Berkeley DB 기반의 DOM API 구현

황명진, 강동완, 박은경, 강병의, 강현석, 배종민
경상대학교 컴퓨터과학과

Implementation of DOM API based on the Berkeley DB

Myung-Jin Hwang, Dong-Wan Kang, Eun-Kyung Park, Byung-Eui Kang, Hyun-Syug Kang, Jong-Min Bae
Dept. of Computer Science, GyoungSang Nat'l University
E-mail : myungjh@hanmail.net, ryder, pek@rtp.gsnu.ac.kr, bekang@cue.ac.kr, hskang, jmbae@nongae.gsnu.ac.kr

요 약

내장형 시스템에서 XML 데이터의 역할과 데이터베이스의 지원이 중요해지면서, 내장형 XML DBMS에 대한 연구가 활발해지고 있다. 버클리 DB[1]는 SleepyCat에서 개발한 것으로 내장형 데이터베이스를 구현하는데 적합한 데이터베이스 시스템이다. XML은 메모리에서 DOM 트리로 구성되어, DOM API를 통해서 조작될 수 있다. 본 논문에서는 내장형 시스템의 부족한 리소스를 고려하여 DOM 트리를 메모리가 아닌 데이터베이스에 구성하는 시스템을 구현하고, 이를 확장하여 내장형 XML DBMS로 발전시킬 수 있는 방안을 검토해 보고자 한다.

1. 서론

인터넷의 성장과 무선 환경의 급속한 발전에 의해 이동 컴퓨팅 기술이 주목을 받고 있다. 즉, 컴퓨터와 통신 기술이 효과적으로 연계되어, 언제, 어디서든지 이동 환경에서 정보 교환 및 검색 등의 처리가 가능하게 되었다. 따라서 이기종 간에 주고받는 데이터 형식과 부가 서비스를 구현하기 위해 내장형 운영체제와 데이터베이스의 중요성이 커지고 있다. 최근 이러한 환경에서 데이터 형식으로 XML의 적용이 활발해짐에 따라, 내장형 데이터베이스에서 XML을 지원하는 것이 필요하다. 그런데, XML 문서를 처리하는 데 있어서 DOM을 기반으로 하는 방법은 이동 단말의 자원의 부족으로 인한 문제점이 있다. 본 연구에서는 이러한 점을 극복하고자 메모리에서 XML을 처리하는 대신에 데이터베이스에 DOM 트리를 구성하고, 이를 DOM API를 통해 다룰 수 있는 시스템을 구현하고자 한다. 그리고, 이 시스템을 확장하여 내장형 XML DBMS로 발전시킬 수 있는 방법을 모색해 본다.

논문의 구성은 다음과 같다. 2장에서는 버클리 DB에 대해서 설명하고, 3장에서는 DOM을 구성하는 요소와 메모리에서 DOM 트리가 어떻게 구성되는지 알아본다. 4장에서는 DOM 트리를 버클리 DB에 구성하기 위한 방법과 구현된 내용을 설명하고, 5장에서는 내장형 XML DBMS로 발전시키기 위한 고려 사항을 검토해 본다.

2. Berkeley DB

버클리 데이터베이스(이하 버클리 DB)[1]는 고성능의 병행 저장과 검색을 요구하는 응용에 사용할 수 있는 내장형 데이터베이스 시스템이다. 이 소프트웨어는 응용 프로그램과 직접 연결될 수 있는 라이브러리로서 배포된다. 이것은 C, C++, Perl, Tcl, Java 등을 위한 API들을 포함하고 있으며, 다양한 프로그램 인터페이스 접속 장치를 제공한다.

버클리 DB는 병행 처리, 트랜잭션 처리 등을 지원하는 하나의 기능으로 볼 때 완전한 DBMS이기는 하지만, 네트워크의 요구를 다루는 데이터베이스 서버가 아니며, 또한 질의를 실행하는 SQL 엔진도 아니다. 그리고 이것은 관계형 또는 객체 지향 데이터베이스

관리 시스템과 같은 고수준의 레이어 모델을 지원하지도 않는다. 단지 키/값의 쌍으로 구성된 레코드들을 저장하고 검색할 수 있는 기능을 제공하고 있다. 키와 값은 바이트 배열이며, 프로그램 언어에서 지원하는 어떠한 데이터 타입이나 구조라도 바이트 배열로 변환하여 저장할 수 있다.

3. 문서 객체 모델(DOM)

DOM[2]은 프로그램에서 XML 문서의 내용과 구조, 그리고 형식을 동적으로 접근하고 조작하기 위한, 언어에 독립적이고 어떠한 프로그래밍 환경에서도 사용할 수 있도록 설계된 프로그래밍 인터페이스이다. 이러한 DOM은 XML 문서를 객체 트리로 구성하고 처리한다. 이때 DOM 파서는 XML 문서 전체를 읽고 트리 구조로 만드는 역할을 수행한다.

DOM 트리를 구성하는 노드의 종류에는 엘리먼트, 텍스트, 처리지시, 주석, CDATA 섹션 노드등이 있다. 노드는 이름과 값을 가질 수 있으며, 부모 노드, 자식 목록, 형제 노드, 속성 등의 정보를 가질 수 있다.[3]

DOM 트리를 구성하기 위해서는 Parent Node(P), First Child Node(F.C), Last Child Node(L.C), Previous Sibling Node(P.S), Next Sibling Node(N.S)에 대한 정보를 하나의 노드에서 유지하면 된다. 그림 1은 DOM 트리를 구성하는 일부 노드들과 이들 노드가 메모리에서 표현되는 모습을 보여주고 있다. 그림 1에서 보는 것처럼 트리가 구성되어 있을 경우 메모리에서는 각 노드들의 참조가 유지되고 있다. 따라서, 노드에 대한 연산이 수행되면 부모 및 자식 노드, 형제 노드에 대한 참조 값이 변경된다.[4]

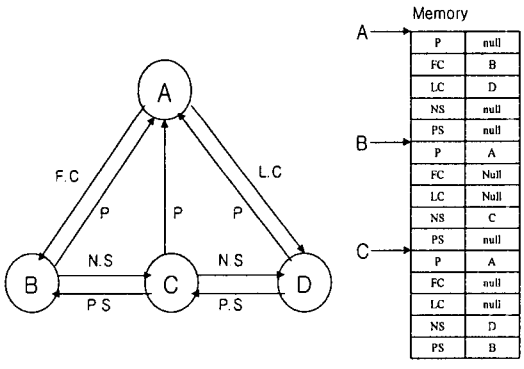

Fig.1 DOM Tree Representation

DOM API의 핵심은 Document와 Node 인터페이스이다. Document는 XML 문서를 나타내는 최상위 객체이다. Node는 위에서 설명한 기본적인 트리 구성 정보를 저장하는 기본 타입이다. 또, Document는 새로운 Node를 생성하기 위한 factory로도 동작한다. Node는 트리를 구성하는 기본적인 데이터를 표현하며, 모든 노드가 가져야 할 기능을 제공한다. 노드에 대해 그 부모, 형제, 자식 노드를 찾을 수 있고, Node를 추가, 삭제함으로써 문서를 수정할 수 있다. DOM API를 구성하는 인터페이스 계층은 그림 2와 같다. Node 인터페이스를 확장하는 인터페이스들은 XML 문서를 구성하는 객체들이 가지는 부가적인 기능을 포함하고 있다.

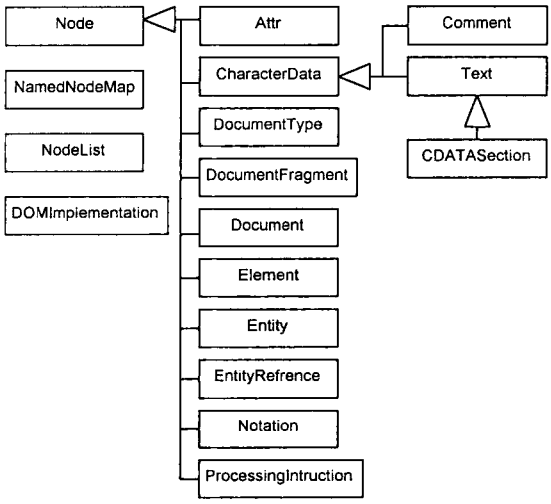

Fig.2 Interface hierarchy of DOM Level 1

4. 구현

우리는 3장의 DOM을 2장에서 살펴본 버클리 DB로 구현한다. 시스템의 구현을 위해 프로그래밍 언어로 자바를 사용하였다. 메모리에 구축되는 DOM 트리는 노드 객체들의 참조가 연결되어 구성된다. 참조는 메모리상의 객체를 식별할 수 있는 정보이고, DOM 트리를 데이터베이스에 구성하기 위해서는 참조 정보를 알 수 있어야 한다. 이를 위하여 객체에 고유한 ID를 부여하고 이것을 객체를 식별하는 정보로서 사용한다. 객체는 메모리에서 만들어 질 때 DB에 저장되어지면서 ID를 부여받고, 내부 데이터가 변경될 때마다 DB에 갱신되어진다.

버클리 DB는 키/값의 쌍으로 데이터를 저장하고 검색할 수 있는 낮은 수준의 API를 제공하기 때문에, 메모리에 생성된 노드 객체를 버클리 DB에 저장하기

위해서는 객체에 부여된 ID를 키로, 객체의 내부 데이터를 값으로 하여 저장해야 된다. 그림 3은 그림 1의 DOM 트리가 DB에 저장된 모습을 나타내고 있다. 이렇게 DB에 저장되어 있는 노드 정보를 사용하기 위해서는, 메모리에 객체로 다시 복원되어 져야 한다.

	Key	P	F.C	L.C	P.S	N.S
A	100		200	400		
B	200	100				300
C	300	100			200	400
D	400	100			300	

Node Object Data

Fig.3 Node Object in the Database

DOM 트리에 대한 연산은 먼저 연산하고자 하는 노드와 연산에 참여하는 노드들이 메모리에 객체로 복원되고, 연산이 수행된 후 다시 DB로 저장되는 방법으로 구현된다. 그림 4는 이러한 기본적인 아이디어를 기반으로 한 시스템의 전체 구조이다.

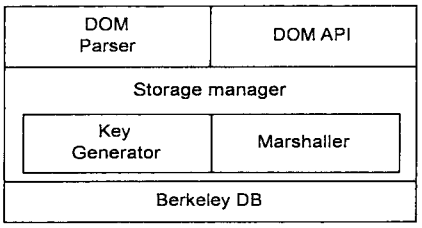

Fig.4 System Architecture

(1) Storage Manager

Storage Manager는 DOM API를 구현한 객체를 버클리 DB에 저장하고 복원하는 역할을 담당한다. Storage Manager는 객체의 고유한 ID를 생성하는 Key Generator, 그리고, 객체의 내부 데이터를 바이트 배열로 변경하고, 반대로 저장되어 바이트 배열을 다시 객체로 변환시키는 역할을 하는 Marshaller를 내부에 포함하고 있으며, 생성된 ID와 변환된 객체 데이터를 버클리 DB에 저장하는 역할을 담당한다.

Key Generator는 주어진 크기의, 임의의 바이트 배열을 생성하여 반환하는 데, 그 값이 유일하지 않다. 그러나 객체의 ID는 고유해야 하기 때문에, 객체를 버클리 DB에 저장하는 과정에서 이 문제를 보완하고 있다.

Marshaller에서 객체를 바이트 배열로 변환할 때, 그 형식은 다양하게 구현될 수 있다. 변환을 수행하기 위해서는 변환될 객체의 내부 데이터에 접근을 해야 한다. 그러나, 객체의 필드에 직접적으로 접근하는 방법은 캡슐화를 해치기 때문에 좋은 방법이 아니다. 그리고, 메쏘드를 통해 데이터에 접근하기 위해서는 새로운 인터페이스가 필요하다. 또는, 자바의 리플렉션을 이용할 수도 있다. 그러나, 새로운 인터페이스를 선언하는 방법은 객체마다 저장될 데이터가 다르므로 어려움이 있으며, 리플렉션을 사용하는 방법은 보안상 객체의 전용 필드에 접근할 수 없는 문제점이 있다. 본 논문에서는 변환 방법으로 자바의 직렬화 메커니즘을 사용한다. 직렬화란 자바 가상머신이 메모리상의 객체를 외부 저장 장치에 저장하기 위해 바이트 배열로 변환시켜주는 것을 말한다. 이 방법은 객체를 복원하기 위해서 역시 자바에서 제공하는 메커니즘을 사용해야 된다는 것인데, 이는 저장된 데이터를 다른 언어에서 접근하거나 또는 데이터의 일부분만을 엑세스하는 데에는 적합하지 않은 단점이 있다.

DB에 변환된 데이터를 저장하기 위해 버클리 DB에서 제공하는 API를 사용하여 객체의 ID를 키로, 변환된 바이트 배열을 값으로 하여 저장한다. 그리고 객체의 ID로 저장되어 있는 객체의 내용을 검색하고, 수정하고, 삭제할 수 있다. 객체의 ID는 객체 내부에 저장된다.

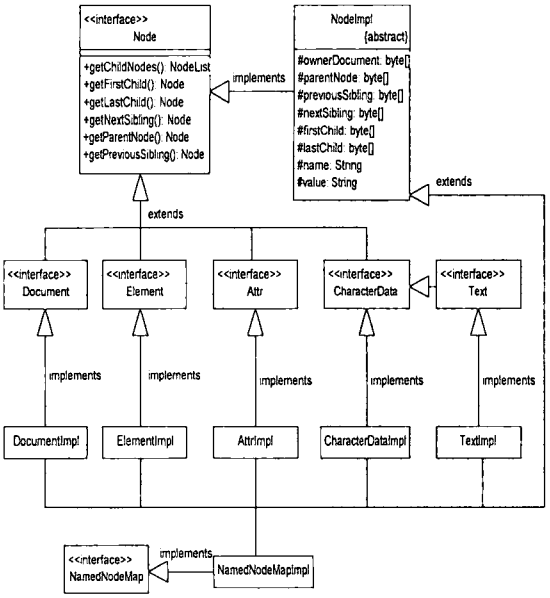

Fig.5 Class hierarchy for DOM Implementation

(2) DOM API 구현

DOM 트리는 DOM API를 구현하는 클래스의 인스턴스들로 구성된다. 그러므로 각 노드의 실제 클래스가 다를 수 있으므로 공통의 데이터와 기능을 한곳으로 모으고, 동일한 데이터 타입으로 접근할 수 있도록 하기 위한 최상위 클래스가 필요하다. 이 클래스는 Node 인터페이스를 구현하며, 노드에 대한 연산을 수행하는 기본 타입이 된다. XML 문서를 구성하는 객체들은 Node 인터페이스를 구현한 클래스를 확장하여 부가적인 기능이 추가된다. 그림 5는 버클리 DB에서 DOM API를 구현한 클래스의 계층도이다.

(3) DOM 파서

DOM 파서는 DOM 트리를 구성한다. 그리고, 내부적으로는 SAX 파서를 이용하고 있다. SAX Parser는 XML 문서에서 태그가 시작되거나 끝날 때, DTD, 주석, PCDATA 등이 나타나면 이벤트를 통지해 주는데, SAX Parser에 등록된 Document Handler에서 이러한 이벤트를 받아서 처리하게 된다. DOM Parser는 SAX Parser에 등록된 Document Handler이다. SAX Parser는 SUN에서 제공하는 JAXP[5]를 사용하였다.

DOM Parser는 DOM 트리를 구성하는 데 필요한 정보들을 유지하고 있다. DOM 트리를 구성하기 위해서는 Document 객체가 필요하고, Document 객체의 Factory 메쏘드를 사용해서 노드를 생성하게 된다. 노드를 추가하기 위해서는 추가될 노드의 부모 노드 객체가 필요하다. DOM Parser 에서는 스택을 이용하여 부모 노드의 ID를 유지하고 있다. XML 문서가 계층형으로 구성되어 있으므로 스택의 최상의 위치에 있는 노드 객체는 현재 노드의 부모 노드이다. 현재 노드를 DOM 트리에 추가하는 방법은 스택의 최상위에 있는 부모 노드의 ID를 이용하여 DB로부터 부모 노드 객체를 복원한 후, 노드 추가 연산을 수행하는 것이다. 연산의 결과는 다시 DB에 저장된다.

5. 결론

본 논문에서는 XML 문서의 DOM 트리를 메모리에서 구성하여 조작하는 대신에, 버클리 DB를 이용하여 구성하여 조작할 수 있는 시스템을 구현하였다. DOM 트리를 구성하는 노드 객체에 ID를 부여하고, 이 ID는 객체의 참조를 대신해서 사용된다. 즉, DOM 트리를 구성하는 노드 객체들 간의 참조는 ID에 의해서 수행된다. 버클리 DB에 객체를 저장할 때, ID를 키로, 객체 내부의 데이터를 바이트 배열로 변경한 것을 값으로 사용한다. 객체 내부 데이터를 바이트 배열로 변환하는 형식은 여러 가지 방법으로 정의될 수 있다. 이 시스템은 XML 문서를 DOM Parser를 사용하여 데이터베이스에 저장할 수 있으며, DOM API를 사용하여 저장된 XML 문서를 사용할 수 있다. 저장되는 데이터 형식을 새롭게 정의하여, 다른 API를 사용하여 XML 문서를 처리할 수도 있고, 데이터의 일부분만을 사용할 수도 있다.

이 시스템은 XML 문서의 구조적인 정보와 객체를 저장하기 위한 부가적인 데이터가 필요하기 때문에 XML 문서를 구성하는 노드의 개수가 늘어날수록 데이터베이스 파일의 크기가 크게 증가하는 단점이 있다. 이 시스템을 XML DBMS로 발전시키기 위해서는 XML 문서를 저장하는 데이터 파일의 개수와 저장되는 바이트 배열의 구조를 어떻게 구성할 것인지 고려해야 한다. 또한 질의를 처리하는데 있어서 DOM API를 사용하는 방법의 효율성에 대한 연구와 데이터 파일의 크기를 최소화하는 연구가 필요하다.

[참고문헌]

[1] M. Olson, K. Bostic, and M Seltzer, "Berkeley DB", Proceeding of the 1999 Summer Usenix Technical Conference, Monterey, California, June 1999

[2] DOM Level 2 Core Recommendation http://www.w3c.org/TR/2000/REC-DOM-Level-2-Core-20001113/

[3] H. Maruyama, K. Tamura, N. Uramoto, "XML and Java" Addison-Wesley 1999

[4] F. Boumphrey, O. Direnzo, et al, "Professional XML Applications", WROX, 1999

[5] Java API for XML Parsing 1.1 http://java.sun.com/xml/xml_jaxp.html