XDBox : XML 객체 저장소의 구현

김종익[O] 최일환 이현숙 김형주
서울대학교 컴퓨터공학부
{jikim[O], ihchoi, hslee, hjk}@oopsla.snu.ac.kr

XDBox : Implementation of XML object repository

Jongik Kim[O] Ilhwan Choi Hyun-Sook Lee Hyoung-Joo Kim
Dept. of Computer Engineering, Seoul National University

요 약

XML은 반 구조 데이터(semi-structured data)의 일종으로 데이터들이 정형화된 구조를 가지고 있지 않다는 것이 특징이다. 기존의 데이터베이스 시스템은 고정된 스키마를 통해 데이터를 저장 관리하기 때문에 정형화된 구조를 가지지 않는 XML 문서를 저장하기에는 한계가 있다.

본 논문에서는 정형화된 구조를 가지지 못하는 XML 문서를 효과적으로 저장하고 관리할 수 있는 저장장치를 구현하고 구현 사항에 대해 설명한다. 본 논문의 저장 장치는 XML을 DOM트리로 표현한 후 DOM트리의 각 노드를 객체로 만들어 저장한다. 본 논문의 저장 장치는 다중 사용자를 지원하기 위해 클라이언트 서버 구조로 구성되어 있으며 각 객체는 전용 파일, 관계형 시스템의 LOB등 다양한 하부 구조에 저장될 수 있다.

1. 서 론

XML[2]이 인터넷 문서 교환의 표준으로 채택 되면서 XML을 효율적으로 저장하고 관리하기 위한 연구가 계속되고 있다. XML은 반 구조 데이터의 일종으로 볼 수 있으며 정형화된 구조를 가지고 있지 않다는 것이 특징이다[1,3,16]. 이러한 특징으로 인해 XML은 많은 종류의 데이터를 쉽게 표현할 수 있다는 장점을 가지는 한편, 데이터의 효과적인 저장 및 관리가 어려워진다는 단점도 가지게 된다. 이에 따라, 현재 XML을 효율적으로 저장 관리하려는 연구가 많이 진행 중이다. XML 데이터를 저장하기 위한 방법은 크게 다음과 같이 나눌 수 있다.

첫 번째 방법은 관계형 데이터베이스 시스템[9,11,14]을 사용하는 방법이다. 이 방법은 기존의 데이터와 연동이 쉽고 성숙된 관계형 시스템의 기술을 모두 사용할 수 있다는 장점을 가지고 있다. 하지만 주어진 XML에 대해 효율적인 관계형 스키마를 만드는 것은 매우 어렵다. 많은 상용 시스템들에서는 XML 문서의 구조에 맞게 사용자가 직접 관계형 스키마를 기술하게 한 후 저장하는 방법을 사용한다[5,7,8]. 이러한 방법을 사용하면 저장된 데이터에 질의하기 위해 관계형 스키마를 알고 있어야 하며 질의를 SQL 형태로 표현해야 하기 때문에 XML 데이터에 투명한(transparent) 접근이 불가능하다. 이에 자동으로 관계형 스키마를 만들어 주는 방법도 제안되고 있다[14]. 하지만 XML에 대한 질의를 SQL 질의로 자동으로 변환해야 하며, 이러한 변환은 매우 어렵고 사실상 불가능한 경우도 발생하게 된다.

두 번째는 객체지향 데이터베이스 시스템을 이용해 XML을 저장하는 방식[6]이다. 하지만 객체지향 데이터베이스 시스템을 사용하는 경우, XML데이터의 구조에 따라 클래스를 만들어야 하므로 클래스의 개수가 크게 증가하는 단점을 가질 수 있다.

세 번째 방법은 전용 저장장치[4,12,13]를 만드는 방법이다. [12]의 경우 DOM 인터페이스 내의 각 클래스를 이용해 객체를 저장하는 방식을 사용하며, [4]의 경우 파일 시스템을 이용해 XML의 각 객체를 저장하는 방식을 사용한다. 엑셀론[10]과 타미노[15]와 같은 시스템도 이러한 범주에 속한다

본 논문에서는 XML 문서의 객체들을 W3C에서 표준으로 제안하고 있는 DOM[17]의 클래스 분류를 이용하여 분류한 후 저장하는 XDBox시스템을 구성하고 구현 방식에 대해 설명한다.

이러한 방식은 PDOM의 저장 방식과 유사하지만 PDOM의 경우 DOM 클래스 객체만을 저장할 수 있는 반면에 XDBox 시스템은 임의의 객체를 저장할 수 있도록 구현되었다. 또한 PDOM의 경우 자바 언어에서 지원하는 쓰레드 동기화 기법을 사용하여 2개 이상의 쓰레드를 사용하는 경우 거의 대부분 교착상태에 빠져 실질적으로 다중 쓰레드 지원을 하지 못한다고 볼 수 있다. XDBox 시스템은 다중 사용자를 지원하며 객체 단위의 락 관리를 구현하고 이를 이용해 서버 내의 쓰레드를 동기화함으로써 이러한 문제를 발생시키지 않으며 다양한 하부 저장소를 지원한다는 것이 큰 차이점이라 할 수 있다.

본 논문의 기여도(contribution)는 다음과 같다.

1. 다중 사용자를 지원하는 클라이언트/서버 구조의 XML저장소를 구현하였다.
2. 효율적인 객체 관리를 위해 객체를 저장 객체와 대리 객체로 나누어 사용하는 방법을 제안하고 구현하였다.
3. 병렬 트랜잭션을 지원하는 트랜잭션 제어를 구현하였다.
4. 다양한 하부 저장소를 지원할 수 있는 유연한 구조로 구성되어 있다.
5. 사용자가 DOM 인터페이스를 사용해 저장 장치에 저장되어 있는 XML 데이터를 투명하게 접근할 수 있도록 하였다.

본 논문의 구성은 다음과 같다.

2장에서는 XDBox 시스템 구조에 대해서 설명하고 3장에서는 클라이언트 시스템에 대해서 설명한다. 4장에서는 서버 시스템을 구성하고 있는 세션 관리자와 버퍼 관리자에 대해서 자세히 설명 한다. 5장에서는 결론 및 향후 연구에 대해 기술한다.

2. 시스템 구조

XDBox 시스템은 클라이언트 시스템, 서버 시스템, 저장소의 3 단계 구조로 이루어진다.

클라이언트 시스템은 기본적으로 사용자에게 DOM 인터페이스를 제공하는 역할을 수행한다. DOM 인터페이스의 각 함수들은 필요한 경우 클라이언트 캐시를 통해 서버 시스템의 기능을 이용하며 사용자에게는 저장 시스템의 내부를 숨기는 역할을 수행한다. 서버 시스템은 세션 관리자와 버퍼 관리자로 이루어져 있다. 세션 관리자는 각 클라이언트 세션을 관리하며 버퍼 관리자를 이용해 클라이언트의 요구를 처리해 준다. 버퍼 관리자는 저장소에 저장되어 있는 객체를 읽어서 버퍼에 유지하고 버퍼 내의 객체를 저장소에 기록하는 일을 담당한다. 하부에는 다양한 저장소가 존재할 수 있지만, 버퍼관리자는 동일한 인터페이스로 하부 저장소를 접근하게 된다. 그림 1은 전체 시스템 구조를 보여준다.

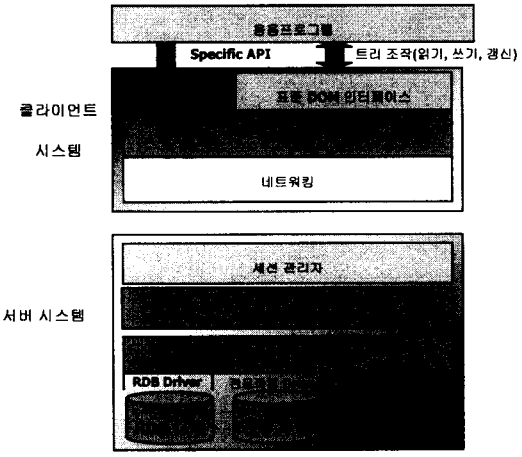

그림 1. XDBox시스템 구조

3. 클라이언트 시스템

클라이언트 프로그램에서 사용하는 모든 객체는 클라이언트 캐시에 의해 관리된다. 이때 응용프로그램이 필요한 객체를 캐시에서 복사해서 사용하는 경우 응용프로그램 작성자가 객체의 일관성(consistency) 유지를 담당해야 하는 문제가 생긴다. 이러한 문제를 막기 위해 응용프로그램은 클라이언트 캐시 내의 객체를 사용하고 각 클라이언트의 캐시는 객체의 일관성 유지를 책임지도록 하면 된다. 하지만 이와 같이 캐시를 이용하는 객체 관리 방법은, 응용프로그램이 동시에 사용하는 객체의 수가 캐시의 크기보다 더 많은 경우에 문제가 생긴다. XDBox 시스템에서는 이러한 문제를 막기 위해 저장 객체(Persistent Object)와 대리 객체(Proxy Object)의 두 가지 타입의 객체를 사용한다. 저장객체는 객체가 저장될 때 필요한 데이터와 그 객체를 유일하게 식별할 수 있는 객체 식별자를 유지하고 있는 객체이다. 클라이언트 캐시 및 서버에서 유지되는 객체는 저장 객체이며 저장 객체는 네트워킹 및 저장의 기본 단위가 된다. 응용프로그램이 객체를 요구하는 경우 클라이언트 캐시는 해당 객체에 대응하는 대리 객체를 생성해 응용프로그램에 넘겨준다. 대리 객체는 자신과 대응하는 저장 객체에 대한 식별자와 저장객체에 필요한 연산(operation)만을 가질 뿐 실제 데이터는 포 함하지 않는다. 따라서 대리 객체는 매우 작은 크기로 생성된다. 이때 DOM 인터페이스의 각 함수들이 대리 객체의 연산이 된다. 즉, DOM 인터페이스는 대리 객체를 통해 구현되어 있다. 대리 객체의 각 연산은 필요한 경우에만 캐시에서 저장 객체를 얻어와 연산을 수행하게 된다. 대리 객체를 사용함으로써, 응용프로그램에서 유지하고 있는 대리 객체에 대응되는 저장 객체가 클라이언트 캐시에서 제거되더라도 응용프로그램은 문제없이 동작하게 된다.

일반적으로 클라이언트와 서버의 메시지 교환은 클라이언트의 요구에 의해 항상 동기화 되어있다. 이러한 환경에서 서버 내에서 변경된 객체에 대한 정보를 각 클라이언트에 알릴 수 있도록 하기 위해서 XDBox 시스템에서는 클라이언트 시스템이 서버의 비동기적인 메시지를 전송받을 수 있는 이벤트 청취자(event listener)를 포함하고 있다. 이벤트 청취자는 서버로부터 이벤트(메시지)를 받은 후 해당 작업을 수행하는 일을 담당한다. XDBox 시스템에서 서버로부터 전송되는 이벤트는 다음의 두 가지이다. 첫 번째 이벤트는 다른 클라이언트에 의해 변경된 객체 리스트이다. XDBox 시스템에서는 클라이언트는 캐시 일관성을 위해 주기적으로 서버의 객체를 폴링(Polling)하지 않고 대신, 서버에서 변경된 객체 리스트가 이벤트 청취자를 통해 전달될 경우에만 캐시의 내용을 수정한다. 클라이언트 캐시는 변경된 객체의 리스트 중 캐시에 포함하고 있는 객체가 있으면 단순히 그 객체를 캐시에서 삭제하고 이 후에 필요한 경우 서버에 객체를 요구하여 사용하므로 캐시 내용의 수정은 전혀 부담(overhead)을 가지지 않는다. 두 번째 이벤트는 서버 장애로 인해 클라이언트 시스템이 종료되어야 하는 경우의 메시지이다.

4. 서버 시스템

4.1 세션 관리자

세션 관리자는 현재 서버에 연결되어 있는 클라이언트 세션(session)들을 유지하면서 각 세션의 요구를 처리해 주는 역할을 수행한다. 세션 관리자는 각 세션에 대해 세션 번호, 락(lock) 리스트, 이벤트 청취자의 정보를 포함하고 있다. 세션 번호는 각 세션을 유일하게 구별할 수 있는 번호이며 락(lock) 리스트는 해당 클라이언트 세션에 의해 락이 걸려있는 객체들의 객체 식별자 리스트이다. 클라이언트 세션이 서버의 동의 없이 임의로 종료되었을 경우 세션 관리자는 그 세션의 락 리스트를 보고 해당 객체들의 락을 풀어준다. 서버는 각 클라이언트의 이벤트 청취자에 연결하여 필요한 메시지를 전송해야 하므로 세션 관리자는 각 클라이언트의 이벤트 청취자 정보를 유지하고 있어야 한 다.

세션 관리자는 위와 같은 세션 관리 외에 클라이언트의 DOM 사용을 최적화하기 위해, DOM트리 순회와 같이 읽기 전용으로 객체를 사용하며 트리의 각 노드를 순회할 때마다 매번 클라이언트와 서버 사이에 객체 교환이 일어나야 하는 경우, 즉 사용자가 단순히 XML 문서 하나를 통째로 읽기 위해 DOM 연산을 사용하는 경우에 서버 시스템에서 해당 DOM 트리를 메모리 DOM으로 구성한 후 한번에 클라이언트에게 전달할 수 있는 API를 제공한다. 이렇게 함으로써 클라이언트와 서버 사이의 통신비용을 최적화 할 수 있으며, 사용자가 임시로 객체를 생성하거나 변경하고자 할 때도 유용하게 사용될 수 있다. 특히, 작은 크기의 파일이 대량으로 저장되어 있고, 읽기 전용 작업이 수행되는 환경을 지원하기 위해서 여러 개의 XML 파일을 메모리 DOM형태로 만든 후 한번에 전송할 수 있는 기능도 제공한다. 이러한 기능은 클라이언트와 서버 사이의 통신부하를 줄여 성능 향상에 기여할 수 있게 된다. XDBox 시스템을 이용해 구축된 XR-03 : 업적관리 시스템[18]의 환경이 이러한 예라 할 수 있다.

4.2 버퍼관리자

버퍼 관리자는 하부의 저장소를 관리하는 역할을 담당한다. 버퍼 관리자는 동일한 인터페이스를 통해 하부 저장소를 접근한다. 하부 저장소의 각 페이지에 저장되는 각 객체는 XML 문서를 파싱(parsing)한 결과로 나오는 파스 트리의 각 노드이다. 앞서 언급한 바와 같이 XDBox 시스템에서는 XML을 DOM으로 파싱한다. 각 노드는 XML 문서 내의 데이터뿐만 아니라 자식 노드들과의 연결(link) 정보등 기타 여러 가지 정보를 가지게 된다. 따라서 대부분의 경우 XML 문서에 대한 파스 트리의 크기는 그 문서의 크기보다 큰 크기를 갖게 되고 따라서 저장되는 데이터의 크기도 원래 문서의 크기보다 커질 수 있게 된다. 버퍼 관리자는 저장 형태의 데이터의 크기에 따른 통신 부담을 줄이기 위해 선택적으로 페이지를 압축하는 기능을 제공한다.

저장되어 있는 각 객체는 자신이 저장되어 있는 저장소에 의해 지역 식별자를 할당받는다. 버퍼관리자에 의해 사용되는 객체의 전역 식별자는 지역 식별자와 저장소 식별자를 합쳐서 만들어진 식별자이다.

버퍼 관리자는 클라이언트의 병렬 트랜잭션의 처리를 위해 각 객체에 대한 락(lock)을 지원한다. 객체에 대한 락은 읽기 락과 쓰기 락의 두 가지 종류가 있을 수 있으며 버퍼 관리자는 이 두 가지 락을 지원하기 위해 객체에 읽기 락을 거는 연산과 쓰기 락을 거 는 연산 그리고 읽기 락이 걸려 있는 객체를 쓰기 락을 걸도록 변환하는 연산의 세 가지 종류의 연산을 수행한다. 각 클라이언트는 객체를 얻어가기 전에 반드시 그 객체에 락을 걸어야한다. 또한 그 객체의 사용이 끝난 후에는 반드시 객체에 걸었던 락을 해제해야 한다. 클라이언트는 자신의 캐시에 보관된 객체일지라도 반드시 서버에 락 요청을 하고 락 승인을 얻은 후에야 그 객체를 사용할 수 있다. 이렇게 함으로써 다중 클라이언트가 하나의 객체를 동시에 갱신하는 것을 막을 수 있다.

5. 결론

XML이 인터넷 문서 교환의 표준으로 채택되면서 많은 데이터들이 XML 형태로 만들어지고 있다. 기존의 문서들과는 달리 문서의 구조를 기술할 수 있는 XML의 특징으로 인해 XML을 체계적으로 저장하고, 이를 효과적으로 질의하려는 노력이 계속되고 있다. 특히, 기존의 데이터베이스 시스템을 이용해 XML을 저장하고 관리하려는 시도가 많이 이루어지고 있다. 하지만 기존의 데이터베이스 시스템은 데이터에 대한 고정된 스키마를 이용해 데이터를 저장하고 그 스키마를 기반으로 하는 질의 언어를 가지고 있기 때문에 유연한(flexible) 구조를 특징으로 하는 XML을 저장하기에는 한계가 있다.

본 논문에서는 XML을 효율적으로 저장 관리할 수 있는 XML 저장 시스템을 구현하고 그 구현 사항에 대해 설명하였다. XDBox 시스템은 시스템과 하부 저장소를 분리하여, 다양한 하부 저장소를 지원한다는 것이 큰 특징이다. 또한 다양한 하부 저장소의 객체를 함께 사용할 수 있도록 전역 객체 식별자를 사용하며, 저장을 위한 객체와 사용을 위한 객체를 분리함으로써 응용 프로그램이 효과적으로 많은 양의 객체를 사용할 수 있다는 장점을 가지고 있다.

향후 연구로 효과적인 색인 구조를 고안하고 색인을 이용해 DOM 인터페이스의 구현함으로써 DOM 인터페이스를 사용하는 응용프로그램(예를 들어, 질의처리기)을 수정하지 않아도 색인의 효과를 얻을 수 있는 색인 방법에 대해 연구할 예정이다.

6. 참고문헌

[1]Serge Abiteboul. Querying semi-structured data. In Proceedings of the International Conference on Database Theory, 1997.

[2]T. Bray, J. Paoli, and C. Sperberg-McQueen. Extensible markup language (XML) 1.0. Technical report, W3C Recommendation, 1998.

[3]Peter Buneman. Semistructured data. In Proceedings of ACM Symposium on Principles of Database Systems, 1997.

[4]G. Moerkotte C. Kanne. Efficient storage of XML data. In IEEE International Conference on Data Engineering, 2000.

[5]J. Cheng and XuJ. XML and DB2. In IEEE International Conference on Data Engineering, 2000.

[6]V. Christophides, S. Abiteboul, S. Cluet, and M. Scholl. From Structured Documents to Novel Query Facilities. In Proceedings of the ACM SIGMOD International Conference on the Management of Data, 1994.

[7]Microsoft Corp. Microsoft SQL Server. http://www.microsoft.com/sql/default.asp.

[8]Oracle Corp. XML Support in Oracle 8i and Beyond. Technical Whitepaper.http://otn.oracle.com/tech/xml/htdocs/xml_twp.html

[9]Alin Deutsch, Mari Fernandez, and Dan Suciu. Storing semistructed data with STORED. In Proceedings of the ACM SIGMOD International Conference on the Management of Data, 1999.

[10]eXcelon Corp. Excelon - the EBusiness Information Server. http://www.exln.com

[11]Daniela Florescu and Donald Kossmann. Storing and querying XML data using an RDBMS. IEEE Data Engineering Bulletin, 1999.

[12]Peter Frankhouser Gerald Huck, Ingo Macherius. PDOM: Lightweight Persistency Support for the Document Object Model. In ACM Conference on Object Oriented Programing, Systems, Languages and Applications, 1999.

[13]J. McHugh, S. Abiteboul, R. Goldman, D. Quass, and J. Widom. Lore: A Database management system for semistructured data. SIGMOD Record, 1997.

[14]Jayavel Shanmugasundaram, H. Gang, Kristin Tufte, Chun Zhang, David DeWitt, and Jeffrey F. Naughton. Relational Databases for Querying XML Documents: Limitations and Opportunities. In Proceedings of the Conference on Very Large Data Bases, 1999.

[15]Software AG. Tamino Information Server for Electronic Business, Technical Whitepaper. http://www.tamion.com/tamino/Download/tamino.pdf

[16]Dan Suciu. Semistructured data and XML. In Proceedings of International Conference on Foundations of Data Organization, 1998.

[17]W3C. Document Object Model(DOM). http://www.w3c.org/DOM

[18]iTcamp, XR-03 : XML기반 업적 관리 시스템, http://oopsla.snu.ac.kr:8088/bib/main.html, 2002