XML 문서의 저장 임계점에 관한 연구

연재훈°, 박현주
한발대학교 정보통신전문대학원
chyon94@hanmail.net°, phj@hanbat.ac.kr

A study of storable-critical point in the XML documents

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

요 약

XML은 W3C에서 웹 문서의 표준으로 제안한 마크업 언어로, 인터넷상에서 데이터를 표현하고 교환하는데 사용되고, 이를 DBMS에 저장하고 관리하려는 연구가 많이 진행되었다. 현재에는 관계형 데이터 모델, 객체지향형 데이터 모델, 그리고 객체관계형 데이터 모델 등 다양한 데이터 모델에서 XML 문서를 관리할 수 있다. 하지만 DBMS가 아무리 뛰어난 기능을 발휘한다하더라도 해당 XML 문서의 구조와 맞지 않는 구조를 가지고 있다면, XML 문서는 효율적으로 관리될 수 없다. 따라서 XML 문서를 DBMS에 저장하기 전에, 그 XML 문서를 어떤 데이터 모델에 저장하는 것이 효율적인지를 미리 판단할 수 있다면 매우 바람직한 일일 것이다. 본 논문에서는 저장 임계점을 사용하여, 일반적인 XML 문서를 어떤 데이터 모델을 사용하여 저장하는 것이 효율적인지를 제시하고자 한다.

1. 서론

XML(Extensible Markup Language)[1]의 핵심적인 특성인 자기 기술 문서화와 동적인 확장성은 다양한 어플리케이션들 사이에서 그리고 느슨히 결합된 분산 업무 처리들 사이에서 필요로 하는 유연성을 제공한다. XML은 특정 언어나 플랫폼에 의존하지 않는다. 따라서 인터넷상에서 데이터를 표현하고 교환하는데 사용한다. 현재 다양한 DBMS들에서 XML을 처리하는데 필요한 도구들을 지원한다[2].

XML 문서의 사용자가 많아짐에 따라, XML 문서를 DB에 저장하려는 연구가 많이 행해졌다. 비용이나 여러 가지 여건상, 기존에 사용하고 있는 하부 저장시스템을 기반으로 하여 XML 문서를 저장하는 DBMS가 대부분이다. 이러한 DBMS들은 대부분 XML 문서의 구조를 해당 DB로 매핑하기 위한 도구를 지원한다. 하지만 DBMS가 아무리 뛰어난 기능을 발휘한다하더라도 해당 XML 문서의 구조와 맞지 않는 구조를 가지고 있다면, 효율적인 데이터 관리를 위한 추가 비용이 들게 된다. 따라서 XML 문서를 DB에 저장하기 전에, 그 문서의 특성을 파악해서, 그 문서에 맞는 데이터 모델을 찾는다면 효율적인 데이터 관리가 이루어질 것이다.

본 논문에서는 저장 임계점(Storable-critical Point)을 정의하고, 저장 임계점을 사용하여 일반적인 XML 문서(General XML Documents)를 어떤 데이터 모델(Data Model)에 저장할 것인가를 결정하는 방법을 제시한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구로 XML 문서의 구조적인 특징과 저장하기 위한 데이터 모델에 대해서 설명한다. 3장에서는 저장 임계점을 사용해서 데이터 모델을 선택하는 방법에 대해서 설명한다. 4장에서는 저장 임계점을 실험을 통해서 계산한다. 마지막으로 5장에서는 결론을 맺는다.

2. 관련연구

2.1 XML 문서의 구조적인 특징

XML 문서는 하나의 루트 엘리먼트가 존재하고, 루트 엘리먼트의 자식 엘리먼트가 존재하며, 다시 그 자식 엘리먼트의 자식 엘리먼트가 존재하는 방식으로, 계층 구조(Hierarchical Structure) 형태를 취한다. 각각의 엘리먼트에는 속성들이 존재한다. 또한 하나의 엘리먼트의 자식으로서 여러 개의 다른 이름의 자식 엘리먼트들이 존재할 수도 있고, 동일한 이름의 자식 엘리먼트들이 존재할 수도 있다. XML 문서의 구조적인 특징은 하나의 루트 아래 모든 데이터 정보가 저장되는 계층 구조라는 것이다.

이러한 계층 구조의 XML 문서를 DB에 매핑할 경우, 다른 이름의 엘리먼트는 각각 이름마다 다른 테이블을 생성시키고, 동일한 이름의 엘리먼트는 하나의 테이블을 생성시켜 데이터를 저장한다. 엘리먼트 안의 속성들은 테이블의 컬럼으로 생성된다.

2.2 XML 문서를 저장하기 위한 데이터 모델

현재 XML 문서를 저장할 수 있는 데이터 모델로는 관계형 데이터 모델, 객체지향형 데이터 모델, 그리고 객체관계형 데이터 모델 등이 있다. 이러한 모델을 사용해서 만든 DBMS들의 대부분은 XML 문서의 구조와 데이터 정보를 해당 DBMS의 구조로 변환하고 저장하는 기능을 제공한다. 또한 그 반대로, DBMS에 저장된 데이터를 XML 문서의 구조와 데이터 정보로 나타내는 변환기능도 제공한다. 하지만 DBMS가 아무리 뛰어난 변환 기능을 발휘한다하더라도 변환하려는 XML 문서의 구조와 맞지 않는 데이터 모델의 구조를 가지고 있다면, 효율적인 데이터 관리를 위한 추가 비용이 들게 된다. 따라서 XML 문서를 DB에 저장하기 전에, 그 문서에 맞는 데이터 모델을 찾는 것이 중요하다. 본 논문에서는 저장 임계점을 사용하여, 일반적인 XML 문서를 어떤 데이터 모델에 저장하는 것이, 저장 측면에서 효율적인지 결정하는 방법을 제시한다. 여기서 저장할 데

이터 모델은 RDB와 ORDB로 한정한다.

3. 저장 임계점을 사용한 데이터 모델의 선택

3.1 저장 임계점의 정의

저장 임계점은 "XML 문서를 DB로 저장시 데이터 모델의 선택 기준이 되는 점"으로 정의한다.

동일한 XML 문서를 RDB와 ORDB로 저장할 경우, ORDB의 네스티드 테이블을 하나씩 증가해서 저장해 보면, RDB의 저장 속도가 ORDB보다 더 빨라지는 때가 생긴다. 이 때, RDB의 테이블 수에 대한 ORDB의 테이블 수의 비율로 저장 임계점을 계산한다. 여기서 RDB의 테이블 수와 ORDB의 테이블 수는, 동일한 XML 문서를 RDB로 저장할 경우에 필요로 하는 테이블 수와 ORDB로 저장할 경우에 필요로 하는 테이블 수를 나타낸다. ORDB의 테이블 수는 네스티드 테이블(Nested Table) 수가 포함되어 있다.

ORDB의 경우 하나의 테이블로 하나의 XML 문서 전체를 표현할 수 있지만, 2.1절에서 설명한 것처럼 XML 문서의 엘리먼트는 자식으로서 여러 개의 다른 이름의 자식 엘리먼트들이 존재할 수도 있고, 동일한 이름의 자식 엘리먼트들이 존재할 수도 있다. 여러 개의 다른 이름의 자식 엘리먼트들의 경우, 자식 엘리먼트를 테이블의 컬럼으로 생성하고, 컬럼의 타입을 객체 타입으로 선언하는데, 이 경우 하나의 테이블로 XML 문서 전체의 표현이 가능하다. 하지만, 동일한 이름의 자식 엘리먼트들이 존재하는 경우는 자식 엘리먼트를 테이블의 컬럼으로 생성하고, 컬럼의 타입을 네스티드 테이블 타입으로 선언한다. 이 경우는 DB 내부에서 그 컬럼을 위해 테이블을 따로 생성함을 의미한다.

저장 임계점은 XML 문서의 구조에 영향을 받는다. 저장 임계점에 영향을 주는 XML 문서의 구조는 동일한 이름을 가지는 엘리먼트의 개수이다. 따라서 저장 임계점을 계산할 때는 동일한 이름을 가지는 엘리먼트의 개수로 구분해서 나타낸다. 즉, 동일한 이름을 가지는 엘리먼트의 개수가 1개인 경우, 2개인 경우, 그리고 3개인 경우의 저장 임계점이 다르다. ORDB의 경우 동일한 이름을 가지는 엘리먼트의 데이터는 컬럼으로 생성되어, 하나의 네스티드 테이블에 저장된다. 이것은 테이블 대비(ORDB의 테이블 수 ÷ RDB의 테이블 수 × 100)와 밀접한 관련을 맺고 있는 저장 임계점에 커다란 영향을 준다. 동일한 이름을 가지는 엘리먼트의 개수가 2개이면 테이블의 레코드 수는 두 배가 되고, 3개이면 테이블의 레코드 수는 세 배가 된다. 일반적인 XML 문서를 어떤 데이터 모델에 저장할 것인가를 결정하는 저장 임계점은 동일한 이름을 가지는 엘리먼트의 개수에 커다란 영향을 받기 때문에, 이를 동일한 이름을 가지는 엘리먼트의 개수에 따라 구분해서 정한다. 동일한 이름을 가지는 엘리먼트의 개수가 세 개 이상인 경우에는 RDB와 ORDB의 특성 때문에, 항상 ORDB에서 빠른 속도를 보인다. 따라서 동일한 이름을 가지는 엘리먼트의 개수가 1개인 경우의 저장 임계점, 동일한 이름을 가지는 엘리먼트의 개수가 2개인 경우의 저장 임계점, 마지막으로 동일한 이름을 가지는 엘리먼트의 개수가 3개 이상인 경우의 저장 임계점으로 나누어 생각한다.

3.2 XML 문서에의 적용

저장 임계점을 XML 문서에 어떻게 적용할 수 있는지 살펴보겠다. 일반적인 XML 문서를 어떤 데이터 모델에 저장할 것인지를 결정하려면 먼저, 테이블 대비를 계산한다. 테이블 대비는 XML 문서의 구조를 살펴보고, RDB로 저장할 때 필요로 하는 테이블 수를 계산하고, ORDB로 저장할 때 필요로 하는 테이블 수를 계산해서 얻는다. 그리고 동일한 이름의 자식 엘리먼트가 몇 개인지 살펴본다. 이 테이블 대비를 앞으로 4장에서 동일한 이름의 자식 엘리먼트의 개수에 구분해서 구할 저장 임계점과 비교해서, 저장할 데이터 모델을 선택한다. 저장 임계점보다 크면, XML 문서를 RDB를 선택하고, 저장 임계점보다 작으면 ORDB를 선택하는 것이 저장시간 측면에서 빠르다.

4. 실험 및 결과

4.1 실험 환경

실험을 위해 사용한 환경은 다음과 같다.

⊙ 서버

- Intel Pentium 4, CPU 1.7GHz × 2, 1GB RAM
- Linux Kernel 2.4.13 version
- DBMS : Oracle9i Database Release 2 Enterprise/Standard Edition for Intel Linux

4.2 실험을 위해 사용한 XML 문서

실험을 위해 사용한 XML 문서는 그림 1의 3개의 문서이다.

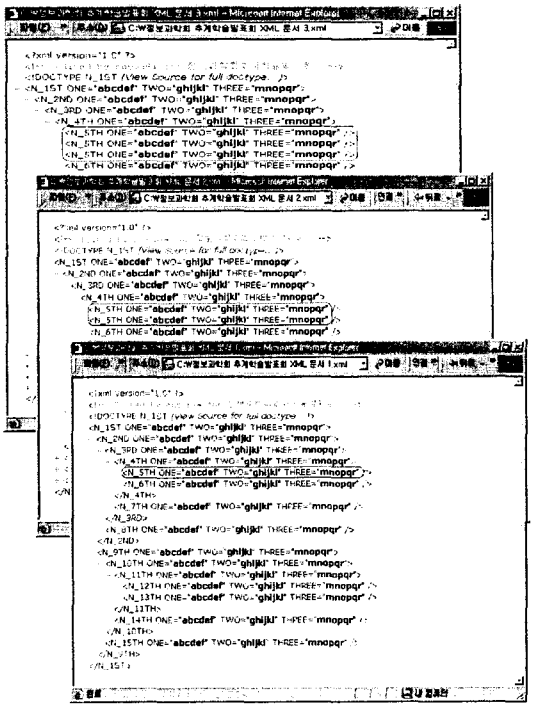

그림 1

그림 1의 첫 번째 문서는 엘리먼트의 자식으로서 여러 개의 다른 이름의 자식 엘리먼트들이 존재하고, 두 번째 문서는 엘리먼트의 자식으로서 여러 개의 다른 이름의 자식 엘리먼트들이 존재하면서 동시에 각각의 동일한 이름의 엘리먼트가 두 개씩 존재하며, 마지막으로 세 번째 문서는 엘리먼트의 자식으로서 여러 개의 다른 이름의 자식 엘리먼트들이 존재하면서 동시에 각각의 동일한 이름의 엘리먼트가 세 개씩 존재한다.

그림 1의 3개의 XML 문서를 도식화하면 그림 2와 같이 깊이 5인 계층 구조임을 알 수 있다.

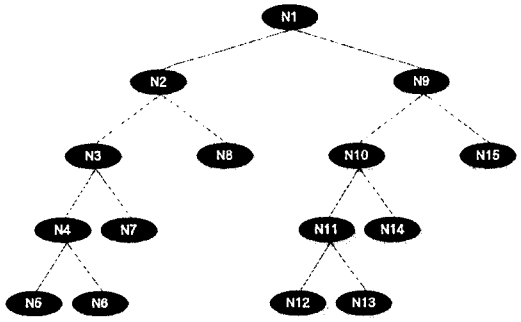

그림 2

4.3 저장 임계점의 계산

본 절에서는 그림 1의 XML 문서들을 RDB와 ORDB로 저장하여, 저장 임계점을 구한다. XML 문서의 DB로의 매핑 절차는 현재 가장 일반적으로 사용되는 방법들을 사용하였다[3,4]. 표 1은 그림 1의 XML 문서를 저장한 결과이다. 표 1에서 가로 항목들(N1, N2, ... , N15)은 문서의 구조에 따라 DB에 생성된 테이블 수를 가리킨다. N1은 하나의 테이블이 사용되었고, N15는 15개의 테이블이 사용되었다. 테이블 수에는 네스티드 테이블의 수가 포함되었다. 세로 항목들(ORDB1, RDB1, ... , ORDB3, RDB3)은 데이터 모델과 동일한 이름의 자식 엘리먼트들의 개수를 가리킨다. 결과 값은 만(萬) 개의 XML 문서를 저장할 때 걸리는 시간이고, 단위는 초(秒)이다.

표 1

데이터 모델	N1	N2	N3	N4	N5	N6	N7	N8	N9	N10	N11	N12	N13	N14	N15
ORDB1	3.7	12.1	16.9	25.3	32.2	38.5	45.2	51.1	57.8	65.4	69.5	78.4	84.8	89.2	98.1
RDB1	48.5	48.5	48.5	48.5	48.5	48.5	48.5	48.5	48.5	48.5	48.5	48.5	48.5	48.5	48.5
ORDB2		15.2	25.7	40.3	59.3	74.9	93.5	125.1	154.6	182.7	216.1	276.4	339.9	399.7	457.9
RDB2		72.3	96.8	103.9	125.5	143.9	178.5	191.7	236.9	250.4	282.8	310.7	332.3	388.1	532.9
ORDB3		18.1	31.9	58.9	86.9	120.9	150.5	225.7	314.5	393.3	481.8	724.9	945.1	1182	1416
RDB3		84.9	141.8	182.8	224.5	281.1	367.3	444.4	652.2	709.9	843.1	1188	1998	2283	2540

3.1절에서는 저장 임계점을 동일한 이름을 가지는 엘리먼트의 개수에 따라 구분한 후, RDB의 저장 속도가 ORDB보다 더 빨라지는 때를 찾아서 저장 임계점을 계산한다고 하였다. 표 1을 살펴보면, 동일한 이름을 가지는 엘리먼트의 개수가 하나인 경우 이를 RDB와 ORDB로 저장했을 때, RDB가 ORDB보다 더 빠르게 저장되는 위치는 테이블을 8개(N8)를 사용했을 때이다. 이 때의 저장 임계점을 계산하면, 저장 임계점은 53 (ORDB의 테이블 수 ÷ RDB의 테이블 수 × 100 = 8 ÷ 15 × 100) 이 된다. 마찬가지로, 동일한 이름을 가지는 엘리먼트의 개수가 2개인 경우의 저장 속도의 변화가 13개(N13)일 때 발생하므로, 저장 임계점은 87 (13 ÷ 15 × 100) 이 된다. 마지막으로 동일한 이름을 가지는 엘리먼트의 개수가 3개 이상인 경우에는 저장 임계점이 존재하지 않는다. 이렇게 구한 저장 임계점을 일반적인 XML 문서에 적용해서 저장할 데이터 모델을 선택할 수 있다.

4.4 저장 임계점의 적용 예

본 절에서는 저장 임계점을 일반적인 XML 문서에 어떻게 적용할 수 있는지 살펴보겠다. 예를 들어, 어떤 XML 문서를 분석해 보았더니, RDB로 저장할 경우 11개의 테이블을 필요로 하고, ORDB 저장할 경우 8개의 테이블을 필요로 한다. 또한, 동일한 엘리먼트의 개수가 1개씩 존재한다. 이 경우 XML 문서의 테이블 대비를 계산하면 73 (8 ÷ 11 × 100) 이 된다. 4.3절에서 구한 저장 임계점 53과 비교하면, 저장 임계점보다 크므로 (73 > 53), 데이터 모델로 RDB를 선택하는 것이 ORDB보다 저장시 소요되는 시간이 더 작다. 만약, 이 XML 문서가 동일한 이름의 엘리먼트의 개수가 2개라면, 저장 임계점 87보다 작으므로 (73 < 87), 데이터 모델로 ORDB를 선택하는 것이 저장 시간 측면에서 바람직하다. 마지막으로 동일한 이름의 엘리먼트의 개수가 3개 이상인 XML 문서라면, 테이블 대비와 상관없이 데이터 모델로 ORDB를 선택하는 바람직하다.

5. 결론

본 논문에서는 저장 임계점을 정의하였다. 저장 임계점은 일반적인 XML 문서를 RDB나 ORDB로 저장할 때 소요되는 시간의 변화가 일어나는 위치이다. 이러한 저장 임계점을 사용하여 일반적인 XML 문서를 어떤 데이터 모델에 저장할 것인가를 선택하는 방법을 제시하였다. 또한 실험을 통해서 저장 임계점을 계산하였다.

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] Oracle9i XML Database Developer's Guide - Oracle XMLDB, http://otn.oracle.com/documentation/oracle9i.html, Otc. 2002.

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

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