 어려움이 있으며, 이는 사용자에게 많은 부담을 증가시키므로 현재의 흐름인 자동튜닝 DBMS 로의 추세에도 맞지 않는다.

참고 문헌 [6](STH 라 표기)에서는 DTD 로부터 관계형 스키마를 자동으로 생성하는 방식을 제안한다. STH 방식에서는 먼저, DTD 로부터 데이타베이스 스키마를 용이하게 생성하기 위해 복잡한 DTD 를 간단한 DTD 로 단순화하는 작업을 수행한다. 그리고, 단순화된 DTD 의 각 엘리먼트를 데이타베이스의 테이블이나 테이블의 애트리뷰트로 대응시키고, 각 엘리먼트 사이의 관계를 주키-외래키 관계(primary key-foreign key relationship)로 대응시켜 관계형 스키마를 생성한다. 그러나 객체관계형 DBMS 는 관계형 DBMS 보다 풍부한 모델링 기능을 제공하므로 이를 이용하면 관계형 DBMS 를 이용하는 것보다 좀 더 자연스러운 스키마를 생성할 수 있다. 또한, STH 에서는 DTD 단순화 과정을 정형적인 알고리즘이 없이 예로 설명하고 있기 때문에, 이에 기반하여 실제 시스템을 구현하기에는 어려움이 있다.

본 논문에서는 오디세우스 객체관계형 DBMS[8]를 이용하여 DTD 로부터 객체관계형 데이타베이스 스키마를 자동적으로 생성하는 XML 스키마 생성자를 설계하고 구현한다. 이를 위하여 DTD 로부터 관계형 스키마를 생성하는 기존의 STH 방법을 DTD 로부터 객체관계형 스키마를 생성하도록 오디세우스 객체관계형 DBMS 에 맞게 수정하여 구현한다. 그리고 STH 에서 예로 기술된 DTD 단순화 과정을 실제 시스템의 구현에 도움이 될 수 있도록 구체적인 알고리즘으로 작성하고 이 알고리즘을 오디세우스 객체관계형 DBMS 에 구현한다.

본 논문의 구성은 다음과 같다. 제 2 장에서는 관련 연구로서 XML 과 STH 방식에 대해서 설명한다. 제 3 장에서는 DTD 로부터 객체관계형 스키마를 생성하는 XML 스키마 생성자의 설계 및 구현에 대해 설명한다. 마지막으로 제 4 장에서 결론을 내린다.

2. 관련 연구

본 장에서는 XML 에 대해 소개하고 DTD 로부터 관계형 데이타베이스 스키마를 생성하는 STH 방법에 대해 설명한다. 제 2.1 절에서는 웹 문서를 기술하는 언어의 새로운 표준으로 제안되고 있는 XML 을 설명한다. 제 2.2 절에서는 DTD 로부터 관계형 데이타베이스 스키마를 생성하는 STH 방법에 대해 설명한다.

2.1 XML 소개

XML[7]은 데이타를 구조적으로 기술할 수 있는 마크업(markup) 언어이다[1]. XML 문서는 XML 스펙(spec)에 따라 기술된 문서로서 엘리먼트들과 애트리뷰트들로 구성된다. 엘리먼트는 XML 문서를 구성하는 기본 단위이고, 애트리뷰트는 엘리먼트에 부가적인 정보를 기술하

* 본 연구는 첨단정보기술연구센터를 통하여 한국과학재단의 지원을 받았음.

는 요소이다. 엘리먼트 안에는 다른 엘리먼트들이 서브 엘리먼트로서 중첩되어(nested) 나타날 수 있다.

DTD 는 XML 문서의 구조를 정의하는 규칙들의 집합으로서 XML 문서에 대해 스키마 역할을 한다[6]. DTD 에서 각 엘리먼트의 구조는 엘리먼트의 타입 선언과 애트리뷰트 선언을 통해 기술되는데, 이 중 엘리먼트의 타입 선언부는 해당 엘리먼트를 구성하는 서브 엘리먼트들의 정규식(regular expression) 형태로 기술된다. 즉, 0 개 혹은 1 개를 나타내는 "?", 1 개 이상을 나타내는 "+", 0 개 이상을 나타내는 "*", 서브 엘리먼트들의 순서를 나타내는 ",", 둘 중의 하나를 나타내는 "|" 연산자들을 이용하여 엘리먼트의 구조를 정규식 형태로 기술한다. 예를 들어, 그림 1은 *book*, *article*, *monograph* 에 대한 XML 문서의 구조를 정의하는 DTD 이다.

```
<!ELEMENT  book          (booktitle, author)>
<!ELEMENT  booktitle     (#PCDATA)>
<!ELEMENT  article       (title, author*, contactauthor)>
<!ELEMENT  contactauthor EMPTY>
<!ATTLIST  contactauthor authorSSN IDREF #IMPLIED>
<!ELEMENT  monograph     (title, author, editor)>
<!ELEMENT  editor        (#PCDATA)>
<!ELEMENT  author        (name, address)>
<!ATTLIST  author        ssn ID #REQUIRED>
<!ELEMENT  name          (firstname?, lastname)>
<!ELEMENT  firstname     (#PCDATA)>
<!ELEMENT  lastname      (#PCDATA)>
<!ELEMENT  address       ANY>
<!ELEMENT  title         (#PCDATA)>
```

그림 1: DTD 의 예.

2.2 STH 방법

STH 에서 DTD 로부터 관계형 데이타베이스 스키마를 생성하는 과정은 DTD 단순화 처리, DTD 그래프 생성, 관계형 스키마 생성의 세 단계로 이루어진다. 먼저, 입력 DTD 에 대해 DTD 단순화 처리를 수행하면 단순화된 DTD 가 생성된다. 다음으로 단순화된 DTD 에 대해 DTD 그래프 생성을 수행하면 DTD 그래프가 생성된다. 마지막으로 DTD 그래프에 대해 관계형 스키마 생성을 수행하면 관계형 스키마가 생성된다.

DTD 단순화 처리 단계에서는 복잡한 구조의 엘리먼트들을 향후 처리를 쉽게 하기 위하여 좀 더 단순한 구조의 엘리먼트로 변환한다. 이 과정에서 엘리먼트의 구조를 나타내는 서브 엘리먼트들의 정규식을 일련의 변환과정을 통해 서브 엘리먼트들의 리스트 형태로 단순화한다. 예를 들어 "<!ELEMENT a (b|c)*>"는 DTD 단순화 처리 단계에서 "<!ELEMENT a (b*, c*)>"로 변환된다. 자세한 DTD 단순화 처리 과정은 제 3.1 절 DTD 단순화 처리에서 설명한다.

DTD 그래프 생성 단계에서는 관계형 데이타베이스 스키마를 용이하게 생성하기 위해 DTD 의 전체적인 구조를 나타내는 자료 구조로서 DTD 그래프를 생성한다. DTD 그래프에서 엘리먼트, 애트리뷰트, "*" 연산자, "?" 연산자는 노드(*node*)로 표현되고, 엘리먼트와 서브 엘리먼트 사이의 관계는 에지(edge)로 표현된다.

관계형 스키마 생성 단계에서는 DTD 그래프로부터 관계형 데이타베이스 스키마를 생성한다. STH 에서는 DTD 그래프의 엘리먼트 노드로부터 테이블 혹은 애트리뷰트로 대응시키는 방식에 따라 기본 인라이닝(basic inlining), 공유 인라이닝(shared inlining), 혼합 인라이닝(hybrid inlining)의 세 가지 방법을 제안한다. 공유 인라이닝 방법이 기본 인라이닝 방법이나 혼합 인라이닝 방법에 비해 데이타 중복이 적게 발생하고, 보다 정규화된(*normalized*) 데이타베이스 스키마를 생성하므로[6], 본 논문에서는 STH 의 세가지 인라이닝 방법을 오디세우스 객체관계형 DBMS 에 맞게 수정하는 방법을 설명하는 데 있어서 공유 인라이닝 방법을 사용한다.

공유 인라이닝 방법은 기본 인라이닝 방법에서 발생하는 엘리먼트 중복을 줄이기 위한 방법으로 루트 엘리먼트 노드와 공유 엘리먼트 노드에 대해서 테이블을 생성하고, 그 노드로부터 도달할 수 있는 애트리뷰트 노드들과 비공유 엘리먼트 노드들을 그 노드가 대응된 테이블의 애트리뷰트로 대응시키는 방법이다. 루트 엘리먼트 노드는 진입 차수(indegree)가 0 인 엘리먼트 노드를 의미하고 공유 엘리먼트 노드는 진입 차수가 1 보다 큰 엘리먼트 노드를 의미한다. 그리고 공유 인라이닝 방법에서는 테이블로 생성된 엘리먼트 노드와 그 노드로부터 도달할 수 있는 공유 엘리먼트 노드들의 관계를 주키-외래키 관계로 대응시킨다. 마지막으로 공유 인라이닝 방법에서는 데이타 중복을 줄이기 위해서 "*" 인접 엘리먼트 노드에 대해 별도의 테이블을 생성하고, "*" 노드로 인접하는 엘리먼트 노드와의 관계는 주키-외래키 관계로 대응시킨다. "*" 인접 엘리먼트 노드는 "*" 노드로부터 인접한 엘리먼트 노드를 의미한다. 참고로 기본 인라이닝 방법과 혼합 인라이닝 방법에서도 데이타 중복을 줄이기 위해서 "*" 인접 엘리먼트 노드에 대해 공유 인라이닝 방법과 같은 방법을 취한다.

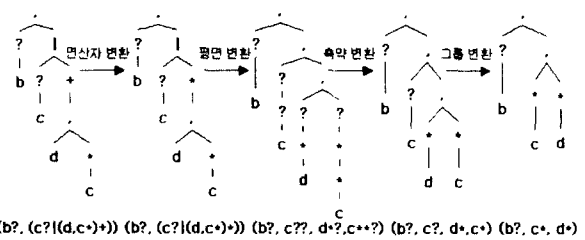
3. XML 스키마 생성자의 설계 및 구현

본 장에서는 XML 문서를 객체관계형 DBMS 인 오디세우스에 저장하기 위해 주어진 XML 문서의 DTD 로부터 객체관계형 데이타베이스 스키마를 자동적으로 생성하는 XML 스키마 생성자의 설계 및 구현에 대해 설명한다. 본 논문에서 설계하고 구현한 XML 스키마 생성자는 관계형 스키마 대신에 객체관계형 스키마를 생성한다는 점을 제외하고는 제 2.2 절의 STH 방법과 동일한 과정을 따른다.

본 장의 구성은 다음과 같다. 제 3.1 절에서는 복잡한 DTD 를 단순화하는 DTD 단순화 처리에 대해 설명한다. 제 3.2 절에서는 단순화된 DTD 로부터 DTD 그래프를 생성하는 DTD 그래프 생성에 대해 설명한다. 제 3.3 절에서는 DTD 그래프로부터 객체관계형 스키마를 생성하는 객체관계형 스키마 생성에 대해 설명한다.

3.1 DTD 단순화 처리

본 논문에서는 STH 에서 예로 설명한 DTD 단순화 방법을 실제 시스템 구현에 도움이 될 수 있는 구체적인 알고리즘으로 작성하고 구현한다. STH 의 DTD 단순화 처리 방법은 DTD 의 각 서브 엘리먼트 정규식에 대해 연산자 변환, 평면 변환(flattening transformation), 축약 변환(simplification transformation), 그룹 변환(grouping transformation)의 네 가지 변환을 적용하는 것으로 이루어진다[6]. 본 논문에서는 DTD 의 각 서브 엘리먼트 정규식을 서브 엘리먼트 이진 트리로 표현하고 서브 엘리먼트 이진 트리에 대해 트리 탐색을 함으로써 연산자 변환, 평면 변환, 축약 변환, 그룹 변환을 수행한다. 예를 들어 그림 2는 서브 엘리먼트 정규식 "(b?, (c?|(d,c*)+))"에 대한 서브 엘리먼트 이진 트리를 단순화하는 과정을 나타낸다.



(b?, (c?|(d,c*)+)) (b?, (c?|(d,c*)*)) (b?, c??, d*?,c*+?) (b?, c?, d*,c*) (b?, c*, d*)

그림 2: DTD 단순화 과정의 예.

서브 엘리먼트 이진 트리는 서브 엘리먼트 정규식을 나타내는 이진 트리(binary tree)로서 내부 노드(internal node)로 ",", "|", "?", "+", "*", 그리고 단말 노드(leaf node)로 서브 엘리먼트 노드를 가진다. 그리고 서브 엘리먼트와 연산자 사이의 관계, 또는 연산자와 연산자 사이의 관계는 에지로 나타낸다.

연산자 변환은 각 서브 엘리먼트 정규식 내의 단항 연산자(unary operator) "+"를 단항 연산자 "*"로 치환하는 변환이다. 예를 들어 서브 엘리먼트 정규식 "(b?, (c?|(d,c*)+))"는 "(b?, (c?|(d,c*)*))"로 변환된다. 연산자 변환 알고리즘은 서브 엘리먼트 이진 트리에 대해 전위 순회(preorder traversal)를 하면서 방문하는 각 노드 P 에 대해, P 가 "+" 노드이면 P 를 "*" 노드로 변환하는 절차를 따른다.

평면 변환은 단항 연산자 "+"가 없는 서브 엘리먼트 정규식에 대해 서브 엘리먼트 정규식이 가지는 구조를 중첩(nested) 구조에서 평면(flat) 구조로 만드는 변환이다. 서브 엘리먼트 정규식이 중첩 구조를 가진다는 것은 서브 엘리먼트 정규식이 "|"를 포함하거나 서브 엘리먼트 정규식에서 ","나 "|"와 같은 이항 연산자(binary operator)들이 다른 연산자 안에 존재하는 것을 의미한다. 반대로 서브 엘리먼트 정규식이 평면 구조를 가진다는 것은 서브 엘리먼트 정규식이 "|"를 포함하지 않고 서브 엘리먼트 정규식에서 ","나 "|"와 같은 이항 연산자들이 어떤 다른 연산자 안에도 존재하지 않는 것을 의미한다. 예를 들어 서브 엘리먼트 정규식 "(b?, (c?|(d,c*)*))"는 평면 변환을 거쳐서 결과적으로 "(b?, c??, d*?, c**?)"가 된다 평면 변환 알고리즘은 전위

순회와 후위 순회(postorder traversal)를 혼합한 VLRV(Visit-Left-Right-Visit) 순서로 서브 엘리먼트 이진 트리를 순회하면서 방문한 노드의 종류에 따라 트리를 변환하는 절차를 따른다.

축약 변환은 평면 변환을 거친 서브 엘리먼트 정규식의 각 서브 엘리먼트에 대해 엘리먼트에 붙어 있는 두 개 이상의 단항 연산자들을 한 개의 단항 연산자로 줄이는 변환이다. 예를 들어 서브 엘리먼트 정규식 "(b?, c??, d*?, c**?)"는 축약 변환을 거쳐서 결과적으로 "(b?, c?, d*, c*)"가 된다. 축약 변환 알고리즘은 평면 변환된 서브 엘리먼트 이진 트리에 대해 전위 순회를 하면서 방문하는 노드의 종류에 따라 트리를 변환하는 절차를 따른다.

그룹 변환은 축약 변환을 거친 서브 엘리먼트 정규식에서 동일한 이름을 가지는 서브 엘리먼트들을 하나의 서브 엘리먼트로 통합하는 변환이다. 예를 들어 서브 엘리먼트 정규식 "(b?, c??, d*?, c**?)"는 그룹 변환을 거쳐서 결과적으로 "(b?, c*, d*)"가 된다. 그룹 변환 알고리즘은 서브 엘리먼트 이진 트리에 대해 전위 순회를 하면서 트리를 구성하는 서브 엘리먼트 노드들에 대한 정보를 저장하기 위한 전위 순회 서브 엘리먼트 리스트를 작성하고, 작성된 리스트로부터 서브 엘리먼트 이진 트리를 재구성하는 절차를 따른다. 전위 순회 서브 엘리먼트 리스트가 저장하는 서브 엘리먼트 노드에 대한 정보는 엘리먼트 이름과 부모 노드의 연산자 타입이다.

3.3 DTD 그래프 생성

DTD 그래프를 생성하기 위해서는 먼저 단순화된 DTD 의 각 엘리먼트에 대한 엘리먼트 노드를 생성한다. 그리고 단순화된 DTD 의 각 엘리먼트에 대해, 해당 엘리먼트의 서브 엘리먼트 정규식에 나타나는 "*", "?" 단항 연산자들에 대한 노드를 생성하고, 엘리먼트 노드로부터 서브 엘리먼트 노드, "*" 노드, "?" 노드로의 관계를 에지로 나타낸다. 마지막으로, DTD 의 각 애트리뷰트에 대해 애트리뷰트 노드를 생성하고, 해당 애트리뷰트가 속하는 엘리먼트에 대응되는 노드와의 관계를 에지로 나타낸다. 예를 들어 그림 1의 DTD 는 이미 단순화 처리된 형태를 가지고 있는데, 이 DTD 에 대한 DTD 그래프를 생성한 결과는 그림 3과 같다.

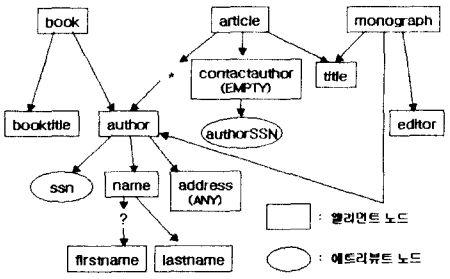


그림 3: DTD 그래프의 예.

3.4 객체관계형 스키마 생성

본 논문에서는 객체관계형 스키마를 생성하도록 공유 인라이닝 방법을 수정하기 위해 집합과 참조 타입을 이용한다. 수정된 공유 인라이닝 방법에서는 STH 의 공유 인라이닝 방법에서 주키-외래키 관계로 대응시키는 부분을 집합 또는 참조 타입을 이용한 애트리뷰트로 대응시키도록 수정한다. 따라서 수정된 공유 인라이닝 방법은 다음과 같다. 첫째, DTD 그래프에 있는 "*" 인접 엘리먼트 노드, 루트 엘리먼트 노드, 공유 엘리먼트 노드에 대해 각각 테이블을 생성한다. 둘째, 어떤 엘리먼트 노드에 대해 테이블 T 가 생성되었을 때, 해당 엘리먼트 노드로부터 도달할 수 있는 비공유 엘리먼트 노드들과 애트리뷰트 노드들은 스트링 타입으로서 T 의 애트리뷰트로 대응시킨다. 셋째, 어떤 엘리먼트 노드에 대해 테이블 T 가 생성되었을 때, 해당 엘리먼트 노드로부터 도달할 수 있는 공유 엘리먼트 노드들은 참조 타입으로서 T 의 애트리뷰트로 대응시킨다. 넷째, 어떤 엘리먼트 노드에 대해 테이블 T 가 생성되었을 때, 해당 엘리먼트 노드로부터 도달할 수 있는 "*" 인접 엘리먼트 노드들은 참조 타입의 집합 타입으로서 T 의 애트리뷰트로 대응시킨다.

이러한 수정된 공유 인라이닝 방법으로부터 작성된 알고리즘은 DTD 그래프의 루트 엘리먼트 노드, 공유 엘리먼트 노드, "*" 인접 엘리먼트 노드에 대해 테이블을 생성하고 그 노드로부터 깊이 우선 탐색(Depth First Search:DFS)을 하면서 만나는 각 노드의 종류에 따라 그에 알맞은 타입의 애트리뷰트로 대응시키는 절차를 따른다. 예를 들어 그림 4는 그림 3의 DTD 그래프에 대해 수정된 공유 인라이닝 방법을 이용하여 생성한 객체관계형 스키마를 나타낸다. OID, varchar, set은 각각 오디세우스 객체관계형 DBMS 에서 지원하는 참조 타입, 가변길이 스트링 타입, 집합 타입이다. book_booktitle_isroot 는 booktitle 엘리먼트가 XML 문서에서 루트 엘리먼트인지 book 엘리먼트의 서브 엘리먼트인지를 구별하기 위한 애트리뷰트이다.

```
book(
  book_booktitle_isroot      char(1),
  book_booktitle             varchar(8192),
  book_author                OID(author)
)
author(
  author_ssn                 varchar(8192),
  author_name_isroot         char(1),
  author_name_firstname_isroot char(1),
  author_name_firstname      varchar(8192),
  author_name_lastname_isroot char(1),
  author_name_lastname       varchar(8192),
  author_address_isroot      char(1),
  author_address             varchar(8192)
)
article(
  article_author             set<OID(author)>,
  article_contactauthor_isroot char(1),
  article_contactauthor_authorSSN varchar(8192),
  article_title              OID(title)
)
title(
  title                      varchar(8192)
)
monograph(
  monograph_title            OID(title),
  monograph_author           OID(author),
  monograph_editor_isroot    char(1),
  monograph_editor           varchar(8192)
)
```

그림 4: 수정된 공유 인라이닝 방법을 이용한 객체관계형 스키마 생성의 예.

STH 의 기본 인라이닝 방법과 혼합 인라이닝 방법을 수정하여 객체관계형 스키마를 생성할 때에도 공유 인라이닝 방법을 수정할 때와 같이 주키-외래키 관계로 대응시키는 부분만을 집합 또는 참조 타입을 이용한 애트리뷰트로 대응시키도록 수정한다. 구체적으로 어떤 엘리먼트 노드에 대해 테이블 T 가 생성되었을 때, 해당 엘리먼트 노드로부터 도달할 수 있는 "*" 인접 엘리먼트 노드들은 참조 타입의 집합 타입으로서 T 의 애트리뷰트로 대응시킨다.

4. 결 론

본 논문에서는 오디세우스 객체관계형 DBMS 를 이용하여 XML 문서를 효율적으로 저장하고 질의할 수 있도록 DTD 로부터 오디세우스 객체관계형 스키마를 자동으로 생성하는 XML 스키마 생성자를 설계하고 구현하였다. 이를 위해 기존의 STH 에서 제안하는 스키마 생성 방법을 오디세우스에 맞게 수정하였다.

본 논문에서 설계하고 구현한 XML 스키마 생성자의 특징은 다음과 같다. 첫째, DTD 로부터 객체관계형 스키마를 생성하도록 STH 의 세 가지 인라이닝 방법을 오디세우스 객체관계형 DBMS 에 맞게 수정하고 이를 구현하였다. 즉, 관계형 DBMS 에서 지원하지 않는 집합과 참조 타입을 스키마 생성시에 활용하여 객체관계형 데이타베이스 스키마를 생성하도록 하였다. 둘째, STH 에서 예로 기술된 DTD 단순화 방법을 실제 시스템 구현에 도움이 될 수 있는 구체적인 알고리즘으로 작성하여 오디세우스 객체관계형 DBMS 에 구현하였다. 이를 위해 서브 엘리먼트 이진 트리 자료 구조를 이용하여 구체적인 알고리즘을 제시하였다.

참고 문헌

[1] Simon, H., *Strategic Analysis of XML for Web Application Development*, Computer Research Corp., 2000.

[2] Williams, K. et al., *Professional XML Databases*, Wrox Press, 2000.

[3] Banerjee, S. et al., "Oracle8i – The XML Enabled Data Management System," In *Proc. 16th Int'l Conf. on Data Engineering*, pp. 561–568, San Diego, California, USA, 2000.

[4] Cheng, J. and Xu J., "XML and DB2," In *Proc. of the 16th Int'l Conf. on Data Engineering*, pp. 569–573, San Diego, California, USA, 2000.

[5] Microsoft Corp., Microsoft SQL Server 2000, *http://www.microsoft.com/sql/default.asp*, 2000.

[6] Shanmugasundaram, J. et al., "Relational Databases for Querying XML Documents: Limitations and Opportunities," *In Proc. 25th Int'l Conf. on Very Large Data Bases*, pp. 302–314, Edinburgh, Scotland, UK, Sept. 1999.

[7] Bray, T. et al., Extensible Markup Language(XML) 1.0 (2nd Ed.), *http://www.w3.org/TR/2000/REC-xml-20001006*, Oct. 2000.

[8] 한욱신, 이민재, 이재길, 박상영, 황규영, "오디세우스 객체관계형 멀티미디어 DBMS 의 아키텍처," 한국정보과학회 추계학술발표회, 2000.