Labeling 방식에 따른 XML 데이터의 갱신 성능 분석

정민옥° 남동선 한정엽 박종현 강지훈
(주)한글과컴퓨터 XML 기술팀, 충남대학교 컴퓨터과학과
{mojung°, speeno, yeobi}@haansoft.com {jhpark, jhkang}@cs.cnu.ac.kr

Analysis on Update Performance of XML Data by the Labeling Method

Min-Ok Jung°, Dong-Sun Nam, Jung-Yeob Han, Jong-Hyen Park, Ji-Hoon Kang
XML Technology Team, HaanSoft, Dept. of Computer Science, Chungnam National University

요 약

XML is situating a standard for data exchange in the Web. Most applications use database to manage XML documents of high-capacity efficiently. Therefore, most applications create label that expresses structure information of XML data and stores with information of XML document. A number of labeling schemes have been designed to label the element nodes such that the relationships between nodes can be easily determined by comparing their labels. With the increased popularity of XML data on the web, finding a labeling scheme that is able to support order-sensitive queries in the presence of dynamic updates becomes urgent. XML documents that most applications use have many properties as their application. So, in the thesis, we present the most efficient updating methods dependent on properties of XML documents in practical application by choosing a representative labeling method and applying these properties. The result of our test is based on XML data management system, so it expect not only used directly in practical application, but a standard to select the most proper methods for environment of application to develop a new exclusive XML database or use XML.

1. 서 론

XML[1]은 인터넷 상에서 데이터 교환을 위한 표준으로 자리 잡고 있다. 대용량의 XML 문서를 효율적으로 관리하기 위해서 대부분의 응용에서는 데이터베이스를 이용한다. 그러나 구조적인 XML 문서를 평평한 구조의 데이터베이스에 저장하기 위해서는 XML 문서의 구조정보를 얼마나 효율적으로 관리하는가 하는 것이 저장과 검색의 효율성을 위하여 매우 중요하다. 이러한 이유에서 대부분의 응용에서는 나누어진 XML데이터의 구조정보를 표현하는 레이블[2, 3, 4](노드의 식별자)을 생성하여 XML 문서의 정보와 함께 저장한다. 이러한 레이블은 XQuery나 XPath와 같은 XML 데이터 질의어에 포함된 구조적 질의를 처리하기 위해서 반드시 필요할 뿐만 아니라, 차후 나누어 저장된 XML 문서를 재구성하기 위하여 반드시 필요하다. 현재 효율적으로 XML 문서를 저장하고 검색하기 위한 방법들은 많이 제안되고 있다. 그러나 아직까지 어떤 방법이 가장 효율적인 방법인지는 검증이 되지 않은 상태이고 이들 각각은 개별적인 방법으로 레이블을 정의하여 사용하고 있다.

XML을 사용하는 많은 응용에서는 실시간 또는 일정간격을 두고 문서의 갱신이 요구된다. 그러나, 데이터베이스에 저장된 XML 문서를 갱신하기 위해서는 레이블도 같이 갱신해야 하지만 현재까지의 대부분의 연구에서는 데이터의 효율적인 저장이나 검색에 초점을 맞추어 연구하였으므로 갱신의 문제는 소홀할 수밖에 없었기 때문에 데이터베이스에 저장된 XML 문서를 갱신하기 위한 방법은 지난 문서 전체를 삭제하고 새로운 문서를 다시 삽입하는 방법을 주로 사용하고 있다. 또한 현재 XML 문서의 갱신을 위한 표준은 없는 상태이므로 어떤 방법으로 문서를 갱신해야 하는지의 기준이 없는 실정이다. 그러나 많은 응용에서 문서의 갱신이 필요한 상태이므로 효율적인 문서의 갱신을 위한 표준은 곧 제안될 것으로 기대되며, 이를 처리하기 위한 방법 또한 반드시 필요하다.

변경된 문서의 일부분만을 갱신하기 위한 방법은 현재 국내외에서 몇몇 연구가 진행 중이다. 그러나 이들 방법 모두는 가장 일반적인 XML 문서의 갱신을 위한 방법을 제안하고 있다. 그러나 대부분의 응용에서 사용하는 XML 문서는 응용에 따라 많은 특성을 가지고 있다. 이러한 경우 어떤 갱신 방법이 XML 문서의 특성에 가장 적절한 방법인지를 확인하는 것은 효율적인 문서의 갱신을 위해서 반드시 필요하다. 또한 기존의 연구들은 연구에서 제안하고 있는 방법을 실제 응용에 적용하였을 경우 발생하는 문제점을 감안하지 않고 있다. 이미 언급한 것처럼 문서의 일부분을 갱신하기 위해서는 레이블의 갱신이 가능해야 하는 것은 물론이고, 전체 문서에서 변경된 부분의 위치를 검색하는 방법 역시 효율적인 문서의 갱신을 위해서는 매우 중요하다. 그러므로 본 논문에서는 이러한 점들을 감안하여 실제 응용에서 XML 문서의 특성에 따른 가장 효율적인 갱신 방법을 실험을 통해 확인하고자 한다.

본 논문의 결과는 XML 데이터 관리시스템을 기반으로 평가한 결과이므로 실제 응용에서 직접 사용이 가능하며 향후, 새로운 XML 전용 데이터베이스의 개발이나 XML을 사용하는 응용에서 응용의 환경에 가장 적절한 방법을 선택할 수 있는 기준이 될 것으로 기대된다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구에 대해 살펴 보고, 3장에서는 문서의 갱신시 고려해야 할 점에 대하여 설명하며 4장에서는 각 레이블링 방법들에 따른 갱신 성능을 비교 분석하고, 마지막으로 5장에서는 본 논문의 결론 및 향후 연구에 대하여 기술한다.

2. 관련 연구

관계형 데이터베이스 시스템에서의 XML 문서의 저장과 검색

에 관한 연구들은 이미 많이 제안 되어 있지만, 문서의 갱신에 관한 연구들은 소홀했던 것이 사실이다. 그러나 관계형 데이터베이스 시스템에서 문서 갱신의 중요성은 이미 여러 연구에서 밝힌바 있다.

eXcelon 에서는 한 문서당 하나의 노드를 갱신할 수 있는 인터페이스를 제공하고 있으며, 대부분 상업적인 관계형 데이터베이스 시스템에서는 문서의 갱신을 위해 제한적인 자료 구조를 제시하여, 갱신이 발생하면 데이터베이스 내의 내용을 재구성하는 방식을 택하고 있다. 하지만, 이러한 방법들은 정확한 문서의 갱신 내용을 반영하지 못하며, 일시적인 효과를 낼 수 밖에 없는 불완전한 방법들이다. 즉, 이는 문서 갱신의 중요성을 인식하지만 갱신 방법에 관한 연구의 미약함을 의미하는 것이다.

본 논문의 연구 이전에 Edge 방식, Binary 방식, Universal 방식 등 데이터의 원시 저장 형식을 비교하였던 연구와 Local order, Global order, Dewey order 등의 Prefix 방식의 비교하였던 연구들이 있었으나 이것들은 정적인 저장 스키마로 너무 오래 되어 효율적이지 못하거나, 같은 종류의 저장 방식들의 비교로는 범위가 한정되어 있기 때문에 새로운 연구가 필요하게 되었다.

대부분의 본 논문과 관련되어 있는 연구들은 XML 문서를 관계형 데이터베이스를 이용하여 저장하고 질의 하는 연구들이다. 하지만 이러한 연구들은 문서의 갱신 측면에서의 연구가 미약하거나 또는 갱신을 지원하지 않으므로 갱신 성능의 비교가 쉽지 않았다. 따라서 이 연구에서는 현대 XML 저장 시스템에서 많이 보여지고 있는 동적인 레이블링 스키마의 세가지 종류 중에서의 대표적인 방식 한가지씩을 뽑아내어 저장, 재구성, 갱신의 성능을 여러 가지 측면에서 비교 평가 하였다.

3. 갱신 성능 비교시 고려해야 할 점

XML 문서의 저장 방법은 문서의 갱신에 가장 큰 영향을 미치는 요인이다. 문서의 갱신에서 문서의 특정 위치를 검색하는 성능도 고려해야 할 요소 중에 하나이다. 문서의 부분 갱신이 수행될 때, 가장 먼저 일어나는 것이 문서가 갱신되어야 할 위치를 찾는 것이기 때문에 검색 성능에 따라 갱신 성능이 달라지기 때문이다. 문서의 갱신 성능도 문서의 저장 방식에 따라 그 성능 차이를 보이므로 저장 방법의 선택이 가장 중요한 요소가 된다.

XML 문서의 갱신 방법은 수정 범위에 따라 크게 두 가지로 분류된다. 첫 번째로 문서의 전체의 삭제 후 새로운 문서를 삽입하는 방법이며, 두 번째는 변경된 부분(변경된 서브트리)만을 갱신하는 방법이다. 문서의 갱신 위치에 따라 갱신의 성능이 달라 질 수도 있다. XML 문서를 DOM 트리 형태로 나타내었을 때, 문서의 갱신 위치가 말단 노드(텍스트, 에트리뷰트), 중간 노드(중간 서브 트리), 상위 노드(전체 트리)가 될 수 있다.

XML 문서는 트리 구조를 가지고 있기 때문에 여러 가지 구조 특징을 가지고 있다. 우선 문서의 깊이와 넓이를 가지고 있으며 문서 트리의 모양을 가지고 있다. 또한 서브 트리의 개수나 문서의 크기 또한 데이터의 특징이 될 수 있다. 이러한 특징은 문서 갱신 시에 상당한 영향을 끼칠 수 있으므로 각각에 대해 알아볼 필요가 있다.

이와 같이, 주로 입력되어지는 XML 데이터의 특징을 파악하는 것이 중요하며, XML 저장 시스템을 설계할 때 이를 고려하는 것이 중요한 일이다.

4. 각 레이블링 방법들에 따른 갱신 성능 비교

4.1 실험 환경

본 논문에서 사용된 운영 체제로는 Windows XP professional 을 사용하였으며, 사용 데이터베이스로는 Oracle(R) Enterprise Manager Version 9.2.0.1.0 Production 을 사용하였다. 실험에 사용된 시스템 사양은 CPU로Intel(R) Pentium(R) 4 CPU 2.8 GHz 를 사용하였으며 메모리는 1 GB RAM을 사용하였다. 그리고 프로그래밍 언어로는 J2SDK 1.4를 사용하였고 JDBC 드라이버를 이용하여 데이터베이스에 접근하도록 구현하였다. 또 한, XML DOM 파서로는 eXerces 2.3을 이용하여 구현하였다.

4.2 데이터 및 질의어의 특징

성능 측정을 위해 사용한 데이터로 3가지 문서를 선택하였다. 하나는 세익스피어의 소설을 XML 문서화 한 Shakespeare XML 문서이고 두 번째는 실제 방송에서 사용하는 TV-Anytime 메타데이터 문서 데이터를 사용하였으며, 세 번째는 전자 상거래에서 실제로 이용한 Commercial XML (cXML) 에서 상품 요청 문서인 Request 문서를 사용하였다. 각 문서의 특징은 [표 1] 에 나타나 있다. 3장에서 설명되었듯이 XML 문서의 특징은 문서 갱신시 영향을 미치기 때문에 여러 특성이 골고루 분포 되어 있는 3개의 문서를 선택하였으며, 각각의 문서의 특징에 나타나는 결과의 특징을 중요하게 살펴 보아야 한다.

이러한 세 문서를 각 5MB, 10MB, 15MB로 분류하여 크기를 변경해 가면서 실험을 진행하였으며, 이것들이 선형적으로 증가하는지 살펴 보았다.

	트리 형태	문서의 크기	서브 트리 수
Shakespeare	넓고 얕다	비교적 크다	비교적 많다
TV-Anytime	좁고 깊다	비교적 작다	비교적 많다
cXML	넓이와 높이가 균형적이다.	매우 작다	매우 적다

[표 1] 테스트 데이터의 특성

이 실험에서 사용되는 질의어의 종류는 다음과 같다. Q1 질의는 문서의 말단 노드를 지칭하며 Q2 는 문서 트리내의 말단이 아닌 비교적 하위 노드를 지칭하고, 마지막 Q3 는 문서의 루트와 가까운 상위 노드를 지칭한다. 이것들은 XQuery로 표현된 질의어이며 처리 과정 시 SQL로 변환하여 사용된다.

4.3 XML 문서 저장 방법

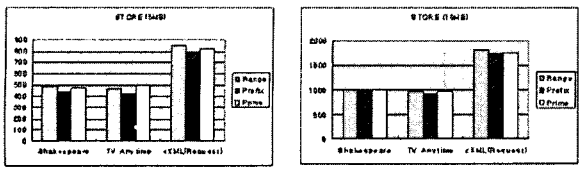
실험에서 사용된 문서의 DB 저장시 Labeling 방법은 실수 기반 방법[2], dewey order 방법[3], 솟수(Prime number)를 이용하는 방법[4] 이다.

4.4 실험 결과

4.4.1 저장

문서의 저장시 Range, Prefix 방식은 문서의 모양에는 큰 영향을 받지 않으며 Prime Number 방식은 문서의 깊이가 깊어질수록 많은 계산을 필요로 하기 때문에 문서의 높이에 많은 영향

을 미친다. 또한 세 방법 모두 문서의 크기에는 비례하여 증가한다.

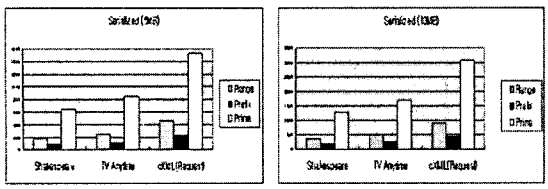


(가) STORE (5MB) (나) STORE (10MB)

[그림 1] 레이블링 방법에 따른 저장 성능 비교

4.4.2 재구성 (Serialized)

문서의 재 구성시 세 방법은 문서의 모양에 큰 영향을 받지 않으며 문서의 크기에는 비례하여 증가한다.

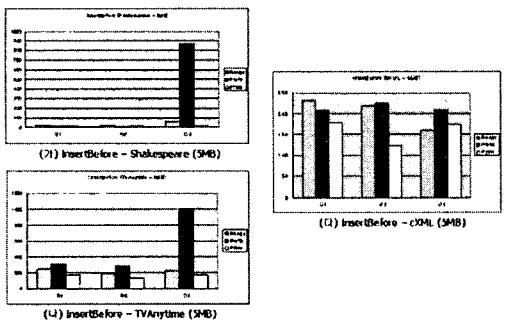


(가) Serialized (5MB) (나) Serialized (10MB)

[그림 2] 레이블링 방법에 따른 재구성 성능 비교

4.4.3 Insert

세가지 방법 모두 문서의 깊이가 깊은 문서일수록 좋지 못한 성능을 보이며, 문서의 모양에 상당히 민감한 결과를 가져온다. 또한, 크기가 작은 문서에는 영향을 받는 노드의 범위가 작기 때문에 세가지 방법이 큰 차이를 보이지 않았다. 문서의 양에는 비례적으로 증가하는 것으로 보인다.



(가) InsertBefore - Shakespeare (5MB)

(나) InsertBefore - TVAnytime (5MB)

(다) InsertBefore - cXML (5MB)

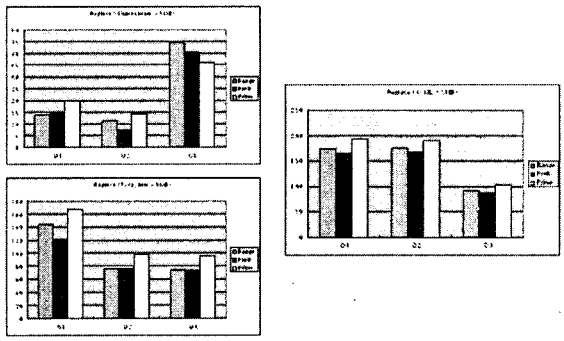
[그림 3] 레이블링 방법에 따른 **Insert** 성능 비교

4.4.4 Delete

삽입과 마찬가지로 세가지 방법 모두 문서의 깊이가 깊은 문서일수록 좋지 못한 성능을 보이며, 문서의 모양에 상당히 민감한 결과를 가져온다. 또한, 크기가 작은 문서에는 영향을 받는 노드의 범위가 작기 때문에 세가지 방법이 큰 차이를 보이지 않았다. 문서의 양에는 비례적으로 증가하는 것으로 보인다.

4.4.5 Replace

노드 교환시 Range, Prefix 방식은 문서의 모양에는 큰 영향을 받지 않으며 Prime Number 방식은 문서의 깊이가 깊어질수록 많은 계산을 필요로 하기 때문에 문서의 높이에 많은 영향을 미친다. 또한 세 방법 모두 문서의 크기에는 비례하여 증가한다.



[그림 4] 레이블링 방법에 따른 **Replace** 성능 비교

5. 결론

본 논문에서는 XML 문서의 갱신을 위하여 이미 제안된 레이블링 방법들을 분석하여 응용에 따른 효율적인 레이블링 방법을 실험을 통해 제안하였다. 또한, 이러한 레이블링 방법을 가반으로 응용에서 사용하는 XML 문서의 특성에 따라 어떠한 갱신 방법이 가장 적절한 방법인지를 확인하였다.

본 논문의 결과는 XML 데이터 관리시스템을 기반으로 평가한 결과이므로 실제 응용에서 직접 사용이 가능하며 향후, 새로운 XML 전용 데이터베이스의 개발이나 XML을 사용하는 응용에서 응용의 환경에 가장 적절한 방법을 선택할 수 있는 기준이 될 것으로 기대된다. 또한, 본 논문의 결과는 XML 문서의 갱신을 위한 새로운 연구의 기준이 될 것으로 사료된다.

향후 연구로는 각 레이블링 방법들의 단점으로 지적된 사항의 개선 방법들이 필요 하며, 본 논문의 결과를 실제 응용에 적용하여 다른 예외 사항을 확인해보는 것이 중요하다. 또한 XML 문서의 활용 효율성을 높이기 위하여 갱신, 검색, 저장, 재구성 등 부분적 혹은 전체적인 성능 개선에 대한 연구가 활발하게 이루어져야 할 것이다.

참고 문헌

[1] W3C, Extensible Markup Language (XML) Version 1.0, Recommendation, Feb. 1998. (http://www.w3.org/TR/1998/REC-xml-19980210)

[2] T. Amagasa, M. Yoshikawa & S. Uemura. "*QRS: A Robust Numbering Scheme for XML Document*", Proceedings of the 19th ICDE'03.

[3] Igor Tatarinov, Statis D. Viglas, Kevin Beyer & Chun Zhang. "*Storing and Querying Ordered XML Using a Relational Database System*" ACM SIGMOD'2002

[4] Xiaodong Wu, Mong Li Lee & Wynne Hsu, "A Prime Number Labeling Scheme ofr Dynamic Ordered XML Trees", Proceedings of the 20th ICDE'04.