분산 객체 관계 데이터베이스 시스템을 이용한 분산 XML 문서 저장 시스템과 분산 XPath 질의 처리기 설계 및 구현*

이창주° 홍의경
서울시립대학교 컴퓨터과학부
{cjlee°, ekhong}@venus.uos.ac.kr

Design and Implementation of Distributed XML Storage System and Distributed XPath Query Processor using Distributed ORDBMSs

Changju Lee° Eui Kyeong Hong
Dept. of Computer Science, University of Seoul

요 약

최근 컴퓨팅 환경은 인터넷 환경의 웹을 기반으로 한 분산 컴퓨팅 환경으로 변화하고 있다. 그에 따라 XML 문서의 사용과 XML 문서의 양이 급속하게 증가하였으며, 언제라도 쉽게 필요한 XML 문서에 접근할 수 있어야 한다. XML 문서에서 정보를 검색하기 위하여 XPath 질의어가 널리 사용 중이며, XML 저장 기법과 XPath를 이용한 질의 처리에 대한 연구가 활발히 진행되고 있다.

본 연구에서는 분산 객체 관계 데이터베이스 시스템을 이용하여 XML 문서를 저장하고 관리하는 시스템을 설계하였으며, 분산된 XML 데이터를 접근할 수 있도록 하기 위해 XPath를 분산 SQL로 변환하여 실행하는 분산 XPath 질의 처리기를 설계 및 구현하였다.

1. 서 론

지식과 정보 교류의 기반이 웹 환경으로 바뀌면서 W3C에서는 기존의 HTML의 단점을 보완하고 SGML의 복잡성을 제거한 XML을 표준으로 제정하였다[1]. XML은 확장성과 문서 관계성 표현이 우수하여 새로운 정보 공유 환경의 표준 매체로 자리 잡아가고 있으며, 이기종 시스템 간의 정보 교환을 용이하게 하는 특성을 가지고 있어 EC/EDI, 전자 도서관, 전자 상거래 등 다양한 응용 분야에서 사용되고 있다.

XML 문서를 저장하기 위하여 관계형 데이터베이스 시스템에 XML 데이터를 통합하는 작업들이 XML 문서의 구조 정보를 저장함으로써 관계형 데이터베이스 시스템이 지니고 있는 질의 수행 능력 및 우수한 성능을 충분히 활용할 수 있고 기존 응용 시스템의 데이터를 활용할 수 장점이 있다[2,3,4].

최근 컴퓨팅 환경은 인터넷 환경의 웹을 기반으로 한 분산 컴퓨팅 환경으로 변화하고 있다. 그에 따라 XML 문서의 사용과 XML 문서의 양이 급속하게 증가하였으며, 언제라도 쉽게 필요한 XML 문서에 접근할 수 있어야 한다. 또한 다양한 형태로 분산 저장된 XML 문서에서 원하는 데이터를 추출하고 변환하며, 단편화된 XML 데이터를 통합하는 작업들이 필요하게 된다. 이를 해결하기 위해서 W3C에서는 XML 문서에 대한 표준 질의어로 구조적 질의어인 XPath를 핵심으로 하는 XQuery를 새로운 질의어의 표준으로 제정하였다[5,6]. 따라서 XML 문서를 분산 객체 관계 데이터베이스 시스템에 효율적으로 저장하는 시스템을 개발하고, 분산 저장된 XML 문서에서 사용자가 필요한 정보를 검색할 수 있도록 하기 위해 XQuery 질의어를 지원하는 연구가 필요하다.

본 논문에서는 분산 객체 관계 데이터베이스 시스템을 이용하여 XML 문서를 저장하고 질의하는 시스템을 설계하였으며, 서울시립대학교에서 설계한 XML 문서 저장 기법[3,4]을 분산 환경에 맞게 재설계하였고, 분산된 XML 데이터를 접근할 수 있도록 하기 위해 XPath를 분산 SQL로 변환하여 실행하는 분산 XPath 질의 처리기를 설계 및 구현하였다.

본 논문의 구성은 다음과 같다. 2절에서는 XML 문서의 저장 기법 및 XPath, 3절에서는 본 논문에서 설계한 분산 객체 관계 데이터베이스 시스템을 이용한 XML 저장 시스템에 대하여 설명한다. 4절에서는 XPath를 분산 객체 관계 데이터베이스 시스템에서 수행 가능한 분산 SQL문으로 변환하여 처리하는 분산 XPath 질의 처리기의 설계 및 구현을 논의한다. 마지막으로 5절에서는 결론 및 향후 연구 방향을 기술한다.

2. 관련 연구

2.1 XML 문서의 저장 방법

XML 문서를 저장하고 관리하기 위하여 파일 시스템, 객체 관계 데이터베이스 시스템을 이용한 방식, 네이티브 XML 데이터베이스 등이 연구되었다. 파일 시스템을 이용한 기법은 확장성이나 질의 처리에 제한이 많고 네이티브 XML 데이터베이스 시스템의 경우, 많은 부분에서 기술적으로 성숙되지 못하였다. 그래서 기존에 많이

* 본 연구는 첨단정보기술 연구센터를 통하여 과학재단의 지원을 받았음

연구되고 기술적으로 성숙한 객체 관계 데이터베이스 시스템을 이용한 방식이 활발히 연구되었다. 기존 데이터베이스 시스템을 이용하여 XML 문서를 저장하고 관리하면 질의 처리, 트랜잭션, 권한 시스템, 인덱스 기능 등 기존 데이터베이스 시스템의 기능을 활용할 수 있다.

기존 데이터베이스 시스템을 이용한 XML 저장 기법들은 접근 방식과 데이터 모델에 따라 분류할 수 있다. 일반적으로 XML 문서는 텍스트 중심 XML 문서와 데이터 중심 XML 문서로 나눌 수 있다. [7]에서는 이러한 XML 문서의 특징에 따라 가능한 저장 기법들을 구분하였다.

XML 문서 전체를 하나의 BLOB(Binary Large Object) 이나 CLOB(Character Large Object) 형태로 저장하거나 XML 문서의 노드를 데이터베이스의 릴레이션과 애트리뷰트로 나누어 저장하는 분할 저장 기법이 주로 사용된다. 분할 저장 기법은 XML 문서가 데이터베이스에 저장되는 구조에 따라 XML 문서의 XML 스키마나 DTD를 이용하는 스키마 종속적인 기법과 스키마 독립적인 기법으로 구분할 수 있다. 스키마 종속적인 기법은 XML 스키마에 따라 데이터베이스에 생성된 데이터베이스 스키마가 비효율적으로 모델링될 수 있는 단점이 있다. 그리고 XML 스키마나 DTD가 없는 문서가 스키마가 존재하는 문서보다 많으므로 본 논문에서는 XML 문서의 스키마와 관계없이 데이터베이스의 스키마를 정의하여 XML 문서의 정보를 저장하는 스키마 독립적인 저장 기법을 사용한다. 또한 다양한 XML 문서를 처리하기 위하여 XML 문서의 노드마다 오더 레이블링을 사용하여 텍스트 중심 XML 문서와 데이터 중심 XML 문서를 동시에 처리할 수 있도록 하였다.

XML 문서의 노드 오더 인코딩 기법으로는 글로벌 오더, 로컬 오더, Dewey 오더가 있다[8]. 글로벌 오더의 경우 XML 문서의 노드 순서대로 레이블링하는 방식으로 질의 처리 성능이 우수한 장점이 있지만 노드의 삽입, 삭제가 발생할 경우 노드의 레이블링을 다시 수행하여야 하는 단점이 있다. 로컬 오더는 노드의 갱신 성능이 우수하지만 노드들 간의 부모-자식 관계를 파악하기 어려운 단점이 있다. Dewey 오더와 같은 하이브리드 오더 기법은 갱신 연산에 따른 오버헤드를 줄이고 질의 처리 성능을 유지시키는 장점을 가진다[9].

2.2 XPath

XPath[5]는 1999년 W3C에서 XPath 1.0으로 권고(Recommendation)되었다. XPath는 XML 문서의 일부분을 지시하기 위해서 사용된다. XPath는 URL 경로 기법을 사용하여 XML 문서의 계층적인 구조를 논리적으로 탐색하는데 이 구조는 요소 노드(element node), 속성 노드(attribute node), 내용 노드(text node)를 사용하여 트리 구조로 형성된다. 또한 XPath는 각 노드의 문자열 값을 계산하는 방법을 정의하고 있다.

XPath는 노드의 경로를 나타내기 위해 위치 경로(location path)를 사용하는데, 위치 경로 표현 방식은 절대 경로와 상대 경로 두 가지 방식이 있다. 절대 경로란 / 로 시작하여 문서의 루트 노드를 시점으로 경로를 지시하는 것을 말하고, 상대 경로는 현재 노드를 기점으로 하여 요구되는 상대적인 노드의 경로를 나타낸다. 이때 위치 경로를 기초부(basis)와 술어부(predicate)로 나누어 설명할 수 있는데, 기초부는 축(axis)과 노드 검사 부분으로 구성된다. 여기서 축이란 문맥 노드(context node)와 경로 단계로 선택된 노드간의 관계를 명시한다. 한편, 술어부는 대괄호([,])로 묶인 표현식을 말하는데, 축에 관련된 노드집합을 새로운 노드집합으로 만들기 위한 필터링 작업을 한다.

본 연구에서는 XPath의 모든 기능을 만족하기보다는 XQuery에서 지원하는 생략된 표기만을 사용한다. 또한 단일 술어부에 대한 지원을 보장하며, 술어부에 사용되는 비교 연산자로는 관계 연산자만을 사용한다.

3. 분산 XML 관리 시스템

분산 객체 관계 데이터베이스에서 XML 문서를 저장 및 질의하기 위해 그림 1과 같은 분산 XML 관리 시스템을 설계하였다.

그림 1. 분산 XML 관리 시스템

분산 XML 관리 시스템은 JAVA 5.0버전을 이용하여 구성하였으며 객체 관계 데이터베이스로는 Oracle 10g를 사용하였다. 데이터베이스와 미들웨어 사이는 JDBC를 사용하여 연결하였다. 분산 객체 관계 데이터베이스의 네트워크 환경은 근거리 통신망(LAN)이다.

분산 객체 관계 데이터베이스를 구성할 때 참여 사이트는 해당 사이트에 저장된 XML 문서와 관련된 카탈로그의 뷰를 카탈로그 뷰 관리기에서 관리하고, 그것을 다른 모든 사이트의 카탈로그 뷰 관리로 전송하여 저장한다. 즉, 카탈로그 뷰는 모든 사이트에 완전 중복하여 저장한다. 완전 중복된 카탈로그 뷰 관리기의 카탈로그 뷰는 ROWA(Read One Write All) 알고리즘을 이용하여 관리한다[10]. 카탈로그 뷰 관리기에 저장되는 각 사이트의 카탈로그 뷰 정보에는 각 사이트에 저장된 XML 문서와 XML 문서에 대한 XML 스키마나 DTD 정보, 각 사이트의 릴레이션에 대한 정보 등이 포함된다.

네트워크 모듈은 분산 객체 관계 데이터베이스 시스템 상에서 각 사이트 간의 통신을 담당한다. 조정자와 참여자 간의 파일 이동, 질의 전달, 제어 흐름 등에 관한 정보가 네크워크 모듈을 통해 이루어진다[11].

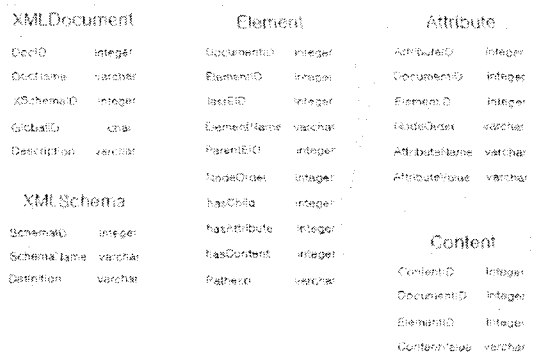


그림 2. 분산 XML 저장 스키마

본 논문의 저장 시스템은 XML 문서의 구조 정보를 유지하기 위해 엘리먼트의 경로를 기록하는 Pathexp 정보와 부모 엘리먼트 ID를 저장하고 있으며, 문서의 구조 검색 시 검색을 용이하게 하기 위하여 하이브리드 오더를 추가적으로 사용하였다[4]. 본 시스템의 분산 XML 저장 스키마는 그림 2와 같이 구성하여 각 노드의 정보를 분산 저장할 수 있게 하였다. XMLDocument 릴레이션의 GlobalID는 <XML 문서의 이름>.<XML 문서를 저장하는 사이트 주소> 형식으로 저장되어 XML 문서가 어느 사이트에 저장되어 있는지 검사하게 된다. 그리고 XML 문서의 이름과 DTD나 XML 스키마가 같더라도 저장되는 사이트가 다른 경우에도 XML 문서의 저장이 가능하도록 하였다.

XML 문서를 저장하는 것은 문서 저장관리기에서 담당한다. XML 문서를 저장하는 과정은 먼저 XML 문서의 유효성 검증 과정을 거친다. XML 문서가 DTD나 XML 스키마에 정의된 문법과 구조화 규칙 및 그 외의 추가적인 규칙을 따르는 Well-formed XML 문서인지를 판단한다. 그 후에 XML 문서를 파싱해야 하는데 파싱 방법에는 노드 중심의 DOM 파서와 이벤트 중심의 SAX 파서가 있다. SAX 파서가 DOM 파서보다 문서를 파싱하는데 걸리는 시간이 더 적고 대용량의 XML 문서를 파싱하는데 유리하기 때문에 본 논문에서는 SAX2 파서를 사용하였다. 문서를 파싱해서 발생된 이벤트 순서에 따라 문서 저장 관리기를 통해 엘리먼트, 애트리뷰트, 콘텐츠 테이블에 각각의 정보를 저장한다.

SAX 파서로 XML 문서를 파싱하게 되면 문서상에서 이벤트가 발생한 순간에 미리 구현된 메소드를 수행하게 된다. 본 시스템에서는 XML 문서의 각 노드의 정보를 이벤트가 발생한 상황에 즉시 처리한다. 이벤트가 일어난 상황에 오더 레이블링이 수행되며 하이브리드 오더링 방법으로 쉽게 고유한 ElementID를 부여할 수 있다. 또한 현재 이벤트의 위치 경로를 기록하기 위해 Pathexp 정보를 저장한다. XML 문서가 분산 객체 관계 데이터베이스 시스템에 저장된 후 XML에 관련되어 수정된 카탈로그의 변경 사항은 모든 사이트의 카탈로그 뷰 관리기에 전송되어 저장된다.

분산 저장된 XML 문서의 검색은 분산 XPath 질의 처리기를 통해서 수행되는데, 분산 XPath 질의 처리기는 XPath 질의를 분산 SQL 문으로 변환하여 처리하는 작업을 수행한다. 그리고 분산 XPath 질의 처리기에 의해 반환된 결과 집합은 XML 문서 생성기에서 DOM 형식의 트리 구조로 재구성되어 XML 문서 조각을 형성하게 된다. 이에 대한 세부적인 내용은 다음 절에서 알아본다.

4. 분산 XPath 질의 처리기 설계

XML 문서에서 정보를 검색하기 위해 XPath 질의어를 주로 사용하고 있다. 따라서 분산 객체 관계 데이터베이스 시스템을 기반으로 한 분산 XML 관리 시스템 역시 XPath 질의어에 대해 적절한 처리를 수행해야 한다. 하지만 XPath에는 분산된 XML 데이터와 관련된 질의 처리에 대한 부분이 없기 때문에 본 절에서는 3절의 분산 XML 관리 시스템 하에서 XPath 질의를 수행하기 위한 분산 객체 관계 데이터베이스 시스템의 질의어인 분산 SQL문으로 변환하여 처리하는 분산 XPath 질의 처리기와 XML 문서 생성기를 그림 3과 같은 구조로 설계하였다.

분산 XPath 질의를 처리하는 것은 분산 XPath 질의 처리기와 XML 문서 생성기에서 수행된다. XPath 질의 처리기는 사용자가 XPath 질의를 입력하면 카탈로그 뷰 관리기를 참조하여 적절한 형태의 분산 SQL문이 생성된다. XPath 질의는 본질적으로 엘리먼트의 범위 검색을 수행하는 질의이기 때문에 변환되어야 하는 분산 SQL문을 반환되는 요소 노드의 노드 오더 값을 구할 수 있도록 설계하였다. 그리고 XML 문서 생성기는 분산 SQL문을 수행한 후 반환된 결과 집합을 이용하여 XSL, XSLT 등 응용 프로그램에서 활용할 수 있는 DOM 트리 구조의 XML 문서를 생성한다.

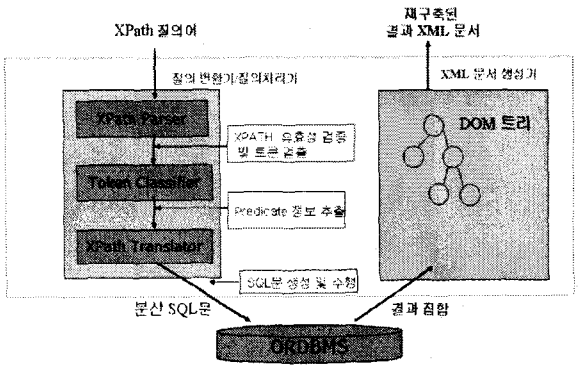


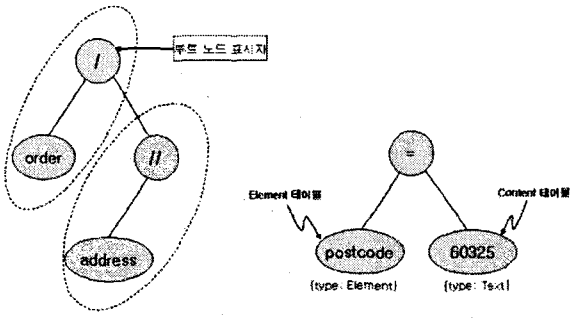
그림 3. 분산 XPath 질의 처리기와 XML 문서 생성기 구조

본 논문에서는 address의 postcode 값이 60325인 주문 사항의 주소를 확인하는 XPath 질의를 작성하면 "/order//address[postcode=60325]"이 되며, 이것을 각 단계별 처리 내용을 설명하는데 활용하였다.

4.1 분산 XPath 질의 처리기

본 논문의 분산 XPath 질의 처리기는 XPath 질의 파서 단계, 토크 분류 단계, 분산 SQL 질의 생성 단계와 같이 3단계로 구성하였다.

XPath 질의 파서 단계는 사용자의 XPath 질의의 유효성을 검증하고 축 타입, 노드 타입, 속성 타입, 문자 타입, 비교 연산자 타입 등의 토큰 정보를 생성한다. 즉, 예제의 경우 { /, order, //, address, [, postcode, =, 60325,] } 과 같은 토큰 정보가 발생한다.



(a) ContextNode 정보 (b) Predicate 정보

그림 4. 토큰 분류

토큰 분류 단계에서는 XPath 파서 단계에서 발생된 토큰을 문맥 노드 정보를 저장하는 ContextNode와 술어부 정보를 저장하는 Predicate로 분류한다. 만약 토큰이 술어부의 시작을 알리는 "[" 이 나타나기 이전이라면 문맥 노드에 해당하기 때문에 ContextNode에 저장하며 "//" 토큰이 있는지 검색한다. 술어부는 "[" 토큰 다음부터 "]" 이전까지 해당되며 이 부분의 토큰은 Predicate에 저장된다. Predicate에는 술어부에 대한 정보를 담으면서 비교 노드의 유형이 요소 노드인지 속성 노드인지를 확인하는 유형 판별 인자를 포함해서 저장한다. 위 예제의 XPath 질의 경우, 토큰 분류기에 의해 ContextNode에는 "/order//address" 정보가 그림 4의 (a)와 같이 저장되며, Predicate에는 "postcode=60325" 정보가 그림 4의 (b)와 같이 저장된다.

분산 SQL 질의 생성 단계는 반환해야 하는 요소 노드 정보를 확인하기 위해 토큰 분류 단계에서 구성된 ContextNode 객체와 Predicate 객체, 카탈로그 뷰 관리기의 카탈로그 뷰를 활용하여 분산 객체 관계 데이터베이스에서 실행될 수 있는 분산 SQL문으로 변환하는 단계이다.

ContextNode 객체는 요소 노드의 집합으로 표현되며 저장 스키마 중 엘리먼트 테이블에 대한 재귀 조인을 이용하여 반환할 범위를 결정하는데 최소 1개, 최대 N개의 엘리먼트 테이블이 사용된다. 경로 표현식을 나타내는 필드인 Pathexp 필드를 이용하여 요소 노드의 위치 정보를 사상하고, 요소 노드의 노드 오더 레이블링 값을 확인하기 위해 하이브리드 오더링 값을 이용하여 요소 노드의 시작 ElementID와 종료 ElementID를 구한다. 만약 ContextNode 객체에 "//" 위치 지시자 토큰이 존재하면 위치 정보를 단일 Pathexp필드에 사상할 수 없으며 여러 개의 Pathexp 필드에 대한 연산을 처리하여야 한다. 예제에서 ContextNode 객체가 "/order//address" 인 경우 "/order"와 "/address"인 두 개의 부요소 노드 정보로 나눠서 각각 문자열 검색을 통해 검색할 수 있도록 사상한다.

Predicate 객체는 비교 연산자의 유형에 따라 분류되는데, 만약 요소 노드의 내용을 검색하는 경우에는 엘리먼트 테이블과 콘텐츠 테이블이 관련이 있으며 엘리먼트 테이블은 최소 0번, 최대 N번 사용되고 콘텐츠 테이블은 최소 0번, 최대 1번만 사용된다. 콘텐츠 테이블에서 내용에 대한 검색 조건이 필요하고 엘리먼트 테이블에서는 해당 요소 노드를 검색하기 위한 조건문이 필요하다. 또한 두 테이블 간의 조인을 위해 조인 조건이 추가적으로 필요하다. 만약 속성 노드 정보를 필터링하는 검색일 경우에는 엘리먼트 테이블과 애트리뷰트 테이블이 관련되며 엘리먼트 테이블은 역시 최소 0번, 최대 N번 사용되고 애트리뷰트 테이블은 최소 0번, 최대 1번 사용된다. 애트리뷰트 테이블은 속성 노드의 이름과 값을 검색하기 위한 조건을 작성할 때 필요로 하고 엘리먼트 테이블은 해당 요소 노드를 검색할 때 필요하다. 두 테이블 간의 조인을 위해 조인 조건이 필요하다. 또한 Predicate 객체 안에 포함된 ContextNode 객체는 앞에서 설명한 ContextNode 객체 처리 단계에서 수행한 절차와 동일한 과정을 거쳐 수행된다.

그리고 분산 SQL 질의 생성 단계에서 카탈로그 뷰 관리기의 카탈로그 뷰를 참조하게 된다. 카탈로그 뷰는 해당 XML 문서가 저장되어 있는 사이트와 XML 문서의 엘리먼트, 애트리뷰트, 콘텐츠 릴레이션이 저장되어 있는 사이트에 대한 카탈로그 정보를 참조하여 분산 SQL로 변환한다.

만약 예제의 XML 문서의 이름이 order이고 저장된 사이트의 주소가 dblab1이라면 GlobalID는 order.dblab1이 되고, 앞의 3단계를 거치면 예제 XPath 질의인 "/order//address[postcode='60325']"는 하부 객체 관계 데이터베이스에서 정보를 검색할 수 있도록 하는 그림 5와 같은 SQL문으로 변환된다.

```
SELECT e1.NodeOrder as OrderLabeling e1.documentID as XML
FROM helement@order.dblab1 as e1, helement@order.dblab1 as e2,
     helement@order.dblab1 as e3, content@order.dblab1 as con1
WHERE instr(e2.pathexp, '/order' ) > 0
    and (e2.ElementID = e1.ElementID and e2.XML = e1.XML)
    and e1.ElementName = 'address' and instr(e1.Pathexp, 'address' ) > 0
    and (e3.ElementName = 'postcode' and instr(e3.Pathexp, 'postcode') > 0)
    and e3.ElementID >= e1.ElementID and e3.lastID <= e1.lastID
    and con1.hContentValue = '60325' and e3.ElementID = con1.ElementID
    and e1.XML = con1.XML
```

그림 5. 변환된 분산 SQL문

4.2 XML 문서 생성기

분산 XPath 질의 처리기에 의해 생성된 분산 SQL 질의는 결과값으로 반환해야 할 요소 노드의 오더 값을 가져온다. 본 예제 질의의 경우에는 오더 값 1.1을 가져온다. 반환된 요소 노드의 오더 레이블링 값을 이용하여

오더 레이블링 값 1.1 아래의 모든 요소 노드, 속성 노드, 내용 노드들의 집합을 구할 수 있으며, 응용 층에서 구해진 결과 값으로 DOM 트리를 구성할 수 있다. 즉, 해당되는 노드 정보를 가져오기 위해 새로운 SQL 문이 필요하다. 해당 노드 정보를 가져오기 위해 외부 조인을 이용하여 그림 6과 같이 하나의 SQL문으로 수행하면, 반환되는 결과 값이 요소 노드를 구성하는데 관련이 있는 속성 노드와 내용 노드를 같이 반환한다[4]. 따라서 하나의 개별 요소 노드를 바로 형성할 수 있다. 이와 같이 하면 그룹화된 요소 노드들 간의 상하 관계를 엮어서 그림 7과 같은 DOM 트리를 형성할 수 있다.

```
SELECT e1.NodeOrder as OrderLabeling e1.documentID as XML
FROM helement@order.dblab1 as e1, helement@order.dblab1 as e2,
     helement@order.dblab1 as e3, content@order.dblab1 as con1
WHERE instr(e2.pathexp, '/order' ) > 0
    and (e2.ElementID = e1.ElementID and e2.XML = e1.XML)
    and e1.ElementName = 'address' and instr(e1.Pathexp, 'address' ) > 0
    and (e3.ElementName = 'postcode' and instr(e3.Pathexp, 'postcode') > 0 )
    and e3.ElementID >= e1.ElementID and e3.lastID <= e1.lastID
    and con1.hContentValue = '60325' and e3.ElementID = con1.ElementID
    and e1.XML = con1.XML
```

그림 6. 문서 재구축을 위한 SQL 문

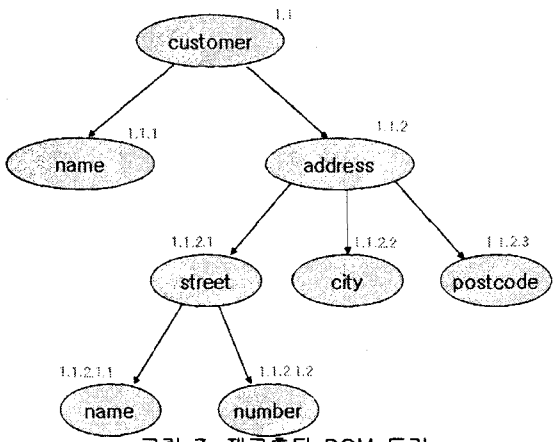


그림 7. 재구축된 DOM 트리

5. 결론 및 향후 연구 방향

본 논문에서는 분산 객체 관계 데이터베이스 시스템을 이용하여 XML 문서 저장하고 질의하는 시스템을 설계하였으며, 서울시립대학교에서 설계한 XML 문서 저장 기법을 분산 환경에 맞게 재설계하였고, 기존의 DFS 오더링을 사용한 것을 하이브리드 오더링을 사용하도록 수정하여 XML 문서를 분산 저장하도록 하였다. 또한 분산된 XML 데이터를 접근할 수 있도록 하기 위해 XPath를 분산 SQL로 변환하여 실행하는 분산 XPath 질의 처리기를 설계 및 구현하였다. 그리고 변환된 분산 SQL의 결과를 DOM형태로 XML 문서를 생성하여 타 응용 시스템에서 바로 사용할 수 있도록 하는 XML 문서 생성기를 설계 및 구현하였다.

향후에는 본 분산 XML 저장 시스템이 XML의 표준 질의어인 XQuery 언어를 지원할 수 있게 함으로써 더 폭넓은 범위의 질의 수행 능력을 갖출 수 있게 할 예정이다.

6. 참고 문헌

[1] World Wide Web Consortium, Extensible Mark Up Language (XML) 1.0", W3C Recommendation 4, Feb. 2004.

[2] J. Shanmugasundaram, et al., "Relational Databases for Querying XML Documents: Limitations and Opportunities," Proc. of the 25th VLDB Conf., pp. 302-314, Edinburgh, Scotland, Sept. 1999.

[3] 고영기, 홍의경, "분산 저장 시스템에 적합한 XPath 질의 처리기 설계," 한국정보과학회 가을학술대회 논문집 29(2), p;p.52-54, 2002.

[4] 김영우, 홍의경, " 객체 관계 데이터베이스 시스템과 하이브리드 오더 인코딩을 이용한 XML 저장 시스템 설계 및 구현," 한국정보과학회 추계학술발표회 논문집 32(2), pp. 154-156, 2005.

[5] World Wide Web Consortium, XML Path Language (XPath) 1.0, W3C Recommendation 16 Nov. 1999

[6] World Wide Web Consortium, XQuery 1.0: An XML Query Language, W3C Candidate Recommendation 8, June 2006.

[7] I. Mlynkava and J. Pokorny, "XML in the World of (Object-)Relational Database Systems," Proc. of the 13th Int'l Conf. on ISD, Vilnius, Lithuania, Sept. 2004.

[8] J. Shanmugasundaram, et al., "A General Technique for Querying XML Documents using a Relational Database System," ACM SIGMOD 30(3), pp. 20-26, 2001.

[9] P. O'Neil and E. O'Neil, "ORDPATHs: Insert-Friendly XML Node Labels," Proc. of ACM SIGMOD, Paris, France, pp. 903-908, 2004.

[10] V. Papadimos and D. Maier, "Distributed Query Processing and Catalogs for Peer-to-Peer Systems," Conf. on Innovative Data Systems Research, 2003.

[11] 정성룡, 홍의경, "분산 환경에서 객체 관계 데이터베이스 시스템을 이용한 XML 문서 저장 시스템 구현 및 성능 비교," 한국정보과학회 추계학술발표회 논문집 32(2), pp.151-153, 2005.