韓國IT서비스學會誌
第7卷 第3號
2008年 9月

대용량 XML 문서의 효율적인 질의 처리를 위한 세그먼트 기반 역 인덱스

정병수* · 이혜자**

Segment-Based Inverted Index for Querying Large XML Documents

Byeong-Soo Jeong* · Hiye-Ja Lee**

■ Abstract ■

The existing XML storage methods which use relational data model, usually store path information for every node type including literal contents in order to keep the structural information of XML documents. Such path information is usually maintained by an inverted index to efficiently process XPath queries for large XML documents. In this study, We propose an improved approach that retrieve information from the large volume of XML documents stored in a relational database, while using a segment-based inverted index for path searches. Our new approach can reduce the number of searching an inverted index for getting target path information. We show the effectiveness of this approach through several experiments that compare XPath query performance with the existing methods.

Keyword : Large XML Documents, Path Search, Segment-based Inverted Index

논문투고일 : 2008년 06월 25일 논문수정완료일 : 2008년 09월 05일 논문게재확정일 : 2008년 09월 18일

* 경희대학교 전자정보학부
** 용인송담대학교 의료정보시스템과

1. 서 론

인터넷상의 정보 표현과 교환의 표준 포맷으로 XML(eXtensible Markup Language)이 제안되면서 XML로 작성되는 문서들의 양은 점점 더 많아지고 있다. 따라서 대용량의 XML 문서들을 효율적으로 저장하고 원하는 정보를 빠르게 찾기 위한 방법에 대한 연구는 그동안 많이 이루어져왔다[19, 22]. 대용량의 XML 문서들을 저장하는 기법은 크게 두 가지로 구분되어 발전되어 왔다. 그 하나는 XML 문서의 구조적 특징을 반영한 즉, XML 데이터 모델을 근간으로 하는 고유한 저장 시스템을 개발하는 방법이고, 다른 하나는 XML 문서의 내용 및 구조적 정보를 기존의 데이터 모델로 매핑하는 과정을 거쳐 저장 시스템은 기존의 데이터 모델(즉, 관계형 데이터 모델)을 사용하는 방법이다[19, 22]. 구조적인 특성을 갖는 XML 문서들을 관계형 데이터베이스로 저장하기 위해서는 XML 문서의 내용뿐만 아니라 문서의 구조적 정보를 트리 구조에 기반 하는 노드들로 분해하여 저장하는 것이 적절하다[22]. 하나의 고정된 데이터베이스 스키마가 모든 XML 문서의 구조를 저장하는 모델 매핑 방법에 따라 관계형 스키마를 설계할 경우의 주된 문제들 중의 하나는 트리 모델의 기본적인 구성체를 어떻게 관계형 스키마로 대응시키는가에 관한 것이다. 이에 대한 기존의 방법들로는, 에지(edge) 저장하기[10], 노드 저장하기[12], 경로와 영역(region)의 조합으로 표현하기[22], 경로와 노드식별자로 표현하기[11] 등의 방법이 제안되고 있다.

일반적으로 XML 문서에 대한 질의는 경로 표현식(path expression)을 포함하고 있다. XRel[22]에서는 모든 가능한 경로 표현식들이 데이터베이스에 스트링으로 저장되어 있어서, 이들 질의를 스트링 매칭으로 처리할 수 있다. 최근의 연구에서는 대용량 XML 데이터베이스에서 스트링 매칭의 횟수를 줄이기 위해 경로 정보에 대한 인덱스를 구성하는 방법들을 제안하고 있다[1].

경로 정보를 이용한 기존의 XML 문서 저장 및 질의 처리 기법들에서는 루트부터 각 노드까지의 모든 엘리먼트 노드들에 대해 경로 정보를 저장함으로써 정보의 중복을 초래하여 경로 정보의 양을 증가시키게 되며, 따라서 경로 정보를 이용하는 XML 질의 처리의 성능을 저하시키는 요인이 되었다. 이를 해결하기 위하여 [5]에서는 경로 정보 중 가장 긴 단말 엘리먼트 노드까지의 경로인 단말 엘리먼트 노드 경로(Leaf element Node Path; LNP)만을 저장하고 단말 엘리먼트 경로만을 대상으로 하여 역 인덱스를 구성함으로써 질의 처리의 성능을 향상시키고 있다.

이와 같이 단말 엘리먼트 경로만을 대상으로 하여 역 인덱스를 구성할 경우 경로정보의 양을 줄일 수 있으나 찾고자 하는 경로를 구성하는 엘리먼트의 개수만큼 인덱스를 검색해야 하는 부담은 여전하다. 특히 찾고자 하는 경로의 깊이(depth)가 깊을 경우, 경로를 구성하는 엘리먼트의 개수가 많으므로 인덱스 검색의 부담은 더 커지게 된다.

본 연구에서는 이러한 문제를 해결하기 위해 세그먼트 기반 경로 검색을 제안한다. 이 방법에서는 단말 엘리먼트 경로만을 대상으로 하면서, 이를 구성하는 엘리먼트들을 필요에 따라 몇 개씩 묶어 세그먼트로 만들고, 그 세그먼트들을 기반으로 하여 역 인덱스를 구성한다. 따라서 찾아야 할 엘리먼트 여러 개를 한꺼번에 검색할 수 있으므로 기존의 역 인덱스 이용 기법에 비해 경로 탐색을 위한 인덱스 검색 횟수를 줄이게 되어 질의 처리의 성능을 향상시킬 수 있다.

제안 방법에서는 저장되는 XML 문서에 대하여 어떤 제약도 부과하지 않으며, XML 문서의 저장 및 질의는 현재 가용한 관계형 데이터베이스를 그대로 이용할 수 있고, 질의 처리는 효율적이어야 한다는 데에 설계 목표를 두고 있다. 본 논문에서 다루는 XML 문서에 대한 질의는 선형으로 표시되는 단순한 XPath[XML] 형태의 질의에 국한하며, 트리 형태를 띠는 Twig 질의[12]에 대한 처리 방법은 여기서는 다루지 않기로 한다.

2. 관련 연구

2.1 XML 문서 저장 기법

기존의 데이터베이스 시스템, 즉 관계형 데이터베이스 시스템 기반의 XML 문서 저장 기법들은 XML 데이터 모델과 기존 데이터베이스 시스템에서 지원하는 데이터 모델이 일치하지 않는다는 단점은 있으나, 기존 관계형 데이터베이스 시스템이 제공하는 여러 저장 기술을 그대로 사용할 수 있고 또한 다양하고 효율적인 질의 처리 방식을 이용하여 대용량의 XML 문서들을 처리할 수 있다는 장점을 가지고 있어서 최근 활발히 연구되고 있다[10, 17].

일반적으로 XML 문서에 대하여 관계 데이터베이스 스키마를 설계하는 방법은 구조 매핑(structure-mapping) 방법과 모델 매핑(model-mapping) 방법으로 분류된다. 구조 매핑 방법은 한정된 숫자의 문서구조나 DTD(Document Type Definition)를 갖는 대량의 XML 문서를 저장할 경우와 문서 구조나 DTD가 비교적 정적일 때에 적합하다. 그러나 수많은 정교한 웹 애플리케이션들은 융통성 있고 동적인 XML 사용을 전제로 하고 있다. 즉, 여러 종류의 XML 문서를 저장할 수 있어야 하고 논리적 구조가 자주 바뀌는 XML 문서를 다룰 수 있어야 한다. 구조 매핑 방법은 이렇듯 동적이고 구조적으로 변하는 대량의 XML 문서를 저장하기에는 적합하지 않다. 모델 매핑 방법은 DTD에 관한 정보 없이 하나의 고정된 데이터베이스 스키마가 모든 XML 문서의 구조를 저장하는 데 사용될 수 있다[20, 22]. 따라서 본 연구에서는 XML 문서 저장 방법 중 구조적으로 변하는 대량의 XML 문서를 저장하기에 적합한 모델 매핑 방법을 이용한다.

2.2 XML 문서 처리 기법

XML 질의를 위해 경로 표현식을 처리하는 기법들은 크게 2가지, 스키마-레벨 방법과 인스턴스-레벨 방법으로 구분할 수 있다[3]. 스키마-레벨 방법은 경로표현식과 매치하는 노드들을 찾기 위해 레이블 경로와 같은 구조적인 정보를 사용하는 방법을 가리키며, 그 중 경로들을 저장하기 위해 관계형 테이블을 사용하는 방법으로는 XRel[[22], Xparent[11], XIR-Linear[3] 등이 있으며, 특별한 목적의 인덱스를 사용하는 방법으로는 Index Fabric[16], APEX[8] 등이 있다. 인스턴스-레벨 방법은 노드 구분 정보만을 사용하는 방법을 가리키며, Element Numbering Scheme[13] 등이 여기에 속한다.

관계형 테이블을 사용하는 스키마 레벨 방법 중 제안 방법의 기초가 되는 XRel에서는 루트로부터 각 노드까지의 경로들을 열거하고, 경로 표현식을 관계형 테이블의 한 속성으로 저장하며, 이들 정보와 영역정보를 조합하여 트리 구조를 표현한다. 모든 가능한 경로 표현식들이 데이터베이스에 스트링으로 저장되어 있어서, 스트링 매칭으로 질의를 처리할 수 있다.

대규모의 이질적인 XML 문서들에 대한 효율적인 질의 처리를 위해 제안된 방법인 XIR-Linear에서는 관계형 테이블을 이용한 스키마 레벨 방법에 기반을 두되, 대용량의 정보를 효율적으로 검색하기 위해 이용하는 역 인덱스 기술을 이용함으로써 질의 처리의 효율성을 향상시켰다. 레이블 경로를 텍스트로 간주하고, 레이블 경로 내의 레이블들을 텍스트 내에 있는 키워드로 간주한 후, 정보검색기술을 이용하여 레이블들을 인덱싱함으로써, 기존의 스트링 매치보다 효율적인 방법으로 레이블 경로들을 찾는다.

경로 정보를 이용한 기존의 기법들에서는 모든 엘리먼트 노드들에 대해 경로 정보를 저장함에 따라 정보의 양이 증가하여 질의 처리의 성능을 저하시키는 요인이 되었다. 이 문제를 해결하기 위해 우리는 경로 정보 중 가장 긴 단말 엘리먼트 노드까지의 경로인 단말 엘리먼트 경로만을 저장하고 단말 엘리먼트 경로만을 대상으로 하여 역

인덱스를 구성함으로써 질의 처리의 성능을 향상시킨 바 있다[5]. 본 연구에서도 관계형 테이블을 사용하는 스키마-레벨 방법에 기반을 두면서 역 인덱스 기술을 이용하고 있다.

2.3 XML 문서의 순서 부여 방법

XML 문서의 각 노드의 순서를 인코딩하는 방법은 순서가 매겨진 XML 문서를 재구성하기에 충분하도록 번호를 붙이는 방법으로, Tatarinov [19]등은 관계형 데이터 모델에서 XML 순서를 표현하는 데 사용될 수 있는 순서 인코딩 방법을 전역 순서 코딩(Global order encoding), 지역 순서 코딩(Local order encoding), 듀이 순서 코딩(Dewey order encoding), 같은 이름의 형제노드 순서 코딩(Same sibling order encoding)으로 구분하여 제안하고 있다.

전역 순서 코딩은 질의 위주 작업에 최적이고, 지역 순서 코딩은 갱신 위주 작업에 최적이며, 듀이 순서 코딩은 질의와 갱신이 혼합된 작업에 최적이다. 같은 이름의 형제 노드들 사이의 순서 코딩은 같은 이름을 가진 형제노드에 대해 부분적으로 순서를 코드화하는 것으로, 다른 3가지 방법들에서 보완적으로 사용될 수 있다[19]. 본 연구에서는 듀이 순서 코딩을 기초로 부분적으로 보완한 데이터 모델에 기반을 두고 있다.

3. 세그먼트 기반 경로 탐색

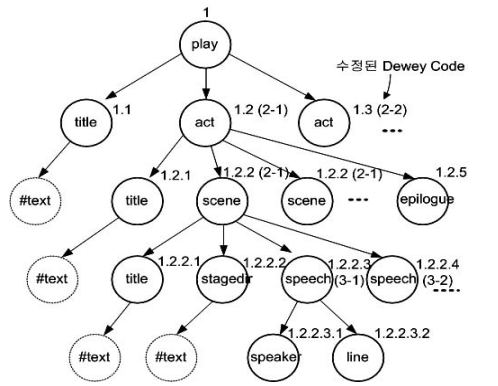
3.1 XML 문서의 데이터 모델

본 논문에서 XML 문서를 표현하기 위해 사용한 데이터 모델은 [5]에서 사용한 모델-매핑을 이용한 방법과 동일하다. XML 문서의 각 노드의 순서를 코드화하는 방법은 질의와 갱신이 혼합된 작업에 최적인 듀이 순서 코딩 방법을 기본으로 하되, 같은 이름의 형제 노드들 간의 순서도 함께 나타낼 수 있도록 수정하였다. [그림 1]은 각 절의

```
<play>
  <title> Hamlet </title>
    <act>
  <title> ACT I </title>
      <scene>
        <title> SCENE I. Elsinore. … </title>
          <stagedir> FRANCISCO at his
                     post … </stagedir>
          <speech>
            <speaker> BERNARDO </speaker>
            <line> Who's there? </line>
          </speech>
          <speech>
          <speaker> FRANCISCO </speaker>
            <line> Nay, answer me : stand,
                   and … </line>
          </speech>
         …
      </scene>
      …
      <epilogue>
      …
      </epilogue>
    </title>
  </act>
  <act>
  …
  </act>
</play>
```

[그림 1] 샘플 XML 문서

설명에서 사용하고 있는 XML 문서의 예이고 [그림 2]는 본 연구에서 사용하고 있는 수정된 듀이 번호로 각 노드의 *nodeID*로 부여하여 트리구조로 나타낸 것이다.



[그림 2] 수정된 듀이번호를 부여한 XML 트리 예

3.2 세그먼트 기반 인덱스 구성 및 경로 탐색

일반적으로 XML 문서에 대한 질의는 경로 표현식을 포함하고 있다. 제안 방법에서도 XML 트리의 분해 단위로서 경로를 사용한다. 경로정보를 저장하는 데 있어서 모든 노드에 대해 루트부터 각 노드까지의 경로 정보를 저장함에 따라 정보의 양이 증가하여 질의 처리의 성능을 저하시키는 문제점을 해결하기 위해, [5]에서는 내부 엘리먼트 노드의 경로들은 저장하지 않고 단말 엘리먼트 경로만을 저장함으로써 경로정보의 양을 줄이고 단말 엘리먼트 경로만을 대상으로 하여 역 인덱스를 구성함으로써 질의 처리의 성능을 향상시키고 있다. 본 연구에서는 단말 엘리먼트의 경로정보에 대해 필요에 따라 엘리먼트들을 몇 개씩 묶어 세그먼트로 구분하고 세그먼트별 역 인덱스 구성을 통하여 질의 처리의 성능을 향상시키고자 한다.

XML 문서의 구조적 정보를 나타내는 각 노드들에 대한 경로정보들은 중복된 데이터를 가지고 있다. <표 1>은 [그림 1]에 나타난 XML 문서에 대하여 모든 경로를 저장한 경로 테이블이다. <표 1>에서 노드 9에 대한 경로정보는 노드 1, 3, 5, 8에 대한 경로정보를 포함하고 있다. <표 2>는 내부 엘리먼트 경로를 제외하고 단말 엘리먼트 경로만을 저장한 경로 테이블이다. <표 2>에서는 <표 1>에 나타난 10개의 경로가 6개로 줄어든 것을 볼 수 있다.

〈표 1〉 모든 경로를 저장한 경로 테이블의 예

번 호	경로 표현식
1	/play
2	/play/title
3	/play/act
4	/play/act/title
5	/play/act/scene
6	/play/act/scene/title
7	/play/act/scene/stagedir
8	/play/act/scene/speech
9	/play/act/scene/speech/speaker
10	/play/act/scene/speech/line

〈표 2〉 단말 엘리먼트 경로만 저장한 경로테이블의 예

번 호	경로 표현식
1	/play/title
2	/play/act/title
3	/play/act/scene/title
4	/play/act/scene/stagedir
5	/play/act/scene/speech/speaker
6	/play/act/scene/speech/line
…	…

〈표 3〉 모든 경로에 대한 역 인덱스 테이블의 예

키워드	포스팅 리스트
play	<1, 1> <2, 1> <3, 1> <4, 1> <5, 1> <6, 1> <7, 1> <8, 1> <9, 1> <10, 1> <2, 2> <4, 3> <6, 4>
title	<3, 2> <4, 2> <5, 2> <6, 2> <7, 2>
act	<8, 2> <9, 2> <10, 2>
scene	<5, 3> <6, 3> <7, 3> <8, 3> <9, 3>
stagedir	<10, 3>
speech	<7, 4>
speaker	<8, 4> <9, 4> <10, 4>
line	<9, 5>
…	<10, 5> …

<표 3>은 샘플 XML 문서의 경로정보에 대한 역 인덱스를 모든 경로를 대상으로 하여 구성한 예이고, <표 4>는 단말 엘리먼트 경로만을 대상으로 하여 구성한 예이다. 역 인덱스를 구성하는 포스팅 리스트(posting list)는 <경로 표현식 번호, 표현식에서의 엘리먼트 위치>의 정보로 나타낸다. 예를 들면 'play' 키워드에 대한 포스팅 리스트의 하나인 <1, 1>은 1번 경로표현식에서 'play' 키워드가 첫 번째 나타난다는 의미이고, <2, 1> 역시 'play' 키워드가 2번 경로표현식에서 첫 번째로 나타난다는 의미이다. 모든 노드들에 대한 경로 정보를 이용한 역 인덱스 파일인 <표 3>에서는 'play'키워드의 포스팅 리스트는 모두 10개이지만, 단말 노드의 경로 표현식을 사용하는 역 인덱스 파일인 <표 4>에서는 10개가 6개로 줄어든 것을 볼 수 있다. 8가지 키워드에 대한 포스팅 리스트의 평균 개수가 모든 경로를 대상으로 한 <표 3>에서는 4.12개이지만, 단말 엘리먼트 경로만을 대상

으로 한 <표 4>에서는 2.87개로 줄어들었다. 이와 같은 역 인덱스 파일의 중복된 정보 제거를 통하여 대용량의 XML 문서에 대한 역 인덱스의 구성 시 인덱스 파일의 크기를 줄여 인덱스 검사의 성능을 향상시킬 수 가 있다.

<표 5>는 단말 엘리먼트 경로만을 대상으로 하되, 엘리먼트들을 2개씩 묶어 세그먼트화하여 인덱스를 만든 예이다. 예를 들어, 'play/act/scene/speech'경로를 찾고자 할 때, 단말 엘리먼트 경로만을 대상으로 하여 역 인덱스를 구성한 <표 4>에서는 4개의 엘리먼트(키워드) 각각에 대하여 즉, 4회에 걸쳐 인덱스를 검색해야 하지만, 세그먼트별 역 인덱스를 구성한 <표 5>에서는 2개의 키워드 'play/act'와 'scene/speech'에 대하여, 즉 2회의 인덱스 검색만 실시하면 되므로 인덱스 검색 횟수를 반으로 줄일 수 있다. 필요에 따라 세그먼트를 3개 또는 4개의 엘리먼트로 구성할 경우, 인덱스 검색 횟수를 최대 3분의 1 또는 4분의 1로도 줄일 수 있다. 물론 세그먼트별 인덱스를 추가로 구성하는 것은 필요하다. 그러나 세그먼트 기반 경로 탐색 기법은 질의 처리 시에 인덱스를 검색하는 횟수를 줄임으로써 질의 처리의 성능을 향상시키게 된다.

〈표 4〉 단말 엘리먼트 경로에 대한 역 인덱스 테이블의 예

세그먼트(키워드)	포스팅 리스트
play/title	<1, 1>
play/act	<2, 1> <3, 1> <4, 1> <5, 1>
act/title	<6, 1>
act/scene	<2, 2>
scene/title	<3, 2> <4, 2> <5, 2> <6, 2>
scene/stagedir	<3, 3> <4, 3> <5, 3> <6, 3>
scene/speech	<4, 3>
speech/speaker	<5, 3> <6, 3>
speech/line	<5, 4>
…	<6, 4>
	…

〈표 5〉 단말 엘리먼트 경로에 대한 세그먼트별 역 인덱스 구성의 예

키워드	포스팅 리스트
play	<1, 1> <2, 1> <3, 1> <4, 1> <5, 1> <6, 1>
title	<1, 2> <2, 3> <3, 4>
act	<2, 2> <3, 2> <4, 2> <5, 2> <6, 2>
scene	<3, 3> <4, 3> <5, 3> <6, 3>
stagedir	<4, 4>
speech	<5, 4> <6, 4>
speaker	<5, 5>
line	<6, 5>
…	…

4. XML 문서에 대한 질의 처리

4.1 테이블 스키마

제안 방법에서 단말 엘리먼트 경로에 대한 정보 및 각 노드에 대한 정보를 저장하기 위한 관계형 스키마는 다음과 같다. 기본적인 구성은 [이혜자, 2005]에서 사용한 스키마와 동일하고, 단말 엘리먼트 경로에 대한 역 인덱스(**PathIndex**) 테이블을 구성하는 키워드가 <표 4>와 같은 단일 엘리먼트 뿐만 아니라 <표 5>와 같이 세그먼트도 포함하고 있다는 점만 차이가 있다.

Element (*docID, nodeID, lnpID, level, name*)
Text (*docID, nodeID, lnp_ID, level, value*)
Attribute (*docID, nodeID, lnpID, level, name, value*)

Path (lnp*ID, pathExp*)
PathIndex(*keyword, {lnpID, level}*)

각 테이블에서, *docID*와 *nodeID*은 각각 XML 문서 번호, 문서내의 노드 번호(즉, 수정된 듀이 코드 번호), *lnpID*와 *level*은 단말 노드 경로 표현식 번호, XML 트리에서의 노드 레벨을 나타내고, *name*은 엘리먼트 혹은 속성(attribute)의 이름, *value*는 속성의 값 또는 엘리먼트 태그안의 텍스트 값을 의미한다. XML 문서 정보를 담고 있는 **Element**, **Text**, **Attribute** 테이블들에서는 *docID*와 *nodeID*들이 키 값을 나타내게 된다. 텍스트 정

보를 나타내는 **Text** 테이블에서 *nodeID*는 부모 노드인 엘리먼트 노드의 *nodeID* 값과 항상 같으며 별도의 노드 번호를 부여하지 않는다. 그 이유는 XML 문서에서 텍스트 값은 항상 하나의 엘리먼트 태그와 짝을 이루게 되기 때문이다.

역 인덱스 파일을 구성하는 **Path**와 **PathIndex** 테이블에서는 *lnpID*와 *pathExp*는 각각 단말 노드 경로 표현식의 번호와 스트링 형태의 표현식을 나타내고, *{lnpID, level}*는 포스팅 리스트로 경로 표현식 번호와 해당 키워드(혹은 세그먼트)의 위치 값을 나타낸다.

4.2 질의 처리 알고리즘

경로 표현식으로 표현되는 XPath 질의를 처리하기 위한 세그먼트 기반 알고리즘은 [그림 3]와 같이 4단계로 구성된다. 알고리즘 설명 및 실험의 편의를 위해 세그먼트는 2개의 엘리먼트로 구성되어 있는 것으로 가정한다. 역 인덱스 테이블에는 기존의 1개의 엘리먼트별로 키워드를 구성하여 만든 인덱스와 2개의 엘리먼트로 구성된 세그먼트를 키워드로 하여 만든 인덱스가 함께 포함되어 있다. 1개의 엘리먼트로 만들어진 키워드도 2개의 엘리먼트로 구성된 키워드와 마찬가지로 여기서는 모두 세그먼트로 칭한다.

단계 1) 경로 표현식에서 세그먼트를 추출한다.

단계 2) 세그먼트별로 PathIndex 테이블에서 세그먼트 이름과 레벨이 같은 단말 엘리먼트 경로 집합을 찾는다.

단계 3) 단계 2)에서 찾아진 단말 엘리먼트 경로 집합 모두에서 나타나는 단말 엘리먼트 경로만을 걸러낸다.

단계 4) Element/Attribute/Text 테이블에서 경로 값과 레벨이 일치하는 노드 집합을 찾는다. 만약 텍스트 노드를 찾는 경우에는 텍스트 값도 동시에 일치하는 노드 집합을 찾는다. 여기서 찾아진 노드 집합이 최종 결과 값이 된다.

[그림 3] 세그먼트 기반 질의 처리 알고리즘

단계 1)에서는 경로 표현식을 레벨 1에서부터 시작하여 2개의 엘리먼트로 구성된 세그먼트로 분할한다. 경로 표현식이 홀수 개로 구성된 경우에는 마지막 남은 엘리먼트 하나는 그것만으로 세그먼트를 만든다. 다음 단계 2)에서는 각 세그먼트별로 레벨이 맞는 경로들을 역 인덱스 테이블에서 찾아서 세그먼트별 단말 엘리먼트 경로(*lnpID*)의 집합을 구한다. 이 때 비교하는 레벨은 세그먼트에서 첫 번째 나타나는 엘리먼트의 레벨이다. 이 때 만들어지는 경로 집합의 개수는 단계 1)에서 만들어진 세그먼트의 개수와 같다.

각 세그먼트별로 레벨이 맞는 경로들을 찾는 SQL 문장은 엘리먼트가 세그먼트로 변경된 것만 제외하고 [13]에서 사용한 것과 동일하다. 경로 표현식의 각 세그먼트별로 레벨이 맞는 경로 집합을 역 인덱스(**PathIndex**) 테이블에서 찾은 다음, 단계 3)에서는 세그먼트별로 찾아진 경로 집합들에서 모든 집합에서 나타나는 경로들만을 찾아낸다. 이렇게 찾아진 단말 엘리먼트 경로(*lnpID*)들이 찾고자 하는 목표 경로 값들이 된다.

목표 경로 값들(*lnpIDs*)을 찾은 다음, 단계 4)에서는 목표 경로 값과 경로 표현식에서의 레벨, 그리고 텍스트 값이 있는 경우에는 텍스트 값까지 일치하는 노드 집합을 **Element, Attribute, Text** 테이블에서 찾는다. 이 때 각 테이블에서 해당 노드를 찾는 SQL 문장은 다음 절의 질의 처리 예제에 잘 나타나있다.

4.3 질의 처리 예제

다음과 같은 일반적인 정규 경로식으로 표현된 질의 예제 Q1을 처리하는 경우를 예를 들어 질의 처리 과정을 살펴본다. 앞서 알고리즘 설명에서도 언급하였듯이 세그먼트별 역 인덱스는 엘리먼트를 2개씩 묶어 구성한 경우를 예로 한다.

예제 1 (Q1) : /PLAY/ACT/SCENE/SPEECH

먼저, 경로에 대한 역 인덱스 테이블에서, 세그먼트 PLAY/ACT, SCENE/SPEECH 각각에 대하여 레벨이 맞는 경로 값들을 찾아서 경로 값의 집합들을 만든다. Q1에서 레벨은 PLAY부터 순서대로 1, 2, 3, 4로 부여되며, 비교하는 레벨은 세그먼트에서 첫 번째 나타나는 엘리먼트의 레벨이다. Q1 질의에서는 비교해야 할 세그먼트가 2개이므로 만들어지는 경로 집합의 개수도 2개이다. 이를 SQL 문장으로 표현하면 다음과 같다.

```
SELECT   lnpID
FROM     PathIndex
WHERE keyword = 'PLAY/ACT' and level
                = 1;

SELECT   lnpID
FROM     PathIndex
WHERE keyword = 'SCENE/SPEECH' and
                level = 3;
```

다음은, 앞 단계에서 만들어진 2개의 경로 값 집합에서 경로 값들이 모두 나타나는 경우의 경로 값들만을 찾아냄으로써 목표 경로 값을 정한다. <표 5>와 같은 역 인덱스 파일일 경우 {<2, 1>, <3, 1>, <4, 1>, <5, 1>, <6, 1>} 과 {<5, 3>, <6, 3>}의 두 집합이 SQL 문장의 수행 결과이고 목표 경로 값들은 {5, 6}이 된다. 그 다음 단계에서는, 목표 경로 값들과 최대 레벨 값(Q1에서는 4)을 이용하여 **Element** 테이블에서 해당 노드들을 찾는다. 해당 경로 표현식에 사용된 텍스트를 찾을 경우에는 **Text,** 속성 값 속성(attribute) 값을 찾을 경우에는 **Attribute** 테이블에서 필요한 값들을 찾으면 될 것이다.

```
SELECT   docID, nodeID, name
FROM     Element
WHERE    (lnpID = 5 and level = 4) or
         (lnpID = 6 and level = 4);
```

세그먼트별 역 인덱스를 구성하지 않는 기존의 방법에서는 4개의 엘리먼트(PLAY, ACT, SCENE, SPEECH) 각각에 대하여 인덱스를 검색해야 하므로 4회의 인덱스 검색이 필요하지만, 제안방법에서는 2개의 세그먼트(PLAY/ACT, EPILOGUE/SPEECH)에 대하여 인덱스를 검색하므로 인덱스 검색 횟수를 2회로 줄일 수 있다.

다음은 연산자 '//'가 있는 질의 예제 Q2를 처리하는 경우를 예로 들어 살펴본다.

예제 2 (Q2) : /PLAY/ACT//SPEECH

Q2 질의에서도 세그먼트별 역 인덱스를 구성하지 않는 기존의 방법에서는 3개의 엘리먼트(PLAY, ACT, SPEECH) 각각에 대하여 인덱스를 검색해야 하므로 3회의 인덱스 검색이 필요하지만, 제안방법에서는 2개의 세그먼트(PLAY/ACT, SPEECH)에 대하여 인덱스를 검색하므로 인덱스 검색 횟수를 2회로 줄일 수 있다. 제안방법에서는 연산자 '//'가 있는 Q2에서도 '//'가 없는 Q1에서와 마찬가지로 같은 알고리즘으로 처리할 수 있다. 다만 '//' 이후에 나타나는 세그먼트의 경우 레벨의 정확한 값을 알 수 없으므로 레벨이 같거나 큰 경우에 해당하는 것으로 처리해야 하는 부분만 차이가 있다.

```
SELECT   lnpID, level
FROM     PathIndex
WHERE    keyword = 'SPEECH' and level
                ≥ 3;
```

예제 3 (Q3) : /PLAY/ACT[4]

같은 이름을 가진 형제노드간의 관계를 알아야 하는 질의에서는 Q1과 같은 방식으로 처리하되, 찾아진 경로 값들과 경로 표현식의 레벨을 이용하여 엘리먼트 테이블에서 해당 노드들을 찾을 때, 노드 번호 즉, *nodeID*를 비교하는 부분만 추가하면 된다. 제안 방법의 노드 번호에는 문서 내에서

의 노드 순서와 이름이 같은 형제노드의 순서가 같이 포함되어 있기 때문이다.

속성(attribute)의 값을 찾거나 텍스트 값을 찾는 질의에서도 Q1과 같은 방식으로 처리하되, 찾아진 경로 값들과 경로 표현식의 레벨을 이용하여 해당 노드들을 찾을 때, **Attribute** 테이블이나 **Text** 테이블의 노드 번호를 비교하는 부분만 추가하면 된다.

연산자 '//'가 있으면서 텍스트 값을 찾는 복잡한 질의에서도 연산자 '//'가 있으면서 단순한 질의인 Q2과 같은 방식으로 처리하되, 찾아진 경로 값들과 경로 표현식의 레벨을 이용하여 **Text** 테이블에서 필요한 정보를 찾을 때, **Text** 테이블의 노드 번호(*nodeID*)를 비교하는 부분만 추가하면 된다.

5. 실험 및 분석

5.1 실험 설정

본 연구에서 실험을 위해 사용한 하드웨어는 Pentium4 3GHz, 512RAM이고, 운영체제는 Windows XP, 데이터베이스 관리 시스템은 MS SQL-Server 2000이다. 프로그래밍 언어는 Java와 Visual Basic을, XML Parser는 APACHE Xerces2를 사용하였다.

실험용 데이터는 본 실험을 위해 가상으로 생성한 XML 문서의 집합으로, 신체구성, 병력, 의학검사, 체력측정 등 10가지 종류의 건강정보를 나타내는 XML 문서들을 사용하였다. 문서의 개수는 총 12,000개 이며, XML 문서의 구조적 형태는 문서의 종류마다 다양하여, XML 트리의 깊이가 깊고 옆으로는 적은 형태, 깊이는 낮고 옆으로는 많은 형태, 깊이나 넓이가 비슷한 형태 등을 포함하고 있으며, 실험 데이터의 상세 내용은 <표 6>과 같다.

성능 평가를 위해 사용한 질의 예제는 대표적인 XPath 질의로, <표 7>와 같이 6가지이다. 본 실험에서는 앞에서 언급하였듯이 선형으로 표시되는 단순한 XPath 질의 외의 복잡한 Twig 질의에 대해서는 다루지 않았다.

성능 평가의 방법은 기존의 모든 경로를 저장하여 역 인덱스를 만드는 방법과 단말 엘리먼트 경로(LNP)만 저장하여 역 인덱스를 만드는 방법, 그리고 본 연구에서 제안하는 단말 엘리먼트 경로만을 저장하되 세그먼트를 기반으로 역 인덱스를 구성하는 방법을 비교하는 방식으로 실시하였다. 성능에 영향을 미치는 경로의 수와 역 인덱스의 포스팅 리스트의 수, 그리고 역 인덱스의 크기를 비교하고, XML 문서 수의 증가에 따라 실제 질의 처리 시간의 변화를 측정하고 비교하였다.

경로를 이용하는 대표적인 방법인 XRel과 성능을 비교하는 것은 역 인덱스를 이용하는 XIR-Linear[8]에서 이미 실험을 통하여 대용량일수록 역 인덱스를 이용하는 것이 성능 측면에서 우수함이 확인되었기에 여기서는 실시하지 않았다.

〈표 6〉 실험에 사용된 XML 문서에 대한 상세정보

구 분	가상 데이터 (다양한 형태의 건강정보)
문서의 종류	10
문서의 개수	12,000
엘리먼트 노드의 수	529,200
텍스트 노드의 수	382,800
애트리뷰트 노드의 수	31,200
문서당 평균 엘리먼트 수	44.1
문서당 평균 텍스트 수	31.9
문서당 평균 애트리뷰트 수	2.6

〈표 7〉 성능 평가를 위해 사용한 질의 예제

경로 표현식
Q1 : FitnessMeasurement/Member/FM_HealthRelatedFitness/VO2max
Q2 : /MedicalHistory//MH_Present/Disease
Q3 : BodyComposition/Member[3]/BC_Present/PercentBodyFat
Q4 : /Members/Member/[@MemberID = '2004010001']
Q5 : ExercisePrescription/Member/EP_SubjectGroup/[Level3Class = 'Obese']

5.2 실험 결과

문서의 개수가 12,000개일 때, 경로의 수는 모든 경로를 저장하여 역 인덱스를 만드는 방법에서는 348개였으나, 단말 엘리먼트 경로만 저장하여 역 인덱스를 만드는 방법과 본 연구에서 제안하는 세그먼트별 역 인덱스를 구성하는 방법에서는 243개로 약 30.2% 정도 줄일 수 있었다.

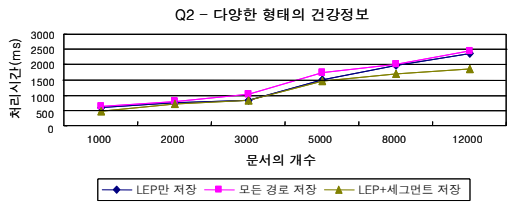
질의 처리 시 비교해야 하는 역 인덱스의 포스팅 리스트의 개수는 <표 8>에서 볼 수 있듯이 본 연구에서 제안하는 세그먼트별 역 인덱스를 구성하는 방법이 다른 방법들에 비해 많이 줄일 수 있었다. 문서의 개수가 12,000개일 때, 기존의 모든 경로를 저장하여 역 인덱스를 만드는 방법에서는 평균 276.5개를 비교해야 하고, 단말 엘리먼트 경로만 저장하여 역 인덱스를 만드는 방법에서는 평균 197개를 비교해야 하는 데 반해, 제안 방법에서는 평균 26.83개만 비교하면 됨에 따라 기존의 2가지 방법들에 비해 약 91.3%, 86.4%를 줄일 수 있었다.

〈표 8〉 질의 처리 시 비교해야 하는 역 인덱스의 포스팅 리스트의 개수

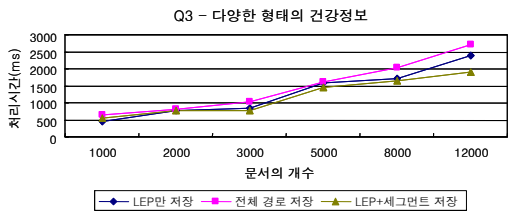
질의 예제	질의 처리 시 비교해야 하는 역 인덱스의 포스팅 리스트의 개수 (문서의 수가 12,000개일 때)		
	모든 경로 저장	LEP만 저장	LEP+세그먼트 저장(제안 방법)
Q1	370	264	16
Q2	63	45	34
Q3	355	256	9
Q4	354	254	12
Q5	454	318	54
Q6	63	45	36
합계	1,659	1,182	161
평균	276.5	197	26.83

한편, 역 인덱스의 크기(레코드의 수)는, 문서의 개수가 12,000개일 때, 기존의 모든 경로를 저장하여 역 인덱스를 만드는 방법과 단말 엘리먼트 경로만 저장하여 역 인덱스를 만드는 방법에서는 132개인 반면, 제안 방법에서는 338개로, 기존의 2가지 방법들에 비해 약 2.56배 증가하였다. 그러나 인덱스의 크기는 데이터의 크기에 비해 상대적으로 작기 때문에 질의 처리의 성능에는 큰 영향을 미치지 않았다.

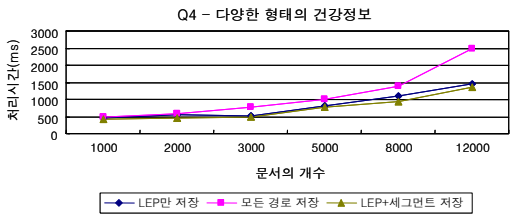
[그림 4]은 XML 문서의 개수의 증가에 따른 질의 처리 시간의 변화를 보여주고 있다. 기존의 모든 경로를 저장하여 역 인덱스를 만드는 방법과



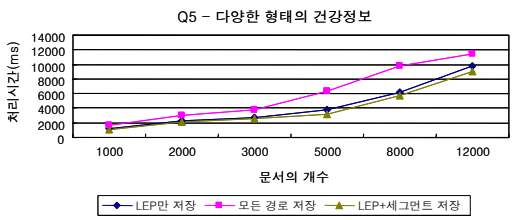
(a) Q2 질의처리시간의 변화



(b) Q3 질의처리시간의 변화



(c) Q4 질의처리시간의 변화



(d) Q5 질의처리시간의 변화

[그림 4] 문서 개수의 증가에 따른 질의처리시간의 변화

단말 엘리먼트 경로만 저장하여 역 인덱스를 만드는 방법과 비교 시, 제안 방법에서는 실험을 한 모든 질의 예제에서 더 좋은 성능을 보이고 있다. 대부분의 경우 문서의 수가 많아질수록 성능의 차이는 커지는 경향을 보였다.

6. 결 론

본 논문에서는 XML 문서에 대한 효율적인 질의처리를 위해 기존의 경로 기반 기법을 좀 더 보완할 수 있는 방법을 제안하였다. 제안 방법에서는 경로 정보 중 반드시 필요하지 않은 내부 엘리먼트 경로들은 제외하고 단말 엘리먼트 경로만을 이용하여 역 인덱스를 구성하되, 단말 엘리먼트 경로를 구성하는 엘리먼트들을 필요에 따라 몇 개씩 묶어 세그먼트화 하여 역 인덱스를 구성함으로써, 기존의 역 인덱스 이용 기법들에 비해 인덱스 검색의 회수를 최소화하여 대용량 XML 문서 집합에서의 검색 성능을 향상시키고 있다.

제안 방법은 상용 관계형 데이터베이스의 타입이나 기능의 확장은 필요하지 않으나 질의처리를 위한 별도의 인덱싱은 필요하다. 향후 연구에서는 XML 문서의 갱신에 대한 지원이 포함되어야 할 것이고, XML 문서 변경 시의 성능비교에 대해서도 추가 실험이 필요하다. 또한 최근 XML 질의처리 기법에서 많이 다루고 있는 경로 조인 기법들과의 성능 비교가 이루어져야 하며, 선형으로 표시되는 단순한 XPath 질의 외에 트리 형태를 띠는 Twig 질의에 대한 처리 방법에 대해서도 향후 추가 연구가 필요하다.

참 고 문 헌

[1] 민경섭, 김형주, “상이한 구조의 XML 문서들에서 경로 질의 처리를 위한 RDBMS 기반 역 인덱스 기법”, 「정보과학회논문지 : 데이터베이스」, 제30권, 제4호(2003), pp.420-428.

[2] 민준기, 박명제, 안재용, 정진완, “다양한 저장소에서의 효율적인 XML 저장기법에 대한 연구”, 「정보과학회 데이터베이스연구」, 제19권, 제1호(2003), pp.1-14.

[3] 박영호, 한욱신, 황규영, “정보 검색 기술을 이용한 대규모 이질적인 XML 문서에 대한 효율적인 선형 경로 질의 처리”, 「정보과학회 논문지 : 데이터베이스」, 제31권, 제5호(2004), pp.540-552.

[4] 배진욱, 문봉기, 이석호, “빠른 XML 질의 처리를 위한 세그먼트 조인 기법”, 「정보과학회 논문지 : 데이터베이스」, 제32권, 제3호(2005), pp.334-343.

[5] 이혜자, 정병수, 김대호, 이영구, “경로정보의 중복을 제거한 XML 문서의 저장 및 질의처리 기법”, 「정보처리학회논문지D」, 제12-D권, 제5호(2005), pp.663-672.

[6] M. G. Bauer, F. Ramsak, and R. Bayer, “Multidimensional Mapping and Indexing”, In Proceedings of BTW Conf., 2003.

[7] N. Bruno, N. Koudas, and D. Srivastara, “Holistic Twig Joins : Optimal XML Pattern Matching”, In Proceedings of ACM SIGMOD Conf., 2002.

[8] V. Christophides, S. Abiteboul, S. Cluet, and M. Scholl, “From Structured Documents and to Novel Query Facilities”, In Proceedings of ACM SIGMOD Conf., 1994.

[9] C. Chung, J. Min, and K. Shim, “APEX : An Adaptive Path Index for XML Data”, In Proceedings of ACM SIGMOD Conf., 2002.

[10] B. F. Cooper, N. Sample, M. J. Franklin, G. R. Hjaltason, and M. Shadmon, “A Fast Index for Semi-structured Data”, In Proceedings of VLDB Conf., 2001.

[11] D. Florescu and D. Kossmann, “Storing and Querying XML Data Using an RDBMS”, *IEEE Data Engineering Bulletin*,

Vol.22, No.3(1999), pp.27-34.

[12] H. Jiang, H. Lu, W. Wang, and J. Yu, "XParent : An Efficient RDBMS-Based XML Database System", In Proceedings of ICDE Conf., 2002.

[13] H. Jiang, W. Wang, H. Lu, and J. X. Yu, "Holistic Twig Joins on Indexed XML Documents", In Proceedings of VLDB Conf., 2003.

[14] Q. Li and B. Moon, "Indexing and Querying XML Data for Regular Path Expression", In Proceedings of VLDB Conf., 2001.

[15] J. McHugh, S. Abiteboul, R. Goldman, D. Quass, and J. Widom, "Lore : A Database Management System for Semistructured Data", *SIGMOD Record*, Vol.26, No.3(1997), pp.54-66.

[16] S. Pal, I. Cseri, O. Seeliger, G. Schaller, L. Giakoumakis, and V. Zolotov, "Indexing XML Data Stored in a Relational Database", In Proceedings of VLDB Conf., 2004.

[17] J. Shanmugasundaram et al., "Relational Databases for Querying XML Documents : Limitation and Opportunities", In Proceedings of VLDB Conf., 1999.

[18] H. Schoning, "Tamino-a DBMS Designed for XML", In Proceedings of IEEE ICDE Conf., 2001.

[19] S. Sundara, Y. Hu, T. Chorma, and J. Srimivasan, "Developing an Indexing Scheme XML Document Collections Using the Oracle8i Extensibility Framework", In Proceedings of VLDB Conf., 2001.

[20] 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 Proceedings of ACM SIGMOD Conf., 2002.

[21] F. Tian, D. J. Dewitt, J. Chen, and C. Zhang, "The Design and Performance Evaluation of Alternative XML Storage Strategies", *SIGMOD Record*, Vol.31, No. 1(2002), pp.5-10.

[22] XML Path Language (XPath) 2.0, http://www.w3c.org/TR/2003/WD-xpath20-20031112.

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

저 자 소 개

정 병 수 (jeong@khu.ac.kr)

서울대학교 공과대학 컴퓨터공학과에서 학사, 한국과학기술원 전산학과에서 전산학 석사, Georgia Institute of Technology, College of Computing에서 박사 학위를 취득하고, 경희대학교 전자정보대학 컴퓨터공학 전공에서 부교수로 재직 중이며, 현재 웹 로그 분석, 스트림 데이터 마이닝, 패턴 탐색 기법 등을 연구 중이다. 관심분야는 데이터 마이닝, 모바일 데이터베이스, 플래쉬 메모리 저장시스템 등이다.

이 혜 자 (hjlee@ysc.ac.kr)

서울대학교 간호학과에서 학사, 연세대학교 산업대학원 전산전공에서 석사, 경희대학교 대학원 컴퓨터공학과에서 박사 학위를 취득하고, 용인송담대학 의료정보시스템과에서 부교수로 재직 중이며, 현재 건강관리 서비스 모델, 지식관리 기법 등을 연구 중이다. 관심분야는 의료정보 시스템, 정보 모델링, 지식관리, 전문가 시스템, 데이터 마이닝 등이다.