Journal of Korea Multimedia Society Vol. 11, No. 7, July 2008(pp. 905-913)

분할된 역 인덱스 테이블에서 부모노드의 정보를 이용한 질의 처리

김명수[†], 황병연[††]

요 약

최근 XML(Extensible Markup Language) 사용의 증가와 함께 다량의 이질적 구조를 가진 XML 문서들이 이용되고 있으며, 이러한 XML 문서들의 효율적인 관리를 위해 데이터 저장 구조에 대한 연구의 필요성이 증가하고 있다. 이에 따라 본 논문에서는 이들 XML 문서들의 효율적인 관리를 위하여 분할된 역 인덱스 테이블에서 부모노드의 정보를 이용하여 질의를 처리하는 방법을 제안한다. 이질적 구조를 가진 대규모의 문서들에 대한 질의 처리 횟수와 처리 데이터 량은 검색 성능에 큰 영향을 주기 때문에 데이터 구조를 설계할 때에 이 두 가지 요소들을 신중히 고려해야 한다. 제안된 방법은 부모 노드의 정보를 이용하여 선형 경로 질의를 위한 질의 처리 횟수를 반감시키고, 역 인덱스 테이블을 XML 트리의 깊이에 따라 적절히 분할하여 탐색 대상이 되는 데이터의 양을 줄이기 때문에 XML 문서에 대한 검색 성능을 향상시킨다. 제안하는 방법의 효율성을 입증하기 위해 인터넷에서 수집한 XML 문서들에 대한 XPath 질의 처리 성능이 기존의 역 색인 기법들에 비해 우수함을 보인다.

Query Processing using Information of Parent Nodes in Partitioned Inverted Index Tables

Myung-Soo Kim[†], Byung-Yeon Hwang[††]

ABSTRACT

Many heterogeneous XML documents are being widely used with the increasing employment of XML, and the importance of data structure research for more efficient document management has been growing steadily. We propose a query processing technique which uses parent node information in a partitioned inverted index tree. The searching efficiency of these heterogeneous documents is greatly influenced by the number of query processing and the amount of target data sets in many ways. Therefore, considering these two factors is very important for designing a data structure. First, our technique stores parent node's information in an inverted index table. Then using this information, we can reduce the number of query processing by half. Also, the amount of target data sets can be lessoned by using partitioned inverted index table. Some XML documents collected from the Internet will be used to demonstrate the new method, and its high efficiency will be compared with some of the existing searching methods.

Key words: XML, Path Storing(경로 저장), Partitioned Inverted Index Tables(분할된 역 인덱스 테이블)

※ 교신저자(Corresponding Author) : 황병연, 주소 : 경기도 부천시 원미구 역곡2동 산 43-1(420-743), 전화 : 02)2164-4363, FAX : 02)2164-4777, E-mail : byhwang@catholic.ac.kr

접수일 : 2007년 10월 4일, 완료일 : 2008년 3월 5일

[†] 준회원, 가톨릭대학교 컴퓨터공학과 학사과정 (E-mail : mail.mskim@gmail.com)
[††] 종신회원, 가톨릭대학교 컴퓨터정보공학부 교수

※ 본 연구는 2008년도 가톨릭대학교 교비연구비의 지원으로 이루어졌음

1. 서 론

1996년 W3C(World Wide Web Consortium)에서 XML(Extensible Markup Language)[1]이 제안된 이후 그 사용이 증가하면서, 점차 대량의 XML 문서를 저장하고 그로부터 원하는 정보를 효율적으로 탐색하는 기법을 개발하는 것이 XML 문서의 저장 및 활용에 관련된 주요 연구 과제로 자리 잡고 있다. 특히, XML 문서가 가지는 구조 또한 매우 다양하여 XML 문서 구조의 표준이 존재하기 어렵기 때문에, 최근에는 이러한 이질적인 구조를 가진 문서들의 관리를 위한 연구가 활발히 진행되고 있다. 개체간의 관계 분석을 통하여 경로 표현식을 처리하는 방법은 크게 스키마-레벨(Schema-level) 방법과 인스턴스-레벨(Instance-level) 방법으로 나눌 수 있다. 스키마-레벨 방법이란 경로 표현식을 이용한 개체 탐색 시 개체의 부모-자식 관계성, 조상-후손 관계성 등 구조적 정보를 사용하는 방법이다[2-4]. 인스턴스-레벨 방법은 개체의 시작 위치, 끝 위치 등 XML 트리 내부의 노드 구분 정보만을 사용하는 방법이다[5,6]. 구조적인 정보를 사용하는 스키마-레벨 방법은 레이블 경로들이 어디에 어떻게 저장되는가에 따라 관계형 데이터베이스를 사용하는 방법[2,3]과 특별한 목적의 인덱스를 사용하는 방법[4]으로 구분된다. 관계형 데이터베이스를 사용하는 방식 중 하나의 고정된 데이터베이스 저장 스키마가 모든 XML 문서의 구조를 저장하는 방법을 모델 매핑 방법(Model-mapping approach)이라 한다[7,8].

제안되어진 기존의 방법들 중 XIR-Linear(XML via Information Retrieval for Lenear Path Expressions)[9] 기법은 관계형 테이블을 사용하는 스키마-레벨 방식에 기반을 두고, 대규모 이질 저장 구조를 가진 문서들에 대한 질의 처리 효율성과 확장성을 향상시키기 위하여 역 인덱스 기술을 XML 문서의 구조 검색에 활용하는 모델 매핑 방법이다. 역 인덱스(inverted index)[10] 기술이란 정보 검색(information retrieval, IR) 분야에서 대규모 문서들을 빠르게 찾기 위해 전통적으로 사용해 온 기술이며, 이 기술을 통하여 XML 문서들의 구조 정보인 레이블 경로들에 대해 역 인덱스를 구축하고 이 인덱스를 사용하여 선형 경로 질의들의 효과적인 처리가 가능하다.

XIR-Linear 방법에서 역 인덱스를 구축하기 위하여 그 기반을 두고 있는 경로 저장 방식은 모든 가능한 경로 표현식들을 데이터베이스에 문자열 속성으로 저장시키는 방식이다. 그러나 이러한 경로 정보의 저장 기법에서는 루트부터 각 노드까지의 모든 엘리먼트 노드들에 대한 경로 정보를 저장하게 되고, 이는 정보의 중복을 초래하여 경로 정보의 양을 증가시키게 되며, 따라서 경로 정보를 이용하는 XML 질의 처리의 성능을 저하시키는 요인이 된다. 이에 따라 대용량 XML 데이터베이스에서 경로정보의 중복을 제거한 효율적인 질의 처리[11]에서는 루트에서 각 노드까지의 모든 경로가 아닌 루트부터 가장 긴 단말 엘리먼트 노드까지의 경로만을 저장하여 경로의 수를 줄이고, 경로 레이블에 대한 역 인덱스를 만들 때도 단말 엘리먼트 경로만을 대상으로 함으로써 키워드별 포스팅 리스트의 수를 줄이는 방법을 통하여 위와 같은 성능저하 요인을 제거했다. 그러나 질의에 포함되는 모든 엘리먼트 노드들에 대하여 키워드별 포스팅 리스트들을 찾는 과정은, 질의에 포함된 해당 노드들의 XML 트리에서의 깊이가 포스팅 리스트에 저장되어져 있으며 그것이 주로 연속적이라는 점을 생각할 때 개선의 여지가 있다.

본 논문에서는 관계형 데이터베이스를 기반으로 대용량 XML 문서 집합에 대한 저장 및 질의 처리 기법 중 경로 정보의 중복을 제거하면서 역 인덱스를 함께 이용하는 방법을 개선하여, 분할된 역 인덱스 테이블에서 부모 노드의 정보를 이용한 효율적인 질의 처리를 제안한다. 제안 방법에서는 XML 문서와 질의 처리에 대하여 어떠한 제약도 부과하지 않으며, 현재 가용한 관계형 데이터베이스를 이용함과 동시에 보다 효율적으로 질의를 처리한다는 데에 그 설계 목표를 두고 있다.

본 논문의 구성은 다음과 같다. 2장에서는 XIR-Linear와 대용량 XML 데이터베이스에서 경로정보의 중복을 제거한 효율적인 질의 처리 방법을 소개하고, 3장에서는 새로운 XML 문서의 데이터 모델 및 관계형 데이터베이스 저장 구조를 제안한다. 4장에서는 성능평가를 통하여 XPath 질의들에 대한 제안 방법의 처리 성능과 효율성을 입증하고, 5장에서 결론 및 향후 연구과제에 대해 기술한다.

2. 관련연구

XIR-Linear 방법은 대규모의 이질적인 XML 문

서들에 대한 선형 경로 질의를 효과적으로 처리하기 위한 방법이다. 선형 경로 질의란 선형 경로 표현식(Linear Path Expression)으로 나타내어지는 질의 표현을 의미하는데, 선형 경로 표현식이란 전체 매치 질의(full match query)와 부분 매치 질의(partial match query)를 통칭한다. 경로 정보의 저장 방식은 관계형 테이블을 이용한 스키마 레벨 방법에 기반을 두고 있으며, 이 경로 정보를 역 인덱스 기술을 이용하여 역 인덱스 테이블에 저장하는 방법으로 질의 처리의 효율성을 향상시킨다. XIR-Linear 방법에서 경로정보에 대한 역 인덱스 구성을 위한 저장 구조는 다음과 같다.

LablePath (pid, labelpath)
LablePath Inverted Index (keyword, posting list)
*posting list = {pid, occurrence_count, offsets, label_path_length}

LabelPath Inverted Index는 LabelPath 테이블의 labelpath 필드에 대하여 구축된다. 이때, XIR-Linear는 각 레이블 경로를 텍스트 문서로 간주하고, 이들 레이블 경로 안에 있는 각 레이블들을 키워드로 사용한다. LabelPath 역 인덱스 구조는 키워드(즉, 레이블)와 포스팅 리스트의 쌍들로 구성되는데 포스팅 리스트의 각 포스팅은 레이블이 존재하는 레이블 경로의 식별자인 pid, 레이블 경로 내에서 그 레이블이 나타난 개수인 occurrence_count, 레이블 경로가 시작하는 위치로부터 해당 레이블들의 위치까지의 거리들의 집합인 offsets, 레이블 경로 내에 존재하는 레이블들의 개수인 label_path_length로 구성되어진다.

대용량 XML 데이터베이스에서 경로정보의 중복을 제거한 효율적인 질의처리[11]는 XIR-Linear와 마찬가지로 역 인덱스 기술을 사용하면서, 루트부터 각 노드까지의 모든 엘리먼트 노드들에 대한 경로 정보를 저장하는 것이 아닌 루트부터 가장 긴 단말 엘리먼트 노드까지의 경로만을 저장하는 방법을 통하여 효율을 얻는 방법이다. 또한 경로 레이블에 대한 역 인덱스를 만들 때도 단말 엘리먼트 경로만을 대상으로 함으로써 키워드별 포스팅 리스트의 수를 줄인다. 역 인덱스 검색 기법을 사용할 경우 레이블 경로 탐색 시간에 영향을 미치는 주된 요소는 경로 표현식에 주어진 레이블들의 개수와 이들 레이블들과 연관된 포스팅 리스트들의 길이의 합인데, 만약 경로 표현식에 주어진 레이블들이 증가하면 이들 레이블들을 통하여 만들어지는 연관된 포스팅 리스트들의 길이는 더욱 큰 비율로 증가하게 된다. 즉, 깊이가 n인 경로 표현식 하나가 증가할 때마다 포스팅 리스트는 n개 씩 증가하게 되는데, 역 인덱스 기법을 사용한 질의 처리에서 경로의 개수를 줄이는 작업이 전체 경로 탐색 시간에 큰 효율을 주는 이유는 이 때문이다. 이 방법에서 사용되는 경로정보에 대한 역 인덱스 구성을 위한 저장 구조는 다음과 같다.

Path (LEP_ID, pathExp)
PathIndex (keyword, {LEP_ID, level})

PathIndex 테이블은 XIR-Linear와 마찬가지로 posting list를 포함하고 있는데, 중요한 차이점으로 경로 ID인 pid 대신 단말 엘리먼트 경로 ID인 LEP_ID(Leaf Element Path ID)가 포함되어 있는 것을 볼 수 있다. XIR-Linear에서의 pid가 모든 내부 엘리먼트의 경로까지 포함한 추출 가능한 모든 경로를 나타내는 반면, LEP_ID는 단말 엘리먼트의 경로 ID만을 나타낸다. 단말 엘리먼트의 경로 ID만을 저장 할 경우의 경로 정보 테이블과 포스팅 리스트는 전체 경로 ID를 저장 할 경우의 경로 정보 테이블 및 포스팅 리스트보다 그 데이터 량이 확연히 줄어들게 되며, 이에 따라 레이블 경로 탐색 시간 역시 줄어든다.

3. XML 문서를 위한 데이터베이스 저장 구조

3.1 경로정보에 대한 데이터베이스 저장 구조

기존의 역 인덱스 검색방법들의 질의처리 과정을 살펴보면, 질의어에 포함된 레이블이 n개일 경우, 총 n번의 질의 처리를 통하여 해당 레이블이 속해 있는 경로 ID를 찾아내고, 이들, 찾아내어진 경로 ID들 중 모든 레이블이 속해 있는 경로 ID만을 선별하는 방법으로 질의를 만족하는 경로 ID를 찾아내게 된다. 그러나 이러한 방법은 질의에 포함된 해당 노드들의 XML 트리에서의 깊이가 주로 연속적이라는 점을 생각할 때 개선의 여지가 있다. 다음 Q1 질의문과 그에 따른 질의 처리를 생각해보자.

Q1 : /PLAY/ACT/SCENE/SPEECH

Q1과 같은 부모-자식 관계성 '/' 만으로 연결된 질의를 수행할 경우, 기존의 방법을 따르면 역 인덱스

테이블의 포스팅 리스트에서 레벨이 '1'이고 키워드가 'PLAY'인 포스팅들의 경로 ID 집합과, 레벨이 '2'이고 키워드가 'ACT'인 포스팅들의 경로 ID 집합 또, 레벨이 '3'이고 키워드가 'SCENE'인 포스팅들의 경로 ID 집합과, 레벨이 '4'이고 키워드가 'SPEECH'인 포스팅들의 경로 ID 집합 즉, 총 네 번의 질의 처리 과정을 거쳐 네 개의 집합을 구한 후, 네 집합의 AND 연산을 통하여 최종적으로 원하는 경로 ID 집합을 찾아내게 된다. 그러나 만약 포스팅 리스트에 한 단계 위의 부모 노드의 정보를 추가해 준다면, 레벨이 '2'이고 키워드가 'ACT'이면서 부모 노드의 키워드가 'PLAY'인 포스팅들의 경로 ID 집합과, 레벨이 '4'이고 키워드가 'SPEECH'이면서 부모노드의 키워드가 'SCENE'인 포스팅들의 경로 ID 집합으로 단 두 번의 질의 처리 과정을 통해 찾아낼 수 있게 된다.

부모 노드의 정보를 포스팅 리스트에 포함시키는 방법은, 단지 질의 처리 횟수를 감소시키는 것 외에도, 역 인덱스 테이블의 분할 및 루트 엘리먼트 노드의 포스팅 리스트에서의 분리를 가능하게 하는, 제안 방법에 있어서의 가장 중요한 핵심이 된다. Q1의 질의처리에서 볼 수 있듯이 만약 한 단계 위의 부모 노드의 정보를 가지고 있고 이에 따라 이원화로 최적화 되어진 역 인덱스 테이블에서는 전체 매치 질의에서 부모노드의 정보를 이용한 검색을 하게 될 경우 질의 처리 시 사용되는 각 레이블들의 level 값은 모두 2의 배수, 즉 짝수만으로 이루어지게 된다. 만약 역 인덱스 테이블을 XML 트리의 깊이에 따라 even 테이블과 odd 테이블로 나누게 된다면, 루트 엘리먼트로 시작되고 부모-자식 관계로 이루어지는 깊이 2n인(단, n은 n≥1인 정수) 질의 처리를 위해서는 단지 even 테이블 만을 검색하는 것으로 충분하다. 즉, 부모노드의 정보와 역 인덱스 테이블의 이원화를 통한 경로 정보의 데이터베이스 저장 구조에서는 Q1 예제와 같은 질의가 주어졌을 경우, 경로 ID를 검색하는데 있어 기존의 방법과 비교하여 약 1/2 수준의 데이터만을 대상으로 검색이 가능하다. 만약 더 높은 차원의 분할을 시도한다면 검색 대상이 되는 데이터의 양도 그에 반비례하게 줄어들게 된다. 이러한 몇 가지 사실들을 종합하여 테이블을 이원화 할 경우를 예로 들어 제안 방법의 경로정보에 대한 관계형 데이터베이스 저장 구조를 다음과 같이 정리한다.

Document (docID, docName, rootElem)
Node (nodeNum, nodeName)
Path (LEP_ID, pathExp)
PathIndexEven (nodeNum, {LEP_ID, level, parent})
PathIndexOdd (nodeNum, {LEP_ID, level, parent})

경로정보의 관계형 데이터베이스 저장 구조를 제시함에 있어 Document 테이블의 저장 구조를 함께 소개한 이유는 모든 경로의 시작이 되는 루트 엘리먼트에 대한 정보를 Document 테이블이 가지고 있기 때문이다. Q1 예제를 통하여 한 가지 더 알 수 있는 사실은, 깊이가 2 이상인 모든 질의 처리에 있어서 루트 엘리먼트의 정보를 사용하는 일이 거의 없다는 것이다. 다음 Q2, Q3 예제를 통하여 이를 알 수 있다.

Q2 : /PLAY/ACT/SCENE

Q3 : //PLAY/PLAYSUBT

Q2와 Q3 모두 깊이가 2 이상인 질의 처리 예제이다. Q2의 경우 '/PLAY/ACT'를 하나로 묶어 Q1과 같은 방법으로 PathIndexEven 테이블에서 검색한 후, 부모노드의 정보 없이 PathIndexOdd 테이블에서 SCENE을 검색하면 된다. Q3의 경우 '//PLAY/PLAYSUBT'를 하나로 묶어 Q1과 같은 방법으로 검색하되, 자손 탐색으로 질의가 시작되는 경우이기 때문에 PathIndexEven 테이블과 PathIndexOdd 테이블의 선택이 불가능하다. 따라서 양쪽 테이블 모두를 검색하게 된다(단, 이 경우 역시 질의 처리 횟수가 1/2로 줄어든다는 점은 동일하다).

앞의 두 가지 예제 모두 질의의 깊이가 2 이상이며, 이와 같은 경우 루트 엘리먼트인 'PLAY'에 대하여서는 부모노드로서의 정보만이 필요할 뿐 PathIndexOdd 테이블에 속해있는 스스로가 검색의 대상이 되는 일은 없음을 알 수 있다. 루트 엘리먼트가 사용되는 유일한 경우는 깊이가 1인 질의가 처리될 경우뿐이며, 단지 이러한 적은 경우의 수를 위하여 루트 엘리먼트의 모든 포스팅을 역 인덱스 테이블에 저장하기에는 그 비용이 너무 크다. 따라서 제안하는 경로정보의 데이터베이스 저장 구조에서는 루트 엘리먼트에 대한 정보를 Document 테이블에 저장한다. 하나의 문서는 오직 하나의 루트 엘리먼트를 가지고 있기 때문에 이는 타당하다. Node 테이블의 nodeNum은 각 nodeName당 하나의 키 값으로 주어

지게 되는데, 이는 역 인덱스 테이블에서의 키워드를 숫자로 대체하기 위해서이며, 또한 포스팅에서의 parent 역시 부모 노드의 nodeName에 해당하는 nodeNum 값을 저장하게 된다.

역 인덱스 테이블 구성의 기본이 되는 Path 테이블은 [11]에서의 LEP_ID를 사용하였으며 제안 방법의 Path 테이블, Node 테이블과 역 인덱스 테이블은 그림 1과 같다. 이 과정에서 구축되는 역 인덱스는 전통적인 역 인덱스의 방법에 따라 키워드(본 논문에서는 nodeNum으로 대체)와 포스팅 리스트들의 쌍들로 구성된다. 특히 제안하는 방법을 위해 사용되는 포스팅 리스트들은 Path 테이블의 pathExp 필드에 대해서 구축되며, 키워드를 대체하기 위하여 Node 테이블을 참조한다. 역 인덱스 구축을 위한 보다 자세한 정보는 전통적으로 사용되어져 온 역 인덱스 기법[10]과 XIR-Linear[9] 등으로부터 참조할 수 있다.

3.2 노드정보에 대한 데이터베이스 저장 구조

제안 방법에서 각 노드에 대한 정보를 저장하기 위한 데이터베이스 저장 구조는 제안하는 방법의 효율성과 직접적인 연관이 없기 때문에 기존의 방법들 중 대용량 XML 데이터베이스에서 경로정보의 중복을 제거한 효율적인 질의 처리 방법에서 사용된 저장 구조와 동일한 구조를 제시한다.

nodeNum	keyword
1	PLAY
2	TITLE
3	PLAYSUBT
4	ACT
5	SCENE
6	STAGEDIR
7	SPEECH
8	SPEAKER
9	LINE
10	RUNNINGTIME
11	@id

(a) Node table

LEP_ID	pathExp
1	PLAY.TITLE
2	PLAY.PLAYSUBT
3	PLAY.ACT.TITLE
4	PLAY.ACT.SCENE.TITLE
5	PLAY.ACT.SCENE.STAGDIR
6	PLAY.ACT.SCENE.SPEECH.SPEAKER
7	PLAY.ACT.SCENE.SPEECH.LINE
8	PLAY.RUNNINGTIME.ACT.SCENE
9	PLAY.@id

(b) Path table

nodeNum	posting list
2	<3,3,4>
4	<8,3,10>
5	<4,3,4> <5,3,4> <6,3,4> <7,3,4>
8	<6,5,7>
9	<7,5,7>

(c) PathIndexOdd table

nodeNum	posting list
2	<1,2,1> <4,4,5>
3	<2,2,1>
4	<3,2,1> <4,2,1> <5,2,1> <6,2,1> <7,2,1>
5	<8,4,4>
6	<5,4,5>
7	<6,4,5> <7,4,5>
10	<8,2,1>
11	<9,2,1>

(d) PathIndexEven table

그림 1. Path 테이블과 PathIndex 테이블의 예

각 노드의 정보를 저장할 때에는 대표 단말 엘리먼트 노드 ID(Representative Leaf Element Path ID, RLEP_ID)를 사용한다. 각 엘리먼트 노드의 단말 엘리먼트 경로는 여러 개가 있을 수 있으므로, 그 중에서 가장 작은 값을 갖는 단말 엘리먼트 경로의 ID를 RLEP_ID로 사용한다. RLEP_ID를 저장한다는 것은 내부 엘리먼트의 경로를 따로 저장하지 않는다는 뜻이므로, 각 노드의 속성 중 RLEP_ID에서의 level이 추가되어 있다. 마지막으로 nodeID는 Tatarinov 등[12]이 제안한 순서화된 XML 데이터모델 중 질의와 갱신이 혼합된 작업에 최적인 듀이 순서 코딩(Dewey order encoding)을 사용하여 나타낸다. 단말 노드는 자료 값 즉, 텍스트를 가지고 있는 노드에 해당되고, 내부노드는 XML 엘리먼트에 해당된다. XML 문서 트리는 엘리먼트 노드와 텍스트 노드 외에 애트리뷰트 노드를 포함할 수 있는데, 애트리뷰트 노드는 서브 엘리먼트가 없는 엘리먼트 노드로 취급하여, XML 트리를 구성할 경우 부모의 제일 마지막 자식 노드로 합한다. 제안 방법에서 각 노드 타입별 관계형 데이터베이스 저장 구조는 다음과 같다.

Element (docID, nodeID, RLEP_ID, level, nodeName)

Text (docID, nodeID, RLEP_ID, level, value)

Attribute (docID, nodeID, RLEP_ID, level, value)

3.3 질의 처리 유형에 따른 분석

제안 방법에서 부모노드의 정보와 이원화된 역 인덱스 테이블을 이용한 질의 처리 유형은 크게 네 가지로 나누어 설명할 수 있다. 질의에 포함된 레이블이 오직 하나이고 그것이 루트 엘리먼트일 경우, 질의에 포함된 레이블이 오직 하나이고 그것이 루트 엘리먼트가 아닌 자손 탐색으로 시작할 경우, 질의에 포함된 레이블이 두 개 이상이고 루트 엘리먼트로 시작될 경우, 마지막으로 질의에 포함된 레이블이 자손 탐색으로 시작하지만 레이블이 두 개 이상이거나 레이블이 하나일지라도 그것이 루트 엘리먼트가 아님이 명확할 경우 등이 그것이다. 이 네 가지 유형을 조상-후손 관계성을 나타내는 '//'로 연결하여 실제 사용되는 모든 질의어를 구성할 수 있다. 네 가지 유형의 예는 그림 2와 같다.

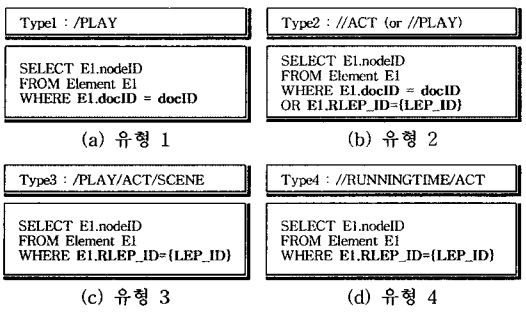
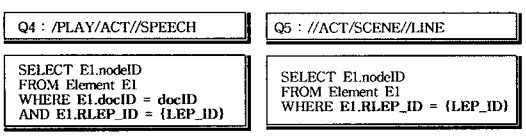


그림 2. 질의 처리의 네 가지 유형

먼저 유형 1과 같은 경우 질의어에 속한 하나의 레이블이 루트 엘리먼트인 것이 명확하기 때문에 포스팅 리스트를 검색할 필요가 없으며 Document 테이블에서 해당 레이블을 rootElem으로 갖는 docID를 찾아 질의한다. 유형 2와 같은 경우 유형 1과 유사하지만 질의어에 속한 레이블의 루트 엘리먼트 여부를 알 수 없기 때문에 Document 테이블과 포스팅 리스트를 모두 검색하여 일치하는 docID나 LEP_ID 집합을 통하여 질의한다. 유형 3의 경우 질의어에 속한 레이블들의 깊이가 명확하므로 2n(단, n은 n≥1인 정수)번째의 레이블들을 PathIndexEven테이블에서 검색하되, 질의어의 전체 깊이가 홀수일 경우 마지막 노드는 PathIndexOdd 테이블에서 검색하여 LEP_ID 집합을 얻는다. 유형 4의 경우 유형 3과 유사하게 검색하되, 자손 탐색으로 질의가 시작되므로 PathIndexEven 테이블과 PathIndexOdd 테이블의 선택이 불가능하다. 따라서 양쪽 테이블 모두를 검색하여 LEP_ID 집합을 얻는다. 실제로 주어지는 다양한 질의어는 위의 네 가지 유형을 '//'로 연결한 것과 동일하게 해석하여 처리할 수 있는데 그 예는 다음과 같다.



Q4의 질의를 '//'를 기준으로 나누면 '/PLAY/ACT'와 '//SPEECH' 두 개의 질의가 된다. 이는 각각 유형 1과 유형 4에 해당하는 것이며, 앞서 설명한 방법으로 docID와 LEP_ID 집합을 찾아 AND 연산으로 질의한다. Q5의 경우 역시 질의어 내부에 '//'가 포함되어 있기 때문에 이를 기준으로 '//ACT/SCENE'와 '//LINE'의 두 개의 질의로 나눈다. '//LINE' 은 질의에 포함된 레이블이 하나뿐이지만 그것이 루트 엘리먼트가 아님이 명확하기 때문에 유형 4에 해당된다. 따라서 나누어진 두 개의 질의를 앞서 설명한 유형 4에 해당하는 방법으로 LEP_ID 집합을 찾아내어 질의한다. 이러한 방법들을 정리하여 그림 3의 알고리즘을 작성하였다.

알고리즘의 처리 과정에 따르면 질의문 Q4는 '//'를 기준으로 두 개의 질의문으로 나누어지게 된다. 만약 나누어진 질의문들 중 첫 번째 질의문이 단 하나의 레이블을 갖고 있다면 XML 문서 수준의 검색을 Document 테이블에서 실행하게 되며, 그 밖의 경우 기준이 되는 노드의 depth와 부모 노드의 nodeID 속성을 이용하여 PathIndex 테이블에서 포스팅 리스트들을 검색한다. 이 작업을 나누어진 질의문들의 수만큼 반복하고, 이를 통해 만들어지는 포스팅 리스트들의 LEP_ID들의 집합을 tempPathIDset 이라고 하면, 모든 tempPathIDset들에 공통으로 포함되어 있는 LEP_ID가 찾고자 하는 노드가 포함된 경로의 LEP_ID임을 알 수 있다. Q4의 예에서 두 개의 질의문 중 첫 번째 질의문은 depth가 2이고 부모노드의 nodeID가 1인 포스팅 리스트들에 대한 검색을 수행하게 되며(이것은 포스팅 리스트의 두 번째 요소가 2이고 세 번째 요소가 1임을 의미한다), 두 번째 질의

```
Algorithm SearchNodes
Input : PEset[]:path expression of query divided by inner '//',
        NOL:number of labels of PEset[]
Output : set of nodes matching the all of PEset[]

begin
  initialize docIDset[], tempPathIDset[];
  n ← 0;
  while n<PEset.length do
  begin
    if (n=0 and NOL[n]=1) then
      if (PEset[n] appears after the first '/') then
        docID ← getDocID(PEset[n]);
      else
        docID ← getDocID(PEset[n]);
        tempPathIDset[n] ← getTempPathIDset(PEset[n]);
    else
      tempPathIDset[n] ← getTempPathIDset(PEset[n]);
  end
  filter LEP_IDs that is stored in all tempPathIDset[] and then store LEP_ID in
  filteredPathIDset;
  return findMatchedNode(docID, filteredPathIDset);
end

Algorithm findMatchedNode
Input : docID, filteredPathIDset
Output : set of nodes matching the docID and filteredPathIDset
begin
  SELECT E1.nodeID AS nodeSet[] FROM Element E1
  WHERE docID={docIDset} AND RLEP_D={filteredPathIDset};
return nodeSet[];
end
```

그림 3. 부분매치 질의 처리 알고리즘

문은 아무런 정보가 없기 때문에 분할되어진 모든 역 인덱스 테이블에서 자신의 nodeID가 7인 포스팅 리스트들에 대한 검색을 수행하게 된다. 첫 번째 검색으로 찾을 수 있는 LEP_ID들의 집합은 {3, 4, 5, 6, 7}이며, 두 번째 검색으로 찾을 수 있는 LEP_ID들의 집합은 {6, 7}이다. 따라서 찾고자 하는 노드가 포함된 경로의 LEP_ID는 6과 7임을 알 수 있다.

4. 성능평가

4.1 실험 환경

본 실험을 위한 실험 환경은 표 1과 같다.

또한, 제안 방법의 효율성을 측정하기 위하여 여러 가지 형태의 경로 질의를 사용하였으며, 이 질의를 위한 실험 데이터로는 약 15Mbyte에 해당하는 용량의 실제 웹상에서 수집 가능한 XML 문서 묶음[출처: http://dblab.catholic.ac.kr/dblab/source/xmldocs/exdocs.zip]을 사용하였다. 이 문서 묶음의 세부적 특징은 표 2와 같다.

표 1. 실험을 위한 환경

server system	
CPU/RAM	Intel(R) Xeon(TM) CPU 3.20GHz / 2.00GB RAM
OS	windows 2003 server
database	mssql server 2005 express
JDBC driver	microsoft sql server 2005 jdbc driver 1.1
client system	
CPU/RAM	Intel(R) Core(TM)2 CPU T5200 1.60GHz / 1.00GB RAM
OS	windows xp pro. sp2
programming language	jdk-1_5_0_12
XML parser	SAX builder with JDOM 1.1 library

표 2. 실험에 사용된 데이터 셋의 세부적 특징

데이터 량	16,733,215 Byte
파일 개수	200 개
총 라인 수	22,272 라인
평균 라인 수	111 라인
중복되지 않는 element 개수	197,397 개
중복되지 않는 attribute 개수	343,706 개
text 개수	55,138 개
XML 트리의 평균 depth	4.23

4.2 실험 결과

그림 4는 성능 평가를 위하여 사용한 경로 질의문들이다. 그림 5와 그림 6은 그림 4의 경로 질의를 이용하여 대용량 XML 데이터베이스에서 경로정보의 중복을 제거한 효율적인 질의 처리와 제안 방법을 각각 처리하는데 걸리는 시간을 보여준다. QR1과 QR2를 비교해 보면, QR1의 경우 제안 방법은 Document 테이블에서 루트 엘리먼트를 찾기 때문에 기존의 방법에 비하여 매우 빠른 검색 속도를 보이게 된다. 반면 QR2의 경우 제안 방법이 기존의 방법에 대하여 아무런 이점을 갖지 않기 때문에 거의 동일한 응답시간을 보인다. 질의문 전체가 부모-자식 관계성 '/'로 이루어져 있는 QR3와 QR4의 경우 제안방법이 현격한 차이의 빠른 응답시간을 갖는다. QR5와 QR6는 루트 엘리먼트를 알 수 없는 형태의 질의문으로서, 분할된 역 인덱스 테이블을 이용한 효율은 살릴 수 없지만 부모노드의 정보를 이용할 수 있다는 이점을 활용하여 마찬가지로 좋은 효율을 보인다.

다음으로 QR7부터 QR10까지의 성능평가 예제는 전체 경로 표현식을 관계성 '//'를 기준으로 두 가지 질의문으로 나누어 적용하는 알고리즘의 테스트를 위한 예제들이다. QR7과 QR8을 비교하면 QR7에 비하여 QR8의 경우 관계성 '//'의 경로 처리가 많아짐에 따라 그 효율이 줄어들게 되는데, 이것은 루트 엘리먼트로 시작되는 질의문의 경우 그 깊이가 명확하

Query#	경로 표현식	특징
QR1	/CIM	질의 처리의 대표적인 네 가지 유형들을 테스트하는 예제
QR2	//INSTANCEPATH	
QR3	/CIM/DECLARATION/VALUE/INSTANCEPATH	
QR4	/CIM/DECLARATION/DECLGROUP	
QR5	//NAMESPACEPATH/HOST	
QR6	//INSTANCENAME/KEYBINDING/KEYVALUE[-'STRING']	
QR7	/CIM/DECLARATION//PROPERTYNAME	전체 경로 표현식을 관계성 '//'를 기준으로 두 가지 질의문으로 나누어 적용시키는 예제
QR8	//INSTANCEPATH//PROPERTY/VALUE	
QR9	/CIM/DECLARATION//NAMESPACEPATH//HOST	
QR10	/CIM//DECLGROUP//INSTANCEPATH/NAMESPACEPATH//LOCAL	

그림 4. 성능 평가를 위한 경로 질의문

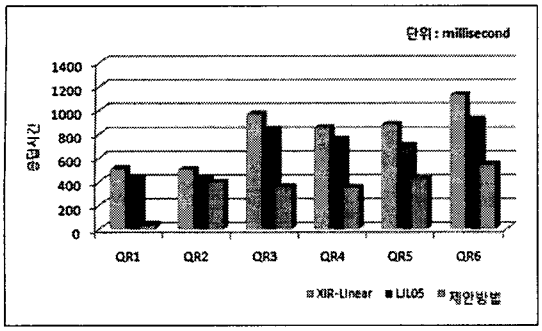


그림 5. 질의 처리 응답시간 QR1-QR6

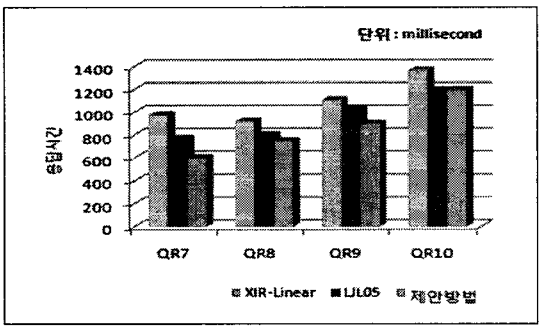


그림 6. 질의 처리 응답시간 QR7-QR10

기 때문이다. 제안하는 방법은 결과 집합을 도출함에 있어서 기존의 [9,11] 방법과 기본적으로 동일한 역 인덱스 기반의 처리 흐름을 가지고 있기 때문에, 관계성 '//'로 이루어진 질의가 많을수록 그 처리 과정이 유사한 흐름을 보이게 된다. 이러한 특징은 제안하는 방법의 검색 범위가 기존의 방법들과 완전히 동일하다는 장점을 갖도록 하는 반면, 관계성 '//'로 이루어진 질의가 많을수록 그 효율이 떨어지는 단점의 원인이 되기도 한다. 마지막으로 QR9과 QR10의 경우 전체 질의문을 세 가지 이상의 질의문으로 나누어 적용하게 되는 예제이며, 이와 같이 XML 트리의 깊이가 깊어지고 복잡한 조건을 가진 질의문일수록 제안 방법의 효율이 줄어든다는 것을 알 수 있다.

그림 7은 삽입된 노드의 개수에 따른 질의 처리 결과이다. 삽입된 노드의 개수는 데이터베이스에 삽입된 서로 다른 이름을 가진 노드의 개수를 의미하며 평균 응답시간은 질의 처리 응답시간 QR1부터 QR10까지의 총 처리 속도의 평균을 나타낸다. 실험 결과에 의하면 삽입된 노드의 개수가 많을수록 그 효율이 높게 나타남을 알 수 있다. 이것은 일반적인 모델 매핑 방법에서 루트노드부터 리프노드까지의

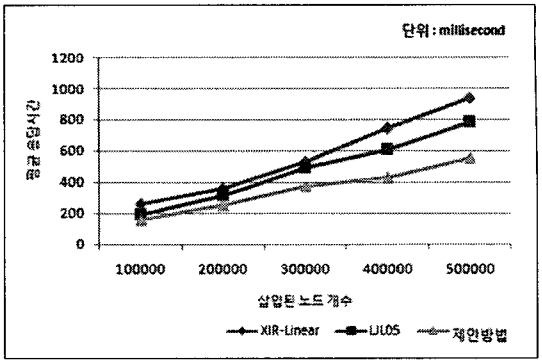


그림 7. 삽입된 노드의 개수에 따른 질의 처리

서로 다른 경로의 개수가 많을수록 그 검색 속도가 느려지게 되는 반면, 제안하는 방법의 검색 기법은 불필요하게 누적된 검색 대상을 최소화함으로써 처리 시간의 증가를 둔화시키기 때문이다.

5. 결 론

본 연구는 관계형 데이터베이스를 기반으로 한 기존의 연구들을 토대로, 경로 정보의 중복 제거 및 역 인덱스 사용과 함께 부모노드 정보의 검색이라는 새로운 검색 기법을 통하여 더욱 효율적인 검색 기법을 제안하였다. 포스팅 리스트에 부모노드의 정보를 저장함으로써 적은 비용으로 부모노드의 정보에 접근하여 질의 횟수를 감소시킬 수 있는 이 방법은, 단지 질의 처리 횟수를 감소시키는 것 외에도, 역 인덱스 테이블의 분할 및 루트 엘리먼트 노드의 포스팅 리스트에서의 분리를 가능하게 한다. 제안방법의 이러한 특성은 전체 질의 처리 속도에 상당한 영향을 미치게 된다.

예를 들어 질의 처리 횟수의 반감과 역 인덱스 테이블 분할의 이점은 각각 그 효율을 약 2배 수준으로 높일 수 있다. 이러한 두 가지 이점을 모두 얻게 되는 전체 매치 질의의 경우 경로 검색 속도는 최대 4배 이상 효율적이다. 또한 검색하고자 하는 노드의 깊이가 불분명할 경우에도 질의 처리 횟수 감소에 따라 얻게 되는 효율은 변함이 없기 때문에 부분 매치 질의의 경로 검색 속도 역시 약 2배의 효율을 보이게 된다. 이러한 경로검색 속도의 증가는 질의 처리 효율에도 상당한 영향을 미친다. 결과적으로 부모노드의 정보를 이용한 질의 처리 횟수 감소, 루트 엘리먼트의 정보를 제거한 역 인덱스 테이블의 분할 등을 통하여, 본 논문에서 제안한 방법이 기존의 단말 엘리먼트 경

로 ID를 이용한 질의 처리에 비해 최대 2배 이상의 효율을 얻을 수 있음을 실험을 통해서 보였다.

향후 연구에서는 테이블의 분할에 따른 XML 문서의 갱신 과정에서의 높은 비용을 해결해야 할 필요가 있으며, 최적의 분할 알고리즘을 밝히는 것 역시 남겨진 연구과제이다. 또한, 선형으로 표시되는 단순한 XPath 질의 외에 트리 형태를 띠는 복잡한 Twig 질의에 대한 처리 방법 역시 추가 연구가 필요한 부분이다.

참 고 문 헌

[1] "Extensible Markup Language(XML) 1.1," *W3C Candidate Recommendation*, 2002.

[2] M. Yoshikawa and T. Amagasa, "XRel: A Path-Based Approach to Storage and Retrieval of XML Documents using Relational Databases," *ACM Transactions on Internet Technology*, Vol.1, No.1, pp. 110-141, 2001.

[3] H. Jiang, H. Lu, W. Wang, and J. Xu Yu, "XParent: An Efficient RDBMS-Based XML Database System," *In Proc. of the 18th Int'l Conf. on Data Engineering (ICDE)*, San Jose, California, pp. 335-336, 2002.

[4] C. Chung, J. Min, and K. Shim, "APEX: An Adaptive Path Index for XML Data," *In Proc. of the 2002 ACM SIGMOD Int'l Conf. on Management of Data*, Madison, Wisconsin, pp. 121-132, 2002.

[5] S. Al-Khalifa, H. V. Jagadish, N. Koudas, and J. M. Patel, "Structural Joins: A Primitive for Efficient XML Query Pattern Matching," *In Proc. of the 18th Int'l Conf. on Data Engineering (ICDE)*, San Jose, California, pp. 141-152, 2002.

[6] H. Jiang, W. Wang, H. Lu, and J. X. Yu, "Holistic Twig Joins on Indexed XML Documents," *In Proc. of the 29th Int'l Conf. on Very Large Data Bases (VLDB)*, Berlin, Germany, pp. 273-284, 2003.

[7] 이윤호, 최일환, 김종익, 김형주, "색인된 XML 문서에서 레벨 정보를 이용한 효과적인 구조 조인 기법," *정보과학회논문지*, 제32권, 제6호, pp. 641-649, 2005.

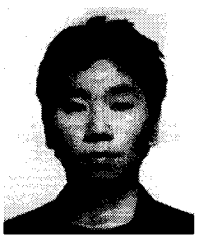
[8] Xiaodong Wu, Mong Li, and Lee Wynne Hsu, "A Prime Number Labeling Scheme for Dynamic Ordered XML Trees," *In Proc. of the 20th Int'l Conf. on Data Engineering (ICDE)*, pp. 66, 2004.

[9] 박영호, 한욱신, 황규영, "정보검색기술을 이용한 대규모 이질적인 XML 문서에 대한 효율적인 선형 경로 질의 처리," *정보과학회논문지*, 제31권, 제5호, pp. 540-552, 2004.

[10] G. Salton and M. McGill, *Introduction to Modern Information Retrieval*, McGraw-Hill, New York 1983.

[11] 이혜자, 정병수, 이영구, "대용량 XML 데이터베이스에서 경로정보의 중복을 제거한 효율적인 질의 처리," *데이터베이스연구*, 제21권 제3호, pp. 95-102, 2005.

[12] I. Tatarinov, S. D. Viglas, K. Beyer, J. Shanmugasundaram, E. Shekita, and C. Zhang, "Storing and Querying Ordered XML Using a Relational Database System," *In Proc. of ACM SIGMOD Conf.*, 2002.



김 명 수

2002년~현재 가톨릭대학교 컴퓨터공학과 재학

관심분야 : XML 인덱싱, 데이터베이스, 정보검색

황 병 연

1986년 서울대학교 컴퓨터공학과(공학사)

1989년 한국과학기술원 전산학과(공학석사)

1994년 한국과학기술원 전산학과(공학박사)

1994년~현재 가톨릭대학교 컴퓨터정보공학부 교수

1999년 University of Minnesota Visiting Scholar

2007년 California State University, Sacramento Visiting Scholar

관심분야 : XML 데이터베이스, 데이터마이닝, 정보검색, 지리정보시스템