DOM 기반의 XML DBMS 저장구조 설계

노현종° 한창현 이규철
충남대학교 컴퓨터공학과
{neoswirl°, hanch1, kclee}@ce.cnu.ac.kr

Design of DOM Based Storage Structure for XML DBMS

Hyun-Jong Noh° Chang-Hyun Han Kyu-Chul Lee
Department of computer engineering, Chungnam National University

요 약

미래의 컴퓨팅 환경인 Ubiquitous 컴퓨팅에서는 USN을 이용하여 사용자의 위치, 환경, 서비스의 요구사항 등을 인지하고 서로 전달한다. USN에서는 XML 기반의 메시지 포맷을 사용한다. 그에 따라 대량의 XML(eXtensible Markup Language) 데이터를 고속처리 해야 하는 요구사항이 발생된다. USN과 같은 Ubiquitous 기술은 앞으로 계속 확산될 전망이며, 실시간으로 XML 데이터를 저장/처리해야 할 경우가 증가할 것이다. 이와 같은 추세에 따라 XML을 처리하기위한 데이터베이스의 요구가 점점 증가하고 있다. 본 연구에서는 그에 적합한 데이터베이스의 저장구조를 Main-Memory 데이터베이스와 XML 문서 처리 방법인 DOM을 바탕으로 하여 XML을 효과적으로 저장/처리하는 저장구조를 설계하였다.

1. 서론

미래의 컴퓨팅 환경인 Ubiquitous 컴퓨팅에서는 대량의 XML(eXtensible Markup Language) 데이터를 고속처리 해야 하는 요구사항이 발생된다. Ubiquitous 환경의 한 예인 USN(Ubiquitous Sensor Network)에서는 도처에 깔린 RFID 센서를 통해 필요한 정보를 수집하게 된다. 이 때, RFID는 XML로 표현된 EPC(Electronic Product Code) 데이터를 전송하도록 되어 있다.

Ubiquitous 컴퓨팅에서는 USN을 이용하여 사용자의 위치, 환경, 서비스의 요구사항 등을 인지하고 서로 전달하며, XML 기반의 메시지 포맷을 사용하여 그 작업을 수행한다. USN과 같은 Ubiquitous 기술은 앞으로도 계속 확산될 전망이며, 실시간으로 저장/처리해야 할 XML 데이터의 양은 급속도로 증가 될 것이다. 따라서 다양하고 대량의 XML 데이터를 신속하게 저장하고 처리할 새로운 형태의 XML DBMS가 필요하게 될 것이며, 이와 같은 DBMS는 차세대 컴퓨팅 환경에서 필수 요소가 될 전망이다.

본 논문에서는 앞으로 요구될 대량의 XML 데이터를 고속처리하기 위한 적합한 기술로서, XML 문서를 효과적으로 저장/처리하는 저장구조를 설계하였다.

본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT연구센터 육성•지원 사업의 연구결과로 수행되었음

2. 관련연구

기존의 XML DBMS의 저장구조는 관계 DBMS에 매핑시켜 저장하는 XML-Enabled DBMS와 XML의 특성을 살려 저장 관리하는 Native XML DBMS로 크게 나눌 수 있다[1]. XML-Enabled DBMS는 XML을 관계 DBMS로 매핑하는 과정에서 XML의 구조 정보를 손실 시킬 수 있어 효율성의 문제가 발생한다. Native XML DBMS는 XML의 특성에 맞는 처리 및 저장구조를 사용하기는 하나, 역시 Disk 기반의 DBMS이기 때문에 실시간 처리에는 한계를 가지게 된다.

이러한 한계점을 극복하려면 XML 처리를 실시간으로 지원할 수 있는 매커니즘을 가진 Main-Memory Native XML DBMS의 개발이 필요하다. Main-Memory DBMS는 Disk 기반의 DBMS보다 고속의 처리가 가능하다는 장점을 가지고 요사이 활발히 개발되고 있다[2]. 즉, Native XML DBMS에 Main-Memory DBMS의 실시간 특성을 결합한 Main-Memory Native XML DBMS는 대량의 XML 정보를 손실 없이 고속으로 처리할 수 있다. 현재 [3,4,5]등의 시스템이 Main-Memory XML DBMS라 주장하고 있으나, Main-Memory Native XML DBMS가 가져야 하는 요구 조건들을 모두 만족하고 있는 진정한 Main-Memory Native XML DBMS 시스템은 개발되지 못하고 있다. 그 이유는 Main-Memory의 실시간성을 살리면서, XML 구조 특성을 그대로 지원하는 저장/처리 기법이 개발 되어 있지 않기 때문이다.

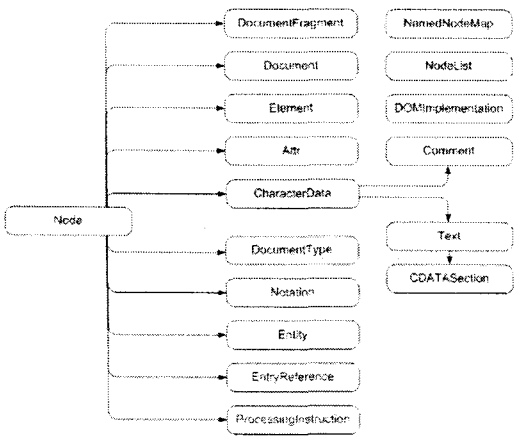
본 논문에서는 XML의 구조 특성을 그대로 지원할 수

있는 저장/처리 기법을 가지는 시스템을 설계하였으며, 고속처리에 적합하도록 하기 위해 Main-Memory DBMS를 기반으로 하여 설계하였다.

3. DOM의 구조

본 논문에서는 단순한 매핑에서 벗어나 저장을 XML 처리에 사용하고 있는 DOM(Document Object Model)[6] 표준을 활용하여 XML의 계층적 구조정보를 유지하면서, 처리 시에 발생하는 추가적인 변환을 최소화 하도록 하는 방법을 제시하고자 한다. 이를 위해서 DOM 표준의 구현을 확장하여 저장구조에서 사용하는 내부의 API를 활용한다. 데이터베이스에 DOM이 융합된 형태의 저장방식이 선택하였다. DOM을 구현한다고 했을 때, DOM의 데이터와, 노드간의 연결정보를 Record로 저장하고 Record를 탐색하는 Cursor를 DOM노드로부터 확장시켜 구현하여 실제로는 Record를 이동하지만 DOM 트리를 탐색하는 것과 같은 효과를 가지는 구조이다. 이와 같은 저장 구조를 구현하기 위해서는 저장하고자 하는 데이터 부분과 확장하여 사용할 DOM에 대한 이해가 필요하다.

W3C에서 정하고 있는 DOM 표준은 플랫폼과 프로그래밍 언어에 중립적인 인터페이스로 프로그램과 스크립트가 동적으로 콘텐츠에 접근하고 업데이트할 수 있도록 하는 문서의 구조이자 스타일이다. DOM은 트리와 같은 구조를 가진다. DOM 구조에서 핵심 인터페이스인 DOM Core가 존재한다. DOM Core의 구조는 다음 [그림2]와 같다.



[그림 1] DOM Core의 구조

2 DOM Core 구조로부터 DOM 트리를 구성하고 있는 요소들을 확인해 볼 수 있으며, Record로 저장해야할 데이터 정보와, DOM 노드와 같은 효과를 가질 Cursor가 수행해야 할 Method 들을 추출할 수 있다. 다음의 [그림 2]가 그중 Record로 저장해야할 Data 이다.

4. XML DBMS 저장구조

4.1. 전체 시스템 구조

본 논문에서 설계하고자 하는 시스템은 [그림 1]과 같은 형태이며 각각의 기능은 다음과 같다.

- XML DBMS Manager : XML DBMS의 최상단 API로서 XQuery 요청, 문서 입력/출력/삭제 등과 같은 기능에 대한 인터페이스를 제공한다. 더불어 데이터베이스의 컨트롤을 수행한다.

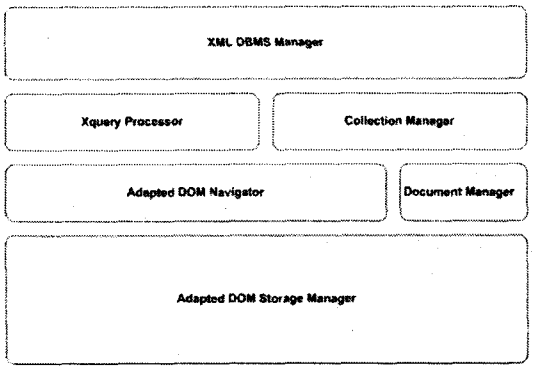
- XQuery Processor : XML DBMS의 요청에 따라 하위의 Adapted DOM Navigator를 이용하여 질의 수행하고 결과를 처리한다.

- Collection Manager : 본 연구에서 XML 문서는 하나의 Collection에 여러 개의 XML 문서를 저장하도록 하는 문서 관리 방법을 사용한다. 그에 따라 상위 XML DBMS Manager로부터의 XML 문서 입력/출력/삭제 등의 요청을 Collection 단위로 처리한다.

- Adapted DOM Navigator : W3C에서 정하고 있는 DOM 인터페이스를 구현함에 있어서 저장의 바탕이 되는 데이터베이스의 내부적인 저장 API를 활용하여 구현할 DOM을 Adapted DOM이라 부르고, 이 DOM을 Cursor처럼 이동하여 필요한 노드를 반환할 수 있도록 하는 처리를 한다.

- Document Manager : Collection Manager가 큰 틀의 문서 모음을 관리하는 역할을 수행할 때 개별 문서에 대한 작업을 수행할 때 이용하게 되는 부분으로, 입력 요청된 문서를 파싱하여 하위에 저장되도록 하는 작업을 수행한다.

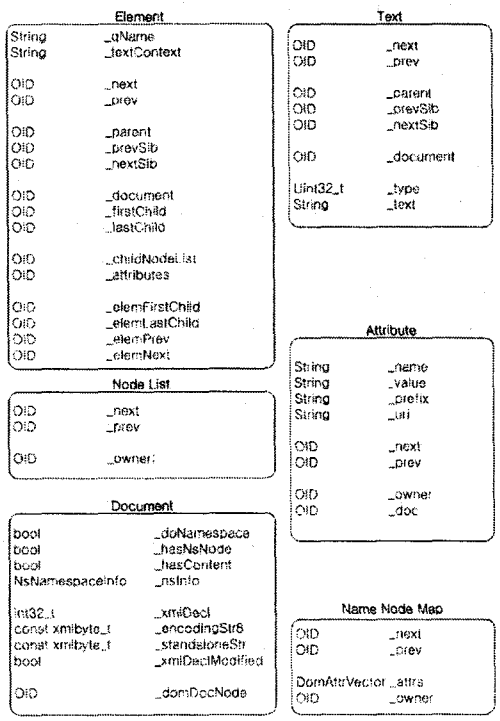
- Adapted DOM Storage Manager : 데이터베이스의 내부적인 API를 활용하여 구현된 DOM의 직접적인 처리와 문서의 인덱스를 요청 했을 때 인덱싱 작업을 수행한다.



[그림 2] 전체 시스템 구조

4.2. 저장구조

본 논문에서 DOM을 적용시킬 데이터베이스로 생각하고 있는 형태가 Main-Memory 데이터베이스이다. 고속처리가 관건인 환경에서 XML을 처리하고자 하는 것이 목표이기 때문에 Main-Memory 데이터베이스를 선택하였으며 [그림 3]에서는 그에 따라, Main-Memory 데이터베이스의 특징인 OID(Object Identifier)를 활용하고 있다.

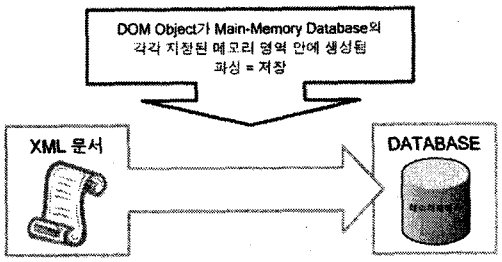


[그림 3] 저장에 필요한 DOM의 데이터

3 OID를 활용하여 레코드간의 직접 링크가 가능하며, 이것은 DOM의 노드간의 탐색을 그대로 반영할 수 있어 DOM 노드로부터 확장시킨 Cursor가 레코드에 바로 접근하여, 접근 자체가 단순한 DOM 구현에서의 노드 이동과 같은 효과를 가지게 된다.

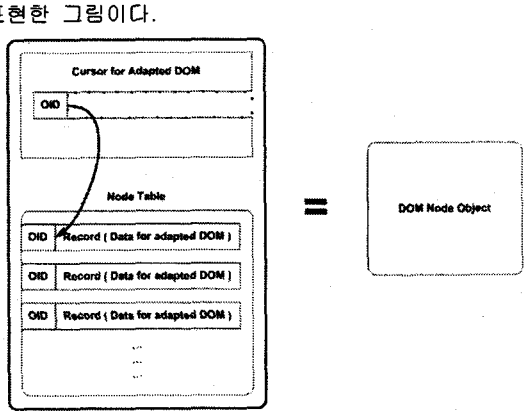
4.3. 시스템의 특징

본 논문에서 제안하고 있는 시스템을 통해서 XML 문서의 저장 처리를 수행한다면 다음의 [그림 4]에서와 같이 처리가 이루어진다.



[그림 4] 파싱 작업을 통한 저장

[그림 4]에서와 같이 처리할 때 파서는 기존의 Xerces파서를 이용한다. 파서에서 DOM을 구현하고 있는 부분의 구현 내용을 적용시키고자 하는 데이터베이스의 내부 저장관련 API를 사용하게 함으로써 파싱처리가 저장으로 이루어 질 수 있다. 저장된 데이터를 다시 활용하고자 할 때에는 Cursor와 같은 역할을 하는 DOM 노드를 확장한 Object를 이용하여 검색과 요청 문서의 Return을 하게 된다. 다음의 [그림 5]는 저장된 record가 DOM 노드 Object의 역할을 하는 Cursor의 모습을 표현한 그림이다.



[그림 5] 저장 데이터와 DOM 노드 Object

저장된 데이터는 DOM 노드의 확장인 Cursor의 하나의 부속으로 작동해서 Cursor가 DOM 노드 Object가 된다. 따라서 DOM으로부터 확장된 Cursor를 통해서 노드의 추가나 삭제등도 가능한 구조가 된다. 상위 XQueryProcessor에서 DOM을 통해서 XQuery 질의 처리를 수행한다면 DOM으로부터 확장된 Cursor가 필요로 하는 문서의 노드에 대한 record위치로 이동하면 곧바로 XQuery 처리로 이어질 수 있다. 그러나 이것은 가장 간편한 형태로 구현하고자 하는 경우 일 때 해당하는 내용이다. 좀 더 시스템의 성능을 향상시키기 위해서는 DOM 노드들을 인덱싱 하여 질의를 처리할 때 처리 속도를 높여줄 필요가 있다.

5. 인덱스의 적용

본 논문에서 제안하고 있는 시스템은 파싱이 저장으로 이어지는 구조이다. 또한 DOM 구조를 이용하고 있기 때문에 파싱은 문서의 구조적 정보를 트리 화 하는 작업과도 같다. 따라서 노드의 레이블링을 하는 작업역시 병행될 수 있는 있다. 파싱 시에 레이블링 이루어져 있으면 이후에 이 레이블 값과 노드의 값을 사용하는 인덱스 테이블을 구성하는 작업이 빠르기 이루어 질 수 있다. 레이블을 이용하는 인덱싱 방법들이 여러 가지가 있겠지만 그중에서 본 시스템에 적합한 방법은 노드의 추가 삽입과 관련해서 문제가 발생하지 않는 방법이 적합하다. ORDPATHs[7]와 같은 레이블링을 적용하여 활용 할 수 있다. ORDPATHs는 Microsoft사의 SQL Server에도 적용된 방법으로 Insert-Friendly한 XML의 노드 레이블링 기법이다.

6. 결론

본 논문에서 제안하고 있는 시스템은 기존의 방식들이 XML 타입의 데이터 타입을 만들어 적용하거나, XML 데이터를 CLOB, BLOB 등의 형태로 데이터베이스에 저장하는 기존의 방식, 혹은 단순 매핑 방식을 통한 XML 저자구조에서 벗어나, 저장구조가 XML 문서를 처리하는데 쓰이는 DOM 그 자체가 되도록 하였다. 그에 따라 저장 시 성능을 높이기 위해 XML 파싱 자체를 저장이 이루어지도록 하였고, 레코드의 검색을 위한 이동이 DOM 노드의 탐색이 되는 형태이다.

계층적 정보를 유지함에 있어서도, 추가적인 노드의 입력이나 삭제에 있어서도 적합하여, 향후에 XQuery에서 업데이트 기능을 지원한다고 해도 바로 적용할 수 있다.

인덱싱을 적용 시 인덱스에서 요구하는 레이블링에 따라 차이가 있을 수 는 있겠지만, 레이블링을 파싱타임에 처리할 수 있기 때문에 인덱스 적용과 생성에서도 효과적인 시스템을 제안하고자 하였다.

참고문헌

[1]R. Bourret, "XML Database Products : Native XML Databases," http://www.rpbourret.com/xml/ProdsNative.htm, 2005.

[2]H. Garcia-Molina and K. Salem, "Main Memory Database Systems :An overview," IEEE Transactions an Knowledge and Data Engineering, Vol.4, Issue 6, pp.509-516, Dec. 1992.

[3]dbXML Group, "dbXML(Native XML Database)," http://www.dbxml.com/product.html, Mar. 2004.

[4]Ellipsis, "DOMSafeXML: state-of-the-art XML database technology," http://www.ellipsis.nl/content/DOMSafeXML.htm, Jun. 2004.

[5]QuiLogic Inc, "SQL/XML-IMDB: In Memory SQL/XML Database Component for Universal Data Management", White Paper - V4.1, http://www.quilogic.cc/ WhiteP.pdf, 2006.

[6]A. L. Hors, P. L. Hegaret, L. Wood, G. Licol, J. Robie, M. Champion, S. Byrne, "Document Object Model (DOM) Level 3 Core Specification" http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/, Apr. 2004

[7]S. Pal, I. Cseri, G. Schaller, N. Westbury, "ORDPATHs: Insert-Friendly XML Node Labels," Proc. ACM SIGMOD Int'l Conf. on Management of Data, 2004, pp. 903-908.