논문 2006-43CI-1-5

효율적인 XML 검색을 위한 상대 위치 좌표 기반의 BitmapInvert Index 기법

(An Efficient BitmapInvert Index based on Relative Position Coordinate for Retrieval of XML documents)

김 택 곤 *, 김 우 생*

(Tackgon Kim and Woosaeng Kim)

요 약

최근 XML 문서의 저장 및 관리에 대한 연구가 활발히 이루어지고 있다. XML 문서를 위한 현재까지 연구된 대부분의 색인기법의 경우 절대좌표를 이용하여 표현하는 방법을 사용하므로 갱신연산이 커다란 부담으로 받아들여지고 있다. 본 논문에서는 이를 개선하기 위하여 상대위치좌표에 기반한 BitmapInvert 색인 기법을 제안하였다. 상대위치좌표는 부모 노드와 좌, 우 형제 노드들 간의 관계만을 표현하므로 갱신연산지 자주 발생하더라도 좋은 성능을 보인다. BitmapInvert 색인 기법은 bitwise 연산을 지원하는 텍스트 검색을 지원하고, PostUpdate 알고리즘을 적용하여 갱신에 따른 성능 저하를 줄이도록 하였다. 따라서, 제안하는 기법은 검색이나 갱신에 대해서 접근하는 노드를 줄임으로써 전체적인 성능을 향상시킬 수 있다.

Abstract

Recently, a lot of index techniques for storing and querying XML document have been studied so far and many researches of them used coordinate-based methods. But update operation and query processing to express structural relations among elements, attributes and texts make a large burden. In this paper, we propose an efficient BitmapInvert index technique based on Relative Position Coordinate (RPC). RPC has good preformance even if there are frequent update operations because it represents relationship among parent node and left, right sibling nodes. BitmapInvert index supports text query with bitwise operations and does not casue serious performance degradations on update operations using PostUpdate algorithm. Overall, the performance could be improved by reduction of the number of times for traversing nodes.

Keywords : 상대위치좌표, Relative Position Coordinate, RPC, BitmapInvert, XML

I. 서 론

최근 W3C (World Wide Web Comsortium)의 권고를 통하여 SGML[1]로부터 파생된 XML[2] 언어가 상호 운영성을 유지하면서 인터넷을 통한 데이터 교환을 제공하기 위한 표준 언어로서 부상하고 있으며, XML 언어의 풍부한 표현능력을 통하여 많은 애플리케이션이 등장하고 있다. 따라서 XML 문서를 데이터베이스에 효율적으로 저장하고 추출하기 위한 주제가 꾸준히 연구되어 왔으며, XML 문서내의 내용뿐만 아니라 구조적 정보 및 위치적 정보 또한 보존할 수 있는 방안이 제공되어야 하며, 이를 효율적으로 검색할 수 있는 방법이 필요하다.

XML 문서에 대한 색인 기법은 다양하게 연구되어 왔으며, 많은 연구가 절대좌표를 이용하는 Position 기법의 색인 기법이나 경로에 기반한 Path 기법의 색인 기법을 적용하였다. 이런 연구들 중에 문서 내의 content,

* 정회원, 광운대학교 컴퓨터과학과
(Dept. of Computer Science, Kwangwoon University)
접수일자 : 2005년7월27일 수정완료일 : 2006년1월3일

structure, attribute에 따른 위치 기반의 색인과 경로(Path)기반의 색인에 대해 언급한 연구[3]가 있으며, 위치기반의 방법으로 GCL(Generalized Concordance Lists) 위치 기반의 색인 모델[4]이나, 절대좌표 방법을 개선한 상대영역좌표 방법[5]에 관한 것등이 있다. 그리고, 좌표나 경로에 기반한 방법외에도 비트맵 색인에 기반한 Bitcube 기법[6][7]등도 연구되고 있다.

이러한 기존의 많은 연구들의 경우 좌표 기반의 방법에 대해서 단어, 엘리먼트(element) 혹은 속성(attribute)의 위치에 대해 시작과 끝에 대한 위치 값을 이용하므로 갱신이 발생할 경우 색인 구조의 상당 부분을 재계산해야 한다. 만약 갱신연산이 빈번한 응용에서 이와 같은 현상이 발생한다면 갱신을 위한 비용은 상당한 부담으로 시스템에 영향을 미치게 될 것이다.

본 논문에서 제안하는 기법은 기존의 절대좌표 기반의 색인기법에서 적용하던 포함질의(Containment Query)[8]를 지속적으로 지원하면서도 노드의 갱신시에 색인 정보가 재계산되는 부분을 감소시키고, 텍스트 검색을 위해 역색인(Inverted Index)[9][10]과 비트맵 인덱스(Bitmap Index)[9][10]를 응용하여 보다 효율적으로 XML 문서의 구조를 표현할 수 있는 기법으로 XML의 구조를 표현하기 위한 상대위치좌표(Relative Position Coordinate) 기법과, 이를 기반으로 하여 텍스트 질의를 수행할 수 있는 BitmapInvert 기법을 제안하였다.

먼저 상대위치좌표(Relative Position Coordinate) 기반의 표현 방법은 Position 기반의 상대영역 좌표에 기반한 기존의 연구를 개선하여 오프셋의 대상을 자식노드에 국한하고, 자식노드들의 상호 위치를 리프 노드들의 경로에 매치시켜 표현하는 방법을 이용한다. 따라서, 갱신이나 삽입 연산이 발생할 경우 해당 노드에 대한 형제 노드들만의 오프셋 정보만 변경하면 되므로 보다 좋은 성능을 얻을 수 있다.

다음으로 BitmapInvert 기법은 XML 문서의 갱신이 일어날 경우 최소한의 정보만을 변경하도록 하는 PostUpdate 기법을 제안하여 텍스트 관련 색인의 갱신 연산에 따른 비용을 최소화하고, 검색시 보다 좋은 성능을 얻을 수 있도록 하였다. 이 기법은 XML 문서들의 여러 어휘들이 각 파일에서의 존재유무를 나타내는 필드와 파일 내에서의 각 리프노드상에서의 존재 유무를 표현하는 필드로 구성을 하여, XML 문서의 추가나 삭제시 해당 내용에 대해서 바로 변경이 가능하도록 하였고, 검색에 bitwise 연산을 이용할 수 있어 복잡한 검색이더라도 큰 성능 저하 없이 검색을 수행할 수 있다.

본 논문의 구성은 다음과 같다. 제 Ⅱ장에서는 관련 연구로서 XML 문서의 색인 기법에 대한 기존의 연구에 대하여 설명한다. 제 Ⅲ장에서는 본 논문에서 제안한 XML 문서의 위치 정보 표현을 위한 상대위치좌표 기반의 방법과 텍스트 색인을 위한 BitmapInvert 방법을 소개한다. 제 Ⅳ장에서는 상대 위치 좌표를 위한 연산들을 제시하고, BitmapInvert Index를 처리하기 위한 방법들을 설명한다. 제 Ⅴ장에서는 포함질의 및 단어기반 질의를 위한 질의 예제를 보여준다. 제 Ⅵ장에서는 기존의 방법과의 비교를 통해 제안 방법에 대한 성능 평가를 보이고, 마지막으로 제 Ⅶ장에서 결론을 맺는다.

Ⅱ. 관련 연구

XML 문서에 대한 대표적인 색인 기법으로는 문서내의 엘리먼트(element), 속성(attribute) 그리고 텍스트 데이터에 대해 이들의 시작과 끝에 대한 위치 값을 색인화하여 처리하는 절대좌표 기반의 Position 방법이 있다. 이 방법에서는 각 엘리먼트, 속성, 텍스트들이 이루는 영역 정보에 대해 질의가 가능하므로 이들간의 포함여부에 대한 처리도 가능하다.

그리고, 절대영역좌표(Absolute Region Coordinate, ARC) 기반의 방법에서 업데이트가 일어나는 경우 많은 갱신 연산이 이루어져야 하므로 이를 감소하기 위해 상대영역좌표(Relative Region Coordinate, RRC) 방법이 제안되었다. 이 방법은 XML 문서의 각 리프노드에 대해서 부모 노드의 위치를 기준으로 하여 새로운 영역값을 설정하여 다른 부분에서 갱신이 발생하더라도 다른 부분의 리프 노드들에 대해서는 값을 계속 유지시킬 수 있어 갱신에 대해 보다 좋은 성능을 보일 수 있다[5].

Path 기반의 색인에서는 단어의 위치가 구조적 요소로서 표현되며, 트리 구조의 경로가 질의처리에 이용된다. 문서내의 단어의 위치를 결정하기 위하여, 루트 노드로부터 단어를 포함하는 리프 노드에 이르는 엘리먼트 이름의 경로를 이용하여 색인을 구성한다. 각 단어에 대한 역색인을 구성하여 단어에 대한 경로의 표현을 구성한다[11].

XML 문서들로부터 정보를 추출하기 위한 수단으로 많은 질의어가 제안되었는데, 분명한 것은 이들 질의어들을 이용하여 XML 문서들로부터 정보를 추출시, 포함질의(Containment Query)[8]가 XML 정보검색 시스템 상에서의 질의의 핵심이 된다는 것이다. 여기서 말하는 포함질의란 XML 문서상에서 엘리먼트들(elements), 속

성들(attributes), 그리고 그것들의 내용을 이루는 텍스트 단어들간의 포함(containment)관계에 기반을 둔 질의를 말한다. 이러한 포함질의가 XML 정보검색 시스템 상에서 XML 문서에 대한 질의의 핵심적인 부분이 되기 때문에 포함질의를 어떤 방법으로 지원하느냐 하는 것이 중요한 문제가 된다.

III. 상대위치좌표에 기반한 BitmapInvert 기법

1. 테이블 스키마

본 연구에서는 질의 처리의 효율을 증가시키기 위해 Position 기반의 색인 기법을 개선한 상대위치좌표 기반의 색인 방식을 이용하여 노드의 갱신에 따른 많은 노드의 색인 정보가 재계산되지 않고, 갱신연산이 빈번한 응용에 최적화될 수 있도록 한다. 그리고 텍스트 기반의 검색을 위해 비트맵 인덱스를 응용한 형태의 색인을 이용하여 텍스트 검색의 성능을 향상시키도록 하였다.

본 논문에서는 노드의 정보를 표현하기 위해 그림 1과 같은 테이블 스키마를 이용한다.

XML의 구조적인 정보와 위치 정보를 저장하기 위한 테이블 스키마는 Document, ElementType, ElementNode, Path, PathMap, Attribute, Text, TextFile, TextContent 테이블로 구성된다.

Document 테이블은 XML 문서들의 기존 정보를 표현하며, 일련번호를 나타내는 docID 필드, 파일명을 나타내는 fileName, 그리고 문서의 존재여부를 나타내는 existFileBit 필드와 각 리프노드들의 존재 여부를 나타내는 existPathBitmap 필드로 구성된다.

ElementType 테이블은 XML 문서에서 발생되는 엘리먼트들에 대해 중복을 제거한 하나의 구조로 표현할 때의 각 엘리먼트들의 정보를 관리한다. etID 필드는 엘리먼트 유형에 대한 일련 번호이고, etName 필드는 엘리먼트 이름, 그리고, level은 최상위 엘리먼트의 레벨값을 0으로 할 때의 해당 엘리먼트에 대한 깊이값을 저장한다.

Document {docID, fileName, existFileBit, existPathBitmap}
ElementType {etID, etName, level}
ElementNode {docID, etID, enID, parentID, preOffset, postOffset}
Path {pathID, etID}
PathMap {docID, pathID, enIDs}
Attribute {docID, pathID, attrName, attrValue}
Text {docID, pathID, textValue}
TextFile {termID, termName, fileBitmap}
TextContent {docID, termID, contentBitmap}

그림 1. 테이블 스키마
Fig. 1. Table schemes.

ElementNode 테이블은 XML 문서의 각 엘리먼트에 대해 상대적인 위치값을 관리한다. 테이블을 구성하는 필드로는 XML 문서의 일련번호를 나타내는 docID, 엘리먼트의 유형을 표시하는 etID, 엘리먼트에 대해 일련의 순서를 붙여서 표시하는 enID, 그리고 해당 엘리먼트의 부모 엘리먼트의 enID값을 표시하는 parentID, 좌우형제노드의 enID값을 표현하는 preOffset, postOffset들이 존재한다.

Path 테이블은 XML 엘리먼트 트리 구조상에서 최상위 노드(root node)로부터 각각의 리프 노드(leaf node)들까지의 경로식들에 대해 식별자를 부여하고, 이를 통해 XML 문서의 트리 구조를 표현하는데 사용한다.

PathMap 테이블은 Path 테이블이 가지고 있는 각 경로들에 대해 해당 경로가 어떤 엘리먼트들을 포함하고 있는지를 관리한다. 구성되는 필드로는 문서의 일련번호를 표시하는 docID, 해당 경로에 대한 값을 표시하는 pathID, 그리고, 각 pathID에 대해 포함되는 엘리먼트들을 표시하는 enID 필드, 각 노드의 존재 유무를 나타내는 existBit 필드가 존재한다.

Attribute 테이블은 XML 문서의 속성에 대한 정보를 저장한다. Attribute 테이블은 XML 문서의 식별자를 위한 docID, 경로식별자를 위한 pathID, 해당 속성의 이름을 표시하기 위한 attrName, 해당 속성의 값을 저장하기 위한 attrValue 필드로 표현된다.

Text 테이블은 완전포함질의를 위한 테이블로 텍스트 내용을 그래도 표현한다. Text 테이블은 문서 식별자인 docID, 경로 식별자인 pathID, 그리고, 텍스트 내용을 표현하는 textValue 필드로 구성된다.

TextFile 테이블은 문서의 텍스트 검색을 효율적으로 하기 위해 각 파일들에 대한 단어의 존재 여부를 표현하는 비트맵 정보를 이용한다. TextFile 테이블은 단어의 식별자를 위한 termID, 각 단어를 나타내는 termName, 그리고, 해당 단어가 포함된 파일의 존재 여부를 나타내는 fileBitmap 필드로 구성된다.

TextContent 테이블은 각 파일 내에서 리프노드의 순서에 따라 해당 단어의 존재 여부를 나타내는 테이블이다. TextContent 테이블은 각 termID에 대해 해당 문서의 식별자를 위한 docID 필드와 리프노드들에서의 존재 여부를 나타내는 contentBitmap 필드로 구성된다.

2. 상대위치좌표

상대위치좌표 기반의 색인 방법[12]은 XML 트리의 각 노드의 정보를 부모 노드와 좌, 우의 형제 노드에 국한

하여 업데이트를 할 경우 갱신 비용을 최소화하고, 질의할 수 있는 방법이다. 본 논문에서는 상대위치좌표 기반의 방법을 위해 각 노드의 정보를 표현하는 상대위치좌표를 이용하고, 전체적인 구조 정보를 나타내기 위해 각 리프 노드에 대응하는 경로식별자를 이용한다.

상대위치좌표는 부모 노드 P와 자식 노드 C_1, C_2, .., C_n이 있을 때 임의의 자식 노드 C의 주소를 (ParentNode, preOffset, postOffset, level)와 같은 형태로 표현하는 것이다. 여기서 ParenNode는 부모 노드 P의 노드식별자 값이며, preOffset, postOffset은 자식 노드 C의 좌, 우 형제 노드의 노드식별자 값을 나타내고, level은 XML 트리에서의 깊이값을 의미한다.

그리고 상대위치좌표만으로는 전체 트리 구조를 표현하기 어려우므로, 경로식별자를 이용한다. 경로식별자로 사용되는 pathID 필드는 루트 노드로부터 각 리프 노드까지의 경로식을 구분하는 식별자로 사용되는데, 다른 XML 색인 기법에 관한 연구들에서는 보통 중간의 노드들까지의 경로식별자들도 유지하고 있으나 본 논문에서는 리프 노드의 위치에서 조상 노드들의 포함 여부를 파악하는 연산이 필요하므로 중간 노드들에 대한 경로식별자를 따로 두지 않는다. 레벨값에 따라 자식 노드들을 CL_0 (=root node), CL_1, CL_2, ... CL_n 으로 표현할 때, 하나의 경로식은 {CL_0, CL_1, CL_2, ... CL_n} 으로 표현될 수 있으며, 식 1과 같이 표현할 수 있다.

$$PathMap=\{\ enID \mid 0 \leq enID_{level} \leq n,\ enID_{pathID=}=PathMap_{pathID}\} \quad (1)$$

예제 1. 부모 노드의 식별자(enID)가 1, 레벨값이 1이고, 자식 노드 C_1, C_2, C_3, C_4, C_5가 존재한다고 할 때 이들의 값은 그림 2와 같이 표현할 수 있다. 그림 2는 부모 노드 P와 자식 노드들에 대한 예를 트리 구조로 표현한 것이며, 이에 대한 ElementNode 테이블의 정보는 그림 3과 같이 나타낼 수 있다. 그리고, 트리 구조를 표현할 수 있는 경로식별자에 대한 정보는 그림 4와 같이 나타낼 수 있다.

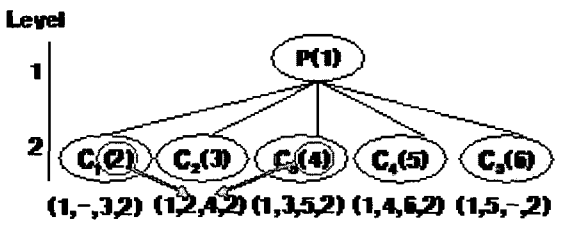


그림 2. 예제 XML 트리
Fig. 2. An example of XML tree.

node	enID	parentID	preOffset	postOffset
P	1	–	–	–
C_1	2	1	–	3
C_2	3	1	2	4
C_3	4	1	3	5
C_4	5	1	4	6
C_5	6	1	5	–

그림 3. ElementNode 테이블 예
Fig. 3. An example of ElementNode table.

node	pathID	docID	enID List
C_1	1	1	2, 1
C_2	2	1	3, 1
C_3	3	1	4, 1
C_4	4	1	5, 1
C_5	5	1	6, 1

그림 4. 경로식별자 할당
Fig. 4. Assignment of pathID values.

3. BitmapInvert 색인

본 논문에서는 텍스트 검색시 XML 문서의 갱신연산에 대해 큰 영향을 받지 않으면서 빠른 검색 성능을 보일 수 있도록 역색인와 비트맵 인덱스를 응용하여 BitmapInvert Index 방법을 제안하였다.

정의 1. (BitmapInvert Index) XML 문서의 텍스트를 구성하는 단어들의 존재 유무와 위치 정보를 표현하기 위해 역색인과 비트맵 인덱스를 응용한 형태로 표현한다. BitmapInvert Index는 TextFile과 TextContent 테이블로 구성되며, TextFile 테이블은 각 어휘가 문서 내에 존재하는지의 유무를 나타내고, TextContent 테이블은 각 문서 내에서 어휘의 존재 위치를 표현하는 비트열로 구성된다. 각 XML 문서들을 D_1, D_2, ..., D_n, XML 문서들 내에 포함되어 있는 어휘들을 T_1, T_2, ..., T_m, 문서내의 리프노드들의 수들 k라 할 때 임의의 어휘 T_i에 대해 식 2와 식 3과 같이 표현할 수 있다. 여기서 '0'은 어휘가 존재하지 않음을 의미하며, '1'은 존재함을 의미한다.

$$Ti_{TextFile}=\{BitString(x)_a,\ x=\{0,\ 1\},\ 1 \leq a \leq n,\} \quad (2)$$

$$Ti_{TextContent}=\{BitString(y)_b,\ y=\{0,\ 1\},\ 1 \leq b \leq k,\} \quad (3)$$

예제 2. XML문서가 그림 5와 같이 2개가 존재할 경우 각 어휘에 대해 BitmapInvert Index는 그림 6, 그림 7과 같이 나타낼 수 있다.

[문서 1]	*[문서 2]*
<xml1> <text>hello</text> </xml1>	<xml1> <text1>world</text1> <text2>hello</text2> </xml1>

그림 5. 예제 XML 문서들
Fig. 5. Examples of XML document.

termID	termName	fileBitmap
1	hello	1 1
2	world	0 1

그림 6. 그림 5에 대한 TextFile 테이블 결과
Fig. 6. Result of TextFile table from fig. 5

termID	docID	contentBitmap
1	1	1
1	2	0 1
2	2	1 0

그림 7. 그림 5에 대한 TextContent 테이블 결과
Fig. 7. Result of TextContent table from fig. 5

Ⅳ. 색인을 위한 연산

본 논문에서는 색인을 위한 연산에 대해 기존의 XML 문서의 갱신 연산에 대한 연구[13]를 기반으로 하여 노드의 접근, 삽입, 삭제, 갱신으로 분류하여 각각의 연산에 대한 방법을 고려하였다.

1. 노드의 접근

XML 문서의 각 노드에 대한 접근은 해당 노드에 대한 존재 여부와 구조적인 정보를 검색하기 위한 것이다. 본 논문에서는 RDBMS에 기반하여 각 노드의 정보를 저장하고 SQL 문을 통해 노드의 정보를 얻어올 수 있다. V장에서 다양한 포함질의 처리를 통하여 구체적인 과정을 자세히 설명한다.

```
☛ InsertBefore | InsertAfter ( ref, content )
1. Get ref_node's information
   (docID,enID,parentID,level,preOffset,postOffset,...)
2. Setg inserted node's offset from ref_node
   If InsertBefore operation then
     content_node_preOffset=ref_node_preOffset && content_node_postOffset=ref_node_enID
   If InsertAfter operation then
     content_node_preOffset=ref_node_enID && content_node_postOffset=ref_node_postOffset
3. Update left and right sibling nodes
   ref_node->left_node_postOffset = content_node_enID
   ref_node->right_node_preOffset = content_node_enID
   Update ElementNode table.
4. Update Path, PathMap tables as follows
   Creatge a new path value for inserted node
   content_node_pathID = max(pathIDs) + 1
5. Update a Document table's existPathBitmap field
   Add a bit as 1
6. Update a TextFile, TextContent tables as follows
   Extract terms of text value
   Check appearance of terms on TextFile table
     If new terms is appeared, set bit value 0 to 1
   Update contentBitmap table for appeared terms
     For each appeared term, get bitmap length
     Call PostUpdate(update, new terms)
     Append a bit as 1 for each appeared term's bitmap
```

그림 8. 삽입을 위한 의사코드 알고리즘
Fig. 8. Pseudo algorithm of insertion operation.

삽입 연산문: InsertBefore (C3, C6)

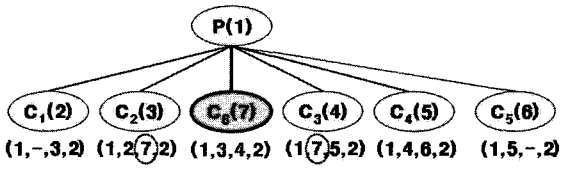


그림 9. 삽입연산 후 ElementNode 리스트
Fig. 9. Result of ElementNode list after insert operation.

node	pathID	enID	docID	enID List
C_1	1	2	1	2, 1
C_2	2	3	1	3, 1
C_3	3	4	1	4, 1
C_4	4	5	1	5, 1
C_5	5	6	1	6, 1
C_6	**6**	**7**	**1**	**7, 1**

그림 10. 삽입 연산 후 경로 식별자
Fig. 10. Result of PathID values after insert operation.

2. 노드의 삽입

노드의 삽입은 그림 8의 알고리즘을 이용하여 처리할 수 있다.

예제 3. 그림 2와 그림 3의 예제에서 C_6 노드가 C_3 노드 앞에 추가되는 경우 그림 8의 알고리즘에 따라 노드 식별자 값과 오프셋 값이 할당되어 그림 9와 같이 새로운 공간에 삽입될 수 있으며, 경로 식별자 값도 그림 10과 같이 갱신된다.

3. 노드의 삭제

노드의 삭제는 그림11의 알고리즘에 따라 노드의 위치를 확인하고 해당 노드의 자식 및 자손 노드들을 삭제한다.

4. 노드의 갱신

노드의 갱신은 그림 12의 알고리즘을 이용하여 해당 노드의 내용을 변경하는 작업을 수행한다. 여기서 갱신

```
☛ Remove ( ref, content )
1. Get ref_node's information
   (docID,enID,paretnId,preOffset,postOffset,level,...)
2. Find sub nodes (including child, descendent)
   If (ref_node_pathID = subnode_pathID) and
     (ref_node_level < subnode_level) then
       Remove sub nodes in ElementNode, Path, PathMap tables
3. Remove ref_node from ElementNode table
4. Update Path, PathMap tables
5. Update a Document table's existPathBitmap field
   Set a bit value 1 to 0 on ref_node enID's order position
6. Call PostUpdate(Renewal)
```

그림 11. 삭제 연산을 위한 의사코드 알고리즘
Fig. 11. Pseudo algorithm of deletion operation.

```
☛ Update ( ref, content )
1. Get ref_node's information
   (docID,enID,parentID,preOffset,postOffset,level,text,...)
2. Update a Document table's existPathBitmap field
   Set a bit value 1 to 0 on ref_node enID's order position
   Append a bit value as 1
3. Change a ref_node's information (like a intsertion a new node)
   new enID = max(enID) + 1
   new Offsets, parentID, 뒤 values = original values
4. Process with insertion algorithm step 3
5. Call PostUpdate(Renewal)
```

그림 12. 갱신 연산을 위한 의사코드 알고리즘
Fig. 12. Pseudo algorithm of update operation.

은 텍스트 내용의 변경을 의미하며, 제안하는 색인 구조에서는 텍스트 내용의 변경에 대해 TextFile, TextContent 테이블의 변경 외에는 다른 부분에서의 변경이 일어나지 않는다.

5. PostUpdate

정의 2. (PostUpdate) BitmapInvert 인덱스에서 갱신 연산을 적용하는 경우 많은 연산이 필요하게 되므로, 이를 감소시킬 수 있는 방법으로 PostUpdate 방법을 이용하여 갱신에 참여하는 데이터의 양을 최소화하도록 하였다. PostUpdate의 기본 알고리즘은 BitmapInvert 색인의 방법에서 모든 내용을 항상 최신의 정보로 유지시키는 것이 아니라 갱신이나 검색 등을 통해 요구되는 색인의 정보들에 대해서만 최신의 정보를 유지시키는 방법을 사용한다. 따라서 자주 사용되는 정보들에 대해서는 최신의 정보를 쉽게 구할 수 있으며 자주 사용되지 않는 정보들에 대해서는 갱신비용이 줄어들게 되어 성능을 높일 수 있게 된다. PostUpdate 방법은 그림 13의 방법을 이용하여 처리할 수 있다.

예제 4. 예제 2의 경우에서 첫 번째 XML 문서에서 <text> 엘리먼트 다음에 <text3> 엘리먼트가 추가되고 텍스트 데이터로 'welcome'이 삽입될 경우, 변경되는 내용은 다음 그림들로 표현될 수 있다. 그림 14와 같이 새로운 어휘가 추가되더라도 다른 부분에 대해서는 변경

```
☛ PostUpdate ( operation, data )
1. Check operation mode
2. If operation is Update then
   // Run at insert, search time
   If length(data's bitmap) < (existPathBitmap - 1)
     Set zero bits on insufficient bitmap
   For other terms, do nothing immediately.
3. If operation is Renewal
   // Run at idle time
   Update a TextFile, TextContent tables as follows
   . Run AND operation with existPathBitmap and all terms bitmap
     Update each terms bitmap with result
```

그림 13. PostUpdate 알고리즘
Fig. 13. Pseudo algorithm of PostUpdate.

[변경전: 문서 1]	*[변경후: 문서 1]*
<xml>	<xml>
<text>hello</text>	<text>hello</text>
</xml>	**<text3>welcome</text3>**
	</xml>

그림 14. XML 문서의 변경이 일어나는 경우
Fig. 14. Update on sample XML document.

termID	termName	fileBitmap
1	hello	1 1
3	**welcome**	**1 0**

그림 15. 그림 14에 대한 TextFile에서의 업데이트 내용
Fig. 15. Update a TextFile table from fig. 14.

termID	docID	contentBitmap
1	1	1
3	**1**	**0 1**

그림 16. 그림 14에 대한 TextContent에서의 업데이트
Fig. 16. Update a TextContent table from fig. 14.

을 하지 않고, 해당 어휘에 대한 접근이 일어나는 경우, 즉 삽입, 삭제, 수정, 검색 등의 작업을 하는 경우에만 변경하도록 하며, 해당되지 않는 어휘들에 대해서는 업데이트를 하지 않는다.

V. 질의 처리

본 절에서는 포함질의에 대한 종류를 [8]의 분류와 마찬가지로 직접포함질의, 간접포함질의, 완전포함질의 및 노드간의 근접 정도에 따른 근접질의 등의 분류를 이용하여 설명하고, 텍스트 기반의 질의에 대해서도 설명한다. 포함질의는 엘리먼트, 속성, 그리고 이들의 컨텐트 사이의 포함환계에 기반하여 구성된다. 이들 질의를 표현하기 위하여 상대위치좌표기반 색인과 BitmapInvert Index을 어떻게 이용하는지 예제를 사용하여 보이고, 단어들간의 조합이나 문서의 검색의 예를 제시한다. 그리고, 텍스트 기반의 질의는 XML 문서내의 단어들에 대해 bitwise 연산을 이용하여 검색을 할 수 있도록 지원한다.

1. 직접포함질의

직접 포함 관계 질의는 엘리먼트, 속성, 그리고 텍스트 간의 관계가 직접 포함관계로 이루어진 질의이다. 엘리먼트들, 속성들 그리고 그것들의 내용을 이루는 텍스트 단어들 간의 포함관계가 부모-자식 관계로 이루어진 질의로서, '/' 표현식이 직접포함 관계의 표현을 위하여 이용된다. 이와 같은 종류의 질의는 식 4의 비교 조건을 통하여 처리된다.

$$ParentNode_{enID}=ChildNode_{parentID} \text{ and}$$
$$ParentNode_{pathID}=ChildNode_{pathID} \quad (4)$$

예제 5. 그림 17은 직접 포함 관계를 표현하기 위한 예제로서 부모노드가 movie이고, 자식노드가 title인 노드를 포함하는 XML 문서를 검색하기 위한 SQL 문장이다.

Query : movie/title

```
SELECT   enC.docID
FROM     ElementType etP, ElementNode enP, Path pP,
         ElementType etC, ElementNode etC, Path pC
WHERE    etP.etName = 'movie' and etP.etID = enP.etID and
         etC.etName = 'title' and etC.etID = enC.etID and
         enP.docID = enC.docID and enP.enID = enC.parentID
```

그림 17. 직접 포함 질의
Fig. 17. Direct containment query.

5.2 간접포함질의

간접 포함 관계 질의는 엘리먼트, 속성, 그리고 텍스트 간의 간계가 간접 포함관계로 구성딘 질의를 가리킨다. 엘리먼트들, 속성들, 그리고 그것들의내용을 이루는 텍스트 단어들 간의 포함관계가 조상-자손 관계로 이루어진 질의로서, '//' 표현식이 간접포함관계의 표현을 위하여 이용된다. 이와 같은 종류의 질의는 식 4의 비교조건에 따라 처리된다.

$$AncestorNode_{pathID}=DescendentNode_{pathID} \quad (5)$$

예제 6. 그림 18의 질의는 조상노드가 movie이고, 자손노드가 title인 노드가 존재하는 XML 문서를 찾는 예제이다. 간접포함질의의 경우는 조상노드와 자손노드가 같은 경로식별자를 가지고 있는지를 검사하여 구할 수 있다.

Query : movie//title

```
SELECT distinct (pmD.docID)
FROM     ElementType etA, ElementNode enA, Path pA,
         ElementType etD, ElementNode enD, Path pD,
         PathMap pmA, PathMap pmD
WHERE    etA.etName = 'movie' and etA.etID = enA.etID and
         etD.etName = 'title' and etD.etID = enD.etID and
         pmA.enID = enA.enID and pmA.docID = enA.docID and
         pmD.enID = enD.enID and pmD.docID = enD.docID and
         pmA.pathID = pmD.pathID and pmA.docID = pmD.docID
```

그림 18. 간접포함질의
Fig. 18. Indirect containment query.

5.3 완전포함질의

완전포함 관계 질의는 노드의 관계가 완전포함관계로 이루어진 질의이다. 완전포함 질의는 주로 텍스트 값과 질의조건 내의 질의 매개변수를 이용하여 동일여부를 파악하는 용도로 이용된다.

예제 7. 그림 19에서는 완전포함질의의 예제를 보여주고 있다. 노드의 텍스트 값으로서 'TOPGUN'이라는 값을 가지는 모든 XML 문서를 선택한다.

Query : //title='TOPGUN'

```
SELECT   Path.docID
FROM     ElementType et, ElementNode en,
         Path p, PathMap pm, Text t
WHERE    et.etName ='title' and et.etID = en.etID and
         en.etID = pm.etID and
         pm.pathID = p.pathID and pm.docID = p.docID
         t.textValue = 'TOPGUN' and
         t.pathID = p.pathID and t.docID = p.docID and
```

그림 19. 완전포함질의
Fig. 19. Complete containment query.

5.4 근접질의

근접 질의는 노드 사이의 거리에 대한 근접 정도에 대한 결과를 목적으로 하는 질의를 가리킨다. 이 질의는 엘리먼트들, 속성들, 그리고 텍스트 내용에 대해서 해당 엘리먼트들의 거리를 측정하여 근접 정도에 만족하는 문서를 구하게 된다. 이와 같은 종류의 질의는 식 6과 같이 표시될 수 있다.

$$Distance\ (content1,\ content2) < distanceValue$$
$$distanceValue=(Ascendent_{level}-Descendent1_{level})+$$
$$(Ascendent_{level}ID-Descendent2_{level}) \quad (6)$$

예제 8. 그림 20은 두 개의 노드에 대해서 노드간의 거리를 측정하여 주어진 값보다 작을 경우 해당 XML 문서를 선택하는 근접 질의에 대한 예이다. 노드간의 거리는 노드 A와 노드 B가 주어질 때, A와 B를 모두 자손노드로 가지고 있는 노드들 중 가장 밑에 위치하고 있는 노드 C를 구하고 $(C_{level}-A_{level})+(C_{level}-B_{level})$ 값을 구하여 표현한다.

Query : Distance ('title', 'studio') < 5

```
SELECT   en2.docID
FROM     ElementType et1, ElementNode en1,
         ElementType et2, ElementNode en2
WHERE    et1.etName = 'title' and et1.etID = en1.etID and
         et2.etName = 'studio' and et2.etID = en2.etID and
         en1.docID = en2.docID
         (2 * (
         SELECT max (pm2.level)
         FROM     ElementType et1, ElementNode en1, PathMap pm1,
                  Elementtype et2, ElementNode en2, PathMap pm2,
         WHERE et1.etName = 'title' and et1.etID = en1.etID and
               et2.etName = 'studio' and et2.etID = en2.etID and
               pm1.enID = en1.enID and pm1.docID = en1.docID and
               pm2.enID = en2.enID and pm2.docID = en2.docID and
               pm1.docID = pm2.docID and pm1.enID = pm2.enID
         ) - en1.level + en2.level) < 5)
```

그림 20. 근접 질의
Fig. 20. Proximity query.

5. 텍스트 검색

XML 문서의 검색시 구조적인 정보를 포함하여 검색하는 경우 외에 질의 단어가 어느 문서에 포함되어 있

는지, 어느 엘리먼트에 포함되어 있는지 검색질의를 가리킨다. 본 논문에서 고려한 질의 형태는 bitwise 연산을 고려하여 식 7과 같이 처리할 수 있다.

$$(Term1\_TextFile_{Bitmap}\ op\ Term2\_TextFile_{Bitmap})\ AND\ (Term1\_TextConten_{tBitmap}\ op\ Term1\_TextContent_{Bitmap})\ AND\ (Term1\_TextContent_{docID}=Term2\_TextContent_{docID}) \quad (7)$$

예제 9. 그림 21는 'phantom'과 'opera' 단어가 한 엘리먼트에 포함되어 있는 XML 문서를 찾는 질의를 나타낸다.

Query : Text ('phantom' & 'opera')

```
1. Query on TextFile table
   Select termID, fileBitmap
   From TextFile
   Where termName = 'phantom' or termName = 'opera'
2. Get result from bitwise operation between step 1's bitmaps
3. Extract the docIDs on result bitmap
4. Query on TextContent table for each term
   Select docID, contentBitmap
   From TextContent
   Where termID=termID_step1 and docID = docID_step1
5. Get result from bitwise operation between step 4's bitmaps
6. Extract the docID on result
```

그림 21. 텍스트 검색
Fig. 21. Text retrieval.

VI. 성능 평가

본 절에서는 제안하는 색인 방법을 이용하여 관계형 데이터베이스를 기반으로 질의 및 저장 작업을 수행할 때, 기존의 좌표기반 색인 기법과 비교되는 모습을 설명하고자 한다.

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 상대위치좌표 기반의 방법에서는 조상노드와 자손노드간의 경로식별자 값이 같은지만을 비교하므로 적은 회수의 비교연산을 통하여 질의를 수행할 수 있다. 그림 22는 기존의 좌표기반 기법과 확장색인 비버사이의 포함관계의 검증을 위한 비교회수 차이를 보여준다.

2 노드 갱신에 의한 참여노드 개수 비교

상대위치좌표 방법에 기반한 색인에서는 XML 트리 구조의 어느 부분에서든지 노드의 삽입 및 삭제 등으로

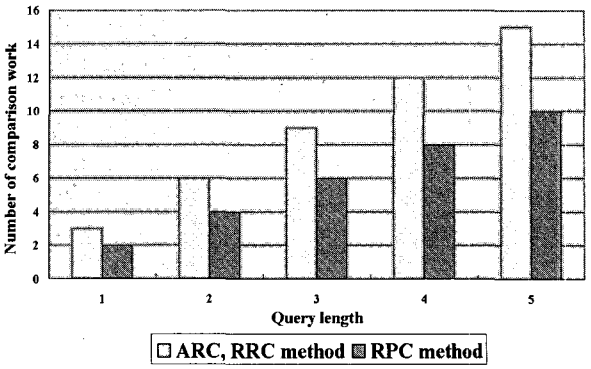


그림 22. 간접포함질의를 위한 비교 연산 회수
Fig. 22. Comparative operation for indirect containment query.

인한 갱신에 대해 고정된 변경 비용만을 요구한다. 반면에 기존의 영역 좌표 기반의 방법에서는 갱신에 대해 상당한 작업이 필요하게 되는데, 여기서는 리프 노드에 국한하여 노드의 삽입이나 삭제 등으로 인한 구조적인 변경이 일어나는 경우에 대해서 비교하였다.

비교를 위해서 XML 트리가 s개의 자식 노드를 가진 균형 트리라 가정하고, 트리의 깊이는 0부터 k까지라 가정하였다. 이 때, 절대영역좌표(ARC) 기반의 방법과 상대영역좌표(RRC) 기반의 방법에 대해 모든 리프노드들 사이로 노드의 삽입이 일어난다고 할 경우 업데이트 되는 노드의 수는 [5]에서 언급된 바와 같이 각각 식 8, 식 9로 나타낼 수 있으며, 상대위치좌표(RPC)의 경우에는 식 10과 같다.

$$\text{ARC method}: \frac{1}{2}\prod_{j=1}^{k} s_j\left(\sum_{m=1}^{k}\prod_{j=1}^{m} s_j + k + 2\right) \quad (8)$$

$$\text{RRC method}: \frac{1}{2}\prod_{j=1}^{k} s_j\left(\sum_{j=1}^{k} s_j + k + 2\right) \quad (9)$$

$$\text{RPC method}: 3\times s^{k-1} \quad (10)$$

이 식을 통해서 자식 노드들의 수인 s값이 증가함에 따라 전체적으로 갱신되어야 하는 노드들의 수를 그림 23과 같이 나타낼 수 있다.

6.3 텍스트 검색에서의 비교

본 논문에서는 텍스트 검색을 위해 BitmapInvert 기법을 제안하였고, 이를 비교하기 위해 역색인에 대한 방법과 비교하였다. 역색인의 경우 일반적인 텍스트 검색에 대해 좋은 성능을 보이고 있으며, [8]의 연구에서도 포함질의에서의 텍스트 질의를 위해 역색인을 적용하였다. 하지만, 역색인의 경우 여러 개의 텍스트 단어들을 비교해야 하는 경우 연산량이 많아지게 된다.

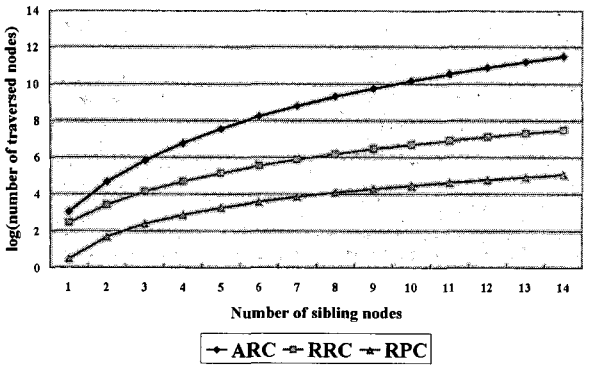


그림 23. 자식 노드 수의 증가에 따른 갱신되는 노드의 수
Fig. 23. Insert operation on leaf nodes according to increase the number of sibling node with depth is 5

비교를 위해 리프노드들의 수를 k라 하고, 질의에 포함되는 단어가 $T_1, T_2, \ldots, T_n$으로 n개이고, 임의 단어 T_i에 대해 하나의 XML 트리의 리프노드들에서 발생하는 수를 m_i라 할 때 비교 연산의 횟수는 각각 식 11, 식 12, 식 13과 같이 표현할 수 있다.

$$\text{Inverted Index based on ARC}: \quad 2n\prod_{i=1}^{n} m_i \qquad (11)$$

$$\text{Inverted Index based on RRC}: \quad n\prod_{i=1}^{n} m_i \qquad (12)$$

$$\text{BitmapInvert based on RRC}: \quad n \times k \qquad (13)$$

역색인의 방법의 경우 단어의 수에 따라 기하급수적으로 비교 연산의 수가 증가하게 되며, BitmapInvert 색인 방법의 경우 단어의 수가 증가하더라도 고정된 연산량만 늘어나게 되므로 문서 내에 발생 빈도가 높은 단어를 검색하거나 질의에 포함되는 단어의 수가 증가할수록 BitmapInvert 방법이 보다 더 좋은 성능을 보일 수 있다.

Ⅶ. 결 론

본 논문에서는 XML 문서에 대한 검색 성능을 유지하면서 갱신에 대한 처리를 고려하고, 텍스트 검색을 효율적으로 처리하기 위한 방법을 연구하였다. XML 문서 구조의 색인화를 위해 엘리먼트들간의 상대적인 관계를 표현할 수 있는 상대위치좌표를 제안하고, 복잡한 텍스트 검색을 수용할 수 있는 BitmapInvert 방법을 설계하였다. 그리고, 비트맵을 이용함으로 인해 발생할 수 잇는 갱신 문제를 해결하기 위해 PostUpdate 방법을 제안하였다.

그리고, 이 방법들을 이용하여 RDBMS에 기반한 색인을 구축하고 SQL문에 기반한 포함질의를 지원하고, 텍스트 검색시 bitwise 연산을 포함한 검색을 할 수 있어, 복잡한 내용을 검색할 경우 좋은 성능을 보일 수 있다. XML 문서의 내용이 변경되는 경우 영역좌표에 기반한 방법의 경우 상당히 많은 부분을 재계산해야 하는 단점이 있는 반면에 본 논문에서 제안한 방법은 적은 비용만을 요구하기 때문에 좋은 성능을 보일 수 있다.

향후 DTD나 XML 스키마 등을 이용하여 색인을 보다 효율적으로 설계하여 색긴의 크기를 감소키기고 보다 빠른 처리를 할 수 있는 방법에 대해서 연구가 필요하다.

참 고 문 헌

[1] ISO, "Information Processing-Text and Office System-Standard Generalized Markup Language (SGML)," ISO/IEC 8879, Oct. 15, 1986

[2] T. Bray, et al, "Extensible Markup Language (XML) 1.0 (Second Edition)," http://www.w3.org/TR/2000/REC-xml-20001006

[3] Ron Sacks-Davis, Tuong Dao, James A. Thom, and Justin Zobel, "Indexing documents for queries on structure, content and attributes," In International Symposium on Digital Media Information Base (DMIB'97), Nov. 1997

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

[5] Dao Dinh Kha, Masatoshi Yoshikawa, Shunsuke Uemura, "An XML indexing structure with relative region coordinate," Proceedings. 17th International Conference on Data Engineering (ICDE'2001), April 2001

[6] Jong P. Yoon, Vijay Raghavan, Venu Chakilam, "BitCube: a three-dimensional bitmap indexing for XML documents," Proceedings. Thirteenth International Conference on Scientific and Statistical Database Management (SDBM'2001), July 2001

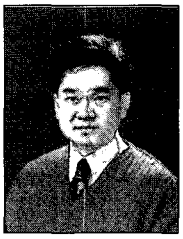
[7] J. Yoon, V. Rahgavan, and V. Chakilam, "BitCube: Clustering and Statistical Analysis for XML Documents," 13th International Conference on Scientific and Statistical Database Management, Virginia, July 2001

[8] C. Cheng, J. Naughton, D. DeWitt, Q. Luo, and G. Lohman, "On supporting containment queries in relational database management system", ACM SIGMOD, pp.425-436, 2001

[9] Hector Garcia-Molina, Jeffrey D.Ullman, Jennifer Wiom, "Database Systems: The Complete Book", Prentice Hall
[10] Silberschatz, Korth, Sudarshan, "Database System Concepts 4th Edition", Mc Graw Hill
[11] Jungsuk Song, Woosaeng Kim, "Extensible index technique for storing and retrieving XML documents", The 4th International Conference on Computer and Information Technology, pp 280-287, Sep. 2004
[12] Tackgon Kim, Woosaeng Kim, "A XML Index Technique with Relative Position Coordinate for Storing and Retrieving XML documents", International Tecnical Conference on Cirrcuits/ Systems, Computers and Communications (ITC-CSCC), 20th Conference, vol 2, pp.787-788, July 2005
[13] Igor Tatarinov, Zachary G. Ives, Alon Y. Halevy, Daniel S. Weld, "Updating XML", ACM SIGMOD, pp413-424, May 2001

저 자 소 개

김 택 곤 (정회원)
1999년 광운대학교 전자계산학과 학사
2001년 광운대학교 전자계산학과 석사
2006년 광운대학교 컴퓨터과학과 박사
<주관심분야 : 데이터베이스, 멀티미디어시스템>



김 우 생 (정회원)
1982년 서울대학교 수료
1985년 Univ. of Texas at Austin 전산학과 졸업
1987년 Univ. of Minnesota 전산학과 석사
1991년 Univ. of Minnesota 전산학과 박사
2001년 UC 버클리 대학 교환 교수
1992년~현재 광운대학교 컴퓨터공학부 교수
<주관심분야 : 멀티미디어시스템>