한국컴퓨터종합학술대회 2005 논문집 Vol. 32, No. 1(B)

문서 단편화 기법을 이용한 XML 저장 관리 시스템 개발

정창후° 최윤수 서정현 진두석 김광영 주원균 이민호
한국과학기술정보연구원 정보시스템부
{chjeong°, armian, jerry, dsjin, kykim, joo, cokeman}@kisti.re.kr

Development of an XML Storage and Management System using Document Fragmentation Method

Chang-Hoo Jeong°, Yun-Soo Choi, Jeong-Hyeon Seo, Du-Seok Jin,
Kwang-Young Kim, Won-Kyun Joo, Min-Ho Lee
Korea Institute of Science and Technology Information

요 약

다양한 응용 분야에서 점차 증가하고 있는 XML 문서를 효과적으로 저장하고 검색하는 방법에 대한 연구가 많이 이루어지고 있다. XML 문서의 정보를 데이터베이스에 저장하는 방법으로 분할 저장 방식과 비분할 저장 방식이 있는데, 본 논문에서는 두 가지 방법의 중간자적인 형태를 취함으로써 문서의 구조적인 정보를 효과적으로 표현하고, 검색 결과 재구성 시에 발생하는 오버헤드를 제거할 수 있는 시스템에 대해서 설명하도록 한다. 복잡한 계층 구조의 XML 문서를 서비스의 목적에 따라서 중요하다고 판단되는 단편노드의 단위로 구분하고 단편노드 안에 존재하는 세부 엘리먼트 및 속성에 대해서 별칭을 사용하는 검색 필드를 미리 구성해 놓음으로써, XML 문서의 구조적 특성을 명확하게 알지 못하거나 질의어 작성에 익숙하지 못한 사용자에게 보다 사용하기 편리한 XML 문서 검색 서비스를 제공할 수 있다.

1. 서 론

웹 상에서의 구조적 데이터 표현 및 문서 교환의 새로운 표준으로 1998년 W3C에서 XML(eXtensible Markup Language)[1]이 채택된 후, 사용이 간편하고 재사용성 및 확장성이 뛰어나다는 장점을 기반으로 전자상거래, 전자책 등 많은 분야에서 활용되고 있다. 또한 XML 관련 소프트웨어나 도구, XML을 지원하기 위한 다른 표준들이 속속 발표됨에 따라 점차 웹 상의 모든 문서가 XML로 대체되거나 빠르게 변경되고 있다. 이러한 추세를 고려하면 앞으로도 XML 관련 문서의 양은 더욱 증가할 것이며 그 사용 범위도 더욱 확대될 것이기 때문에, XML 문서를 효과적으로 저장하고 관리하는 방법에 대한 연구가 필요하다. 이러한 XML 문서 저장 관리 시스템은 원본 XML 문서를 저장 시스템에 저장한 후에 다시 XML 문서로 변환했을 때, 원래의 XML 문서대로 복원되어야 하는 라운드 트리핑 조건을 만족시켜야 한다.

2. 관련 연구

XML 문서의 정보를 데이터베이스에 저장하는 방법으로 분할 저장 방식과 비분할(가상 분할) 저장 방식이 있다[2]. 분할 저장 방식은 XML 문서를 엘리먼트 단위로 나누어 저장하고, 문서 복원 시 구조정보를 참조하여 엘리먼트를 재조합하여 처리하는 방식이다. 이 방식은 문서의 편집과 관리가 쉽다는 장점이 있으나, 검색 결과를 재구성하는 과정이 복잡하고 시간이 많이 소요되는 단점이 있다. 비분할 저장 방식은 LOB(Large OBject) 형태의 필드에 XML 문서 전체를 저장하고, 각각의 엘리먼트에 대한 위치 정보를 추가적으로 저장하는 방식이다. 검색 결과를 재구성하는 과정이 필요없어서 문서 참조를 빨리 할 수 있지만, XML 문서의 일부분에 대한 변경이 발생하면 문서 내에서 영향을 받는 모든 엘리먼트들의 위치 정보도 수정해야 하므로 이에 따른 오버헤드가 발생할 수 있다. 또한 데이터베이스의 일관성 유지에 문제가 발생할 수 있고, 많은 사용자가 같은 문서에 동시에 접근하는데 어려움이 있다.

본 논문에서는 분할 저장 방식과 비분할 저장 방식의 중간자적인 형태를 취함으로써 문서의 구조적인 정보를 효과적으로 표현하고, 검색 결과 재구성 시에 발생하는 오버헤드를 제거할 수 있는 XML 문서 저장 및 관리 시스템에 대해서 설명하도록 한다. 복잡한 계층 구조의 XML 문서를 서비스의 목적에 따라서 중요하다고 판단되는 단편노드의 단위로 구분하고 단편노드 안에 존재하는 세부 엘리먼트 및 속성에 대해서 별칭(alias)을 사용하는 검색 필드를 미리 구성해 놓음으로써, XML 문서의 구조적 특성을 명확하게 알지 못하거나 질의어 작성에 익숙하지 못한 사용자에게 보다 사용하기 편리한 XML 문서 검색 서비스를 제공할 수 있다. 또한 관계형 데이터베이스에서 제대로 지원하지 못하는 집합 값(set-value)[3]을 어떻게 다룰 수 있는지도 함께 살펴보도록 한다.

3 XML 문서 저장 관리 시스템

3.1 문서 분할 및 저장

문서 분할은 복잡한 계층의 XML 문서를 의미 있는 엘리먼트를 중심으로 계층을 단순화시켜서 검색 및 계층적 결과 표현에 사용하도록 하는 작업으로, XML 문

서의 부모-자식 관계, 형제 관계 등의 계층 정보를 유지하면서 문서를 단편화 한다. 이때 계층 관계 및 병합 정보, 그리고 검색에 관련된 각종 정보를 정의하고 있는 변환 규칙을 이용하여 XML 문서를 구조적 특징을 포함한 단편화된 XML 문서 노드들(Document Fragments)로 변형시키는데, 이 각각을 단편노드라고 부르도록 하겠다. 또한 사용자가 의미 있게 생각하지 않아서, 단편노드로 분리되지 않고 남아있는 XML 문서의 나머지 부분은 문서 복원 시에 라운드 트리핑 조건을 만족시키기 위해서 가상 루트 단편노드(Level 0)로 지정하여 저장 시스템에 함께 저장한다.

분할된 각각의 XML 단편노드는 그림 1과 같은 구조 정보를 가지고 Berkeley DB에 저장된다.

구조 정보	의미	구조 정보	의미
SYS.RECID	단편노드 아이디	SYS.FIRSTCHILD	처음 자식 단편노드 아이디
SYS.MERGE	단편노드 병합 식별 값	SYS.ORDER	형제 단편노드 순서
SYS.TYPE	단편노드 타입	SYS.ROUTE	부모 단편노드에서의 분리된 위치
SYS.LEVEL	단편노드 레벨	DOCUMENT	레벨로 지정된 XML 문서의 단편노드 내용 (Document Fragment of XML)
SYS.PARENT	부모 단편노드 아이디		
SYS.PREVIOUS	이전 단편노드 아이디	*USER-DEFINED*	검색을 위한 별칭 정보
SYS.NEXT	다음 단편노드 아이디		

그림 1 단편노드의 구조 정보

문서 단편화 기법을 사용하면 원본 XML 문서가 가지고 있는 복잡한 엘리먼트 계층 구조를 의미 있는 엘리먼트를 중심으로 간략하게 재구성할 수 있다. 또한, 재구성된 트리 역시 원래의 XML 문서가 가지고 있는 노드 간의 부모-자식 관계 및 형제 관계를 그대로 수용할 수 있어서 원본 문서 구조 형태로의 복원도 가능하다.

그림 1에서 SYS.ROUTE는 원본 XML 문서로의 복원을 위해서 사용되는 정보로서 부모 단편노드에서의 현재 단편노드가 분리된 위치를 나타낸다. 표현은 유닉스 파일 시스템에서 디렉토리를 나타내는 구조와 비슷한 방법을 사용한다. 예를 들어, "/2/3/1"은 부모 단편노드의 가장 상위 엘리먼트로부터 2번째 자식 노드로 이동하고, 다시 3번째 자식 노드로 이동하고, 다시 1번째 자식 노드로 이동하라"라는 것을 의미한다.

데이터 중심 XML 문서의 경우에는 데이터의 물리적인 구조나 형제 엘리먼트의 순서가 중요한 고려 대상이 아니지만, 논문이나 연구보고서와 같은 문서 중심 XML 문서의 경우에는 상대적으로 문서 구조의 정형화가 낮고 내용들의 순서가 중요하다는 특성을 가지기 때문에, 문서 재구성 과정에서 SYS.ROUTE 정보를 사용하여 문서 분할 시의 정확한 위치로 복원되어야 한다.

3.2 문서 변경

XML 문서를 변경하기 위한 사용자 작업으로는 새로운 노드의 삽입, 기존 노드의 제거, 기존 노드의 수정이 있다. 이 각각의 작업에 대해서 단편노드 단위로 문서를 변경하는 방법과 단편노드의 범위를 벗어난 노드그룹 단위로 문서를 변경하는 방법으로 구분해서 살펴볼 필요가 있다.

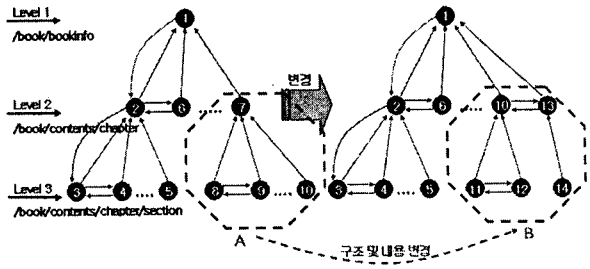

그림 2 XML 문서의 변경

먼저 단편노드 단위로 문서를 변경하는 작업에 대해서 살펴보도록 하겠다. 단편노드 삽입의 경우에는 관리기를 통하여 새로운 단편노드를 추가해주면 된다. 관리기를 통해서 단편노드를 생성하고, 이 단편노드가 삽입될 위치와 기준 노드를 지정하여 단편노드를 새롭게 추가한다. 단편노드 삭제의 경우에는 관리기를 통하여 제거될 단편노드를 지정해주면 된다. 관리기는 사용자가 지정한 단편노드의 아이디와 단편노드가 속한 테이블 아이디를 이용하여 단편노드를 제거한다. 단편노드 수정의 경우에는 단순히 단편노드로 표현되는 DOCUMENT의 내용만 수정하면 된다. 이와 같은 작업은 레코드 기반의 데이터 중심 XML 문서의 변경에서 유용하게 사용될 수 있다.

다음으로 단편노드의 범위를 벗어난 노드그룹 단위로 문서를 변경하는 작업에 대해서 살펴보도록 하겠다. 노드그룹 삽입이나 노드그룹 제거는 그룹의 가장 상위 단편노드에 대해서만 변경해주면 되기 때문에, 단편노드 단위로 이루어지는 방법과 동일하다. 그러나 노드그룹 변경의 경우에는 단순히 가장 상위 단편노드 하나만 변경하는 것이 아니라 그룹에 포함되어있는 모든 단편노드간의 계층 정보까지도 함께 변경해야 한다. 이와 같은 작업은 계층 관계가 중요한 문서 중심 XML 문서의 변경에서 유용하게 사용될 수 있다.

그림 2에서 보여지는 것과 같이 여러 개의 단편노드를 한꺼번에 수정하려고 하는 경우 XML 문서는 단편노드의 DOCUMENT 내용뿐만 아니라 단편노드 간의 계층 관계도 함께 변경이 일어나기 때문에 단편노드 단위의 수정이 아니라 노드그룹 단위의 수정이 일어나야 한다. 그림 2의 경우에 관리기를 통하여 노드그룹 A를 수정했기 때문에 새롭게 생성된 노드그룹 B를 기존에 수정하고자 했던 노드그룹의 가장 상위 단편노드 위치에 연결시켜 준다. 이 작업은 기존 노드그룹을 삭제하고 새로운 노드그룹을 삽입하는 작업으로 구현된다.

대용량 데이터베이스 상에서 단편노드 혹은 노드그룹을 삭제하거나 삽입할 경우 색인을 갱신하는 시간이 많이 걸릴 수 있기 때문에, 삭제는 삭제 플래그를 사용하고 삽입은 보조 테이블을 이용하여 빠르게 작업을 수행한다.

3.3 문서 복원

문서 복원은 검색 결과 생성을 위한 부분 문서 복원

과 원본 문서 재현을 위한 전체 문서 복원이 있다. 부분 문서 복원은 검색된 단편노드들이 변환 규칙에서 지정된 레벨에 맞게 계층화된 트리 구조로 구성될 때 발생하는데, 이때 단편노드의 DOCUMENT 내용은 이미 XML로 구성되어서 저장되었기 때문에 엘리먼트들을 재조합해주는 작업은 필요하지 않다. 따라서 검색 결과 생성에 대한 오버헤드를 획기적으로 제거할 수 있다. 그리고 각각의 검색된 단편노드를 보다가 XML 문서 전문 보기를 원할 경우에는 전체 문서 복원 기능을 이용한다. 단편화되어서 저장된 XML 문서 전문을 원래대로 복원하는 알고리즘은 다음과 같다.

```
String XML_Reproduce_Algorithm(DocumentFragment node) {
  // 부모 단편노드의 내용으로 초기화한다.
  String document = node.parent.document;
  // 형제 단편노드가 존재하지 않을 때까지 반복한다.
  while (true) {
    // 자식 단편노드가 존재하면 그것을 파라메터로 하여 재 호출한다.
    if (node.firstchild != null)
      node.document = XML_Reproduce_Algorithm (node.firstchild);
    // route를 이용하여 parent에서 node가 분리된 위치를 찾아 복원시킨다.
    document = mergeDocument(document, node.document, node.route);
    // 다음 단편노드가 존재하면 다음 단편노드로 이동한다.
    if (node.next != null) node = node.next;
    // 다음 단편노드가 존재하지 않으면 함수를 종료한다.
    else break;
  }
  // 현재까지 복원된 문서를 반환한다.
  return document;
}
```

4. 실험

실험은 XML로 작성된 한의약 데이터베이스를 이용하여 수행하였다. 실험에 사용된 데이터는 처방정보, 천연약물, 화학정보, 병증사전, 그리고 수치사전에 관련된 5개의 XML 문서 테이블과 약재형상, 분자구조식을 나타내는 2개의 이미지 문서 테이블로 구성되어 있다. 실험에 사용된 데이터베이스의 특징으로는 각각의 XML 테이블 간에 복잡한 연결 관계가 존재한다는 것이고, 검색 결과 생성 시에 XML 테이블 사이의 네비게이션을 지원해야 한다는 것이다.

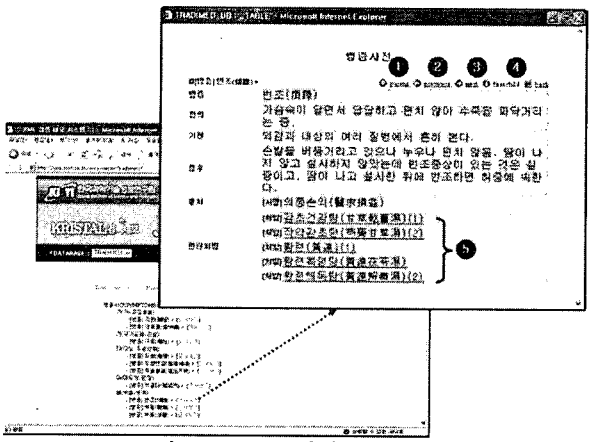

그림 3 XML 문서의 검색 결과

그림 3은 병증사전 테이블에서 "가슴&답답"으로 검색한 결과의 한 예이다. 그림 3에서 보여지는 것과 같이 XML 검색 결과의 단위인 단편노드는 각 노드를 기준으로 해서 자신을 포함하고 있는 부모 단편노드로의 이동(①), 자신과 형제 관계인 이전 단편노드와 다음 단편노드로의 이동(②,③), 자신의 자식들 중 첫 번째 단편노드로의 이동(④)을 지원한다. 또한 ⑤에서 보여지는 것과 같이 하나의 필드에 여러 개의 값을 가지고 있는 집합 값의 형태도 효과적으로 표현할 수 있다. 만일 분할 모델에서 ⑤와 같은 집합 값을 표현하려고 하면 저장 시에 집합 값을 모두 분리했다가 추출 시에 다시 재조합해야 하는 번거로움이 발생한다. 또한 비분할 모델에서도 ⑤와 같은 집합 값을 표현하려고 하면 집합 값 중의 하나가 추가, 삭제 및 수정이 발생하였을 경우 위치 값을 모두 재조정해줘야 하는 번거로움이 있다. 그러나 문서의 의미 있는 엘리먼트를 기준으로 문서를 분할하고, 그 분할된 단편노드들 간의 계층 관계를 유지하는 저장 구조에서는 과학 데이터나 카탈로그와 같은 레코드 기반의 데이터 중심 XML 문서를 서비스할 때 단순히 하나의 단편노드에 존재하는 DOCUMENT의 내용만 수정해주면 되기 때문에 훨씬 효율적인 시스템을 구성할 수 있다.

5. 결론 및 향후 연구

XML 문서에 대한 저장 및 관리 서비스를 제공하기 위해서 문서 저장, 문서 변경, 그리고 문서 복원에 관련된 내용에 대해서 살펴보았다. 분할 저장 방식과 비분할 저장 방식의 중간자적인 입장을 취함으로써 얻는 장점은 문서의 중요한 엘리먼트를 중심으로 구조 정보를 그대로 유지하면서도 검색 결과 재구성 시에 엘리먼트 재조합을 위한 오버헤드가 전혀 발생하지 않는다는 것이다. 이것은 실제적으로 XML 문서 검색 서비스를 실용화 시키는데 있어서 가장 중요한 두 가지 요소라고 생각된다.

향후 연구로는, 첫째로 노드그룹 단위의 문서 변경을 편리하게 수행할 수 있는 GUI(Graphic User Interface)의 개발이 필요하고, 둘째로 XML 문서 처리의 각 단계에 있어서 다른 시스템과의 정밀한 비교 분석 작업이 필요하다.

6. 참고 문헌

[1] Extensible Markup Language(XML) 1.0, "http://www.w3.org/XML"

[2] 김훈, 한상웅, 홍의경, "XML 문서 저장 시스템", 데이터베이스 연구회지, 16권 2호, 2000

[3] J. Shanmugasundaram, K. Tufte, G. He, C. Zhang, D. DeWitt, and J. Naughton, "Relational Database for Querying XML Document: Limitations and Opportunities", Proc. of the 25th VLDL Conf, pp.302-314, 1999