ODYSSEUS/XMLStore: 오디세우스 객체관계형 DBMS 를 이용한 XML 저장 시스템

이기훈[o†] 한욱신[†] 김민수[†] 이종학[‡] 황규영[†]

[†]한국과학기술원 전산학과/첨단정보기술연구센터 [‡]대구가톨릭대학교 컴퓨터정보통신공학부

{drlee, wshan, mskim, kywhang}@mozart.kaist.ac.kr jhlee11@cuth.cataegu.ac.kr

ODYSSEUS/XMLStore: An XML Storage System using the ODYSSEUS Object-Relational DBMS

Ki-Hoon Lee[o†] Wook-Shin Han[†] Min-Soo Kim[†] Jong-Hak Lee[‡] Kyu-Young Whang[†]

[†]Department of Computer Science & Advanced Information Technology Research Center, Korea Advanced Institute of Science and Technology

[‡]School of Computer & Information Communications Engineering, Catholic University of Daegu

요 약

XML 문서가 웹에서 많이 사용됨에 따라, 기존의 DBMS 를 이용하여 XML 문서를 저장하고 관리하는 XML 저장 시스템에 대한 연구가 활발히 진행 중이다. 그러나 지금까지의 연구는 대부분 풍부한 모델링 기능을 제공하는 객체관계형 DBMS 보다는 관계형 DBMS 에 기반하여 이루어져 왔다. 본 논문에서는 오디세우스 객체관계형 DBMS 를 위한 XML 저장 시스템인 ODYSSEUS/XMLStore 를 설계하고 구현한다. 첫째, XML 문서 구조에서 객체관계형 데이타베이스 스키마로의 매핑에 대해 분석한다. 둘째, 분석된 매핑을 기술하는 방법을 표준 언어인 XML Schema 를 활용하여 제안한다. 셋째, 사용자가 정의한 매핑 정보에 의거하여 XML 문서를 객체관계형 데이타베이스에 저장하는 세부 알고리즘을 제안한다.

1. 서 론

XML 문서는 구조 정보를 추가할 수 있는 문서로서, 웹의 새로운 표준 문서로 많이 사용되고 있다[1]. 이에 따라, 대량의 XML 문서들을 효율적으로 저장하고 관리하는 XML 저장 시스템에 대한 필요성이 증가하고 있다.

XML 저장 시스템을 구현하는 방법들 중에서 기존의 관계형 DBMS 를 이용하는 방법에 대한 연구가 활발히 진행 중이다. 즉, 관계형 DBMS 를 이용하여 XML 문서를 데이타베이스의 레코드로 어떻게 변환하여 저장하느냐에 대한 연구가 활발히 진행 중이며[2, 3], 상용 DBMS 에서도 이러한 XML 문서 저장 방법을 제공하고 있다[4, 5].

XML 문서를 데이타베이스의 레코드로 변환하여 저장하기 위해서는, XML 문서 구조가 데이타베이스 스키마에 어떻게 매핑되는지를 정의하고, 정의된 매핑에 의거하여 XML 문서를 저장하는 기능이 필요하다[4, 6]. 이와 같은 기능을 지원하기 위해 상용 DBMS 에서는 XML 문서 구조에서 데이타베이스 스키마로의 매핑을 정의하기 위한 매핑 언어를 제공하고 있다. 그리고 매핑 언어로 작성한 매핑 정보를 이용하여 XML 문서를 데이타베이스의 레코드로 변환하여 저장하는 기능을 제공하고 있다.

그러나 기존 상용 DBMS 에서는 관계형 DBMS 에 기반한 매핑만을 제공하므로, 객체관계형 DBMS 에서 제공하는 참조(reference) 타입과 컬렉션(collection) 타입을 활용한 매핑을 제공하지 못하는 단점이 있다. 또한, 상용 DBMS 에서는 표준이 아닌 독자적인 매핑 언어를 사용하여 매핑을 기술하게 하므로 사용자에게 부담을 주는 단점이 있다.

본 논문에서는 오디세우스 객체관계형 DBMS[7]를 위한 XML 저장 시스템인 ODYSSEUS/XMLStore 를 설계하고 구현한다. ODYSSEUS/XMLStore 는 객체관계형 DBMS 에서 제공되는 참조 타입과 컬렉션 타입을 활용한 매핑을 제공하므로, 트리(tree) 구조를 가지는 XML 문서를 보다 자연스럽게 표현할 수 있는 장점이 있다. 그리고 표준 언어인 XML Schema 를 활용하여 매핑을 기술하므로, 상용 DBMS 에 비해 사용자가 좀 더 쉽게 매핑을 기술할 수 있는 장점이 있다.

본 논문에서는 먼저 XML 문서 구조에서 객체관계형 데이타베이스 스키마로의 매핑에 대해 분석한다. 다음으로 분석된 매핑을 기술하는 방법을 표준 언어인 XML Schema 를 활용하여 제안한다. 마지막으로 사용자가 정의한 매핑 정보에 의거하여 XML 문서를 객체관계형 데이타베이스에 저장하는 세부 알고리즘을 제안한다.

* 본 연구는 첨단정보기술연구센터를 통하여 한국과학재단의 지원을 받았음.

본 논문의 구성은 다음과 같다. 제 2 장에서는 관련 연구로서 XML Schema 와 상용 DBMS 의 XML 문서 저장 기능에 대해서 설명한다. 제 3 장에서는 ODYSSEUS/XMLStore 의 설계 및 구현에 대해 설명한다. 마지막으로 제 4 장에서는 결론을 내린다.

2. 관련 연구

본 절에서는 관련 연구로서 XML Schema 와 상용 DBMS 의 XML 문서 저장 기능에 대해 설명한다. 제 2.1 절에서는 XML 문서 구조를 기술하는 표준 언어인 XML Schema 에 대해 설명하고, 제 2.2 절에서는 상용 DBMS 에서 제공하는 XML 문서 저장 기능에 대해서 설명한다.

2.1. XML Schema

XML Schema 는 XML 문서 구조를 기술하는 XML 형식의 표준 언어이다[8]. XML Schema 는 XML 문서 구조를 기술하기 위하여 엘리먼트를 선언하는 *element* 엘리먼트, 애트리뷰트를 선언하는 *attribute* 엘리먼트, 그리고 추가적인 정보를 기술하는 *appinfo* 엘리먼트 등을 제공한다.

먼저 *element* 엘리먼트는 엘리먼트를 선언하기 위해 사용된다. *element* 엘리먼트는 엘리먼트의 이름과 타입을 나타내기 위해 *name* 과 *type* 애트리뷰트를 가지며, 문서 상에서 엘리먼트가 나타날 수 있는 최소 횟수와 최대 횟수를 나타내기 위해 *minOccurs* 와 *maxOccurs* 애트리뷰트를 가진다. 다음으로 *attribute* 엘리먼트는 애트리뷰트를 선언하기 위해 사용된다. *attribute* 엘리먼트는 애트리뷰트의 이름과 타입을 나타내기 위해 *name* 과 *type* 애트리뷰트를 가진다. 마지막으로 *appinfo* 엘리먼트는 XML Schema 에 응용 프로그램에 필요한 정보를 추가적으로 기술하기 위해 사용된다.

2.2. 상용 DBMS 의 XML 문서 저장 기능

본 절에서는 상용 DBMS 에서 제공하는 XML 문서 저장 기능에 대해 설명한다. 먼저 IBM DB2 의 XML 문서 저장 기능에 대해 설명하고, 다음으로 Microsoft SQL Server 의 XML 문서 저장 기능에 대해 설명한다.

■ IBM DB2

IBM DB2 는 XML Extender 를 통하여 XML 문서를 데이타베이스에 저장하는 기능을 제공하고 있다[4]. XML Extender 는 XML 문서 구조에서

데이타베이스 스키마로의 매핑을 기술하기 위해 자체 매핑 언어로 DAD(Document Access Definition)를 제공하며, 사용자는 DAD 를 사용하여 XML 문서 구조에서 데이타베이스 스키마로의 매핑을 기술한다.

DAD 에서 XML 문서의 엘리먼트는 데이타베이스 테이블이나 컬럼으로 매핑하고, XML 문서의 애트리뷰트는 데이타베이스 컬럼으로 매핑하도록 한다. 그리고 XML 문서의 엘리먼트 사이의 릴레이션쉽은 데이타베이스 테이블 사이의 주 키-외래 키 릴레이션쉽(primary key-foreign key relationship)으로 매핑하도록 한다.

그러나 XML Extender 에서 매핑 언어로 사용되는 DAD 는 관계형 DBMS 에 기반한 매핑만을 제공하므로, 객체관계형 DBMS 에서 제공하는 참조 타입과 컬렉션 타입을 활용한 매핑을 지원하지 못하는 단점이 있다. 또한 XML 문서 구조에서 데이타베이스 스키마로의 매핑을 기술하기 위해 독자적인 매핑 언어를 사용하므로, 사용자에게 부담을 주는 단점이 있다.

■ **Microsoft SQL Server**

Microsoft SQL Server 는 XML Bulk Load 유틸리티를 통하여 XML 문서를 데이타베이스에 저장하는 기능을 제공하고 있다[5]. XML Bulk Load 는 XML 문서 구조에서 데이타베이스 스키마로의 매핑을 기술하기 위해 자체 매핑 언어로서 annotated XDR Schema 를 제공하며, 사용자는 annotated XDR Schema 를 사용하여 XML 문서 구조에서 데이타베이스 스키마로의 매핑을 기술한다.

그러나, annotated XDR Schema 도 IBM DB2 XML Extender 의 DAD 와 같이 관계형 DBMS 에 기반한 매핑만을 제공하므로, 객체관계형 DBMS 에서 제공하는 참조 타입과 컬렉션 타입을 활용한 매핑을 지원하지 못하는 단점이 있다. 또한 XML 문서 구조에서 데이타베이스 스키마로의 매핑을 기술하기 위해 독자적인 매핑 언어를 사용하므로, 사용자에게 부담을 주는 단점이 있다.

3. ODYSSUES/XMLStore 의 설계 및 구현

본 절에서는 오디세우스 객체관계형 DBMS 를 위한 XML 저장 시스템인 ODYSSEUS/XMLStore 의 설계 및 구현에 대해 설명한다. 제 3.1 절에서는 시스템 아키텍춰에 대해 설명하고, 제 3.2 절에서는 매핑 파일의 설계에 대해 설명한다. 그리고 제 3.3 절에서는 객체관계형 데이타베이스로의 XML 문서 저장에 대해 설명한다.

3.1. 시스템 아키텍춰

ODYSSEUS/XMLStore 는 크게 매핑 파일 저장 모듈, XML 문서 저장 모듈, 그리고 오디세우스 객체관계형 DBMS 로 구성된다. 그림 1 은 ODYSSEUS/XMLStore 의 아키텍춰를 나타내고 있다. 그림에서 보듯이 매핑 파일의 저장은 매핑 파일 저장 모듈을 통해 이루어지고, XML 문서의 저장은 XML 문서 저장 모듈을 통해서 이루어진다. 그리고 저장된 XML 문서의 검색, 변경, 삭제는 OOSQL 질의를 통해서 이루어진다. XML 문서의 검색, 변경, 삭제는 향후 XML 질의 언어인 XQuery[9] 질의를 통해 지원되도록 확장할 예정이다.

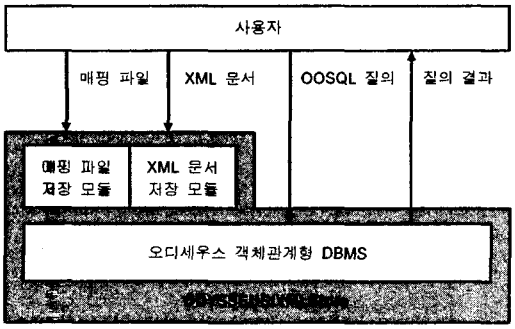

그림 1. ODYSSEUS/XMLStore 의 아키텍춰.

매핑 파일 저장 모듈은 매핑 파일을 파싱하여 매핑 파일에 기술된 XML 문서 구조에 대한 정보와 각 구조의 매핑에 대한 정보를 데이타베이스에 저장하는 기능을 담당하는 모듈이다. 매핑 파일에 기술된 정보들을 데이타베이스에 저장해 두면 매번 정보를 얻기 위해 매핑 파일을 파싱할 필요가 없고, 정보를 보다 효율적으로 검색하고 관리할 수 있는 장점이 있다. 데이타베이스에 저장된 정보는 XML 문서 저장 모듈과 향후 구현될 XQuery 질의 처리 모듈에서 사용된다.

XML 문서 저장 모듈은 XML 문서를 파싱하여 문서에 나타나는 엘리먼트와 애트리뷰트들의 값을 매핑에 따라 데이타베이스 테이블의 레코드로 변환하여 저장하는 기능을 담당하는 모듈이다. XML 문서 저장 모듈은 오디세우스의 OOSQL API 를 통해 직접 데이타베이스에 저장하는 기능과 오디세우스의 벌크 로드 입력 포멧으로 변환하여 벌크 로딩하는 기능을 제공한다.

3.2. 매핑 파일의 설계

본 절에서는 ODYSSEUS/XMLStore 에서 사용되는 매핑 파일에 대해 설명한다. 매핑 파일은 XML 문서 구조에서 데이타베이스 스키마로의 매핑을 기술하는 파일이다. 먼저 XML 문서 구조에서 데이타베이스 스키마로의 매핑에 대해 분석하고, 다음으로 분석에 기반하여 매핑 파일에서 객체관계형 데이타베이스 스키마로의 매핑을 기술하는 방법에 대해 설명한다.

■ **XML 문서 구조에서 데이타베이스 스키마로의 매핑 분석**

XML 문서 구조에서 데이타베이스 스키마로의 가능한 매핑은 그림 2 와 같다. 그림에서 왼쪽 사각형은 XML 문서 구조의 구성 요소를 나타내고 오른쪽 사각형은 데이타베이스 구조의 구성 요소를 나타낸다. 그리고 화살표는 두 구조 사이에 가능한 매핑을 나타낸다. 화살표 중에서 점선으로 표시된 화살표는 두 구조 사이에 가능한 매핑이지만 ODYSSEUS/XMLStore 에서는 허용하지 않는 매핑을 나타낸다.

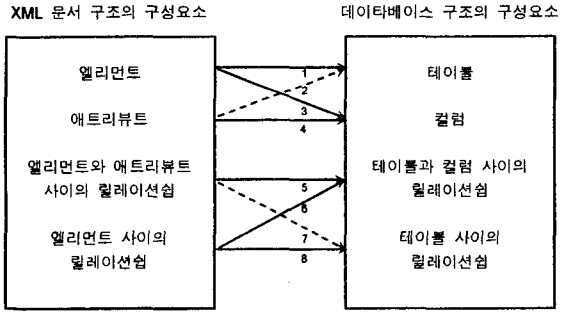

그림 2. XML 문서 구조에서 데이타베이스 스키마로의 매핑.

그림에서 보듯이 XML 문서의 각 엘리먼트나 애트리뷰트는 데이타베이스의 테이블이나 컬럼으로 매핑될 수 있다(화살표 1-4). 그러나, XML 문서의 애트리뷰트를 데이타베이스의 테이블로 매핑하면, 질의 처리 시에 조인 연산이 불필요하게 발생되므로 본 논문에서는 이를 허용하지 않도록 매핑 파일을 설계하였다(화살표 2).

엘리먼트와 애트리뷰트 사이의 릴레이션쉽은 테이블과 컬럼 사이의 릴레이션쉽이나 테이블 사이의 릴레이션쉽으로 매핑될 수 있다(화살표 5, 7). 그러나, ODYSSEUS/XMLStore 에서 애트리뷰트는 컬럼으로만 매핑되므로 엘리먼트와 애트리뷰트 사이의 릴레이션쉽은 테이블과 컬럼 사이의 릴레이션쉽으로만 매핑된다(화살표 5). 그리고 엘리먼트 사이의 릴레이션쉽은 테이블과 컬럼 사이의 릴레이션쉽이나 테이블 사이의 릴레이션쉽으로 매핑된다(화살표 6, 8).

그런데 릴레이션쉽의 매핑(화살표 5, 6, 8) 중에서 테이블과 컬럼 사이의 릴레이션쉽으로의 매핑(화살표 5, 6)은 기술할 필요가 없고, 테이블 사이의 릴레이션쉽으로의 매핑(화살표 8)만 기술하면 된다. 왜냐하면 테이블과 컬럼 사이의 릴레이션쉽은 데이타베이스 스키마에서 이미 유지하고 있으므로, 엘리먼트가 어떤 테이블로 매핑되고 애트리뷰트가 어떤 컬럼으로 매핑되는지를 기술하면, 엘리먼트와 애트리뷰트 사이의 릴레이션쉽이 어떤 테이블과 컬럼 사이의 릴레이션쉽으로 매핑(화살표 5)되는지를 알 수 있기 때문이다.

마찬가지로 엘리먼트들이 각각 어떤 테이블과 어떤 컬럼으로 매핑되는지를 기술하면, 엘리먼트 사이의 릴레이션쉽이 어떤 테이블과 컬럼 사이의 릴레이션쉽으로 매핑(화살표 6)되는지를 알 수 있다. 하지만 테이블 사이의 릴레이션쉽을 나타내기 위해서는 두 테이블을 연결하는 방법에 대한 정보가 추가적으로 필요하므로, 엘리먼트 사이의 릴레이션쉽에서 테이블 사이의 릴레이션쉽으로의 매핑(화살표 8)을 기술하기 위

해서는 두 테이블을 연결하는 방법에 대한 정보를 추가적으로 기술하여야 한다.

이와 같은 매핑(화살표 1, 3, 4, 8 에 해당)을 기술하기 위해서는 1) XML 문서 구조에 대한 정보, 2) 데이타베이스 스키마에 대한 정보, 3) XML 문서 구조에서 데이타베이스 스키마로의 매핑에 대한 정보가 필요하다. 매핑 파일에서 XML 문서 구조에 대한 정보는 XML Schema 를 사용하여 기술하고, 데이타베이스 스키마에 대한 정보 및 XML 문서 구조에서 데이타베이스 스키마로의 매핑에 대한 정보는 XML Schema 의 *appinfo* 엘리먼트 내에 이러한 정보들을 기술하는 새로운 엘리먼트들을 추가하여 기술한다. 이렇게 함으로써 사용자는 표준 XML Schema 언어의 기능만을 사용하여 매핑 정보를 기술할 수 있다.

■ 객체관계형 데이타베이스 스키마로의 매핑 기술 방법

객체관계형 데이타베이스 스키마로의 매핑을 기술하는 매핑 파일에서 데이타베이스 스키마에 대한 정보는 데이타베이스의 테이블을 나타내는 *Table* 엘리먼트와 컬럼을 나타내는 *Column* 엘리먼트를 사용하여 기술한다. *Table* 엘리먼트는 테이블의 이름을 나타내는 *name* 애트리뷰트와 컬럼을 나타내는 *Column* 자식 엘리먼트를 가진다. 그리고 *Column* 엘리먼트는 컬럼의 이름과 타입을 나타내는 *name* 과 *type* 애트리뷰트를 가진다.

XML 문서 구조에서 데이타베이스 스키마로의 매핑에 대한 정보에서 1) 엘리먼트와 애트리뷰트의 매핑(화살표 1, 3, 4)에 대한 정보는 해당 엘리먼트와 애트리뷰트를 선언하는 *element* 와 *attribute* 엘리먼트의 자식 엘리먼트로 *Table* 과 *Column* 엘리먼트를 사용하여 기술하며, 2) 엘리먼트 사이의 릴레이션쉽에서 테이블 사이의 릴레이션쉽으로의 매핑(화살표 8)에 대한 정보는 *Relationship* 엘리먼트를 사용하여 기술한다.

엘리먼트 사이의 릴레이션쉽에서 테이블 사이의 릴레이션쉽으로의 매핑은 참조 타입과 컬렉션 타입을 조합하여 표현할 수 있다. 릴레이션쉽의 종류에는 카디널리티에 따라 참조 타입만으로 나타내는 일대일(one-to-one) 릴레이션쉽, 참조 타입의 컬렉션 타입으로 나타내는 일대다(one-to-many) 릴레이션쉽이 있다. 이때, 순서 정보를 유지할 경우에는 컬렉션 타입으로 리스트(list) 타입을 사용하고, 순서 정보를 유지하지 않을 경우에는 집합(set) 타입을 사용한다.

릴레이션쉽은 다시 릴레이션쉽의 방향에 따라 양방향 릴레이션쉽과 단방향 릴레이션쉽으로 구분될 수 있다. 양방향 릴레이션쉽에는 부모 엘리먼트에서 자식 엘리먼트으로 가는 릴레이션쉽과 자식 엘리먼트에서 부모 엘리먼트로 가는 릴레이션쉽이 있으며, 단방향 릴레이션쉽에는 부모 엘리먼트에서 자식 엘리먼트로 가는 릴레이션쉽만 있다.

이러한 릴레이션쉽의 매핑을 나타내는 *Relationship* 엘리먼트는 *parent, child, cardinality, isOrdered* 애트리뷰트를 가진다. *parent* 애트리뷰트는 부모 엘리먼트가 매핑된 테이블에서 자식 엘리먼트를 참조하는 컬럼을 나타내고, *child* 애트리뷰트는 자식 엘리먼트가 매핑된 테이블에서 부모 엘리먼트를 참조하는 컬럼을 나타낸다. *cardinality* 애트리뷰트는 릴레이션쉽의 카디널리티(cardinality)를 나타내며, *isOrdered* 애트리뷰트는 릴레이션쉽을 맺을 때에 순서 정보를 유지할 지의 여부를 나타낸다. 단방향 릴레이션쉽일 때는 매핑 파일에서 *Relationship* 엘리먼트의 *child* 애트리뷰트가 생략된다.

3.3. 객체관계형 데이타베이스로의 XML 문서 저장

XML 문서를 객체관계형 데이타베이스에 저장하는 알고리즘 StoreXML_ORDB 는 그림 3 과 같다. 알고리즘 StoreXML_ORDB 는 XML 문서의 각 엘리먼트를 하나씩 읽어가며 이를 데이타베이스에 저장한다. 라인 3-15 에서는 엘리먼트 E 가 테이블로 매핑된 경우 E 가 매핑된 테이블의 객체를 하나 생성하고, 부모 엘리먼트를 저장하는 객체 O_p 와 릴레이션쉽을 맺는다. 이때, 릴레이션쉽이 일대일 릴레이션쉽인 경우에는 자신의 OID 를 자신을 참조하는 객체 O_p 의 컬럼에 저장하고(라인 9), 일대다 릴레이션쉽인 경우에는 자신의 OID 를 자신을 참조하는 객체 O_p 의 컬럼의 원소로 저장한다(라인 11). 그리고 양방향 릴레이션쉽인 경우에는 객체 O_p 의 OID 를 객체 O_p 를 참조하는 자신의 컬럼에 저장한다(라인 13). 라인 16-22 에서는 엘리먼트 E 가 컬럼으로 매핑되는 경우 E 의 값을 매핑되는 객체의 컬럼에 저장한다. 라인 23-24 에서는 엘리먼트 E 에 속한 각 애트리뷰트의 값을 매핑되는 객체의 컬럼에 저장한다.

4. 결 론

본 논문에서는 오디세우스 객체관계형 DBMS 를 이용하여 XML 문서를 데이타베이스에 효율적으로 저장하고 관리하는 XML 저장 시스템인 ODYSSEUS/XMLStore 를 설계하고 구현하였다. ODYSSEUS/XMLStore 는 표준에 기반한 매핑 기술 방법을 제공하며, XML 문서를 객체관계형 데이타베이스에 효과적으로 저장하는 방법을 제공한다.

본 논문의 공헌은 다음과 같다. 첫째, XML 문서 구조에서 데이타베이스 스키마로의 매핑에 대해 분석하였다. 이를 위해 XML 문서 구조의 각 구성 요소들이 데이타베이스 구조의 각 구성 요소들에 어떻게 매핑되는지를 분석하였다. 둘째, 분석된 매핑을 기술하는 방법을 표준 언어인 XML Schema 를 활용하여 제안하였다. 셋째, 사용자가 정의한 매핑 정보에 의거하여 XML 문서를 객체관계형 데이타베이스에 저장하는 방법을 제안하였다. 이를 위해 XML 문서를 저장하는 세부 알고리즘을 제안하였다.

Algorithm StoreXML_ORDB
Input:
XML 문서

```
1: for (XML 문서에 속한 엘리먼트 E)
2: {
3:   if (E가 테이블로 매핑됨)
4:   {
5:     E가 매핑된 테이블의 객체 O를 생성;
6:     if (부모 엘리먼트를 저장하는 객체 O_p와 릴레이션쉽 R이 존재함)
7:     {
8:       if (R이 일대일 릴레이션쉽임)
9:         O의 OID를 O를 참조하는 O_p의 컬럼에 저장;
10:      else if (R이 일대다 릴레이션쉽임)
11:        O의 OID를 O를 참조하는 O_p의 컬럼의 원소로 저장;
12:      if (R이 양방향 릴레이션쉽임)
13:        O_p의 OID를 O_p를 참조하는 O의 컬럼에 저장;
14:    }
15:  }
16:  else if (E가 컬럼 C로 매핑됨)
17:  {
18:    if (C의 타입이 단순 타입임)
19:      E의 값을 매핑되는 O의 컬럼에 저장;
20:    else if (C의 타입이 단순 타입의 컬렉션임)
21:      E의 값을 매핑되는 O의 컬럼의 원소로 저장;
22:  }
23:  for (E에 속한 애트리뷰트 A)
24:    A의 값을 매핑되는 O의 컬럼에 저장;
25: }
```

그림 3. XML 문서를 객체관계형 데이타베이스에 저장하는 알고리즘.

참고 문헌

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

[2] Florescu, D. and Kossmann, D., A Performance Evaluation of Alternative Mapping Schemes for Storing XML Data in a Relational Database, Technical Report RR-3680, INRIA, May 1999.

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

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

[5] Microsoft Corp., Microsoft SQL Server 2000, 2000 (available from http://www.microsoft.com/sql/default.asp).

[6] Shanmugasundaram, J. et al., "A General Technique for Querying XML Documents Using a Relational Database System," *ACM SIGMOD RECORD*, Vol. 30, No. 3, 20-26, Sept. 2001.

[7] 한욱신, 이민재, 이재길, 박상영, 황규영, "오디세우스 객체관계형 멀티미디어 DBMS 의 아키텍처," 한국정보과학회 추계학술발표 논문집, pp. 45-47, 2000 년 10 월.

[8] Thompson, H. et al.: XML Schema Part 1: Structures, May 2001 (available from http://www.w3.org/TR/xmlschema-1).

[9] Chamberlin, D. et al., XQuery 1.0: An XML Query Language, June 2001 (available from http://www.w3.org/TR/xquery).