제20회 한국정보처리학회 추계학술발표대회 논문집 제10권 제2호 (2003. 11)

XML 문서에 대한 RDB와 ORDB의 비교

연재훈*, 박현주
한밭대학교 정보통신전문대학원
e-mail : chyon@dblab.hanbat.ac.kr*, phj@hanbat.ac.kr

A comparison of RDB & ORDB about the XML Documents

Chai-hun Yon*, Hyun-ju Park
Graduate School of Information & Communications, HANBAT National University

요 약

XML이 인터넷상에서 데이터의 교환 수단으로 널리 사용됨에 따라, 데이터를 처리하고 관리하는 다양한 데이터 모델의 DBMS에서 이를 수용하고 관리하는 도구를 제공하고 있다. XML이 데이터의 교환 수단으로 사용되는 가장 큰 이유는 자기 기술 문서화의 기능과 동적인 확장성을 가지고 있기 때문인데, 이를 통해 서로 다른 언어나 플랫폼에서 다른 형식의 데이터를 자신의 시스템의 맞게 변환할 수 있다. 문제는 XML 문서의 데이터를, 문서의 구조와 상관없이 자신이 사용하고 있는 데이터 모델로 저장하고 관리하려고 한다는 점이다. 이로 인해 데이터 관리상의 비용은 증가하고, DBMS의 성능은 저하된다. 본 논문에서는 XML 문서를 관리할 데이터 모델을 RDB와 ORDB로 한정하고, XML 문서가 지니는 구조적 특성에 따라 그에 맞는 데이터 모델을 제시한다. 고려되는 XML 문서의 구조적 특성은 평면 구조의 XML 문서와 계층 구조의 XML 문서이다.

1. 서론

XML(Extensible Markup Language)[1]은 인터넷상에서 기업간의 데이터 정보 교환의 표준으로 급속히 발전하였다. 이에 따라, 기업에서도 데이터 처리를 통한 정보를 얻기 위해서 XML 형태로 만들어진 데이터를 DB에 저장하거나, 반대로 DB에 저장된 데이터를 XML 형태로 만들 필요가 생긴다. XML이 자기 기술 문서화의 기능과 동적인 확장성이라는 특성을 가지고 있기 때문에[2], 어떤 데이터 모델에서도 데이터를 변환하여 저장 가능하다. 이로 인해, 저장 후 데이터를 관리하는 비용은 간과될 수 있다. XML 표준을 사용하는 문서가 많아지면 많아질수록, 이러한 비용은 증가하게 되고 DBMS의 성능은 떨어지게 된다. XML 표준을 사용하는 문서가 증가하고 있는 지금, XML 문서에 어울리는 데이터 모델(Data Model)을 고려해 보는 것은 바람직한 일이다.

본 논문에서는 XML 문서와 어울리는 데이터 모델을 찾기 위해서, XML 문서의 구조적인 측면에서 접근한다. 즉, XML 문서를 평면(flat)인 경우와 계층(hierarchical)인 경우로 나누고, 데이터 모델을 RDB와 ORDB로 한정하여, XML 문서의 구조에 따라 RDB와 ORDB에 미치는 영향을 비교 분석한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구로 XML 문서의 구조 종류와 저장을 위한 데이터 모델에 대해서 설명한다. 3장에서는 XML 문서의 구조를 평면 구조와 계층 구조로 구분하여, 각각의 구조에 알맞은 데이터 모델에 관하여 설명한다. 4장에서는 실험을 통하여 3장의 타당함을 보인다. 마지막으로 5장에서는 결론을 맺는다.

2. 관련 연구

2.1 XML 문서의 구조

XML 문서의 구조는 크게 평면 구조와 계층 구조로 나눌 수 있다. 평면 구조의 XML 문서는 루트 엘리먼트와 루트 엘리먼트의 자식 엘리먼트들로 구성되어 있다. 모든 자식 엘리먼트들은 루트 엘리먼트와 직접 관계를 맺고 있고, 서로 동등한 위치의 형제 관계로서 존재한다. 자식

엘리먼트들은 루트 엘리먼트와의 관계만을 통해서 문서의 일부분이 된다. 반면에 계층 구조의 XML 문서는 루트 엘리먼트와 루트 엘리먼트의 자식 엘리먼트, 자식 엘리먼트는 다시 하위에 자식 엘리먼트를 갖고, 하위의 자식 엘리먼트도 자식 엘리먼트를 가진다. 이런 식으로 엘리먼트들이 상위·하위 관계를 맺어 하나의 문서로 결합되어, 계층 구조를 이룬다.

2.2 데이터 모델

본 논문에서 고려할 데이터 모델은 관계형 데이터 모델과 객체관계형 데이터 모델이다. 먼저 RDB에 관련해서 살펴보겠다. RDB에서는 데이터를 테이블, 레코드, 그리고 컬럼으로 나누어 저장하고, 테이블 간에 관계를 만들어 준다. 일반적으로 테이블마다 그 테이블을 식별하는 주키를 설정하고, 참조하는 테이블을 가리키기 위한 외래키를 만든다. 이렇게 만든 외래 키는, 제약 조건을 통해서 참조하는 테이블을 가리키게 하므로, 테이블 간의 관계가 설정된다.

다음으로 ORDB에 관해서 살펴보겠다. ORDB는 RDB와 OODB의 장점들을 가지고 있어서, XML 문서의 가장 큰 특징인 계층 구조를 가장 자연스럽게 표현할 수 있다. 일반적으로 XML 문서의 루트 엘리먼트를 주 테이블로 생성하고, 그 자식 엘리먼트들을 주 테이블의 컬럼으로 생성한다. 이 컬럼의 데이터 타입은 객체 타입이나 네스티드 테이블 타입으로 선언한다. 객체 타입은 동일한 이름을 가지는 엘리먼트의 개수가 1개인 경우에 선언한다. 네스티드 테이블 타입은 동일한 이름을 가지는 엘리먼트의 개수가 2개 이상인 경우에 선언한다.

본 논문에서는 살펴본 RDB와 ORDB에서 평면 구조와 계층 구조의 XML 문서를 관리할 경우, 어떠한 영향을 미치는지 비교 분석하고 대안을 제시한다.

3. XML 문서의 관리

3.1 XML 문서의 예

3.1.1 평면 구조의 XML 문서

그림 1은 평면 구조의 XML 문서이다.

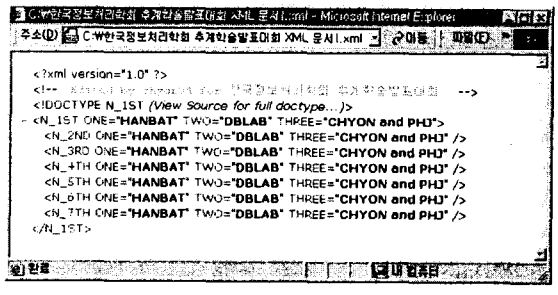

그림 1

그림 1의 XML 문서를 도식하면 그림 2와 같다.

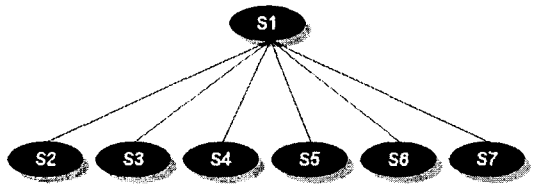

그림 2

루트 엘리먼트 S1이 있고, 자식 엘리먼트들로 S2, S3, S4, S5, S6, S7이 있다. 자식 엘리먼트들은 서로 동등한 위치의 형제 관계로서 존재한다. 자식 엘리먼트들은 루트와의 관계만을 통해서 문서의 일원으로 결합된다. 이러한 구조의 문서를 평면 구조의 XML 문서라고 한다.

3.1.2 계층 구조의 XML 문서

그림 3은 계층 구조의 XML 문서이다.

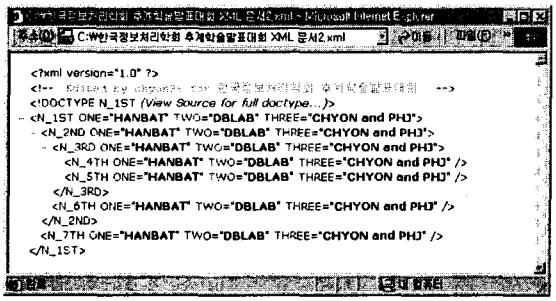

그림 3

그림 3의 XML 문서를 도식하면 그림 4와 같다.

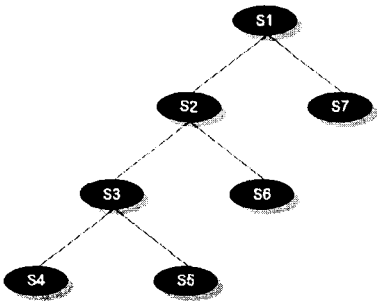

그림 4

루트 엘리먼트 S1의 자식으로 엘리먼트 S2와 S7이 있고, 엘리먼트 S2의 자식으로 엘리먼트 S3과 S6가 있으며, 엘리먼트 S3의 자식으로 엘리먼트 S4와 S5가 있다. XML 문서의 엘리먼트들은 상위·하위의 엘리먼트와 관계를

맺어 하나의 문서로 결합된다. 이러한 구조의 문서를 계층 구조의 XML 문서라고 한다.

3.2 XML 문서의 저장

XML 문서의 저장을 위한 데이터베이스는 RDB와 ORDB로 나눌 수 있다. 일반적인 XML 문서는 계층 구조이기 때문에 ORDB에 더 적합하다. 본 절에서는 그림 1의 평면 구조와 그림 3의 계층 구조를 가진 XML 문서를 RDB와 ORDB로 저장할 경우에 어떤 영향을 미치는지 살펴본다.

XML 문서를 RDB에 저장하는 경우 각 엘리먼트마다 하나의 테이블을 생성해야 한다. 그림 1과 3의 XML 문서의 경우, 7개의 엘리먼트가 존재하므로 7개의 테이블에 데이터를 나누어 저장한다. 하나의 문서를 저장하려면 7개의 입력 명령문을 필요로 한다. 또한 주 키와 외래 키의 제약 조건을 만족하는지 검사해야 하기 때문에, 트랜잭션 처리가 복잡해진다. 반면에 ORDB는 객체 개념을 사용하여 한 문서를 한 테이블 만으로도 저장할 수 있다. 따라서 한 문서당 하나의 입력 명령문만을 필요로 하기 때문에 트랜잭션 처리가 단순하고 빠르다. 결국 특수한 형태의 XML 문서가 아닌 일반적인 XML 문서의 경우에, 저장 시간은 RDB보다 ORDB가 더 빠르다.

3.3 XML 문서의 검색

본 절에서는 RDB와 ORDB에 저장된 XML 문서를 검색할 경우를 설명한다. 특정 컬럼에 저장된 데이터를 검색하는 경우와 특정 테이블에 저장된 데이터를 검색하는 경우를 평면 구조와 계층 구조로 구분하여 설명한다.

3.3.1 특정 컬럼 검색

그림 2와 같은 평면 구조에는 RDB가 적합하다. 각각의 테이블이 모두 주 테이블과 관계를 맺고 있어서, 검색시 필요로 하는 조인 비용이 계층 구조보다 훨씬 적게 든다. 또한 RDB에서는 엘리먼트마다 하나의 테이블을 생성해서 따로 저장하기 때문에, 조인이 필요 없는 테이블을 검색할 때에는 좋은 성능을 보인다. 깊이 1인 테이블의 컬럼을 검색할 때에는 조인이 필요하지 않기 때문에, RDB에서 가장 빠르게 검색한다. 깊이가 2인 테이블의 컬럼을 검색할 때에는, 주 테이블과 해당 컬럼이 속한 테이블과의 조인을 필요로 하기 때문에 RDB의 성능은 약간 저하된다. 이 경우에는 ORDB가 RDB보다 약간 빠른 성능을 보인다. 하지만 그다지 큰 차이가 나지 않고, 테이블간의 조인 비용도 작다. 따라서 평면 구조의 XML 문서는 RDB에서 관리하는 것이 바람직하다.

다음으로 그림 4와 같은 계층 구조를 살펴보겠다. 계층 구조에는 ORDB가 적합하다. ORDB에서는 XML 문서의 복잡한 트리 구조를 자연스럽게 표현 가능하고, 복잡한 질의를 안정성 있게 처리할 수 있다. 반면에 RDB는 상당한 성능 저하가 생긴다. 평면 구조와는 달리, 테이블의 깊이도 깊을 뿐만 아니라 테이블 간의 관계도 수직관계에 있다. 따라서 검색시 생기는 조인 비용이 상당히 커지고, 이에 따라 테이블의 깊이가 깊어질수록 검색 비용도 커진다. 또한 저장되어 있는 문서의 양에 커다란 영향을 받는데, 문서의 양이 선형적으로 증가하면, 조인되는 레코드의 수는 기하급수적으로 증가한다. 그림 4의 경우, 테이블의 최대 깊이는 4이다. 깊이 1인 테이블의 컬럼은 평면 구조와 마찬가지로 아무런 조인을 필요로 하지 않기 때문에, RDB에서 가장 빠르게 검색한다. 깊이 2인 테이블의 컬럼도 평면 구조와 비슷한 양상을 보인다. 하지만, 깊이 3인 테이블부터는 RDB의 검색 성능이 확연하게 떨어진다. 테이블의 깊이가 깊어질수록 계층 특성을 잘 반영하는 ORDB에서 두각을 나타내고, 결국 일정 깊이 이상이 되면 ORDB가 가장 좋은 검색 성능을 보인다.

3.3.2 특정 테이블 검색

특정 테이블의 검색은 특정 깊이에 있는 테이블의 컬럼 전체를 검색하는 것을 의미한다. 따라서 특정 컬럼을 검색하는 것과 비슷한 결과가 나타난다. 깊이 1인 테이블의 경우, 조인을 필요로 하지 않기 때문에 RDB가 ORDB보다 더 나은 성능을 보인다. 깊이 2인 테이블부터는 조인을 필요로 하기 때문에, RDB의 성능이 점차 떨어진다. 깊이 3부터는 조인으로 인한 검색 비용의 증가로 RDB의 성능이 확연히 떨어지고, ORDB는 깊이 1이나 깊이 2의 테이블을 검색할 때와 비교해서 별다른 저하를 보이지 않는다.

3.3.3 결론

조인을 필요로 하지 않는 컬럼이나 테이블 검색에는, 평면 구조든지 계층 구조든지, RDB에서 가장 빠르다. 일단 조인을 필요로 하는 깊이 2이상인 테이블의 검색부터는 RDB의 성능이 저하되나, ORDB와 별다른 차이가 나지 않는다. 하지만 깊이 3부터는 조인으로 인한 검색 비용의 증가로 RDB의 성능이 확연히 떨어지고, ORDB는 별다른 저하를 보이지 않는다. 따라서 최대 깊이가 2인 평면 구조의 XML 문서는 RDB에서 관리하는 것이 적합하며, 최대 깊이가 정해지지 않은 계층 구조의 XML 문서는 ORDB에서 관리하는 것이 적합하다.

4. 실험 및 결과

본 장에서는 3장에서 살펴본 평면 구조의 XML 문서와 계층 구조의 XML 문서를 직접 DBMS에 저장하고 검색하여 나온 결과를 통하여 본 논문에서 다룬 내용과 일치하는지를 살펴보겠다.

4.1 실험 환경

실험 환경은 다음과 같다.

■서버

■Intel Pentium 4, CPU 1.7GHz X 2, 1GB RAM

■Linux kernel 2.4.13 version

■DBMS : Oracle9i Database Release 2 Enterprise /Standard Edition for Intel Linux

■클라이언트

■Intel Pentium 4, CPU 2.4GHz, 512MB RAM

■Microsoft Windows 2000 Advanced Server

■Oracle9i Database Release 2 Client for Windows 98/NT/2000/XP

■Java(TM) 2 Runtime Environment, Standard Edition 1.3.1_08

4.2 실험 방법

실험에 기본이 되는 문서는 그림 1과 그림 3의 XML 문서이다. 실험시 고려 사항은 다음과 같다.

■실험시 고려 사항

■XML 문서 구조의 구분 : 평면 구조, 계층 구조

■문서의 저장 수 : 10만개

■깊이에 따른 구분 : 깊이 1부터 4까지 존재

■문서의 검색 수 : 10만개

■문서의 검색 대상 : 특정 컬럼, 특정 테이블

■저장된 문서의 양이 미치는 영향

문서를 저장할 때는 프로시저를 사용하여 서버 측에서 저장하였고, 검색할 때는 클라이언트 측에서 JDBC를 사용해서 데이터를 검색하였다.

4.3 문서의 저장과 검색

XML 문서를 DB로 저장하기 위한 매핑 절차는 현재 가장 일반적으로 사용되는 방법들을 사용하여[3,4], 그림 1과 그림 3의 XML 문서가 최적으로 DB에 저장되도록 필자가 수작업 하였다.

표 1과 표 2는 각각 그림 1과 그림 3의 XML 문서를 검색한 결과 값이다. 표에서 세로항목으로 검색대상이 되는 컬럼과 테이블이 있고, 괄호안의 숫자는 동일한 이름의 엘리먼트의 개수를 의미한다. 가로항목으로는 데이터 모델과 테이블이 있고, 괄호안의 숫자는 테이블의 깊이를 의미한다. 결과 값은 십만(十萬) 개의 XML 문서를 검색할 때 걸리는 시간이고, 단위는 초(秒)이다.

표 1

평면구조	S1(1)		S2(2)		S3(2)		S4(2)	
검색대상	RDB	ORDB	RDB	ORDB	RDB	ORDB	RDB	ORDB
컬 럼(1)	5,001	5,389	6,697	6,271	6,422	6,087	6,936	5,271
테이블(1)	8,712	6,962	7,742	6,935	7,853	7,166	7,819	7,071
컬 럼(2)	5,077	5,931	13,328	10,51	12,468	10,587	12,396	10,574
테이블(2)	6,506	6,793	14,577	13,561	15,388	13,035	15,468	13,445

표 2

계층구조	S1(1)		S2(2)		S3(3)		S4(4)	
검색대상	RDB	ORDB	RDB	ORDB	RDB	ORDB	RDB	ORDB
컬 럼(1)	4,183	5,402	6,056	5,415	7,488	5,447	9,009	5,361
테이블(1)	6,619	6,986	8,674	7,097	9,353	7,318	10,278	7,201
컬 럼(2)	4,353	4,838	14,154	11,506	29,568	24,435	77,373	72,73
테이블(2)	6,727	7,022	15,44	13,616	36,258	31,454	83,283	74,568

위의 결과를 보면, 조인이 필요 없는 깊이 1인 테이블을 검색할 때에는 XML 문서의 구조와 상관없이 RDB에서 성능이 좋음을 알 수 있다. 깊이 2인 테이블부터는 RDB의 성능이 떨어지고, ORDB의 성능이 향상된다. 깊이가 깊어질수록 RDB의 성능은 급격히 떨어지고, ORDB의 성능은 그대로 유지된다. 따라서 3장에서 설명했듯이, 평면 구조의 XML 문서는 RDB에서 관리하는 것이 적합하며, 계층 구조의 XML 문서는 ORDB에서 관리하는 것이 적합하다.

5. 결론

본 논문에서는 XML 문서의 효율적인 관리를 위해 RDB와 ORDB를 비교 분석하였다. XML 문서를 평면 구조와 계층 구조로 구분하고, 이를 실제로 RDB와 ORDB로 저장하여, 어떤 데이터 모델에 적합한가를 실험을 통해 보여 주었다.

참 고 문 헌

[1] T.Bray et al.,"Extensible Markup Language (XML) 1.0 (Second Edition)", http://www.w3.org/TR/2000/REC-xml-20001006, 2000.

[2] Jayavel S. et al., "Efficiently publishing relational data as XML documents", The VLDB Journal 2001. 10. pp.133~154

[3] Kevin W. et al., Professional XML Databases, Wrox Press, 2001.

[4] David H. et al., Beginning XML 2nd Edition, Wrox Press, 2002.