대용량 XML 문서의 효율적인 검색과 관리를 위한 SCOF 모델

Service-centric Object Fragmentation Model for Efficient Retrieval and Management of Huge XML Documents

정 창 후* 최 윤 수** 진 두 석*** 김 진 숙**** 윤화묵*****
Chang-Hoo Jeong Yun-Soo Choi Du-Seok Jin Jinsuk Kim Hwa-Mook Yoon

요 약

XML 문서가 증가하면서 XML 문서를 처리하는 방법론에 대한 많은 논의가 있어왔다. 본 논문에서는 두 가지 중요한 목적을 가지고 XML 정보 검색 및 관리 시스템을 개발하는데, 첫 번째는 질의에 적합한 내용을 쉽고 빠르게 검색해서 제공하는 것이고, 두 번째는 시스템의 부담을 최소화하면서 효율적이고 안정적인 관리 기능을 제공하는 것이다. 이렇게 실용적인 시스템을 개발하는 핵심 기술은 XML 문서를 어떻게 효과적으로 분할하여 구조적으로 서비스하는가에 달려 있다. 이러한 목적을 달성하기 위하여 본 논문에서는 SCOF(Service-centric Object Fragmentation) 모델을 제안한다. SCOF 모델은 XML 데이터베이스 관리자에 의해서 정의되는 변환 규칙(conversion rule)을 이용하여 문서를 분할하는 준분할(semi-decomposition) 저장 방식이다. SCOF 모델을 사용한 키워드 기반 검색은 전형적인 XML 질의 언어처럼 문서의 특정 엘리먼트나 속성 값을 이용하여 검색을 수행할 수 있다. 비록 이러한 접근법이 XML 문서 컬렉션에 대한 관리자의 지식을 필요로 한다고 하더라도, 개별 문서의 크기나 전체 문서의 양에 상관없이 검색과 관리를 효율적으로 수행할 수 있기 때문에 실용적인 시스템을 구축할 수 있다는 장점이 있다.

Abstract

Vast amount of XML documents raise interests in how they will be used and how far their usage can be expanded. This paper has two central goals: 1) easy and fast retrieval of XML documents or relevant elements; and 2) efficient and stable management of large-size XML documents. The keys to develop such a practical system are how to segment a large XML document to smaller fragments and how to store them. In order to achieve these goals, we designed SCOF(Service-centric Object Fragmentation) model, which is a semi-decomposition method based on conversion rules provided by XML database managers. Keyword-based search using SCOF model then retrieves the specific elements or attributes of XML documents, just as typical XML query language does. Even though this approach needs the wisdom of managers in XML document collection, SCOF model makes it efficient both retrieval and management of massive XML documents.

☞ keyword : SCOF, 변환 규칙(Conversion Rule), 준분할 저장 방식(Semi-decomposition), XML 키워드 검색(XML Keyword Retrieval)

1. 서 론

인터넷상에서의 구조적 데이터 표현 및 문서 교환의 표준으로 채택된 XML은 사용이 쉽고 재사용성 및 확장성이 뛰어나다는 장점을 기반으로 전자상거래, 전자 책, 학술자료 등 많은 분야에서

* 정 회 원 : 한국과학기술정보연구원 정보시스템개발팀
chjeong@kisti.re.kr
** 정 회 원 : 한국과학기술정보연구원 선임연구원
armian@kisti.re.kr
*** 정 회 원 : 한국과학기술정보연구원 선임연구원
dsjin@kisti.re.kr
**** 정 회 원 : 한국과학기술정보연구원 선임연구원
jinsuk@kisti.re.kr
***** 정 회 원 : 한국과학기술정보연구원 정보시스템개발팀장
hmyoon@kisti.re.kr
[2007/10/24 투고 - 2007/10/30 심사 - 2007/11/16 심사완료]

활용되고 있다. 또한 XML 관련 소프트웨어나 도구, XML을 지원하기 위한 다른 표준들이 속속 발표됨에 따라 점차 인터넷상의 많은 문서가 XML로 변경되고 있다. 이러한 추세를 고려하면 앞으로도 XML 관련 문서의 양은 더욱 증가할 것이며 그 사용 범위도 확대될 것이기 때문에, XML 문서를 효과적으로 이용하는 방법에 대한 연구가 필요하다. XML 문서를 이용한다는 말의 의미는 단순히 생성된 문서의 검색에 국한되지 않고, 관리를 효율적으로 수행하여 좀 더 유용한 문서를 생성하는 문서 편찬 기능을 포함한다.

본 논문의 구성은 2장에서 관련 연구에 대해서 기술하고, 3장에서는 SCOF 모델과 이 모델을 적용하여 개발한 XML 정보 검색 관리 시스템에 대해서 설명한다. 구현된 시스템의 성능을 평가하기 위해서 수행한 여러 가지 실험 결과는 4장에서 살펴보며, 결론 및 향후 연구에 대한 내용은 5장에서 논의한다.

2. 관련 연구

XML 문서가 증가하면서 XML 문서를 처리하는 방법론에 대한 많은 논의가 있어왔다. 특히 XML 문서를 저장하는 방법론은 관점에 따라서 다양하게 구분될 수 있는데[1, 2], 본 논문에서는 XML 문서를 데이터베이스에 저장하는 방식에 따라서 살펴보도록 한다. 이러한 관점에서 살펴보면 크게 분할 저장(decomposition) 방식과 가상 분할(virtual fragmentation) 저장 방식이 있다[3]. 분할 저장 방식은 XML 문서를 엘리먼트 단위로 나눈 후에 연관된 구조 정보와 함께 데이터베이스에 저장한다. 이 방법은 XML 문서의 일부분에 변경이 발생해도 문서 내의 다른 엘리먼트들에게 영향을 미치지 않으므로 오버헤드가 별로 발생하지 않는다. 반면에 XML 문서를 가져올 필요가 있을 때, 필요한 구조 정보를 참조하면서 엘리먼트를 재조합하기 때문에 문서를 재구성하는 과정이 복잡하고 시간이 많이 소요되는 단점이 있다. 이와 반대로 가상 분할 저장 방식은 XML 문서 전체를 LOB(Large OBject) 형태의 필드에 저장하고, 각각의 엘리먼트에 대한 위치 정보를 추가적으로 추출하여 저장한다. 이 방법은 XML 문서를 가져올 필요가 있을 때, 구조 정보를 참조할 필요가 없기 때문에 문서를 빠르게 재구성할 수 있다. 반면에 XML 문서의 일부분에 변경이 발생하면 문서 내에서 영향을 받는 모든 엘리먼트들의 위치 정보도 수정해야 하므로 많은 오버헤드를 발생시킬 수 있다.

최근 INEX[4]에서 XML 검색에 관련된 다양한 연구가 이루어지고 있는데, 본 논문에서는 검색 모델에 관련된 상세한 내용은 언급하지 않고 주로 대용량 XML 문서의 효과적인 처리에 중점을 두어 설명한다. 대용량의 XML 문서를 효과적으로 처리하기 위해서 SCOF 모델을 제안하는데, SCOF 모델은 변환 규칙(conversion rule)에 기반한 준분할(semi-decomposition) 저장 방식이다. 이 방식을 사용함으로써 질의에 적합한 내용을 쉽고 빠르게 검색해서 제공할 수 있고, 시스템의 부담을 최소화하면서 효율적이고 안정적인 관리 기능을 제공할 수 있다.

3. SCOF 모델을 이용한 XML 정보 검색 관리 시스템

3.1 XML 정보 검색의 특징

XML 정보 검색은 일반적으로 XML 문서의 계층적인 구조에 기반을 두고 있다. 명확하게 정의된 계층 구조가 문서에 존재하기 때문에 문서의 내용뿐만 아니라 구조에 대해서도 검색을 수행할 수 있다. 이렇게 세분화된 검색을 명시하기 위해서 XPath[5]나 XQuery[6]와 같은 XML 질의 표현식을 사용하는데, 이러한 질의는 사용자가 XML 문서상의 계층 구조(XML DTD 또는 XML 스키마)를 어느 정도 알고 있어야만 사용이 가능하다. 이와 같이 표현력이 풍부한 구조적 질의가 XML

문서의 검색 성능 향상에 기여할 수 있다는 부분은 잘 알려져 왔으나, 구조 검색을 표현하는 사용자의 미숙성 혹은 관련 있는 정보를 어떤 구조로 찾아야하는지를 모르는 데이터에 대한 비전문성으로 인해 실제적으로 그 효과를 보는 경우는 드물다. 더욱이 최근에는 질의에 포함되어 있는 구조적 힌트가 검색의 정확도를 향상시키지 못한다는 연구 결과도 발표되고 있다[7]. 그럼에도 불구하고 대부분의 사용자는 검색 결과를 브라우징할 때 XML 문서 전체보다는 그 문서를 구성하는 일부 중요한 영역만을 보기를 바라고 있다[8, 9]. 사용자들은 전체 문서보다 특정 엘리먼트에서 유용한 정보를 쉽게 찾을 수 있기 때문에 엘리먼트가 그들의 작업을 위해서 좀 더 유용하다고 생각한다[8, 10, 11]. 따라서 XML 정보 검색 시스템은 XML 문서로부터 질의에 적합한 엘리먼트를 검색하고 브라우징해줄 필요가 있다. 이와 같은 사용자의 복잡한 요구사항으로 인해 XML 정보 검색 시스템을 개발하는 작업은 많은 어려움에 직면하게 된다.

의미 있는 정보를 포함하면서 계층적으로 잘 구조화된 엘리먼트는 문서 컬렉션의 중요한 기능이지 사용자 질의의 기능이 아니다[7]. 따라서 관리자가 XML 문서의 의미 있는 내용이 쉽게 검색될 수 있도록 사용자 친화적인 시스템을 구축하는 것이 복잡한 질의 언어에 대한 적절한 대안일 수 있다. 그래서 키워드를 이용하여 XML 문서를 구조적으로 검색하는 다양한 연구가 이루어지고 있다[6, 12, 13, 14, 15].

XML 정보 검색에 관한 대부분의 연구에서 구조적 정보가 엘리먼트로 표현되지만, 엘리먼트의 속성 값을 이용하여 문서 구조를 기술하는 경우도 있기 때문에 XML 정보 검색 시스템은 이러한 속성도 함께 고려해줘야 한다[16].

본 논문에서는 XML 문서 컬렉션 관리자가 전처리 과정으로 서비스에 적합한 구조를 미리 정의해 놓고, XML 문서의 계층 관계와 해당 내용을 검색 필드로 지정해서 키워드 검색을 가능하도록 하는 SCOF 모델을 제안한다. 대부분의 정보 서비스 시스템이 주어진 XML 문서 컬렉션 내에서 구조가 고정되어 있기 때문에 이러한 방법을 쉽게 적용할 수 있다. 그 결과 최종 사용자에게 복잡한 XML 질의 언어 없이 키워드 방식의 검색 서비스를 제공할 수 있을 뿐만 아니라, XML 데이터베이스 관리자에 의해서 의미 있게 재구성된 문서의 구조적 검색 결과도 함께 제공할 수 있다.

3.2 SCOF(Service-centric Object Fragmentation) 모델 정의

XML 문서를 효과적으로 검색하기 위한 보편화된 모델이나 데이터 구조에 대한 의견일치가 아직까지는 명확하게 이루어지고 있지 않다[17]. 또한 고전적인 정보 검색과 달리, XML 검색은 명확하게 문서나 색인 단위를 결정하기가 쉽지 않다. 구조화된 문서 검색을 제공하기 위해서 시스템은 질의에 부합하는 가장 적합한 문서의 부분을 검색해야 하는데, 이러한 종류의 시스템을 알고리즘적으로 구현하기에는 많은 어려움이 있다. 그래서 본 논문에서는 XML 문서를 단편 노드로 분할하여 처리하는 SCOF 모델을 제안한다.

SCOF 모델은 XML 문서의 부모-자식 관계, 형제 관계 등의 계층 정보를 유지하면서 의미 있는 엘리먼트를 중심으로 여러 개의 중복된 영역이 없는 단편 노드(document fragment)로 분할하는 준분할 저장 방식이다. 이때 XML 문서의 단편 노드는 계층적으로 구조화된 트리 형태를 가지는데 이것을 FOT(Fragment Object Tree)라고 부른다. 분할 저장 방식이 모든 엘리먼트 노드의 구조 정보를 저장하는 반면에 준분할 저장 방식은 단편 노드 간의 구조 정보만을 저장한다. 그리고 단편 노드 내에 존재하는 엘리먼트 노드들은 가상 분할 저장 방식처럼 있는 그대로 저장한다. 하지만 위치 정보를 따로 추출하지는 않고 검색에 사용될 엘리먼트나 속성에 대해서 검색 필드를 미리 지정하여 향후에 키워드 검색에 이용될 수 있도록

한다.

SCOF 모델에서 XML 문서는 변환 규칙을 통하여 단편 노드로 분할된다. 문서의 내용과 형식을 잘 알고 있는 관리자가 변환 규칙을 이용하여 XML 문서의 효율적인 서비스 구조를 기술함으로써, 최종 사용자는 해당 문서에 가장 적합한 구조로 XML 문서의 검색 및 관리 서비스를 이용할 수 있다.

SCOF 모델을 도식화하면 다음과 같이 표현된다.

$f(D, R) = \{(F_i, S_i, I_i) \mid 1 <= i <= n\}$

f : XML 문서를 단편 노드로 분할하는 함수

D : XML 문서

R : 관리자에 의해서 정의된 변환 규칙

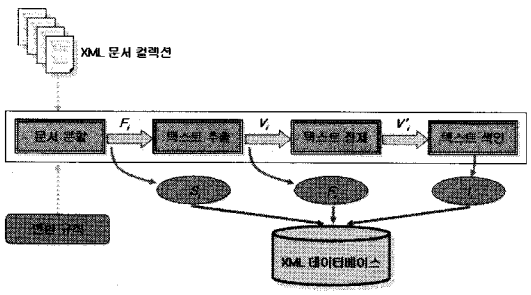
n : $f(D, R)$에 의해서 분할된 단편 노드의 개수

F_i : i번째 단편 노드

S_i : F_i의 구조 정보(주변 단편 노드와의 관계 설정을 위한 정보)

I_i : F_i의 검색을 위한 색인 정보

SCOF 모델을 이용하여 XML 문서를 처리하는 과정을 살펴보면 그림 1과 같다.

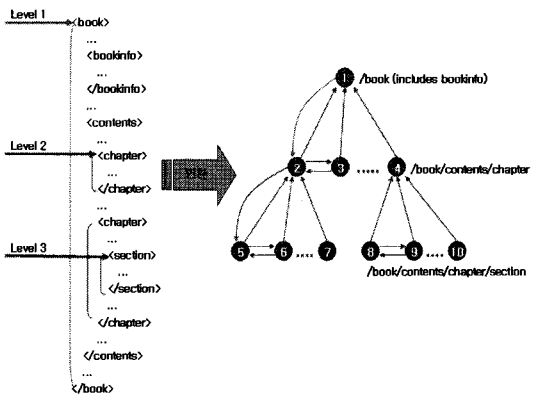


(그림 1) XML 문서를 데이터베이스에 저장하는 과정

그림 1에서 보이는 것과 같이 XML 문서를 서비스하기 위해서는 XML 문서와 해당 문서를 분할하기 위한 변환 규칙이 필요하다. 서비스할 XML 문서에 대한 변환 규칙이 정의가 되면 XML 문서를 분할하여 단편 노드(F_i)와 구조 정보(S_i)를 생성한다. 다음 작업으로 단편 노드로부터 검색에 사용할 엘리먼트나 속성 값(V_i)을 추출한다. 추출된 값은 정규 표현식과 같은 문자열 처리 기술을 이용하여 사용자가 원하는 포맷(V'_i)으로 정제가 된다. 정제된 데이터는 관리자가 지정한 색인 방식에 따라서 데이터 특성에 맞는 색인 정보(I_i)를 생성하게 된다. 이때 색인 정보를 생성하는 오버헤드가 발생할 수 있으나 대용량 문서의 빠른 검색을 위해서는 필수불가결한 작업이다. 최종적으로 단편 노드(F_i)와 구조 정보(S_i), 색인 정보(I_i)는 데이터베이스에 함께 저장되어 검색 및 관리 서비스에 사용된다.

일반적으로 사이즈가 큰 XML 문서들을 다루는 데이터베이스는 검색과 관리 양 측면에서 많은 어려움을 겪게 된다. 그러나 SCOF 모델에서는 XML 문서가 데이터베이스 관리자에 의해서 작성된 변환 규칙을 이용하여 크기가 작은 단편 노드들로 분할되기 때문에, XML 문서의 크기에 상관없이 효과적으로 검색 및 관리 서비스를 제공할 수 있다.

SCOF 모델에서 XML 문서는 단편 노드의 구조 정보를 이용하여 그 자체보다 처리하기 쉬운 FOT로 재구성된다. FOT는 단편 노드의 순회나 관리와 같은 다양한 기능들을 제공하는데, 이러한 기능들은 단편 노드를 조작하는 것을 쉽게 만든다. 이러한 측면에서 FOT는 비록 모든 기능들이 완전히 일치하지는 않더라도 구조적으로 DOM 트리와 유사하다.



(그림 2) XML 문서와 FOT

그림 2에서 XML 문서의 book, chapter, 그리고 section이 단편 노드로 지정되었다. 이것들은 서로 간의 데이터의 중복을 허용하지 않는다. 그림 2를 통해 알 수 있듯이 SCOF 모델은 원래 XML 문서의 계층 관계를 있는 그대로 유지하면서 단편 노드로 분할하기 때문에 새롭게 생성된 FOT도 원래 문서의 구조적인 형태를 그대로 가지고 있다. FOT는 의미 있는 엘리먼트를 중심으로 구성된 이해하기 쉽고 직관적인 구조를 가지는데, 트리의 노드(node)들은 단편 노드(F_i)를 나타내고 트리의 간선(edge)들은 구조 정보(S_i)를 나타낸다.

3.3 SCOF 모델을 이용한 XML 문서 관리의 특징

SCOF 모델을 이용한 문서 관리의 특징은 단편 노드의 삽입, 수정, 삭제뿐만 아니라 단편 노드의 이동, 병합, 재생성이 가능하다는 것이다. 단편 노드의 삽입은 FOT의 특정 위치에 단편 노드 혹은 노드 그룹을 추가하는 기능을 말한다. 단편 노드의 삭제는 FOT의 특정 위치에 있는 단편 노드 혹은 노드 그룹을 삭제하는 기능을 말한다. 단편 노드의 수정은 FOT에 있는 특정 단편 노드의 내용을 수정하는 기능을 말한다. 단편 노드의 이동은 FOT의 특정 위치에 있는 단편 노드 혹은 노드 그룹을 FOT 내의 다른 위치로 이동하는 기능을 말한다. 이때 원본 문서의 전체 구조를 위배하지 않는 경우에만 이동이 허용된다. 단편 노드의 병합은 하나의 FOT에 또 다른 FOT를 병합하는 기능을 말한다. 이것은 다른 말로 하나의 XML 문서에 또 다른 XML 문서를 결합시켜서 서비스할 수 있다는 것을 의미한다. 예를 들어, 조선왕조실록[18]과 같은 고전 문서 전산화 작업에서 한 번에 모든 시대의 내용을 전산화하여 서비스할 수도 있지만, 시간적 혹은 경제적 이유로 인하여 단계적으로 전산화를 수행한다. 이런 경우에 각 단계마다 해당 시기의 내용을 담고 있는 XML 문서가 새롭게 생성되지만 각각의 XML 문서가 서로 독립적인 문서는 아니다. 그렇기 때문에 각각의 XML 문서를 서로 연관시켜줄 방법이 필요한데, SCOF 모델에서는 병합을 이용하여 이 작업을 수행한다. 병합 시에는 병합 키를 사용하는데, 병합 키에 따라서 FOT의 다양한 레벨에서 통합 작업이 이루어질 수 있다. 노드의 재생성은 FOT를 XML 문서로 재생성하는 기능을 말한다.

단편 노드의 삽입, 수정, 삭제와 같이 문서의 내용을 변경하는 관리 작업이 발생하면 실제 문서의 내용을 보존하고 있는 단편 노드(F_i)뿐만 아니라 단편 노드를 FOT에 유지하도록 해주는 구조 정보(S_i), 그리고 단편 노드를 검색하도록 해주는 색인 정보(I_i)도 동시에 변경이 일어난다. 관리 작업으로 인해 영향을 받는 SCOF 모델의 구성 요소는 표 1과 같다.

(표 1) 관리 작업으로 인해 영향을 받는 SCOF 모델의 구성 요소(O: 영향 받음, X: 영향 받지 않음)

	Fi	Si	Ii	재귀 적용
삽입	O	O	O	O
수정	O	X	O	X
삭제	O	O	O	O
이동	X	O	X	O
병합	O	O	O	O
재생성	X	X	X	O

표 1에서 "재귀 적용"은 단편 노드에 관리 작업이 발생하였을 경우에 자동으로 자식 노드까지 처리해주는 것을 의미한다. 즉, 하나의 단편 노드뿐만 아니라 그 단편 노드가 가지고 있는 자식 단편 노드들을 포함한 노드 그룹을 한 번에 처리할 수 있는지의 여부를 나타낸다.

3.4 문서 변환 규칙

변환 규칙은 XML 문서를 단편 노드로 분할하는 중요한 역할을 수행하는데, 크게 3부분으로 구성되어 있다. 첫 번째는 XML 문서를 단편 노드

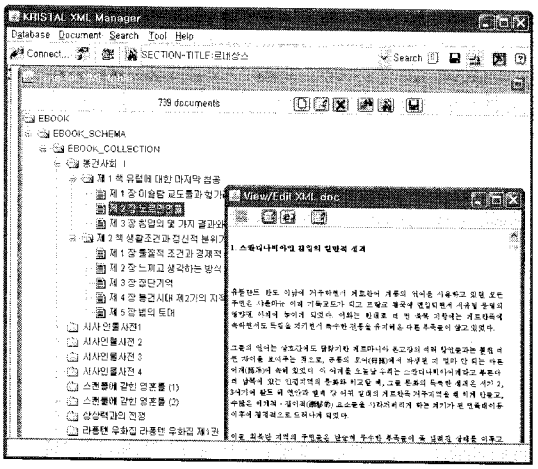
로 분할하는 방법을 정의하는 부분이다. 관리자가 원래 XML 문서의 복잡한 구조상에서 중요한 엘리먼트 및 속성을 중심으로 XML 문서의 구조를 새롭게 재구성하는 것이다. 두 번째는 기존에 존재하는 FOT로의 병합을 위하여 현재 생성되는 FOT의 병합 키를 생성하는 부분이다. 여기서 생성된 병합 키를 이용하여 기존 FOT의 어떤 레벨에 통합 작업이 이루어질 것인가를 결정할 수 있다. 만일 병합 키가 존재하지 않으면 병합 작업은 무시되고 새로운 FOT가 데이터베이스에 추가된다. 세 번째는 FOT의 각 단편 노드들을 검색하기 위한 색인 정보를 생성하는 부분이다. 데이터베이스 관리자가 XPath 표현식을 이용하여 각각의 중요한 엘리먼트나 속성을 미리 지정하고, 시스템에서 제공하고 있는 다양한 색인 방식 중에서 데이터 특성에 맞는 적절한 색인 방식을 적용하여 효율적인 검색을 가능하도록 한다. XPath 표현식으로 지정된 검색 필드는 사용자에게 친숙한 새로운 이름으로 대체되고, 최종 사용자는 이 이름을 사용하여 검색을 하기 때문에 구조 검색에 익숙하지 않은 사용자에게도 쉽게 서비스를 제공할 수 있다.

여러 개의 DTD나 스키마를 가지고 있는 복합 문서 컬렉션의 경우에는 각각의 경우를 포괄할 수 있는 변환 규칙을 작성해야 하기 때문에 작업이 다소 복잡해질 수 있다. 따라서 XML 문서의 구조가 다른 경우에는 새롭게 변환 규칙을 작성하여 서비스하는 것이 효과적이다.

변환 규칙은 XML 포맷으로 작성되는데, DTD는 다음과 같다.

```
<?xml version="1.0" encoding="EUC-KR"?>

<!-- 레벨, 병합, 색인에 관련된 변환 규칙을 정의 -->
<!ELEMENT Rule (LevelInfo+, PatternGroupList)>
<!ATTLIST Rule
    nodeRelation    (yes | no) #IMPLIED
    nodeInclusion   (yes | no) #IMPLIED
    normalization   (yes | no) #IMPLIED>

<!-- 레벨 정보를 정의하여 계층 관계를 새롭게 생성 -->
<!ELEMENT LevelInfo (MergeSectionList?, IndexSectionList?)>
<!ATTLIST LevelInfo
    no              CDATA   #REQUIRED
    path            CDATA   #REQUIRED
    constraint      CDATA   #IMPLIED
    nodeDeletion    (yes | no) #IMPLIED>

<!-- 병합 정보를 정의하여 Fragment Object Tree를 서로 통합 -->
<!ELEMENT MergeSectionList (MergeSection+)>
<!ATTLIST MergeSectionList
    delimiter   CDATA   #IMPLIED>
<!ELEMENT MergeSection EMPTY>
<!ATTLIST MergeSection
    path            CDATA   #IMPLIED
    attr            CDATA   #IMPLIED
    type            (SELF-TEXT | SINGLE-TEXT | MULTI-TEXT |
ALL | ALL-SE) #IMPLIED
    delimiter   CDATA   #IMPLIED
    length      CDATA   #IMPLIED
    textDelimiter   CDATA   #IMPLIED
    trim            (yes | no) #IMPLIED
    newlineDeletion (yes | no) #IMPLIED
    patternGroupName    CDATA   #IMPLIED>

<!-- 색인 정보를 정의하여 Document Fragment를 검색 -->
<!ELEMENT IndexSectionList (IndexSection+)>
<!ELEMENT IndexSection EMPTY>
<!ATTLIST IndexSection
    name            CDATA   #REQUIRED
    path            CDATA   #IMPLIED
    attr            CDATA   #IMPLIED
    type            (SELF-TEXT | SINGLE-TEXT | MULTI-TEXT |
ALL | ALL-SE) #IMPLIED
    delimiter   CDATA   #IMPLIED
    textDelimiter   CDATA   #IMPLIED
    groupDelimiter  CDATA   #IMPLIED
    trim            (yes | no) #IMPLIED
    newlineDeletion (yes | no) #IMPLIED
    patternGroupName    CDATA   #IMPLIED>

<!-- 정규 표현식 패턴을 정의하여 텍스트를 정제 -->
<!ELEMENT PatternGroupList (PatternGroup+)>
<!ELEMENT PatternGroup (Pattern+)>
<!ATTLIST PatternGroup
    name            CDATA   #REQUIRED
    sourceReuse (yes | no) #IMPLIED
    defaultValue        CDATA   #IMPLIED>
<!ELEMENT Pattern (#PCDATA)>
<!ATTLIST Pattern
    extractFormat   CDATA   #REQUIRED>
```

3.5 검색 결과의 구조적 표현



(그림 3) 간략하게 계층화된 XML 문서

그림 3은 관리기를 이용하여 XML로 작성된 전자 책 컬렉션의 데이터를 브라우징한 것이다. 전자 책 문서를 책(메타데이터), 장, 절로 분할하여 서비스를 구성하였는데, 각각의 분할된 단편 노드들은 원본 XML 문서가 가지고 있는 스타일 시트를 이용하여 사용자에게 보기 좋은 형태로 표시된다. XML 문서를 검색하였을 경우에 검색된 단편 노드뿐만 아니라 해당 단편 노드의 조상 정보를 함께 보여주기 때문에 문서의 전체적인 구조도 쉽게 파악할 수 있다. 또한, SCOF 모델이 현재 검색된 단편 노드의 부모, 형제, 자식을 순회할 수 있는 기능을 제공하기 때문에, 검색된 내용의 주변 문맥 정보도 쉽게 파악할 수 있다.

그림 3에서 보이는 데이터베이스를 생성하기 위한 변환 규칙은 다음과 같다.

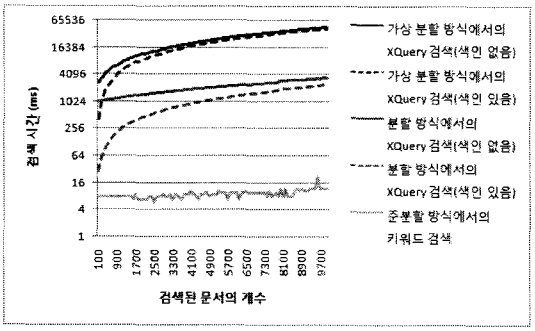
```
<?xml version='1.0' encoding='EUC-KR'?>
<!DOCTYPE Rule SYSTEM "rule.dtd">
<Rule>
    <LevelInfo no="1" path="/book">
        <IndexSectionList>
            <IndexSection name="BOOK_TITLE"
path="bookinfo/cover/title" type="MULTI-TEXT"/>
            <IndexSection name="AUTHOR"
path="bookinfo/cover/author[@role='writer']"/>
        </IndexSectionList>
    </LevelInfo>
    <LevelInfo no="2" path="/book/contents/chapter">
        <IndexSectionList>
            <IndexSection name="CHAPTER_TITLE" path="chapter-title"
type="MULTI-TEXT"/>
        </IndexSectionList>
    </LevelInfo>
    <LevelInfo no="3" path="/book/contents/chapter/section">
        <IndexSectionList>
            <IndexSection name="SECTION_TITLE" path="section-title"
type="MULTI-TEXT"/>
        </IndexSectionList>
    </LevelInfo>
</Rule>
```

인터넷에 공개되어 있는 수많은 XML 문서에는 엘리먼트 태그나 속성과 같은 마크업(markup)이 문서의 실제 내용과 비교해서 매우 많다[19]. 이러한 상황에서 사용자가 어떤 엘리먼트와 속성이 검색을 위해서 유용한 것인가를 알아내는 것은 거의 불가능하다. 그러므로 검색 인터페이스의 단순함은 일반 사용자들을 위해서 매우 중요하다. SCOF 모델을 이용하면 사용자들은 키워드 기반 검색 인터페이스를 사용하면서도 검색 결과들을 계층적으로 살펴볼 수 있기 때문에 컬렉션에 있는 문서들의 전체 구조를 쉽게 파악할 수 있다.

4. 실험 및 고찰

본 논문에서는 SCOF 모델을 사용한 XML 정보 검색 및 관리 시스템의 성능을 평가하기 위해서 문서 검색 시간, 문서 입력 시간, 그리고 문서 수정 시간을 측정해보았다. 이 실험에서 우리는 4GB의 메인 메모리를 가지는 듀얼 펜티엄 재온 2.80GHz 장비를 사용하였다. 또한 저장 매체로는 RAID-5 스카시 스토리지를 사용하였다. 그리고 테스트 컬렉션으로 XML 포맷으로 작성된 약 700권의 전자 책을 사용하였는데, 대용량 데이터의 테스트를 위해서 실험에서는 일부 문서를 중복해서 사용하였다. 전자 책 컬렉션의 전체 소스 사이즈는 220Mbytes이고, 이 컬렉션에는 1,709,004개의

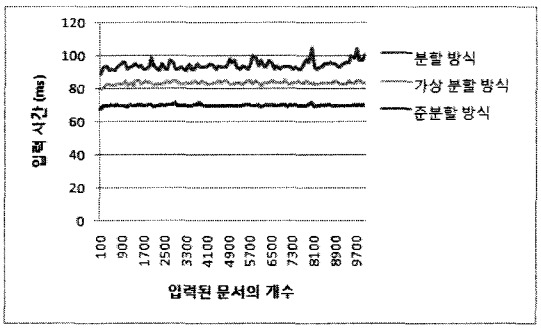
엘리먼트 노드가 존재하고 책, 장, 절을 구성하는 9,180개의 단편 노드가 존재한다. 개별 XML 문서의 사이즈는 46Kbytes로부터 1.2Mbytes까지 다양하게 존재하는데, 문서의 평균 사이즈는 300Kbytes이다. 그리고 단편 노드의 평균 개수가 12개인 반면에 엘리먼트 노드의 평균 개수는 2,334개이다. 이것은 문서의 내용과는 별 상관없이 단지 포맷팅을 위한 태그가 무수히 많이 존재한다는 것을 의미한다.



(그림 4) 검색 시간 비교

그림 4는 XQuery 기반 검색과 SCOF 모델을 사용한 키워드 기반 검색의 검색 시간을 보여준다. 순수한 검색 시간만을 나타내기 위해서 검색 결과를 가져오는 시간은 제외하였다. 준분할 저장 방식을 사용한 키워드 검색이 분할 저장 방식을 사용한 XQuery 검색보다 평균적으로 100배 정도 빠르고, 가상 분할 저장 방식을 사용한 XQuery 검색보다 평균적으로 2000배 정도 빠르다. 따라서 준분할 저장 방식을 사용한 키워드 검색의 성능이 훨씬 우수함을 알 수 있다. 더욱이 SCOF 기반의 키워드 검색의 성능은 검색 결과 문서의 수가 증가하더라도 크게 영향을 받지 않는 반면에, XQuery를 이용한 검색의 성능은 검색 결과 문서의 수가 증가함에 따라서 급격하게 저하되는 것을 볼 수 있다. 이러한 측면에서 SCOF 접근법이 대용량의 XML 문서를 처리하는데 있어서 훨씬 효율적이라는 것을 알 수 있다.

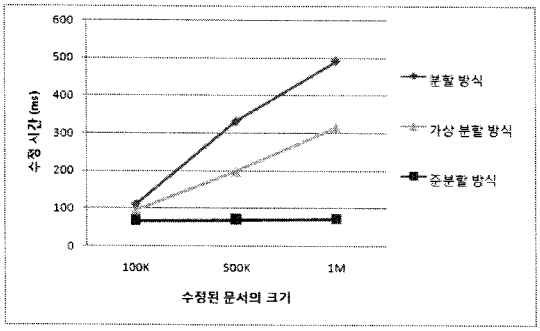
그리고 그림 4의 XQuery 기반 검색 중에서 점선으로 표현된 곡선과 실선으로 표현된 곡선은 색인을 구성한 것과 그렇지 않은 것을 나타낸다. 일반 XML 전용 검색 엔진에서는 자주 사용되는 질의 패턴에 대해서 색인을 미리 구성해 놓음으로써 시스템의 성능을 향상시킨다. 다시 말해서 가장 자주 검색되는 엘리먼트나 속성에 대해서 색인 방식을 미리 설정해놓아야 한다. 그렇지 않으면 문서의 양이 많아질수록 검색 성능이 급격하게 떨어지기 때문이다. 따라서 대용량 컬렉션에서 색인 전략을 미리 구성하지 않은 XQuery 검색은 실효성이 전혀 없기 때문에, XQuery를 사용할 경우에는 반드시 색인 전략을 미리 구성해야 한다. 이러한 측면에서 살펴볼 때, 엘리먼트나 속성에 대해서 검색 필드를 미리 구성하고 데이터 특성에 적합한 색인 방식을 지정하는 SCOF 모델의 검색 방법은 시스템의 성능을 최적화하기 위해서 필요한 작업임을 알 수 있다.



(그림 5) 입력 시간 비교

그림 5는 각 방법의 문서 입력 시간을 보여준다. 준분할 저장 방식의 문서 입력 시간이 분할 저장 방식이나 가상 분할 저장 방식보다 10~20ms 정도 적게 걸리는 것을 볼 수 있다. 문서 입력 시간은 문서의 크기에 따라서 달라질 수 있지만, 같은 크기의 문서를 입력할 때의 성능 그래프는 그림 5와 동일한 순서를 따른다. 여기에서 준분할 저장 방식의 입력 시간이 다른 방법들보다 빠른

이유는 XML 문서를 저장하는 방법과 밀접하게 관련되어 있다. 분할 저장 방식이나 가상 분할 저장 방식이 XML 문서의 모든 엘리먼트를 일일이 고려해서 저장해야 하는 반면에, 준분할 저장 방식은 의미 있게 분할된 단편 노드만을 고려해서 저장하면 되기 때문이다.



(그림 6) 수정 시간 비교

그림 6은 각 방법의 문서 수정 시간을 보여준다. 이 실험에서 우리는 전자 책 문서의 한 절을 수정했다. 책에서 차지하는 각 절 영역의 크기가 비슷하기 때문에, 준분할 저장 방식의 문서 수정 시간은 XML 문서의 크기에 상관없이 70ms 정도로 거의 비슷한 시간이 소요되었다. 이것은 SCOF 모델의 준분할 저장 방식이 문서 수정 작업에 의해서 영향을 받는 단편 노드만을 갱신하기 때문이다. 또한 단편 노드 안에 존재하는 엘리먼트 간의 구조 정보를 일일이 고려하지 않기 때문에 다른 방법들의 문서 수정 시간보다 빠르다는 것을 알 수 있다. 이런 측면에서 문서 수정 시간은 수정 영역이 넓어질수록 문서 입력 시간과 밀접한 관계가 있다.

비록 XQuery가 XML 문서를 다루는 강력한 기능을 가지고 있다고 하더라도, 대용량 XML 데이터를 다루는 데 있어서 XQuery의 성능 테스팅은 별로 이루어져 있지 않은 상태이다. 명세서(specification)의 복잡성 때문에, 사용자들이 기대하고 만족할 만한 성능을 가진 XQuery 기반의 시스템을 구현하는 것은 아직까지는 어려운 상황이라고 볼 수 있다. 이것에 대한 실용적 대안으로 SCOF 접근법은 검색 및 관리 측면에 있어서 대용량의 XML 문서를 처리하는데 훨씬 적합한 방법론임을 알 수 있다.

5. 결론 및 향후 연구

지금까지 우리는 SCOF 모델을 사용한 XML 정보 검색 관리 시스템에 대해서 살펴보았다. 이러한 접근법의 장점은 서비스 공급자인 관리자와 서비스 수요자인 사용자 입장에서 살펴볼 수 있다. 먼저 관리자 입장에서는 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 보다 효과적으로 검색 및 관리 서비스를 제공할 수 있다. 사용자 입장에서는 XML 문서의 구조적 특성을 명확하게 알지 못하거나 질의어 작성에 익숙하지 않더라도, 키워드 방식의 인터페이스를 사용해 쉽게 검색 서비스를 이용할 수 있다.

그리고 시스템 측면에서 얻을 수 있는 장점은 문서의 중요한 엘리먼트를 중심으로 구조 정보를 그대로 유지하면서도 검색 결과 재구성 시에 엘리먼트 재조합을 위한 오버헤드가 거의 발생하지 않는다는 것이다. 또한 문서 관리 작업이 주로 단편 노드 기반으로 이루어지기 때문에 대용량의 문서를 처리함에 있어 오버헤드를 분산시키는 이점을 얻을 수 있다. 이것은 실제적으로 XML 정보 검색 관리 시스템을 실용화 시키는데 있어서 가장 중요한 두 가지 요소라고 생각된다. 실제로 "역사정보통합시스템"[20]이나 "조선왕조실록"[18]과 같이 대용량의 XML 문서를 다루거나 "향토문화전자대전"[21]과 같이 실시간 문서 편찬 작업을 수행하는 사이트에서 SCOF 모델을 적용한 KRISTAL[22]이 효과적으로 활용되고 있다.

향후 연구로는 사용자의 다양하고 복잡한 요구사항을 충분히 반영할 수 있도록 변환 규칙에 대한 보다 상세하고 다양한 명세 작업이 필요하다. 대부분의 시스템들이 검색 시에 XML 질의 표현식을 사용하는 반면에 본 시스템은 변환 규칙을 통하여 질의에 관련된 대부분의 작업을 처리하기 때문에, 변환 규칙에 좀 더 융통성을 부여할 필요가 있다. 또한 관리자가 변환 규칙을 작성함에 있어서 지침이 될 수 있는 원칙을 만들어서 제공하는 것도 일관성 있는 변환 규칙 작성을 위해서 필요하다.

참 고 문 헌

[1] 김훈, 한상웅, 홍의경, "XML 문서 저장 시스템", 데이터베이스연구회지, 제16권, 제2호, pp. 29-34, 2000

[2] 박성진, "XML 데이터베이스", 한국인터넷정보학회지, 제2권, 제3호, pp. 38-46, 2001

[3] P. Francois, "Generalized SGML repositories: Requirements and modelling", Computer Standards & Interfaces 18, pp. 11-24, ELSEVIER, 1996.

[4] INEX, http://inex.is.informatik.uni-duisburg.de

[5] XPath, http://www.w3.org/TR/xpath

[6] XQuery, http://www.w3.org/TR/xquery

[7] A. Trotman, and M. Lalmas, "Why Structural Hints in Queries do not Help XML-Retrieval", SIGIR'06, pp. 711-712, Seattle, Washington, USA, 2006.

[8] B. Larsen, A. Tombros, and S. Malik, "Is XML Retrieval Meaningful to Users? Searcher Preferences for Full Documents vs. Elements", SIGIR'06, pp. 663-664, Seattle, Washington, USA, 2006.

[9] J. Kamps, M. Koolen, and M. Lalmas, "Where to Start Reading a Textual XML Document?", SIGIR'07, pp. 723-724, Amsterdam, The Netherlands, 2007.

[10] H. S. Kim, and H. J. Son, "Users Interaction with the Hierarchically Structured Presentation in XML Document Retrieval", INEX 2005, pp. 422-431, Glasgow, Scotland, 2005.

[11] S. Betsi, M. Lalmas, A. Tombros, and T. Tsikrika, "User Expectations from XML Element Retrieval", SIGIR'06, pp. 611-612, Seattle, Washington, USA, 2006.

[12] D. Florescu, D. Kossmann, and I. Manolescu, "Integrating Keyword Search into XML Query Processing", WWW 2000, Amsterdam, The Netherlands, 2000.

[13] T. Shimizu, N. Terada, and M. Yoshikawa, "Kikori-KS: An Effective and Efficient Keyword Search System for Digital Libraries in XML", ICADL 2006, Kyoto, Japan, 2006.

[14] L. Guo, F. Shao, C. Botev, and J. Shanmugasundaram, "XRANK: Ranked Keyword Search over XML Documents", SIGMOD 2003, pp. 16-27, San Diego, CA, 2003.

[15] R. Schenkel, and M. Theobald, "Structural Feedback for Keyword-Based XML Retrieval", ECIR 2006, pp. 326-337, London, England, 2006.

[16] S. K. Ko, and Y. C. Choy, "A Structured Documents Retrieval Method supporting Attribute-based Structure Information", SAC 2002, pp. 668-674, Madrid, Spain, 2002.

[17] C. D. Manning, P. Raghavan, and H. Schutze, "Introduction to Information Retrieval", pp. 149-164, Online Book, 2007.

[18] The Annals of The Choson Dynasty, http://sillok.history.go.kr

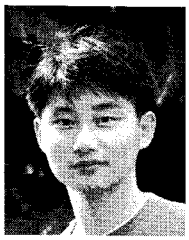
[19] L. Mignet, D. Barbosa, and P. Veltri, "The XML Web: a First Study", WWW 2003, pp. 500-510, Budapest, Hungary, 2003.

[20] Korean History On-line, http://www.koreanhistory.or.kr

[21] Digital Encyclopedia for Local Areas, http://www.grandculture.net

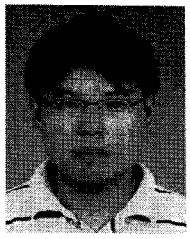
[22] KRISTAL, http://www.kristalinfo.com

◐ 저 자 소 개 ◑

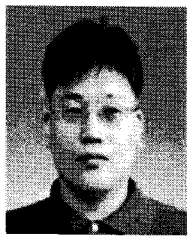


정 창 후(Chang-Hoo Jeong)
1999년 충남대학교 컴퓨터과학과 졸업(학사)
2002년 충남대학교 대학원 컴퓨터과학과 졸업(석사)
2003년~현재 한국과학기술정보연구원 정보시스템개발팀
관심분야 : 정보검색 및 추출, 디지털 도서관, 메타데이터 레지스트리
E-mail : chjeong@kisti.re.kr

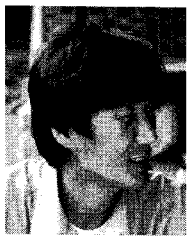
최 윤 수(Yun-Soo Choi)
1993년 충남대학교 컴퓨터공학과 졸업(학사)
1995년 충남대학교 대학원 컴퓨터공학과 졸업(석사)
1995년~현재 한국과학기술정보연구원 선임연구원
관심분야 : 데이터베이스, 정보검색
E-mail : armian@kisti.re.kr



진 두 석(Du-Seok Jin)
1999년 전북대학교 컴퓨터공학과 졸업(학사)
2001년 전북대학교 대학원 컴퓨터공학과 졸업(석사)
2000년~현재 한국과학기술정보연구원 선임연구원
관심분야 : 데이터베이스, 정보검색
E-mail : dsjin@kisti.re.kr



김 진 숙(Jinsuk Kim)
1993년 한국과학기술원 생물공학과 졸업(학사)
1995년 한국과학기술원 생명과학과 졸업(석사)
1995년~2001년 한국과학기술정보연구원 선임연구원
2001년~2002년 (주)서치솔루션 정보시스템연구소 선임연구원
2002년 한국과학기술원 전자전산학과 전산학전공 졸업(석사)
2002년~현재 한국과학기술정보연구원 선임연구원
관심분야 : 정보검색, 문서분류, 생물서열정보처리
E-mail : jinsuk@kisti.re.kr



윤 화 묵(Hwa-Mook Yoon)
1992년 서울산업대학교 전자계산학과 졸업(학사)
1997년 공주대학교 대학원 전자계산학과 졸업(석사)
2005년~현재 배재대학교 재학(박사)
현재 한국과학기술정보연구원 정보시스템개발팀장
관심분야 : 데이터베이스, 정보검색, 온톨로지
E-mail : hmyoon@kisti.re.kr