Journal of Korea Multimedia Society Vol. 8, No. 2, February 2005(pp. 154-163)

XML 문서의 저장과 추출을 위한 색인 기법

송정석[†], 김우생[††]

요 약

XML 문서를 위한 현재까지 연구된 대부분의 색인기법에서는 절대좌표를 표현하는 방법을 이용하기 때문에 갱신연산이 커다란 부담으로 받아들여지고 있다. 또한 이 기법에서 XML 문서내의 엘리먼트, 애트리뷰트, 텍스트 사이의 상호 구조적 관계를 표현하려면 좌표를 재구성해야 한다. 이와 같은 재구성 작업은 갱신이 이루어지는 노드에 국한하지 않고 XML 문서 전반에 걸쳐 연쇄적으로 일어나기 때문에, XML 문서의 갱신이 빈번할 경우 심각한 성능 문제를 야기하게 된다. 본 연구에서는 갱신이 빈번한 상황에서도 성능 저하가 많지 않은 확장 색인에 기반한 색인 기법을 제안하고자 한다. 이 방법은 갱신으로 인한 트리의 재구성 연산에 참여하는 노드의 수를 제한하여 전체적으로 성능을 많이 향상시킬 수 있다. 또한 확장색인 기법은 SQL 문장을 이용한 간결한 표현을 통하여 포함관계질의를 처리할 수 있다.

An Index Method for Storing and Extracting XML Documents

Jungsuk Song[†], Woosaeng Kim[††]

ABSTRACT

Because most researches that were studied so far on XML documents used an absolute coordinate system in most of the index techniques, the update operation makes a large burden. To express the structural relations between elements, attributes and text, we need to reconstruct the structure of the coordinates. As the reconstruction process proceeds through out the entire XML document in a cascade manner, which is not limited to the current changing node, a serious performance problem may be caused by the frequent update operations. In this paper, we propose an index technique based on extensible index that does not cause serious performance degradations. It can limit the number of node to participate in reconstruction process and improve lots of performance capacities on the whole. And extensible index performs the containment relationship query by the simple expression using SQL statement.

Key words: XML, Indexing(색인), Extensible Indexing(확장 색인), Containment Query(포함 질의)

1. 서 론

멀티미디어 응용이 기존의 이미지, 오디오, 비디오의 제한된 영역의 단순한 재생에 만족하지 않고, 최근 폭발적으로 증가한 인터넷 보급 및 이를 통한 사용자의 다양한 요구 증대에 따라 인터넷을 통한 멀티미디어 자원의 복합적 활용을 통한 서비스로의 발전으로 나아가고 있다. 최근 W3C (World Wide Web Consortium)의 권고를 통하여 XML[1] 언어가 SGML[2]로부터 파생되어 상호운영성을 유지하면서도 인터넷을 통한 데이터 교환을 제공할 수 있는 표준 언어로서 부상하고 있다.

XML 언어의 풍부한 표현능력을 통하여 많은 애플리케이션이 등장하고 있다. 각종 비즈니스 응용프로그램에서 XML을 이용하여 자료를 표현하려는 추세가 두드러지고 있다. 따라서 XML 문서를 데이터베이스에 효율적으로 저장하고 추출하기 위한 주제

※ 교신저자(Corresponding Author) : 송정석, 주소 : 서울시 중구 장교동 한화빌딩(100-797), 전화 : 02)729-5011, FAX : 02)729-4934, E-mail : songjs@hanwha.co.kr
접수일 : 2004년 3월 26일, 완료일 : 2004년 8월 17일
[†] 정회원, 한화S&C 기술연구소
[††] 정회원, 광운대학교 컴퓨터공학부 교수
(E-mail : kwsrain@cs.kw.ac.kr)

가 꾸준히 연구되어 왔다[3-5]. 아울러 XML 문서내의 내용뿐만 아니라 구조적 정보 및 위치적 정보 또한 보존할 수 있는 방안이 제공되어야 한다.

XML 문서내의 각 노드의 위치 정보를 표현하기 위하여 기존의 많은 연구에서 절대좌표를 이용하는 Position 기반의 색인 기법을 이용하였다. 이들 색인 기법에서는 어디에 데이터가 위치하고 있는가를 지정하기 위하여 절대주소를 이용하기 때문에 갱신이 발생할 경우 노드의 색인 구조를 상당부분 재계산해야 한다. 만일 갱신연산이 빈번한 응용에서 이와 같은 현상이 발생한다면 갱신을 위한 비용은 상당한 부담으로 시스템에 영향을 미치게 될 것이다.

제안하는 시스템의 내용은 다음과 같다. XML 문서내의 각 노드의 위치 정보를 표현하기 위하여 기존 Position 기반의 색인 기법을 개선하여 노드 간의 운영 및 질의를 위한 새로운 확장색인 방법을 적용한다. 상대영역 좌표에 기반한 기존의 연구와 유사하게 오프셋의 대상을 자식노드에 국한하여 자식노드들의 상호 위치를 표현하는 기법을 이용한다. 오프셋과 함께 부모노드의 주소를 위한 값을 할당하고, 오프셋의 크기를 표시하기 위한 헤더를 별도로 구성한다.

기존의 좌표기반 색인기법에서 제공되는 포함질의를 지속적으로 지원하면서도 노드의 갱신시에 다량의 색인 정보가 재계산되는 단점을 완화할 수 있다. 갱신 연산이 발생했을 경우에는 인접 노드의 위치정보에 영향을 미치지 않으면서 노드의 위치 정보를 변경하며, 상대영역 좌표에 의한 방식에 비해서도 적은 양의 정보를 이용하여 좌표를 표현하기 때문에 보다 경제적으로 노드의 위치를 표현할 수 있다.

본 논문의 구성은 다음과 같다. 제2절에서는 관련 연구로서 XML 문서의 색인 기법에 대한 기존의 연구에 대하여 설명한다. 제3절에서는 XML 문서의 위치 정보 표현을 위한 효율적인 확장색인 기법을 소개한다. 제4절에서는 확장색인 기법을 위한 연산들을 제시한다. 제5절에서는 포함질의를 위한 질의예제를 설명한다. 제6절에서는 기존의 좌표기반 색인기법과 제안하는 기법간의 성능평가를 비교한 후, 마지막으로 제7절에서 결론 및 향후 연구과제를 제시한다.

2. 관련연구

XML 문서에 대한 색인 기법은 다양하게 연구되어 왔다. [6]에서는 문서내의 content, structure, attribute에 따른 position 기반의 색인과 path 기반의 색인을 설명하였다.

Position 기반의 색인에서는 단어, 엘리먼트 혹은 애트리뷰트의 오프셋 범위를 다루면서 질의가 처리된다. XML 문서를 이루는 노드를 1차원으로 나열한 후, 등장하는 노드의 순서에 따라서 절대적인 좌표값을 부여하는 방법으로서 해당 노드의 시작점과 종료점의 위치정보를 부여하여 추후에 시작점과 종료점 사이의 영역에 해당하는 값을 비교하여 노드간 포함관계의 형성 여부에 따라 포함관계를 알 수 있는 방법이다. GCL(Generalized Concordance Lists) position 기반의 모델은 [7]에서 제안되었는데, GCL은 extent라고 부르는 텍스트 간격으로 구성된 일치 목록이라고 불리는 자료구조에 기반한다. 각 extent는 시작점과 길이로 기술된다. 위치의 범위를 표현할 수 있기 때문에 포함관계를 표현하는 질의처리가 가능하다.

Path 기반의 색인에서는 단어의 위치가 구조적 요소로서 표현되며, 트리 구조의 경로가 질의처리에 이용된다. 문서내의 단어의 위치를 결정하기 위하여, 루트로부터 단어를 포함하는 리프 노드에 이르는 엘리먼트 이름의 경로를 이용하여 색인을 구성한다. 각 단어에 대한 역화일을 구성하여 단어에 대한 경로의 표현을 구성한다.

[8]에서는 상대영역 좌표에 기반하여 XML 문서의 색인을 위한 자료구조를 제안하였다. 영역좌표는 XML 문서내의 컨텐트 자료의 위치를 표시하며, XML 문서내의 문자열의 시작과 종료를 표현한다. 영역좌표는 색인구조내의 부모노드의 영역좌표에 대응하는 상대적 오프셋에 의하여 결정된다. 갱신연산이 발생하였을 경우에 이웃하지 않은 외부 노드에 영향이 미치지 않음에 따라 절대좌표 기반 방식의 단점을 개선한 연구이다.

[9]에서는 비트맵 색인에 기반한 새로운 기법이 소개되었다. XML 문서는 비트맵 색인기법에 의하여 색인되어 표현된다. 비트맵 색인의 가능한 연산에 대한 동일성과 근접성이 정의되고, XML 문서집합을 분할하는 방법이 제안되었다. BitCube라 불리는 3차원 비트맵 색인이 2차원 비트맵 색인으로부터 확장되어 통계적 수치와 상관계수를 정의한다.

3. 확장색인기법

3.1 확장색인을 위한 테이블 스키마

본 연구에서는 질의 처리의 효율을 극대화하기 위하여 position 기반의 색인 기법을 개선한 확장색인 방식을 적용하여 노드의 갱신에 따른 많은 노드의 색인정보가 재계산되지 않고, 갱신연산이 빈번한 응용에 최적화될 수 있도록 한다.

본 논문에서는 [10]의 연구내용을 확장하여 노드의 위치정보를 저장하기 위하여 그림 1과 같은 테이블 스키마를 이용한다.

위치 정보를 저장하기 위한 스키마는 Element, Attribute, Text, Path 등의 테이블로 구성된다. Element 테이블은 엘리먼트 노드에 관한 정보를 저장한다. 테이블을 구성하는 필드로는 XML 문서의 식별자를 위한 docID, 경로 식별자를 위한 pathID, 현재노드의 부모노드에 대한 주소 정보를 갖는 parentAddr, 헤더 값을 위한 header, 그리고 현재의 형제노드 사이의 오프셋을 위한 offset 등이 있다.

Attribute 테이블은 애트리뷰트 노드에 대한 정보를 저장한다. 구성되는 필드로는 XML 문서의 식별자인 docID, 경로 식별자를 위한 pathID, 애트리뷰트 값을 위한 value, 현재 노드의 부모 노드 주소를 위한 parentAddr, 헤더 값을 위한 header, 그리고 현재의 형제 노드간의 오프셋을 가리키는 offset 등이 존재한다.

Text 테이블은 텍스트 노드에 관한 정보를 저장한다. Text 테이블은 XML 문서의 식별자를 위한 docID, 경로 식별자를 위한 pathID, 텍스트 내용을 위한 value, 현노드의 부모노드의 주소를 위한 parentAddr, 헤더 값을 위한 header, 그리고 현노드의 형제노드간의 오프셋을 위한 offset 등의 필드로 구성된다.

Path 테이블은 Path 기반의 기법을 수용하기 위한 정보를 제공하는 테이블로, XML 노드들의 단순 경로를 저장하기 위한 필드이다. 경로 식별자인 pathID와 단순경로의 내용인 pathExp로 구성된다.

3.2 확장색인 기법의 개념

확장색인 기법을 정의하기 위하여 다음과 같이 가정한다. XML 문서를 D라 하고, r 노드가 루트라고 한다. 다음으로 P는 트리의 노드를 의미하고, C_1, C_2, ..., C_n 각각을 노드 P의 자손 노드들이라고 명명한다.

```
Element (docID, pathID, parentAddr, header, offset)
Attribute (docID, pathID, value, parentAddr, header, offset)
Text (docID, pathID, value, parentAddr, header, offset)
Path (pathID, pathExp)
```

그림 1. 확장색인을 위한 스키마

정의 1. (확장색인) 확장색인은 XML 문서내의 노드간 주소를 표현한다. XML 문서의 형제 노드 C_1, C_2, ..., C_n 중의 하나인 자식노드 C의 주소는 (A_1, A_2)와 같은 형태의 조합에 의하여 표현된다. 이때 A_1은 부모노드의 주소이며, A_2는 자식노드의 오프셋이다. C_1과 C_2는 비트스트링 표현에 의하여 표현된다. 연속되는 비트스트링 A_1 과 A_2는 자식노드의 확장색인을 의미한다.

$$AddressOfChild = AddressOfParent + OffsetOfChild \quad (1)$$

즉, 자식노드의 주소를 표시하기 위한 비트 스트링은 부모 노드의 주소를 위한 비트 스트링으로 시작한다. 확장 색인은 이 부모 노드의 상대적인 영역 내에서 자식 노드의 위치를 나타낸다. 이는 XML 문서의 시작인 루트 노드의 지점으로부터의 거리에 따른 노드의 위치를 나타내는 절대 좌표를 이용하는 색인 기법과는 달리 상대적 영역좌표를 이용한 기법과 비슷하다.

정의 2. (오프셋) 자식노드 {C_1, C_2, ..., C_n} 사이의 오프셋은 그림 2와 같이 A = {000, 001, 010, 011, 100, ..., n-1} 등의 방식으로 이진 비트 형태로 순서에 따라 부여한다.

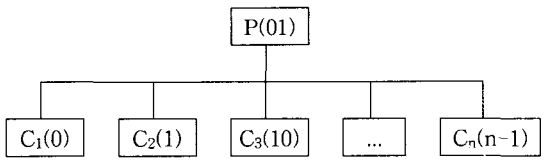


그림 2. 예제 트리와 오프셋 리스트

예제 1. 부모 노드의 주소가 (01)이고, 자식 노드의 개수 n이 5라고 가정했을 때, 자식노드 C_1, C_2, C_3, C_4, C_5의 오프셋은 (000), (001), (010), (011), 그리고 (100)이 된다. 즉 그림 2의 자식노드 C_1, C_2, C_3, C_4,

C_5의 주소는 각각 (01000), (01001), (01010), (01011), 그리고 (01100)이 된다.

정의 3. (헤더) 자식노드들의 오프셋 처리를 위한 비트의 개수를 저장하기 위하여 디렉터리 헤더의 초기치를 할당한다. 헤더의 초기 할당치는 다음의 식에 의하여 결정한다.

$$Header=ceiling(log_2 NumberOfChildNode) \quad (2)$$

예제 2. 예를 들어 부모노드 P의 주소가 '01'이고 P가 3개의 자식노드를 갖는다면, 헤더 값은 식 (2)에 의하여 '2'로 설정된다. 그리고 자식노드를 위한 오프셋 값은 그림 3과 같이 테이블에 저장된다.

Node	Address of P	Header	Offset
C_1	01	2	00
C_2	01	2	01
C_3	01	2	10

그림 3. 헤더와 오프셋의 할당

4. 확장색인 기법을 위한 연산

4.1 노드의 접근

확장색인에 의한 노드의 접근은 궁극적으로 해당 노드를 보유하고 있는 XML 문서의 검색을 위한 것이다. 본 논문에서는 5장의 다양한 포함질의 처리를 통하여 구체적인 과정을 자세히 설명한다.

4.2 노드의 삽입

노드의 삽입에 따른 색인구조의 변경은 그림 4의 알고리즘을 단계별로 적용하여 처리한다.

1. Calculate the location to insert a new node into the sibling nodes.
2. Compare and operate through the header and offset of previous node.
3. If (offset of previous node < $2^{\text{header of previous node}}$ -1)
   3.1. NewOffset=offset of previous node + 1
   3.2..NewHeader=header of previous node
4. If (offset of previous node >= $2^{\text{header of previous node}}$ -1)
   4.1. NewHeader of previous node = OldHeader of previous node + 1
   4.2. NewOffset of previous node = OldOffset of previous node << 1
   4.3. NewHeader of inserted node = Oldheader of previous node + 1
   4.4. NewOffset of inserted node = (OldOffset of previous node << 1)+1

그림 4. 삽입연산을 위한 의사코드 알고리즘

예제 3. 그림 3의 예제에서 C_4 노드가 테이블에 추가되는 경우, 그림 4의 알고리즘에 따라 새로운 헤더 값과 새로운 오프셋 값이 연산되어 그림 5와 같이 새로운 공간에 삽입된다.

Node	Address of P	Header	Offset
C_1	01	2	00
C_2	01	2	01
C_3	01	2	10
C_4	**01**	**2**	**11**

그림 5. 노드 C_4의 삽입 후 테이블 모습

예제 4. 그림 5의 C_4 노드의 추가와 달리, C_5 노드를 연속적으로 삽입하는 경우에는 전위노드의 오프셋이 "$2^{\text{전위노드헤더}}$ - 1"의 결과와 같기 때문에 이전 헤더에 따른 새로운 오프셋을 위한 여유 공간이 충분하지 않기 때문에 삽입되는 노드는 이전 노드와 함께 노드의 분기에 참여한다. C_5 노드의 삽입이 이루어진 후의 모습은 그림 6과 같다.

Node	Address of P	Header	Offset
C_1	01	2	00
C_2	01	2	01
C_3	01	2	10
C_4	**01**	**3**	**110**
C_5	**01**	**3**	**111**

그림 6. C_5의 삽입 후 테이블 모습

4.3 노드의 삭제

본 연구에서는 노드의 삭제를 리프노드에 국한하여 이루어진다는 전제를 이용한다. 그림 7의 알고리

1. Calculate the location to delete a node from the sibling nodes.
2. If target node does not appear, then stop and send the appropriate msg.
3. Compare and operate through the offset of adjacent sibling node.
4. If (the first n-1 bit string of offset data of deleted node where n is the length of offset data = the first n bit string of offset data of adjacent node of deleted node)
   4.1. New offset for adjacent node = first n-1 bit string of former ones.
5. Delete the node.

그림 7. 삭제연산을 위한 의사코드 알고리즘

즘을 통하여 노드의 삭제가 이루어지게 되는데, 먼저 검색을 통하여 노드의 위치를 확인한 후, 해당 노드의 위치를 추적하여 존재하지 않을 경우 오류를 출력한다. 해당 노드가 존재할 경우에는 인접한 형제노드와의 비교를 통하여 알고리즘에 따라서 삭제 연산을 수행한다.

4.4 분할연산 이후 사후처리

좌표기반의 색인기법에서는 갱신이 일어나면 관련된 많은 노드의 위치 정보가 연쇄적으로 변경되지만 확장색인 기법의 경우에는 노드가 분기되는 경우에만 분기되는 노드의 자손노드에 국한하여 위치 정보의 변경작업이 일어난다. 따라서 그림 8과 같은 단계를 통한 작업을 처리해 주어야 한다.

▫ 분기노드의 모든 자손노드가 보유하고 있는 부모 노드 주소에 대하여
- 분기노드의 이전 주소길이 만큼의 상위비트를 제거
- 분기노드의 새로운 주소를 상위비트로서 결합

그림 8. 분할연산 후 처리과정

즉 그림 6의 예제에서는 분기전 '0111'로 시작하던 자손들의 부모노드 필드값을 '01110'으로 대치한다는 뜻이다. 결과적으로 노드의 갱신으로 인한 변경연산을 자손 노드에 국한하여 처리하기 때문에 보다 효율적으로 처리할 수 있다.

5. 질의 처리

본 절에서는 포함질의에 대한 종류를 [11]의 분류와 마찬가지로 직접포함질의, 간접포함질의, 완전포함질의 및 노드간의 근접 정도에 따른 근접질의 등의 분류를 이용하여 설명한다. 포함질의는 엘리먼트, 애트리뷰트, 그리고 이들의 컨텐트 사이의 포함관계에 기반하여 구성된다. 이들 질의를 표현하기 위하여 예제를 이용하여 확장색인을 어떻게 이용하는지를 보이고, 이들 포함 질의를 수행하기 위한 질의 문장을 표현하고자 한다.

5.1 직접포함질의

정의 5. (직접 포함) 직접 포함 관계 질의는 엘리먼트, 애트리뷰트, 그리고 텍스트 간의 관계가 직접 포함관계로 이루어진 질의이다. 엘리먼트들 애트리뷰트들 그리고 그것들의 내용을 이루는 텍스트 단어들 간의 포함관계가 부모-자식 관계로 이루어진 질의로서, '/' 표현식이 직접포함 관계의 표현을 위하여 이용된다. 이와 같은 종류의 질의는 다음과 같은 비교 조건을 통하여 처리된다.

$$Parent(parentAddr+offset)=Child(parentAddr) \quad (3)$$

예제 5. 그림 9는 직접 포함 관계를 표현하기 위한 예제로서 부모노드가 movie이고, 자식노드가 title인 노드를 포함하는 XML 문서를 검색하기 위한 SQL 문장이다. 질의에 이용되는 concat() 함수는 두 필드의 비트열을 연결시켜서 하나의 비트열로 반환해주는 기능을 수행한다.

Query: movie/title

```
SELECT parent.docID
FROM   element parent, element current, path
WHERE  concat(parent.parentAddr,parent.offset)
         = current.parentAddr
         and parent.docID=current.docID
         and path.pathID=parent.pathID
         and path.pathExp like '%/movie'
         and path.pathID=current.pathID
         and path.pathExp like '%/title'
```

그림 9. 직접포함질의

5.2 간접포함질의

노드간의 포함관계가 조상-자손 관계로 이루어진 질의로서 다음과 같이 포함관계를 비교하여 알아낼 수 있다.

정의 6. (간접 포함) 간접 포함 관계 질의는 엘리먼트, 애트리뷰트, 그리고 텍스트 간의 관계가 간접포함관계로 구성된 질의를 가리킨다. 엘리먼트들, 애트리뷰트들 그리고 그것들의 내용을 이루는 텍스트 단어들 간의 포함관계가 조상-자손 관계로 이루어진 질의로서, '//' 표현식이 간접포함 관계의 표현을 위하여 이용된다. 이와 같은 종류의 질의는 식(4)와 같은 비교식에 의하여 처리된다.

$$Ancestor(parentAddr+offset)\subset Descendant(parentAddr) \quad (4)$$

예제 6. 그림 10의 질의는 조상노드가 movie이고, 자손노드가 title인 노드가 존재하는 XML 문서를 찾는 예제이다. contain(b1, b2) 함수는 b1 비트열의 상위비트 부분에 b2 비트열이 포함되어 있는지 여부를 확인해주는 기능을 수행한다. 결과적으로 자손 노드의 부모주소가 조상의 주소로 시작하는지를 확인하는 절차만으로 간접포함관계 질의를 처리할 수 있다.

Query: movie//title

```
SELECT   descendant.docID
FROM     element ancestor, element descendant, path
WHERE    contain(descendant.parentAddr,
  concat(ancestor.parentAddr,ancestor.offset))
           and ancestor.docID=descendant.docID
           and path.pathID= ancestor.pathID
           and path.pathExp like '%/movie'
           and path.pathID=descendant.pathID
           and path.pathExp like '%/title'
```

그림 10. 간접포함질의

5.3 완전포함질의

정의 7. (완전 포함) 완전포함 관계 질의는 노드의 관계가 완전포함관계로 이루어진 질의이다. 완전포함 질의는 주로 텍스트 값과 질의조건 내의 질의 매개변수를 비교하여 동일여부를 파악하는 용도로 이용된다.

예제 7. 그림 11에서는 완전포함 질의의 예제를 보여주고 있다. 노드의 텍스트 값으로서 'TOPGUN'이라는 값을 갖는 모든 XML 문서를 선택한다.

Query: //title='TOPGUN'

```
SELECT   element.docID
FROM     element, text, path
WHERE    text.Value='TOPGUN'
           and element.docID=text.docID
           and path.pathID=element.pathID
           and path.pathExp like'%/title'
```

그림 11. 완전포함질의

5.4 근접질의

근접질의는 노드 사이의 거리에 대한 근접 정도에 따른 결과를 목적으로 하는 질의로서 본 논문에서는 알고리즘의 특성에 알맞게 수평거리와 수직거리의 두 가지 방식을 분리하여 수행한다.

식(5)는 수직거리와 수평거리를 모두 고려한 근접관계를 찾기 위한 조건으로서 수직, 수평 거리를 합한 전체거리가 K 이하인 노드를 나타내기 위한 조건식을 표현하는 예제이다.

$$HorizontalProximity + VerticalProximity \leq K \quad (5)$$

정의 8. (수평거리) 수평거리는 형제노드 사이의 오프셋 간의 간격을 의미한다. 이 때 형제노드란 동일 부모노드 아래의 모든 자식노드들을 의미한다.

식 (6)은 두 개의 자식 노드에 대하여 형제 노드 사이의 거리가 K 이하인 수평거리를 표현하는 것으로서 수평근접관계를 나타낸다.

$$NumberOfNodesBetweenTwoSiblingNodes \leq K \quad (6)$$

정의 9. (수직거리) 수직거리는 조상노드와 자손노드 사이의 거리로서 조상-자손 간의 거리는 다음의 조건에 따라 결정된다.

$$NumberOfNodesBetweenAncestorAndDescendant \leq K \quad (7)$$

자식노드로부터 조상 노드를 찾아나가는 과정에서 빈번한 테이블의 접근이 필요하기 때문에 근접관계 연산은 수직거리 계산에 있어서 복잡하면서 부담이 큰 연산을 수행하게 된다.

예제 8. 그림 12는 형제노드간의 수평거리가 3 이하인 근접 정도를 가진 노드를 검색하는 질의 예제를 설명한다. 임의의 노드 자체와 텍스트 값이 'TOPGUN'인 노드 사이의 수평거리를 계산하여 3 이하의 조건에 맞는 노드를 모두 검색한다.

Query: hDistance ('TOPGUN') ≤ 3

```
SELECT   txtComp.Value
FROM     text txtBase, text txtComp
WHERE    txtBase.docID=txtComp.docID
         and txtBase.Value='TOPGUN'
         and txtBase.parentAddr=txtComp.parentAddr
         and ((select count txtTemp.pathID
         from text txtTemp
         where txtTemp.docID=txtBase.docID
         and txtTemp.offset>txtBase.offset
         and txtTemp.offset<txtComp.offset)<= 3) or
         ((select count txtTemp.pathID
         from text txtTemp
         where txtTemp.docID=txtBase.docID
         and txtTemp.offset>txtComp.offset
         and txtTemp.offset<txtBase.offset)<= 3)
```

그림 12. 근접질의

6. 성능 비교

본 절에서는 제안하는 알고리즘을 이용하여 질의 및 저장 작업을 수행할 때, 관계형 데이터베이스를 이용하는 경우, 기존의 좌표기반 색인기법에 의한 작업량을 비교하여 상대적인 단위작업의 단계가 줄게 되어 성능면에서 개선이 일어날 수 있음을 설명하고자 한다.

6.1 간접포함질의를 위한 비교회수

좌표기반의 색인 기법에서는 노드의 위치를 나타내기 위하여 시작, 종료에 해당하는 2개의 위치값을 이용하여 표현한다. 두 노드의 포함여부를 확인하기 위하여 각 노드의 시작값과 종료값의 위치값 사이의 크기비교를 수행해야 한다. 제안하는 확장색인 기법에서는 조상의 주소와 부모노드 주소 사이의 포함여부 만을 비교하므로 적은 회수의 비교연산을 통하여 질의를 수행할 수 있다. 그림 13은 기존의 좌표기반 기법과 확장색인 기법사이의 포함관계의 검증을 위한 비교회수 차이를 보여준다.

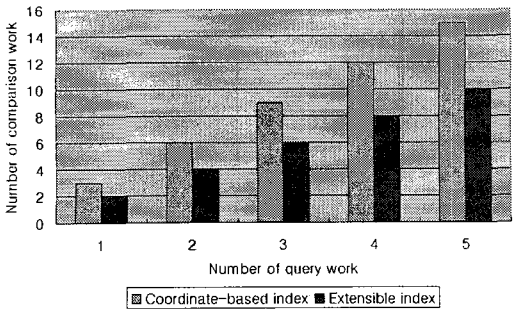


그림 13. 간접포함질의를 위한 비교작업 회수

6.2 형제노드 간의 순서정보 갱신

형제노드 사이의 순서를 표현하기 위하여 수치를 이용하여 앞과 뒤의 노드를 구분한다. 절대주소를 이용하는 방법에서는 이들의 순서를 일련번호를 이용하여 표현하기 때문에 형제노드 사이에 새로운 노드가 삽입될 경우 이후의 노드들은 모두 순서정보를 변경해야 한다. 따라서 다음의 개수만큼의 노드가 평균적으로 변경에 참여하게 된다.

$$NumberOfSiblingNodes \;/\; 2 \qquad (7)$$

확장색인의 경우, 오프셋의 비교만으로 순서정보의 크기를 비교할 수 있다. 새로운 노드를 기존의 형제노드 사이에 삽입하는 과정에서 분기작업 만이 추가되면서 순서정보를 유지할 수 있기 때문에 형제노드의 개수에 관계없이 일정한 회수의 연산이 적용된다.

그림 14에서 보여주듯이 형제노드의 개수가 증가하더라도 일정한 수의 변경이 이루어지므로 형제노드의 개수가 많아짐에 따라서 상대적으로 비교회수의 절감효과를 얻을 수 있다.

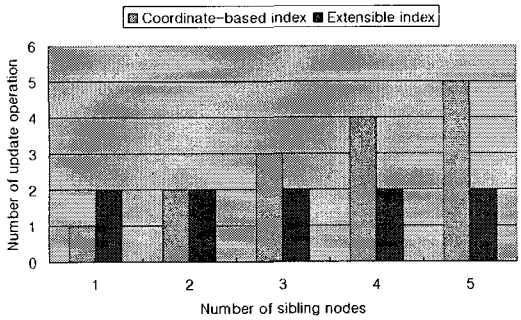


그림 14. 순서정보를 위한 갱신연산 회수비교

6.3 노드 갱신에 의한 참여노드 개수 비교

본 절에서는 예제 XML 문서를 이용하여 노드의 텍스트가 갱신되는 상황에서의 실질적인 비교를 통하여 세가지 방식의 색인기법을 비교하고자 한다. 이용될 예제 XML 문서는 그림 15와 같다.

```
<document>
    <report>
        <author>Video database</author>
        <date>June 12, 2000</date>
    </report>
    <paper>
        <title>XML query data model</title>
        <author>Don Robie</author>
        <source>W3C, June 2000</source>
    </paper>
</document>
```

그림 15. 예제 XML 문서

그림 15의 예제를 절대영역 좌표기반, 상대영역 좌표기반, 확장색인 세 가지 기법에 의하여 색인정보를 구성하면 다음의 그림 16과 같이 주소정보가 부여된다. 이때, 이탤릭으로 표현된 문자열은 XML 문서를 구성하는 엘리먼트, 텍스트 등의 내용이고, 사각

형 내의 수치정보는 엘리먼트의 절대영역 좌표기반의 색인값을 의미하며, 중괄호(brace) 내의 수치정보는 엘리먼트의 상대영역 좌표기반의 색인정보를 의미하고, 대괄호(bracket) 내의 수치정보는 확장색인에 의한 색인값을 나타낸다.

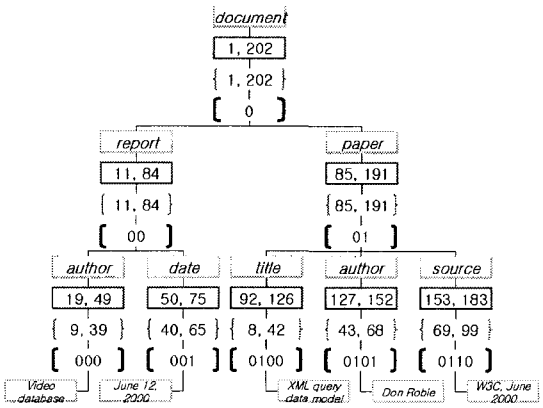


그림 16. 절대영역좌표기반, 상대영역좌표기반, 확장색인 기법에 따른 색인정보 부여

[8]의 연구에서 노드의 변경을 리프노드의 경우로만 가정하였으므로 본 연구에서도 같은 가정을 하여 비교한다. 또한 노드의 갱신시에 변경전의 문자열의 개수와 변경 후의 문자열의 개수가 동일한 경우에는 모든 기법에서 색인값이 변경되지 않으므로 비교의 대상에서 제외하기로 한다. 즉 문자열의 개수가 증가 혹은 감소되는 경우의 갱신만을 가정하여 상호 기법의 결과를 보이고자 한다.

갱신의 가정은 그림 15의 XML 문서내에 있는 /document/report/author 노드의 텍스트 값이 'Video database'에서 'Heijo Video database'로 변경되는 경우를 이용하도록 한다.

문자열이 변경되었을 경우 세 가지 기법에 의하여 구성되는 색인값이 변경된 결과는 그림 17을 통하여 알 수 있다.

'Video database' 노드 한 개의 텍스트 값의 갱신을 통하여 XML 문서내의 모든 노드의 색인값은 절대영역 좌표기반의 경우 8개, 상대영역 좌표기반의 경우 5개, 확장색인의 경우 리프노드의 구문노드 자체만 변경이 이루어진다는 결과를 알 수 있다.

예제 XML 문서에서 'Video database' 이외의 다른 노드에서 갱신이 일어나는 경우에도 비슷한 수준의 개수만큼의 노드에서 색인값이 변경되며 갱신에

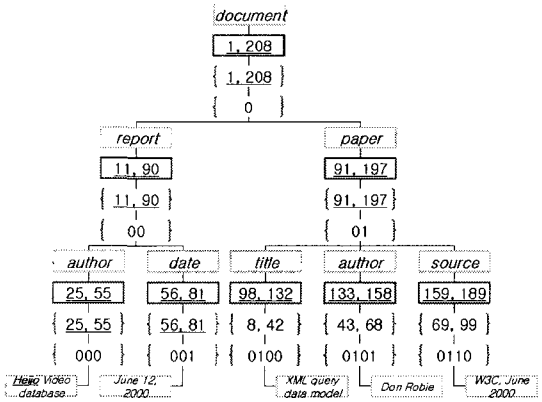


그림 17. 절대영역좌표기반, 상대영역좌표기반, 확장색인 기법에 따른 색인정보 변경결과

참여하는 노드의 개수를 누적하여 평균값을 비교한 결과는 그림 18과 같다.

확장색인 기법의 경우에는 노드의 위치정보가 변경되는 것이 아닌 텍스트 값을 저장하고 있는 테이블의 값을 직접 변경하고, 색인값의 변경은 발생하지 않기 때문에 갱신연산이 발생할 경우 이로 인한 추가 부담이 경감되는 특성이 있음을 살펴보았다.

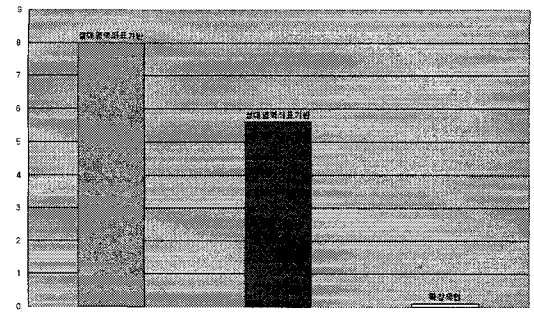


그림 18. 노드의 갱신으로 인한 절대영역, 상대영역, 확장색인 기법간 평균 변경참여노드 개수

7. 결론 및 향후 연구과제

본 논문에서는 XML 문서의 노드간 위치정보를 표현하기 위한 확장색인 기법을 제안하였다. 일반적으로 널리 이용되는 SQL 문장의 표현을 통하여 기존의 직접포함질의, 간접포함질의, 완전포함질의, 근접질의 등의 지원을 유지하면서도, 검색질의에 적용되는 비교대상을 단순화하여 좌표기반의 색인기법에 비하여 성능을 향상시킬 수 있으며, 갱신 연산에 의

한 색인의 재구성을 최소화할 수 있는 효율적인 색인 기법이다.

향후 연구과제로서 트리의 깊이가 깊어짐에 따라 비트열의 길이가 증가함에 따른 성능저하의 문제를 해결하기 위한 고려가 필요하다.

참 고 문 헌

[1] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, "Extensible Markup Language (XML) 1.0 (Second Edition)," http://www.w3.org/TR/2000/REC-xml-20001006.

[2] International Organization for Standardization, "Information Processing - Text and Office System - Standard Generalized Markup Language (SGML)," ISO/IEC 8879, Oct. 1986.

[3] Alin Deutsch, Mary Fernandez, Dan Suciu, "Storing Semistructured Data with STORED," Proc. ACM SIGMOD Conf. Management of Data, June 1999.

[4] Jayavel Shanmugasundaram, Kristin Tufte, Gang He, "Relational Databases for Querying XML Documents: Limitations and Opportunities," Proc. 25th International Conference on VLDB, Edinburth, Scotland, September 7-10, 1999.

[5] Philip Bohannon, Juliana Freire, Prasan Roy, Jerome Simeon, "From XML Schema to Relations: A Cost-based Approach to XML Storage," ICDE 2002.

[6] R. Sacks-Davis, T. Dao, J. A. Thom, and J. Zobel, "Indexing documents for queries on structure, content and attributes," In International Symposium on Digital Media Information Base (DMIB'97), Nov. 1997.

[7] C. L. Clarke, G. V. Cormack, and F. J. Burkowski, "An algebra for structured text search and a framework for its implementation," The Computer Journal, 38(1): pp. 43-56, 1995.

[8] D. D. Kha, M. Yoshikawa, S. Uemura, "An XML indexing structure with relative region coordinate," Proceedings. 17th International Conference on Data Engineering, April 2001.

[9] J. P. Yoon, V. Raghavan, V. Chakilam, "BitCube: a three-dimensional bitmap indexing for XML documents," Scientific and Statistical Database Management, SSDBM 2001. Proceedings. Thirteenth International Conference on, July 2001.

[10] T. Shimura, M. Yoshikawa, S. Uemura, "Storage and retrieval of XML documents using object-relational databases," In Proc. DEXA Conf., pp. 206-217, Florence, Italy, Sept, 1999.

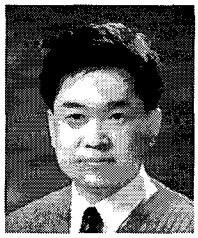
[11] C. Zhang, J. Naughton, D. DeWitt, Q. Luo, and G. Lohman, "On supporting containment queries in relational database management systems," ACM SIGMOD, 2001.

[12] J. Bremer, M. Gertz, "An Efficient XML Node Identification and Indexing Scheme," Technical Report CSE-2003-04. Dept.artment of Computer Science, University of California at Davis, January 2003.

[13] 서치영, 이상원, 김형주, "XML 문서에 대한 RDBMS에 기반을 둔 효율적인 역색인 기법," 정보과학회논문지, 제30권, 제1호, pp. 27-40, 2003.

송 정 석

1991년 2월 세종대학교 지구과학과 졸업
1995년 2월 광운대학교 전자계산학과 석사
2005년 2월 광운대학교 컴퓨터과학과 박사
2001년 10월 현재 한화S&C 기술연구소 선임연구원
관심분야 : 멀티미디어, 데이터베이스, XML, 홈네트워크



김 우 생

1985년 서울대학교 수료 및 University of Texas at Austin 전산학과 졸업
1987년 University of Minnesota 전산학과 석사
1991년 University of Minnesota 전산학과 박사 및 Post Doctor
1987년~1988년 현대전자. Zeus Computer 과장
2000년~2001년 미션텔레콤(주) 이사
2001년 UC 버클리 대학 교환 교수
1992년~현재 광운대학교 컴퓨터공학부 교수
관심분야 : 멀티미디어 시스템, 데이터베이스, 영상/비디오 처리 및 패턴인식