XML 문서의 상향식 질의처리를 지원하는 효율적인 색인구조☆

An Efficient Index Structure for Bottom-Up Query Processing of XML Documents

서 동 민* 김 은 재** 성 동 욱*** 유 재 수**** 조 기 형*****
Dong-Min Seo Eun-Jae Kim Dong-Ook Seong Jae-Soo Yoo Ki-Hyung Cho

요 약

XML에서의 질의는 경로 질의를 사용하고, 경로 질의를 효과적으로 처리하기 위한 여러 가지 색인 기법들이 연구되었다. 최근에는 구조 조인 기법과 더불어 접미사(suffix) 트리를 이용한 기법이 제안되고 있다. 그 중에서 가장 대표적인 기법이 ViST(Virtual Suffix Tree)이다. ViST는 질의 처리 시간을 줄이기 위해서 접미사 트리와 B+ 트리를 이용하여 질의 처리에 참여하는 엘리먼트만을 비교한다. 그러나 실제 문서에서 조상-후손 관계가 아닌 엘리먼트도 후손으로 보고 처리하는 문제점으로 인해 디스크 접근이 많아지는 비효율성을 지닌다. 따라서 본 논문에서는 ViST의 문제점을 해결하는 색인구조를 제안하고, 이 색인구조에 알맞은 질의 처리기법을 제안한다. 그리고 다양한 질의 처리 실험을 통해 기존에 제안된 색인구조에 비해 향상된 질의 처리 성능을 나타냄을 보인다.

Abstract

A path query is used in XML. Several index structures have been studied for processing the path query efficiently. In recent, the index schemes using suffix tree with structure join method were proposed. ViST is the most representative method among such methods. ViST processes the query using suffix tree and uses B+-tree to reduce the search time of the documents. However, it significantly degrades the search performance when processing the path query. The reason is that it regards the element that is not ancestor-descendant relation in the document as a descendent. In this paper, we propose an efficient index structure to solve the problem of ViST. The query processing method suitable to the index structure is also proposed. It is shown through various experiments that the proposed index structure outperforms the existing index structure in terms of the query processing time.

☞ Keyword : XML 색인구조, XML 질의처리, 상향식 질의처리

1. 서 론

XML(eXtensible Markup Language)은 문서 구성 요소들 사이에 계층적인 구조를 가지고 있으나 그 형태가 일정하게 고정된 스키마(scheme)를 따를 필요가 없는 반구조적인(semi structured) 특성을 지닌다[1]. 이러한 특성을 지니는 XML 데이터의 처리를 위해 일반적으로 트리 구조의 모델을 사용한다. 그리고 XML 질의 처리를 위해 XPath[2], Quilt[3], XML-QL[4], XQuery[5] 등과 같은 질의 언어(query language)들이 연구되었고 이러한 질의 언어들은 그래프(graph)로 표현할 수 있다. XML 문서에 대한 경로 질의(path query)를 효율적으로 처리하기 위한 색인 기법에 대한 연구도 많이 수행되었는데, 초기에는 주로 경로 색인(path index) 기법에 대한 연구가 이루어졌다. 이 기법은 XML 데이터 구조에서 발생 가능한 모든 경로에 대한 색인 그래프를 별도로 구축하고 경로 질의 처리를 위해 원본

* 정 회 원 : 충북대학교 정보통신공학과 박사과정
dmseo@chungbuk.ac.kr(제 1저자)
** 준 회 원 : (주)다음커뮤니케이션 R&D센터 연구원
defeat@daumcorp.com(공동저자)
*** 준 회 원 : 충북대학교 정보통신공학과 석사과정
sergei@netdb.cbnu.ac.kr(공동저자)
**** 정 회 원 : 충북대학교 전기전자컴퓨터공학부 교수
yjs@chungbuk.ac.kr(공동저자)
***** 정 회 원 : 충북대학교 전기전자컴퓨터공학부 교수
khjoe@chungbuk.ac.kr(교신저자)
☆ 이 논문은 2005학년도 충북대학교 학술연구지원사업의 연구비지원에 의하여 연구되었음

데이터 그래프의 탐색이 아닌 경로 색인 그래프를 탐색함으로써 탐색 공간의 크기를 줄여 질의 처리 비용을 줄인다[6,7].

최근에는 조상-후손 관계(ancestor-descendant relationship) 등의 포함 질의를 효율적으로 처리할 수 있는 구조 조인(structure join) 기법에 대한 연구가 많이 이루어졌다[8-11]. 하지만 구조 조인 기법은 단순 경로 질의를 효율적으로 처리하지만 분기 구조(branch structure) 질의는 항상 두 개의 부 질의(sub query)로 분해한 뒤에 각각 경로 질의를 통해 처리하고 최종 결과는 많은 비용이 요구되는 부 질의 결과들에 대한 조인(join) 연산을 통해 얻는다.

구조 조인 기법의 문제를 해결하기 위해 최근에는 접미사 트리를 이용해 엘리먼트 간의 포함 관계를 살펴 질의를 처리하고, 문서 전체에 대한 탐색을 줄이기 위해 B+ 트리를 이용하여 해당하는 엘리먼트들만을 비교하여 질의를 처리하는 기법들이 제안되었다[12,13]. 하지만 접미사 트리에서 사용하는 노드 번호 부여 기법의 특성으로 인해 질의 처리시 실제 문서에서 자식이 아닌 엘리먼트도 자식으로 보고 불필요한 노드 접근이 발생해 질의 처리 속도가 감소하는 문제가 발생한다.

이러한 문제를 해결하기 위해 본 논문에서는 효율적인 번호 부여 기법을 사용하는 색인 구조를 제안하고, 이 색인구조에 알맞은 상향식 질의 처리 기법을 제안한다. 그리고 다양한 환경에서 성능평가를 수행하여 제안하는 색인구조가 ViST보다 우수함을 보인다.

본 논문은 다음과 같이 구성되어 있다. 2장에서는 기존에 제안되었던 구조 조인 기법과 이를 개선시킨 ViST에 대하여 논의한다. 3장에서는 ViST의 질의 처리 문제점을 분석하고, 이를 해결하기 위해 제안하는 색인 구조에 대해 설명한다. 4장에서는 다양한 실험을 통해 제안하는 색인 구조의 우수성을 보이고, 마지막으로 5장에서 결론을 맺는다.

2. 관련연구

본 장에서는 두 엘리먼트들 간의 조상-후손 관계를 빠르게 검사하기 위한 번호 부여 기법과 이를 이용하여 질의 처리를 효율적으로 하기 위한 구조 조인 기법들과 접미사 트리를 이용한 기법에 대해 살펴본다.

2.1 XML

XML 문서는 데이터(data), 구조(structure), 표현(presentation)의 속성을 가진다. 데이터는 문서 자체에서 표현하고자 하는 정보를 나타내고, 구조는 문서의 구조 자체를 나타냄으로써 문서의 구조적인 정보를 제공한다. 표현은 XML문서가 표현되는 방식을 나타내는데 표현 방식에 따라 여러 상이한 형태(HTML, WML)로 정보의 손실 없이 표현할 수 있다.

```
<Purchases>
  <Seller>
    <Name>dell</Name>
    <Item>
      <Name>part#1</Name>
      <Manufacturer>ibm</Manufacture>
    </Item>
    <Location>boston</Location>
  </Seller>
  <Buyer>
    <Name>Smith</Name>
    <Location>newyork</Location>
  </Buyer>
</Purchases>
```

〈그림 1〉 견본 XML 문서

그리고 XML 문서는 일정하게 고정된 스키마를 따를 필요가 없는 반구조적인 특성을 지닌다. 이러한 특성을 지니는 XML 데이터 처리를 위해 일반적으로 트리 구조 모델을 사용한다. 그림 2는 그림 1의 XML 문서를 트리 구조 모델로 표현한 것이다. 이러한 반구조적인 XML 데이터를 처리하기 위해 XPath, Quilt, XML-QL, XQuery 등과 같은 여러 종류의 질의 언어들이 연구되어 왔으며 이러한 질의 언어들은 그래프

로 표현할 수 있다.

그림 3은 그래프로 표현한 4가지 간단한 XML 질의의 예로서 그림 3의 (a)를 XPath로 표현하면 "/Purchases/Seller/Item/Manufacturer"이다.

2.2 번호 부여 기법(Numbering Scheme)

그림 1과 같은 XML 문서에서 경로 질의의 처리를 빠르게 하기 위해서는 XML 문서내의 계층에서 노드들 간의 조상-후손 관계를 빠르게 결정해야 한다. 예를 들면 "//Seller/Location"이라는 질의를 처리하기 위해서는 그림 2의 트리 구조에서 Seller 엘리먼트 아래의 부 트리(sub tree)를 모두 순회해야 한다. 하지만 이 방법은 너무 비효율적이기 때문에 번호 부여 기법을 이용하여 부 트리에 대한 전체 순회를 피할 수 있다.

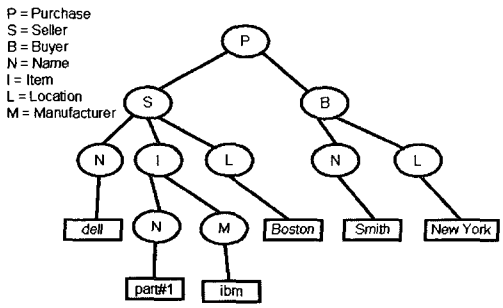
2.2.1 Dietz의 번호 부여 기법

그림 4는 Dietz의 번호 부여 기법으로 XML 문서의 각 엘리먼트를 preorder, postorder로 순회하면서 번호를 부여하는 방식이다[14].

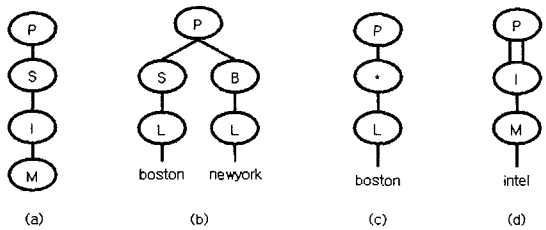
x, y의 두 엘리먼트가 주어졌을 때, x의 preorder가 y의 preorder보다 작고 x의 postorder가 y의 postorder보다 크면 x는 y의 조상이다. 그리고 level의 차이가 1이면 그 엘리먼트가 부모가 된다. 이렇게 번호 부여 기법을 활용하면 두 엘리먼트가 주어졌을 때, 한 엘리먼트의 부 트리를 모두 순회하지 않고도 노드의 preorder와 postorder만을 비교함으로써 효과적으로 구조적 관계를 알아낼 수 있다. 그러나 Dietz의 번호 부여 기법은 XML문서의 삽입, 삭제로 인해 구조가 변화할 때마다 번호 부여를 다시 해야 하는 문제점이 있다.

2.2.2 Durable 번호 부여 기법

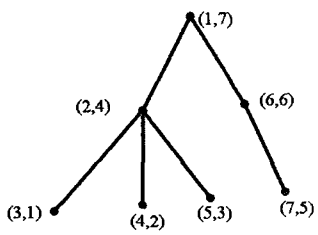
Dietz의 번호 부여 기법의 문제점을 보완한 것이 그림 5와 같은 Durable 번호 부여 기법 또는 Extended Preorder 번호 부여 기법이다[8].



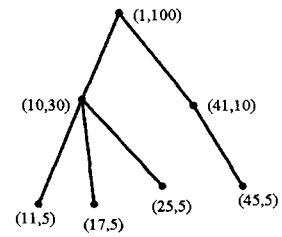
〈그림 2〉 XML 문서의 트리 구조 표현



〈그림 3〉 그래프로 표현한 XML 질의들



〈그림 4〉 Dietz의 번호 부여기법



〈그림 5〉 Durable 번호 부여 기법

이 기법은 각 엘리먼트에 preorder 방식으로 order가 후손들의 size 값들의 합보다 크거나 같은 size값을 주도록 size 값을 부여하여 (order, size)쌍을 생성한다. 그리고 x, y 두 엘리먼트가 주어졌을 때, x의 order가 y의 order보다 작고, y의 order+size가 x의 order+size보다 작거나 같으면 x는 y의 조상이 된다. 이 기법은 새로 삽입될 노드들을 고려하여 order 값과 size 값 사이에 여유를 크게 둠으로써 Dietz의 번호 부여 기법이 가지고 있는 문제점을 해결할 수 있다.

2.3 구조 조인 기법

XML 질의를 효과적으로 처리하기 위해서는 번호 부여 기법을 이용하여 두 엘리먼트의 조상-후손 관계를 빠르게 판단하는 것과 이러한 구조적 관계를 검사할 후보 리스트를 색인을 통해 빠르게 찾아내야 한다. 또한 색인의 검색 결과로 주어진 후보 리스트에서 구조적 관계를 만족하는 엘리먼트들을 효과적으로 찾아야 한다.

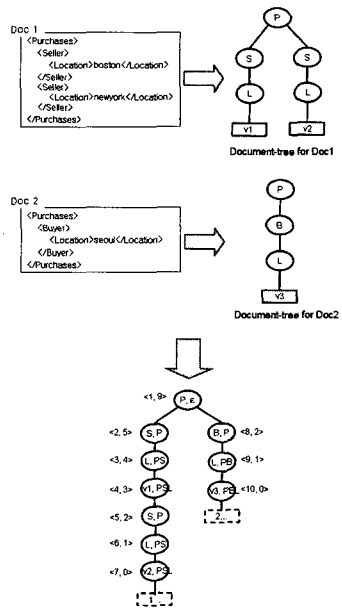
STJ(Stack Tree Join) 알고리즘은 대표적인 구조 조인 기법으로 트리의 레벨과 같은 크기의 스택을 사용하여 깊이 우선 순회 탐색 시간을 줄이는 방법을 이용한 것이다. 이 순회 기법은 트리의 모든 조상-후손 관계에 있어서 항상 후손 노드가 조상 노드보다 스택 상에서 더 높은 곳에 나타난다는 것을 응용했다. 이를 이용 TMJ(Tree-Merge-Join) 알고리즘이 모든 노드들을 비교하는 비용을 줄였다[9].

하지만 이 기법도 조인에 필요한 각 엘리먼트에 대응되는 모든 데이터를 찾는 비용이 크다는 점과, TMJ 조인보다는 불필요한 비교 연산을 줄였지만 여전히 리스트 내에 불필요한 요소들이 있다는 것이 단점이다.

2.4 접미사 트리를 이용한 질의 처리 기법

구조 조인 기법에서 발생하는 조인 비용을 줄이기 위해 최근에는 접미사 트리를 이용해 엘리먼트들의 포함 관계를 살펴 질의를 처리하는 기법이 제안되었다[12]. 이 기법은 XML 문서 내의 구조적인 관계를 접미사 트리로 표현하고, 색인을 이용 경로 매칭(sequence matching)을 통해 처리하는 기법이다. XML 질의를 경로 매칭을 통해 모델링하는 목적은 질의 처리에 있어 제거 가능한 불필요한 조인 연산을 제거하는 것이다. 즉, 경로 매칭을 통해 분기가 있는 질의를 분해하지 않고 처리함으로써 조인 연산의 비용을 줄인다.

그림 6은 두 XML 문서에 대한 접미사 트리를 나타낸 것이다. 접미사 트리에서의 질의 처리는 문자열 매칭 알고리즘을 통해 트리의 모든 노드들을 순회하면서 질의를 구성하는 노드들이 접미사 트리에 포함되는지를 살펴 처리한다. 하지만 이렇게 질의를 처리하기 위해서는 많은 양의 부 트리를 순회해야 하기 때문에 비용이 많이 드는 단점이 있다. 또한 접미사 트리는 메인 메모리 기반의 색인구조이고 상업용 DBMS에 적합하지 못한 문제가 있다.



〈그림 6〉 XML 문서 Doc1, Doc2에 대한 접미사 트리

ViST(Virtual Suffix Tree)는 접미사 트리의 문제점을 개선한 가장 대표적인 기법이다. ViST는 기존에 제안되어진 질의 처리 기법의 문제점인 조인 비용을 줄이기 위해서 문서를 접미사 트리로 변환한다. 그리고 질의 처리 시 요구되는 접미사 트리의 순회 비용을 줄이기 위해 각각의 노드마다 번호를 부여하고, 번호가 할당된 접미사 트리의 노드들을 B+ 트리에 구축함으로써 최소 노드 접근 기법을 제공한다[13].

접미사 트리에서는 두 노드 간에 D-Ancestorship 또는 S-Ancestorship 관계를 가진다. D-Ancestorship은 두 노드가 XML 문서에서 조상-후손 관계를 가지는 경우를 나타내고, S-Ancestorship은 접미사 트리에서 조상-후손 관계를 가지는 경우를 나타낸다. D-Ancestorship은 두 노드의 접두사(prefix)를 비교하여 결정할 수 있다. 하지만 S-Ancestorship은 추가적인 정보를 필요로 한다. 이를 위해 ViST는 두 노드간의 S-Ancestorship 관계를 결정하기 위해 추가적으로 각각의 접미사 트리의 노드 x마다 (n_x, $size_x$)로 번호를 붙인다. n_x는 접미사 트리의 전위 순회를 통해 번호를 부여하고, $size_x$는 접미사 트리 안에서 노드 x의 전체 자식의 수로 표현된다. 이렇게 번호를 부여한 후에는 S-Ancestorship 관계는 x와 y 두 노드가 각각 (n_x, $size_x$), (n_y, $size_y$)라고 가정했을 때 $n_y \in (n_x, n_x+size_x)$를 만족하면 노드 x는 노드 y의 S-Ancestor이다.

B+ 트리의 구축은 먼저 (symbol, prefix)쌍을 키 값으로 하여 모든 접미사 트리의 노드를 D-Ancestor B+ 트리에 삽입한다. 모든 노드 x는 같은 (symbol, prefix)에 삽입된다. 그리고 각 노드의 n_x를 키 값으로 하여 S-Ancestor B+ 트리에 삽입한다. 마지막으로 노드 x의 DocID를 DocID B+ 트리에 삽입한다. 그림 7은 그림 6의 접미사 트리를 D-Ancestor B+ 트리와 S-Ancestor B+ 트리로 변환한 것이다.

그림 8은 그림 7의 D-Ancestor B+ 트리와 S-Ancestor B+ 트리를 이용하여 나온 결과 값으로 해당 문서를 찾을 수 있는 DocID B+ 트리를 나타낸 것이다.

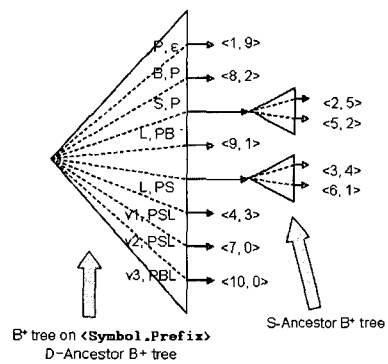
3. 제안하는 색인구조

본 장에서는 ViST의 문제점과 이를 해결하기 위한 색인구조를 제안하고, 제안하는 색인구조에서 사용하는 번호 부여 기법과 이에 적합한 질의 처리 기법인 상향식 질의 처리 기법에 대해 설명한다.

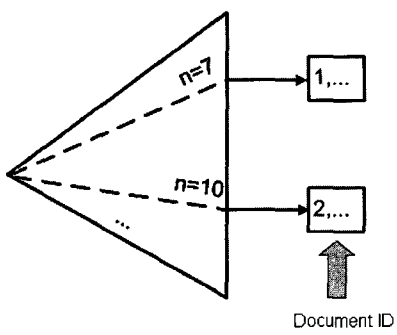
3.1 ViST의 문제점

3.1.1 질의 처리시 자식이 아닌 노드 접근 문제

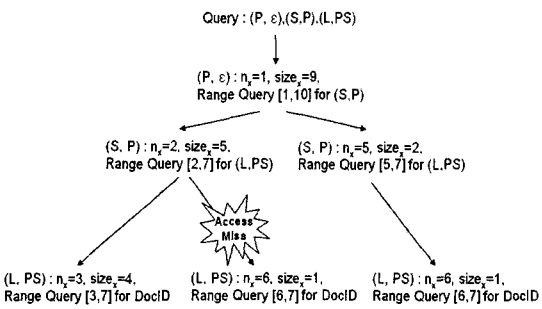
ViST에서의 질의 처리 기법은 그림 9와 같다.



〈그림 7〉 ViST의 색인 구조



〈그림 8〉 ViST의 DocID B+ 트리



〈그림 9〉 ViST의 질의 처리

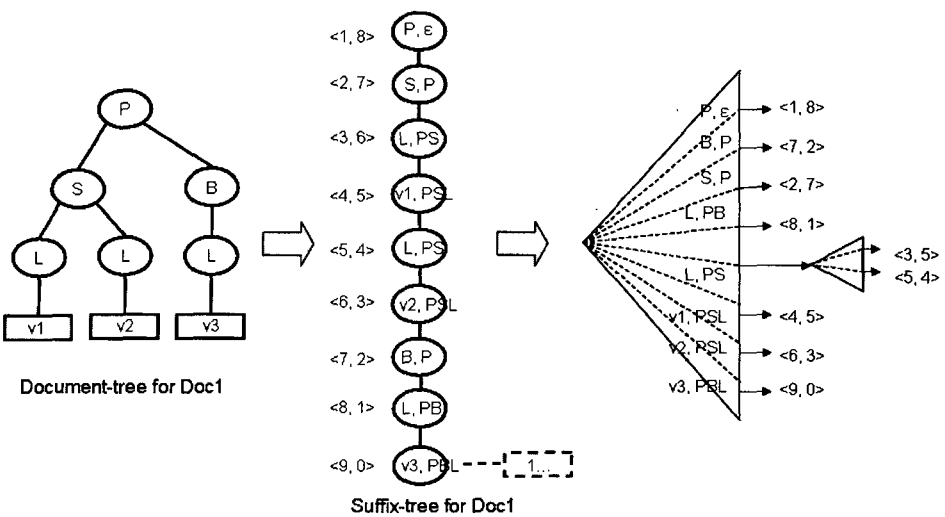
"Purchase/Seller/Location" 질의가 들어오면 먼저 질의를 분석 후 Structure-Encoded 경로를 만든다. 그리고 처음 나오는 (symbol, prefix)쌍인 (P, ε)을 그림 7의 D-Ancestor B+ 트리를 통해 검색하여 해당 n_x, $size_x$의 값을 획득한다. 이 값을 이용하여 다음 순서쌍인 (S, P)에 대해 (n_x, n_x+$size_x$)로 범위 질의를 한다. 그리고 범위 질의 결과들을 마지막 순서쌍인 (L, PS)에 대해 (n_x, n_x+$size_x$)로 각각 범위 질의를 수행한다. 마지막으로 요청된 질의에 부합되는 문서를 찾기 위해 질의 대상이 되는 경로의 마지막 노드의 n_x 값을 이용하여 그림 8의 DocID B+ 트리를 질의하여 처리한다.

하지만 ViST에서 두 노드의 S-Ancestorship 관계를 표현하기 위해 사용하는 번호 부여 기법으로 인해 그림 9처럼 실제 문서 트리에서 자식이 아닌 노드에 대한 노드도 자식으로 보는 문제가 발생한다. 즉, n_x가 2인 SP의 자식은 n_x가 3인 LPS만이 존재하지만 n_x가 6인 LPS도 자식으로 보고 비교하는 연산을 하게 된다.

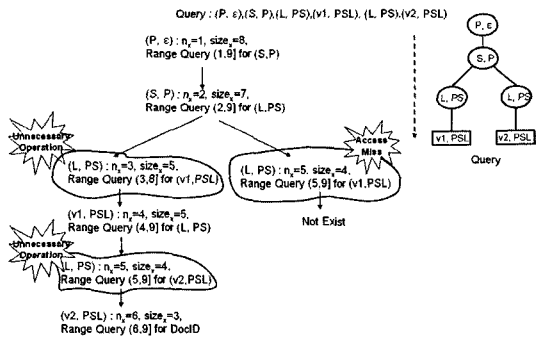
3.1.2 접미사의 특징을 고려하지 않은 문제

그림 10과 같은 문서에 대한 질의 처리는 그림 11처럼 먼저 질의를 분석한 후 순서대로 조상-후손 관계를 판별해 나간다.

하지만 접미사 트리의 접두사가 루트 엘리먼트부터 현재 엘리먼트까지의 경로 정보를 나타낸다는 특성을 고려하지 않고 질의를 처리하기 때문에 그림 11처럼 불필요한 노드를 접근하는 문제점이 발생한다. 즉, (v1, PSL)의 정보에 자신의 상위 노드에 (L, PS)를 포함하고 있다는 정보가 있으므로 (L, PS)를 처리하지 않아도 되지만 처리를 한다.



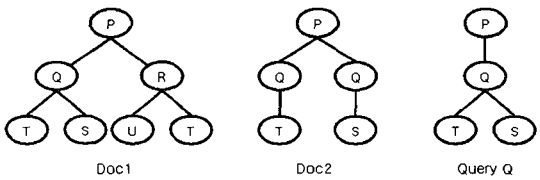
〈그림 10〉 XML 문서에 대한 ViST 구축 기법



〈그림 11〉 ViST의 분기 질의 처리

3.1.3 False Alarm 문제

분기 질의 처리 시 문서와 질의 Q의 구조가 틀려도 질의의 Structure-Encoded 경로가 문서에 포함되어 결과로 나타내는 False Alarm이 발생할 수 있다[13]. 예를 들면 그림 12의 Q와 같은 질의를 보면 문서 1과는 같은 구조를 같지만 문서 2와는 다른 구조이다. 하지만 문서 2의 Structure-Encoded 경로는 (P, ε)(Q, P)(T, PQ)(Q, P)(S, PQ)가 되므로 질의 Q의 Structure-Encoded 경로인 (P, ε)(Q, P)(T, PQ)(S, PQ)를 포함하게 된다. 그래서 구조는 틀리지만 같은 문서로 판단하고 결과로 나타내는 문제가 발생한다.



D1 = (P, ε)(Q,P)(T,PQ)(S,PQ)(R,P)(U,PR)(T,PR)
D2 = (P,ε)(Q,P)(T,PQ) (Q,P)(S,PQ)
Q = (P,ε)(Q,P)(T,PQ) (S,PQ)

〈그림 12〉 ViST에서 발생할 수 있는 False Alarm

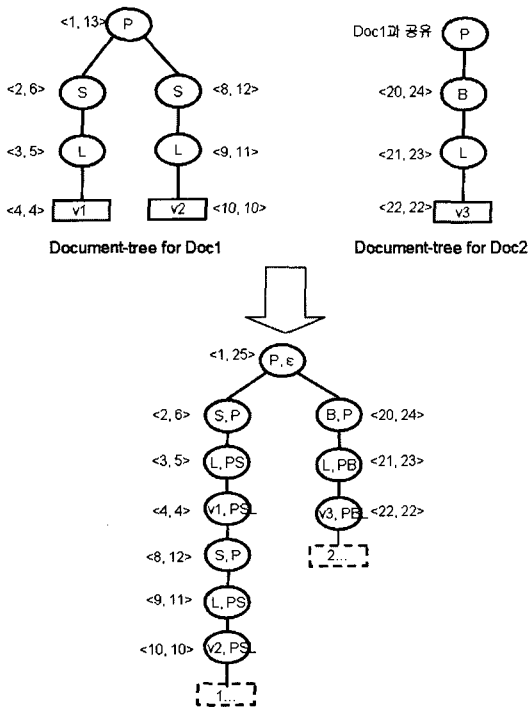
3.2 제안하는 색인구조의 구축 및 질의 처리 기법

앞서 언급한 문제점을 해결하기 위해 제안하는 색인구조는 새로운 번호 부여 기법과, 색인구조에 적합한 상향식 질의 처리 기법을 제안한다.

3.2.1 제안하는 번호 부여 기법

접미사 트리에서 사용하는 번호 부여 기법에 따른 문제를 해결하기 위해 Durable 번호 부여 기법[8]을 사용하였다. 그림 13은 제안한 번호 부여 기법을 이용하여 그림 6과 같은 문서들에 대한 접미사 트리이다.

일례로, 그림 13에서 <2, 6>값을 가지는 노드 (S, P)와 <9, 11>값을 가지는 노드 (L, PS)의 조상-후손 관계를 Durable 번호 부여 기법을 통해 비교하면 조상-후손 관계가 성립되지 않는다, 즉, ViST의 문제인 자식이 아닌 노드 접근 문제가 해결된다.

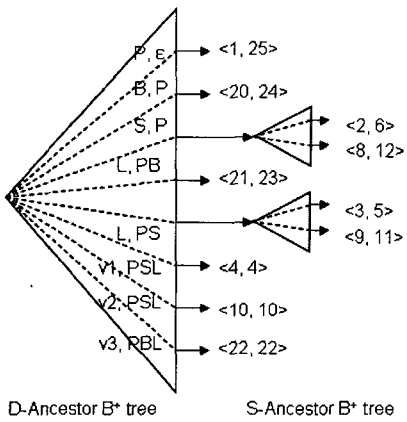


〈그림 13〉 제안하는 번호 부여 기법을 사용해서 구축된 접미사 트리

3.2.2 제안하는 색인 구축 기법

D-Ancestor B+ 트리는 ViST의 기법과 같은 방법 (symbol, prefix)쌍을 키 값으로 하여 삽입한다. 그리고 S-Ancestor B+ 트리는 ViST와 같

은 방법으로 삽입하지만 키 값을 Durable 번호 부여 기법에서 사용하고 있는 (order, size)로 변경하여 삽입한다. 하지만 번호 부여 기법이 달라 질의 대상이 되는 경로의 마지막 노드를 검색할 수 없어 ViST에서 사용한 DocID B+ 트리는 사용할 수 없게 된다. 따라서 제안한 번호 부여 기법을 이용 문서를 검색할 수 있는 방법이 필요하다. 그래서 영역 질의를 할 수 있도록 DocID B+ 트리를 사용하지 않고, DocID R 트리를 사용한다. 그림 14는 제안된 D-Ancestor B+ 트리와 S-Ancestor B+ 트리를 나타낸 것으로 그림 13과 같은 문서를 트리로 만든 예이다.

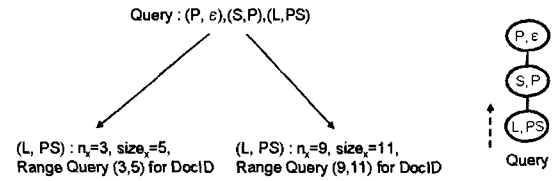


〈그림 14〉 제안하는 색인 구조

3.2.3 제안하는 상향식 질의 처리 알고리즘

각 노드의 접두사는 현재 노드까지의 절대 경로를 나타내므로, 이 정보를 사용하면 ViST에서 질의 처리시 요구되는 중간 노드 검색 비용을 줄일 수 있다. 따라서 질의를 상향식으로 처리하는 기법을 사용하게 되면 단일 경로 질의의 경우는 마지막 노드만 처리하면 된다.

그림 15와 같이 상향식 질의 처리는 그림 9에서 일어나는 중간 계산 과정을 생략하고 마지막 노드만을 검색함으로써 검색 시간을 줄일 수 있다. 즉, (S, P)와 (L, PS)에 해당하는 검색 과정이 생략된다. 예를 들면, 그림 9에서 처리하고 있는 "Purchase/Seller/Location"와 같은 질의를 처리하는 경우는 마지막 노드인 (L, PS)만을 처리하면 된다. 따라서 D-Ancestor B+ 트리에서 (L, PS)를 키로 하여 만족하는 S-Ancestor B+ 트리를 찾은 뒤, 이 S-Ancestor B+ 트리의 모든 값들을 DocID R 트리에서 영영 질의를 통해 만족하는 문서를 찾게 된다.



〈그림 15〉 상향식 질의 처리

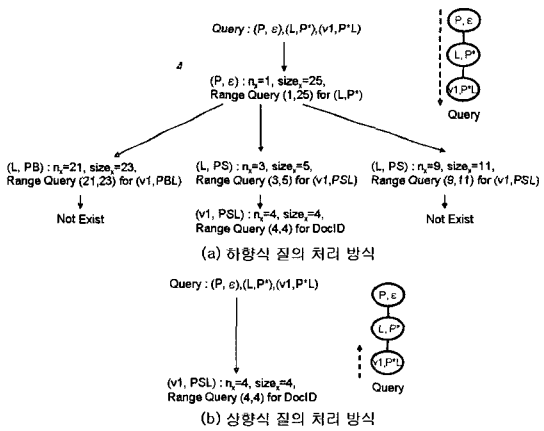
그림 16은 상향식 질의 처리 모듈의 의사코드를 나타내고 있다.

```
상향식 질의 처리 알고리즘
입력 : Q = q, 마지막 질의 sequence
       (symbol, prefix)가 삽입된 D-Ancestor
       B+ 트리
       <n, size>가 삽입된 S-Ancestor B+
       Tree
       노드의 n과 document IDs가 삽입된
       DocID R 트리
출력 : query sequence와 동일한 데이터가 있는
       문서번호
처리과정 : Search(<0, size>); /*
          <0, size> 는 Suffix 트리의 루트노드 /*

Function Search(<n, size>)
1: T D-Ancestor B+ 트리를 탐색 q로 표현되는
   S-Ancestor B+ 트리를 찾는다.
   N 검색된 S-Ancestor B+ 트리에서 [n,
   n+size]인 노드를 범위 질의하여 찾는다.
2: DocID R 트리에서 [n, n+size]로 범위 질
   의하여 해당 문서 번호를 출력한다.
3: end
```

〈그림 16〉 상향식 질의 처리 모듈의 의사코드

그리고 그림 17과 같이 와일드-카드(wild-card)가 포함된 질의의 경우도 상향식으로 처리함으로써 하향식으로 처리할 때 발생하는 많은 중간 검색 과정을 줄일 수 있다. 예를 들면, 그림 17의 질의를 하향식으로 처리하게 되면 (L, P*)을 그림 14의 D-Ancestor B+ 트리에서 검색을 하게 되므로 (L, PB), (L, PS) 두 가지 경우가 나오게 되며, 그 각각의 경우에 대하여 나머지 질의를 처리하게 되므로 많은 비용이 발생한다. 하지만 같은 질의를 상향식으로 처리하면 먼저 (v1, P*L)을 D-Ancestor B+ 트리에서 검색하여 (v1, PSL)이라는 결과 값만 나오게 되므로 중간 질의 처리 과정을 처리 하지 않는다.



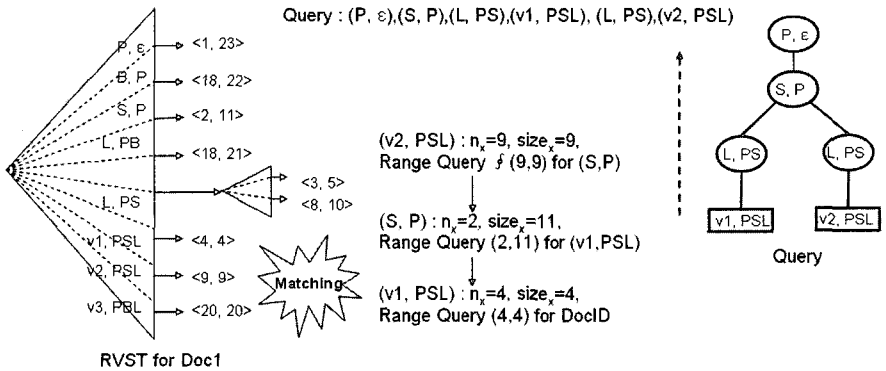
〈그림 17〉 '*'가 포함된 질의의 상향식 질의 처리

3.2.4 분기 질의에 대한 상향식 질의 처리 알고리즘

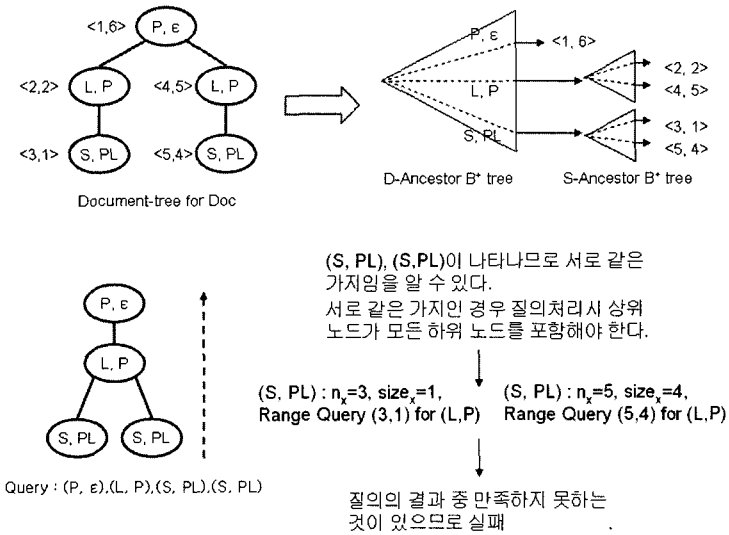
그림 18은 제안하는 색인구조에서 분기 질의를 처리하는 과정을 보여준다. 분기가 있는 질의의 경우에는 먼저 마지막 노드인 (v2, PSL)을 D-Ancestor B+ 트리에서 검색 후 (L, PS)에 대해 범위 질의를 수행한다.

하지만 (L, PS)는 (v2, PSL)에 의해 경로를 보장 받는다. 그리고 (v1, PSL)과 (L, PS)의 접미사를 통해 서로 다른 가지에 있음을 알 수 있으므로 (L, PS)에 대한 범위 질의를 (S, P)에 대한 범위 질의로 변경하여 수행한다. 그리고 나머지 (v1, PSL)에 관해 범위 질의를 수행한다.

또한 상향식으로 질의를 처리하므로 그림 19와 같이 질의의 마지막부터 필요한 노드들만을 접근하여 처리하므로 구조적으로 틀린 문서를 판별해 나갈 수 있게 된다. 따라서 자연스럽게 False Alarm이 해결된다. 예를 들면, 그림 19의 질의 처리시 마지막 노드에서 (S, PL)이 중복되어 나타나므로 부모 노드가 같음을 알 수 있다. 따라서 (L, P)안에 모든 (S, PL)이 포함되는지 비교한다. 하지만 (S, PL)의 값인 <5, 4>는 (L, P)의 값인 <2, 2>와 비교했을 때 order인 5는 2보다 크지만 size 4는 2보다 크므로 결과를 만족하지 못한다. (S, PL)의 값인 <3, 1>은 (L, P)의 값인 <4, 5>와 비교했을 때 order인 3이 4보



〈그림 18〉 제안하는 색인구조를 이용한 분기 질의 처리



〈그림 19〉 분기 질의 처리시 False Alarms의 해결

다 크므로 결과를 만족하지 못한다. 따라서 요청한 질의는 문서에 존재하지 않는다.

4. 구현 및 성능 평가

본 장에서는 설계한 색인구조의 구현 내용과 실험 환경에 대해 기술하고 ViST와의 성능 평가를 통해 제안한 색인 구조의 우수성을 입증한다.

4.1 실험 환경

성능 평가에 사용된 시스템은 Pentium(R) 4 3.00GHz 프로세서에 512Mbytes의 메모리를 가지며, 운영체제는 Microsoft Windows XP Professional을 사용하였다. 사용한 언어는 C++를 사용하였으며, 컴파일러는 Microsoft Visual Studio .Net 2003을 사용하였다. 그리고 XML 문서내의 엘리먼트와 애트리뷰트 등을 분석하여 트리에 삽입하기 위해 Microsoft사에서 제공하는 SAX 파서를 사용하였다[15]. 실험에 사용된 데이터 집합은 Niagara 프로젝트에서 사용된 XML 문서 데이터들을 사용하였다[16].

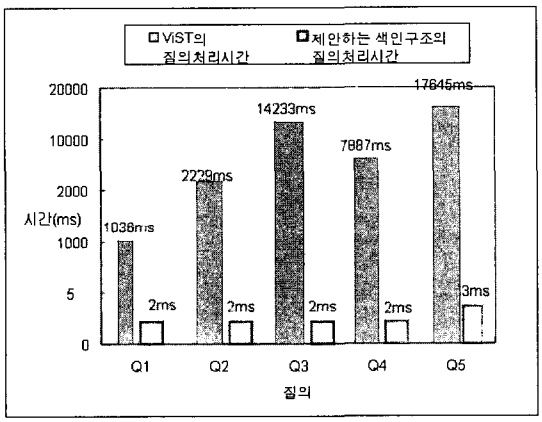
4.2 실험 결과

표 1은 실험에서 사용한 다양한 종류의 질의 형태를 보여준다. 단순 경로 질의 성능 평가를 위해서 Q1~Q3처럼 질의의 깊이를 한 단계씩 증가시키며 질의 처리 시간을 측정하였다. 그리고 Q4와 같이 와일드-카드가 포함된 질의 성능 평가는 단순 경로 질의 성능 평가처럼 질의의 깊이를 증가시키고 와일드-카드의 위치를 변경시키면서 질의 처리 시간을 측정하였다. Q5와 같이 분기가 포함된 질의 성능 평가는 분기가 일어난 곳부터 하위 노드의 수를 다르게 하며 질의 처리 시간을 측정하였다.

〈표 1〉 실험에 사용한 질의 형태

질의	질의 표현식
Q1	/W4F_DOC/Actor
Q2	/W4F_DOC/Actor/Name/FirstName/Robert
Q3	/W4F_DOC/Actor/Filmography/Movie/Title/Heat
Q4	/W4F_DOC/Actor/Filmography/*/Title/Heat
Q5	/W4F_DOC/Actor/Filmography/Movie[Title='Ronin']/[Year='1998']

그림 20은 제안하는 색인구조가 ViST보다 질의 처리시간이 확연히 감소함을 통해 성능이 우수함을 나타내고 있다. 또한 깊이가 낮은 질의의 경우는 비슷한 성능을 보이고 있지만, 상향식 질의처리의 장점으로 인해 깊이가 높아질수록 더 좋은 성능을 보임을 알 수 있다. 이는 제안하는 번호 부여 기법이 불필요한 노드의 접근을 피하고, 동시에 상향식으로 질의를 처리함으로 중간 처리과정시 드는 노드 접근 비용을 감소시켰기 때문이다.



〈그림 20〉 ViST와 제안하는 색인구조의 질의 처리 시간 비교

그리고 ViST에서는 False Alarm 문제로 인해 질의에 대한 잘못된 결과가 나타나는데 반해 제안하는 색인구조에서는 전혀 False Alarm 문제가 발생하지 않는 것을 확인할 수 있었다.

5. 결 론

본 논문에서는 경로 질의를 효율적으로 처리하기 위한 새로운 색인 구조를 제안하고 이에 적합한 질의 처리 기법을 설계하고 구현하였다. 이 기법은 Durable 번호 부여 기법을 이용하여 문서 내의 엘리먼트 간의 조상-후손 관계를 표현하는 D-Ancestor B+ 트리를 구축하고, 접미사 트리 내에서의 조상-후손 관계를 나타내기 위해 S-Ancestor B+ 트리를 구축하였다. 또한 질의의 결과가 어떤 문서에 있는지 알아내기 위해 DocID 트리는 제안하는 색인구조에 적합한 R 트리 사용하였다. 그리고 색인 구조에 알맞게 질의를 상향식으로 처리함으로써 ViST에서 발생하는 많은 중간 검색 과정과 불필요한 노드 접근을 감소시킴으로써 질의 처리 시간을 줄일 수 있었다. 또한 이렇게 상향식 질의를 처리함으로써 분기 질의 시 발생할 수 있는 False Alarm 문제 또한 자연스럽게 해결할 수 있음을 증명하였다.

그리고 성능평가를 통해 제안하는 색인구조가 ViST보다 질의 처리시간이 확연히 감소함을 보임으로써 성능이 우수함을 증명하였다. 향후 연구는 다양한 질의를 더욱 효율적으로 처리할 수 있는 기법들에 대한 연구들을 진행할 예정이다.

참 고 문 헌

[1] Tim Bray, Jean Paoli, C.M. Sperberg-Mc Queen, et. al., "Extensible Markup Lan- guage (XML) 1.0(Third Edition)", W3C Consortium 2004, http://www.w3.org/TR/- REC-xml, 2004.

[2] J.Clark and S. DeRose, "XML Path Language(XPath) 2.0", W3C Working Draft, http://www.w3.org/TR/xpath20, 2005.

[3] D. Chamberlin, J. Robie, and D. Florescu, "Quilt: An XML query language for heterogeneous data sources", In Proc. WebDB, pp. 53-62, 2000.

[4] A. Deutsch, M. Fernandez, D. Florescu, A Levy, and D. Suciu, "A query language for XML", In Proc. the 8th International World Wide Web Conference, pp.77-91, 1999.

[5] S. Boag, D. Chamberlin, Mary F. Fernández, D. Florescu, J. Robie, and J. Simé-on, "Query

1.0: An XML Query Language", W3C Working Draft, http://www.w3.or-g/TR/xquery, 2005.

[6] M. Fernandez and D. Suciu, "Optimizing regular path expressions using graph schemas", In Proc. ICDE, pp.14-23, 1998.

[7] C.W. Chung, J.K. Min, K.S. Shim, "APEX: an adaptive path index for XML data", In Proc. ACM SIGMOD, pp.121-132, 2002.

[8] Quanzhong Li, Bongki Moon, "Indexing and Querying XML Data for Regular Path Expressions", In Proc. the 27th VLDB Conference, pp.361-370, 2001.

[9] S. Al-Khalifa, "Structural Joins: A Primitive for Efficient XML Query Pattern Matching", In Proc. ICDE, pp.141-152, 2002.

[10] Shu-Yao Chien and Zografoula Vagena, "Efficient Structural Joins on Indexed XML Documents", In Proc. the 28th VLDB, pp.263-274, 2002.

[11] Jiang, H., Lu, H., Wang, W., and Ooi, B.C., "XR-Tree : Indexing XML Data for Efficient Structural Joins", In Proc. IEEE International Conference on Data Engineering, pp.253-264, 2003.

[12] H. Wang, S. Park, W. Fan, P. S. Yu, "ViST: A Dynamic Index Method for Querying XML Data by Tree Structures", In Proc. SIGMOD, pp.110-121, 2003.

[13] Haixun Wang, Xiaofeng Meng, "On the Sequencing of Tree Structures for XML Indexing", In Proc. ICDE, pp.372-383, 2005.

[14] Paul F. Dietz., "Maintaining order in a linked list", In Proc. the 14th Annual ACM Symposium on Theory of Computing, pp.122-127, 1982.

[15] Microsoft, "MSXML 4.0 SDK", http://msdn.microsoft.com/library, 2005.

[16] The Niagara Project Group, "The Niagara Project Experimental Data", http://-/www.cs.wisc.edu/niagara/data, 2005.

◐ 저 자 소 개 ◑

서 동 민 (Dong-Min Seo)
2002년 충북대학교 정보통신공학과(공학사)
2004년 충북대학교 정보통신공학과(공학석사)
2004~현재 :충북대학교 정보통신공학과 박사과정
관심분야 : 데이터베이스 시스템, 에이전트 시스템, XML, 이동 객체 데이터베이스, 시공간 색인 구조 등
E-mail : dmseo@chungbuk.ac.kr

김 은 재 (Eun-Jae Kim)
2006년 충북대학교 정보통신공학과(공학석사)
2006~현재 : (주)다음커뮤니케이션 R&D센터 연구원
관심분야 : 데이터베이스 시스템, XML, 멀티미디어 데이터베이스, 객체 기반 파일 시스템, 정보검색 등
E-mail : defeat@daumcorp.com

성 동 욱 (Dong-Ook Seong)
2005년 충북대학교 전기전자컴퓨터공학부(공학사)
2005~현재 : 충북대학교 정보통신공학과 석사과정
관심분야 : 데이터베이스 시스템, 무선 센서 네트워크 등
E-mail : sergei@netdb.cbnu.ac.kr

유 재 수 (Jae-Soo Yoo)
1989년 전북대학교 컴퓨터공학과(공학사)
1991년 한국과학기술원 전산학과(공학석사)
1995년 한국과학기술원 전산학과(공학박사)
1995~1996.8년 목포대학교 전산통계학과 전임강사
1996.8-현재 : 충북대학교 전기전자컴퓨터공학부 및 컴퓨터정보통신연구소 교수
관심분야 : 데이터베이스 시스템, 정보검색, 멀티미디어 데이터베이스, 분산 객체 컴퓨팅 등
E-mail : yjs@chungbuk.ac.kr

조 기 형 (Ki-Hyung Cho)
1866년 인하대학교 전기공학과(공학사)
1984년 청주대학교 산업공학과(공학석사)
1992년 경희대학교 전자공학과(공학박사)
1988-현재 : 충북대학교 전기전자컴퓨터공학부 및 컴퓨터정보통신연구소 교수
관심분야 : 데이터베이스 시스템, 소프트웨어 시스템 설계 및 구현, 컴퓨터 네트워크 설계 등
E-mail : khjoe@chungbuk.ac.kr