XML 질의 처리를 위한 효율적인 시퀀스 매칭 기법

(An Efficient Sequence Matching Method for XML Query Processing)

서 동 민 † 송 석 일 †† 유 재 수 †††
(Dong Min Seo) (Seok Il Song) (Jae Soo Yoo)

요 약 인터넷 상에서 정보 표현 및 교환의 표준으로 XML이 대두되면서 데이타베이스 연구 분야에서는 XML 질의 처리에 대한 중요성이 커지고 있다. 그리고 과거 몇 년간 빠른 XML 질의 처리를 위해 XISS, XR-트리와 같은 구조적 XML 질의 처리 기법이 제안되었다. 하지만 구조적 XML 질의 처리는 가지 경로 질의 처리를 위해 많은 조인 비용이 요구되는 문제를 가지고 있다. 최근에는 구조적 XML 질의 처리 기법의 조인 문제를 해결하기 위해 ViST와 PRIX와 같은 시퀀스 매칭 기반의 XML 질의 처리 기법이 제안되었다. 시퀀스 매칭 기반의 XML 질의 처리 기법은 가지 경로 질의를 다수의 부질의로 분리하지 않고 질의 시퀀스가 문서 내에 포함되는지만 비교하기 때문에 조인 비용이 요구되지 않는 장점을 가지고 있다. 하지만 ViST는 최적화되지 못한 번호 부여 기법을 사용함으로써 질의 처리 시 구조 관계를 정확하게 판단하지 못하고, PRIX는 질의와 문서의 *NPS*와 *LPS*를 비교하는데 많은 비용이 요구된다. 따라서 본 논문에서는 XML 질의 처리 성능 향상을 위해 상향식 질의 처리를 사용하는 효율적인 시퀀스 매칭 기법을 제안한다. 또한 본 논문의 성능 평가에서는 제안하는 기법을 ViST, PRIX와 비교하여 제안하는 기법이 와일드-카드('*'와 '//')를 포함하는 선형 경로 질의뿐만 아니라 가지 경로 질의 처리에 대해 향상된 성능을 나타냄을 보인다.

키워드 : XML, 구조-인코드 시퀀스, 상향식 질의 처리

Abstract As XML is gaining unqualified success in being adopted as a universal data representation and exchange format, particularly in the World Wide Web, the problem of querying XML documents poses interesting challenges to database researcher. Several structural XML query processing methods, including XISS and XR-tree, for past years, have been proposed for fast query processing. However, structural XML query processing has the problem of requiring expensive join cost for twig path query. Recently, sequence matching based XML query processing methods, including ViST and PRIX, have been proposed to solve the problem of structural XML query processing methods. Through sequence matching based XML query processing methods match structured queries against structured data as a whole without breaking down the queries into sub queries of paths or nodes and relying on join operations to combine their results. However, determining the structural relationship of ViST is incorrect because its numbering scheme is not optimized. And PRIX requires many processing time for matching *LPS* and *NPS* about XML data trees and queries.

· 이 논문은 2008년도 정부(과학기술부)의 재원으로 한국학술진흥재단(지방연구중심대학 육성사업 / 충북BIT연구중심대학 육성사업단)과 한국과학재단 특정기초 연구(과제번호R01-2006-000-10809-0)의 지원에 의하여 연구되었음
· 이 논문은 제34회 추계학술대회에서 'XML 질의 처리를 위한 효율적인 시퀀스 매칭 기법'의 제목으로 발표된 논문을 확장한 것임

† 학생회원 : 한국과학기술원 전산학과 연수연구원
dmseo@chungbuk.ac.kr
†† 정 회 원 : 충주대학교 컴퓨터공학과 교수
sisong@cjnu.ac.kr
††† 종신회원 : 충북대학교 전기전자컴퓨터공학부 교수
yjs@chungbuk.ac.kr
(Corresponding author임)

논문접수 : 2007년 12월 7일
심사완료 : 2008년 4월 28일

정보과학회논문지: 데이타베이스 제35권 제4호(2008.8)

Therefore, in this paper, we propose efficient sequence matching method using the bottom-up query processing for efficient XML query processing. Also, to verify the superiority of our index structure, we compare our sequence matching method with ViST and PRIX in terms of query processing with linear path or twig path including wild-card('*' and '//').

Key words : XML, Structure-Encoded Sequence, Bottom-Up Query Processing

1. 서 론

인터넷 상에서 정보 표현 및 교환의 표준으로 XML (eXtensible Markup Language)이 대두되면서 데이타베이스 연구 분야에서는 XML 문서에 대한 저장, 색인 그리고 질의 처리에 대한 중요성이 커지고 있다[1]. XML은 문서 구성 요소들 사이에 계층적인 구조를 가지고 있으나 그 형태가 일정하게 고정된 스키마를 따를 필요가 없는 반구조적인(semi-structured) 특징을 지닌다. 이러한 특성을 지니는 XML 데이타의 처리를 위해 일반적으로 트리 구조의 모델을 사용한다. 그리고 XML 질의 처리를 위해 XPath[2], XQuery[3] 등과 같은 질의 언어들이 연구되었고 이러한 질의 언어들은 선형 경로(linear path) 질의와 가지 경로(twig path) 질의로 표현된다.

XML 문서에 대한 XML 질의를 효율적으로 처리하기 위한 색인 기법에 대한 연구도 많이 수행되었다. 초기에는 주로 경로 색인 기법에 대한 연구가 이루어졌다. 이 기법은 XML 문서 트리에서 발생 가능한 모든 경로에 대한 색인 그래프를 별도로 구축하고 경로 질의 처리를 위해 원본 XML 문서 트리의 탐색이 아닌 경로 색인 그래프를 탐색함으로써 탐색 공간의 크기를 줄여 질의 처리 비용을 줄였다[4,5]. 하지만 와일드-카드('*'와 '//')를 포함하는 질의를 처리하기 위해서는 트리 전체를 순회하기 때문에 많은 탐색 비용이 요구된다.

경로 색인 그래프 기반의 XML 질의 처리 기법의 문제를 해결하기 위해 부모-자식 또는 조상-후손 관계를 효율적으로 처리할 수 있는 구조 색인(structural index)과 번호 부여 기법(numbering scheme)에 대한 연구가 많이 이루어졌다[6-8]. 하지만 구조 조인 기법은 선형 경로 질의를 효율적으로 처리 하지만 가지 경로 질의는 항상 두 개 이상의 부질의(sub-query)로 분해한 뒤, 각각의 부질의에 대한 선형 경로 질의를 통해 처리하고 최종 결과는 많은 비용이 요구되는 부질의 결과들에 대한 조인 연산을 통해 얻는다.

최근에는 구조 색인의 문제를 해결하기 위해 XML 문서와 XML 질의에 대한 시퀀스 매칭을 기반으로 한 색인 구조가 연구되고 있다. *Wang et al.*은 XML 문서와 XML 질의를 구조-인코드 시퀀스(structure-encoded sequence)로 변환하는 ViST를 제안하였다[9]. *Rao et al.*은 XML 문서와 XML 질의를 *LPS*(*Labeled Prufer Sequence*)와 *NPS*(*Numbered Prufer Sequence*)로 변환하는 PRIX를 제안하였다[10]. ViST와 PRIX는 가지 경로 질의를 처리하기 위해 가지 경로 질의를 부질의로 분해하지 않고 시퀀스 매칭만을 통해 질의를 처리함으로써 질의 처리 시 요구되는 조인 비용을 피했다. 하지만 ViST는 최적화되지 못한 번호 부여 기법을 사용함으로써 질의 처리 시 불필요한 디스크 접근이 발생하고 PRIX는 질의와 문서의 *NPS*와 *LPS*를 비교하는데 많은 비용이 요구되어 질의 처리 성능이 감소하는 문제를 가진다.

본 논문에서는 기존 시퀀스 매칭 기반의 색인 구조들에 비해 향상된 XML 질의 처리 성능을 제공하기 위해 상향식 질의 처리를 사용하는 효율적인 시퀀스 매칭 기법을 제안한다. 그리고 다양한 환경에서 성능 평가를 수행하여 제안하는 시퀀스 매칭 기법이 ViST와 PRIX에 비해 우수한 성능을 가짐을 보인다.

본 논문에서 제안하는 시퀀스 매칭 기법의 주요 공헌은 다음과 같다.

- XML 문서와 XML 질의를 구성하는 노드들의 구조 관계를 정확히 표현할 수 있는 구조-인코드 시퀀스를 제안한다. 제안하는 구조-인코드 시퀀스는 질의 처리 시 불필요한 연산에 의해 발생하는 많은 페이지 접근 횟수를 줄이고 *false alarms* 문제를 해결해 질의 결과의 정확성을 높인다.
- 제안하는 XML 색인 구조는 B+-트리를 사용한다. 현재 상용화 되어 사용되는 많은 데이타베이스 관리 시스템에서 B+-트리 색인 구조를 사용한다. 제안하는 색인 구조는 현재 사용되는 데이타베이스 관리 시스템에 쉽게 수용될 수 있다.
- 제안하는 XML 색인 구조는 삽입, 삭제에 융통성 있는 번호 부여 기법을 제안한다. 제안하는 번호 부여 기법은 문서의 삽입, 삭제에 대한 비용을 최소화하여 질의 처리 성능을 향상시킨다.
- 제안하는 XML 색인 구조는 구조-인코드 시퀀스의 접두사(prefix) 정보를 활용한 상향식 질의 처리 기법을 제안한다. 제안하는 상향식 질의 처리 기법은 질의 처리 시 불필요한 노드들의 구조 관계 판단 비용을 현저하게 줄여 질의 처리 성능을 향상시킨다.

본 논문의 구성은 다음과 같다. 2장에서는 XML 질의

처리를 위해 기존에 제안된 색인 기법들에 대해 서술한다. 3장에서는 PRIX에서 기술된 ViST의 *false alarms* 문제 외에 본 논문에서 분석된 ViST의 중요한 문제를 기술하고 4장에서는 본 논문에서 제안하는 색인 구조와 상향식 질의 처리 기법에 대해 기술한다. 5장에서는 다양한 실험을 통해 기존에 제안된 색인 구조와의 성능 평가를 수행하고 마지막 6장에서는 결론 및 향후 연구 방향에 대해 기술한다.

2. 관련 연구

2.1 XML 번호 부여 기법(Numbering Scheme)

XML 문서 트리를 구성하는 두 노드들 사이의 구조 관계를 빠르게 판단하기 위해 XML 문서 트리의 각 노드들은 트리에서의 자기 위치를 기반으로 <*start*, *end*>에 대한 번호가 부여된다. 또한 두 노드 u와 v가 있다고 가정하면, u와 v에 부여된 번호들은 식 (1) 또는 식 (2) 중 하나의 식을 만족한다.

$$u.end < v.start \text{ OR } v.end < u.start \quad (1)$$

$$u.start < v.start \text{ AND } v.end < u.end \quad (2)$$

OR

$$v.start < u.start \text{ AND } u.end < v.end$$

식 (1)은 두 노드가 구조적으로 관계가 없음을 나타내며 식 (2)의 $u.start < v.start$ AND $v.end < u.end$는 u가 v의 조상 노드임을 나타낸다. 그리고 식 (2)의 $v.start < u.start$ AND $u.end < v.end$는 v가 u의 조상 노드임을 나타낸다. 지금까지 번호 부여 기법과 관련된 많은 연구가 진행되었다. *Dietz's* 번호 부여 기법은 트리 순회 값인 <*preorder*, *postorder*>을 통해 각 노드에 번호를 부여한다[11]. *Dietz's* 번호 부여 기법은 $u.preorder < v.preorder$ AND $v.postorder < u.postorder$인 경우 u가 v의 조상 노드가 된다. *Durable* 번호 부여 기법은 <*order*, *size*>을 통해 각 노드에 번호를 부여한다. *order*는 트리의 전위 순회 값을 *size*는 해당 노드가 포함할 수 있는 자식 노드의 수로 문서의 갱신을 고려해서 여유 있는 값이 부여된다[6]. *Durable* 번호 부여 기법은 $u.order < v.order < u.order+u.size$인 경우 u가 v의 조상 노드가 된다.

2.2 ViST

*Wang et al.*은 구조 색인에서 가지 경로 질의 처리 시 요구되는 조인 비용을 줄이기 위해, XML 문서와 XML 질의를 구조-인코드 시퀀스로 변환하고 시퀀스 매칭을 통해 질의를 처리하는 ViST를 제안하였다. 구조-인코드 시퀀스는 각 노드들에 대한 (*symbol*, *prefix*)의 2차원 시퀀스로 구성된다. *symbol*은 각 노드의 이름을 나타내고 *prefix*는 루트 노드로부터 각 노드까지의 경로를 나타낸다. 또한, ViST는 두 노드들의 구조 관계를 빠르게 판단하기 위해 구조-인코드 시퀀스를 구성하는 각 노드에 <n_x, $size_x$>를 부여한다. n_x는 트리의 전위 순회를 통해 부여되며 $size_x$는 해당 노드가 포함하는 후손 노드들의 수로 부여된다.

그림 1은 XML *Doc1*과 *Doc2*에 대한 ViST를 만들기 위해 각 문서를 suffix 트리에 색인 후 트리의 각 노드에 시퀀스와 번호를 부여한 것을 보여준다. 예를 들어, <*3, 4*>가 부여된 시퀀스 노드 (*L, PS*)는 노드의 이름이 *L*이고 루트 노드부터 해당 노드까지의 경로는 *PS*이며 후손 노드 4개를 포함한다는 것을 나타낸다.

ViST는 질의 처리를 위한 시퀀스 매칭 시 질의에 참여하는 시퀀스 노드와 해당 노드들의 번호를 빠르게 찾기 위해 그림 2와 같이 시퀀스 노드 색인을 위한 D-Ancestor B+-트리, 시퀀스 노드들에 대한 번호 색인

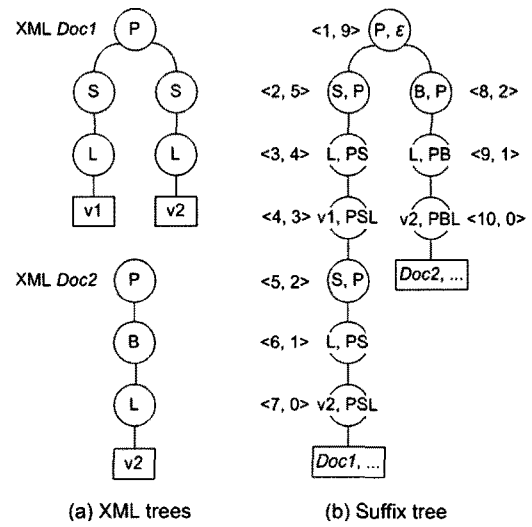


그림 1 XML *Doc1, Doc2*에 대한 ViST의 Suffix 트리

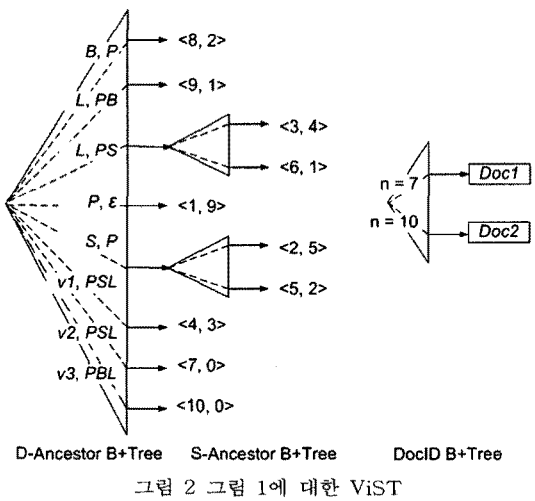


그림 2 그림 1에 대한 ViST

을 위한 S-Ancestor B+-트리 그리고 문서 식별자 색인을 위한 DocID B+-트리를 사용한다.

이와 같이, ViST는 B+-트리를 통한 시퀀스 매칭을 통해 기존 색인 구조에 비해 빠른 질의 처리 성능을 제공한다. 하지만, 문서와 질의의 트리 구조가 틀려도 문서와 질의의 시퀀스가 매칭되면 결과로 만족되는 *false alarms* 문제를 가진다[10].

2.3 PRIX

*Rao et al.*은 ViST의 *false alarms* 문제를 지적하고 XML 문서와 XML 질의를 *LPS*와 *NPS*로 변환하는 PRIX를 제안하였다. PRIX는 문서와 질의를 *LPS*와 *NPS*로 변환하기 위해 그림 3과 같이 문서와 질의 트리에 대해 트리의 후위 순회를 통해 번호를 부여한다. 그리고 가장 낮은 번호를 가지는 노드부터 루트 노드 이전의 번호를 가지는 노드를 순차적으로 삭제하면서, 삭제되는 노드의 부모 노드에 대한 이름을 나열해 *LPS*를 만들고 해당 부모 노드에 부여된 번호를 나열해 *NPS*를 만든다. 예를 들어, 그림 3의 문서 트리 T의 $LPS(T) = A\ C\ B\ C\ C\ B\ A\ C\ A\ E\ E\ E\ D\ A$이고 $NPS(T) = 15\ 3\ 7\ 6\ 6\ 7\ 15\ 9\ 15\ 13\ 13\ 13\ 14\ 15$이다. 그리고 질의 트리 Q의 $LPS(Q) = B\ A\ E\ D\ A$이고 $NPS(Q) = 2\ 6\ 4\ 5\ 6$이다.

PRIX는 문서와 질의를 *LPS*와 *NPS*로 변환 후 질의를 처리하기 위해 $LPS(Q)$가 $LPS(T)$의 서브 시퀀스에 매칭 되는지 검사한다. 그리고 *false alarms* 문제를 해결하기 위해 $NPS(T)$와 $NPS(Q)$에 대한 *gap consistent*와 *frequency consistent*를 검사한다. 마지막으로 문서와 질의의 단말 노드들에 대한 매칭을 통해 최종 결과를 얻는다. 이와 같이, PRIX는 *LPS*와 *NPS*를 통한 시퀀스 매칭을 통해 ViST의 *false alarms* 문제를 해결하고 향상된 질의 처리 성능을 제공한다. 하지만, 질의 처리 시 수행되는 *LPS*와 *NPS*에 대한 많은 매칭 연산 비용은 질의 처리 성능을 저하시킨다.

3. ViST의 질의 성능 저하 문제

본 논문에서는 PRIX에서 지적된 ViST의 *false alarms* 문제 외에도 ViST의 질의 처리 성능을 저하시키는 중요한 문제들을 분석했다. 먼저, 노드들의 구조 관계를 빠르게 결정하기 위해 사용하는 ViST의 번호 부여 기법이 ViST에서 제안하는 색인구조에 적합하지 못하다. 그래서 문서에서는 자식 또는 후손 노드가 아닌 노드들도 질의 처리 시 자식 또는 후손 노드로 보고 접근하는 *Unacceptable Accesses* 문제가 발생한다.

예제 1. 그림 4는 그림 2의 ViST에서 XPath 질의 *P/S/L/v1*를 처리하는 과정을 보여준다. 그림 1의 XML *Doc1*에서 n_x=2를 가지는 (S, P) 노드의 자식 노드는 n_x=3을 가지는 (L, PS) 노드뿐이다. 하지만, n_x=2를 가지는 (S, P) 노드의 자식 노드인 (L, PS)의 $(symbol, prefix)$에 대한 범위 질의 시 $(S, P).n_x < (L, PS).n_x \le (S, P).n_x + (S, P).size_x$에 위해 n_x=6을 가지는 (L, PS) 노드도 자식 노드로 보고 질의 처리 시 접근한다.

다음으로 ViST의 시퀀스를 구성하는 접두사는 루트 노드로부터 각 노드까지의 경로 정보를 나타낸다. 하지만, ViST는 접두사의 정보를 활용하지 않기 때문에 질의 처리 시 불필요하게 단말 노드의 부모 노드와 조상 노드들에 대한 범위 질의를 수행하는 *Unnecessary Accesses* 문제가 발생한다.

예제 2. 그림 4에서 문서 시퀀스 노드 $(v1, PSL)$의 접두사인 PSL은 $v1$ 노드가 조상 노드로 P와 S 노드를 가지고 부모 노드로 L 노드를 가진다는 것을 알 수 있다. 그러므로 그림 4에서 질의 처리 시, $(v1, PSL)$를 포함하는 조상 노드와 부모 노드를 찾는 (P, ε), (S, P) 그리고 (L, PS)에 대한 범위 질의는 불필요한 연산이다.

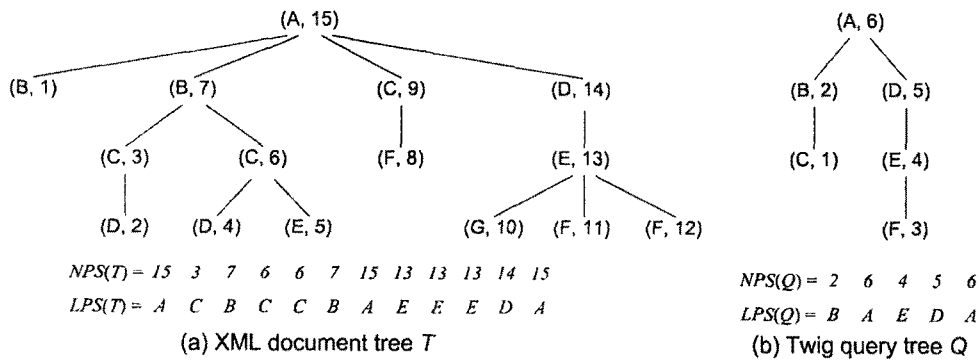


그림 3 XML 문서와 질의 트리에 대한 PRIX의 *LPS*와 *NPS*

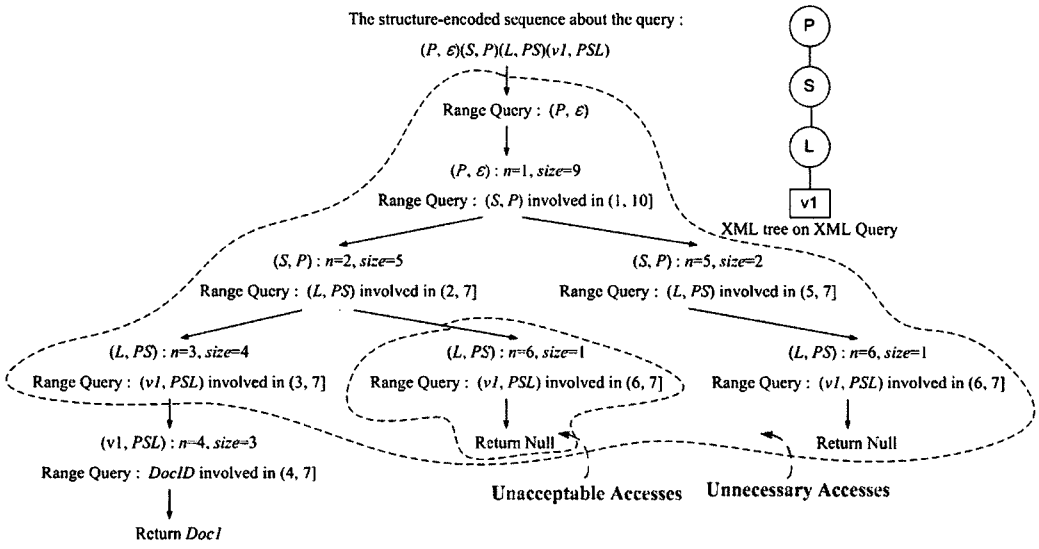


그림 4 ViST에서의 질의 처리

4. 제안하는 XML 시퀀스 매칭 기법

4.1 제안하는 색인 구조

본 논문에서 제안하는 색인 구조는 ViST의 번호 부여 기법 문제를 해결하고 향후 문서 삽입과 삭제에 대해 융통성 있는 번호 부여를 위해 통계적(statistical) *Durable* 번호 부여 기법과 동적(dynamic) *Durable* 번호 부여 기법을 제안한다. *Durable* 번호 부여 기법에서 루트 노드는 모든 자식 노드와 후손 노드에 부여된 번호를 포함해야 하기 때문에 [1, *MAX*)의 범위를 가진다. 여기서, *MAX*는 루트 노드가 모든 자식 노드와 후손 노드를 포함할 수 있는 최대 값을 의미한다. 통계적 *Durable* 번호 부여 기법은 부모 노드에 대해 자식 노드가 발생할 수 있는 확률 정보를 이용해 번호를 부여하는 방법으로, 노드 x에 대해 노드 u가 자식 노드로 발생할 수 있는 확률 값을 $p(u|x)$로 정의하면 부모 노드 x에 대해 각 자식 노드들이 가지는 노드 발생 확률 값은 식 (3)으로 나타낼 수 있다.

$$P_x(y_i) = p(y_i|x)\prod_{k=1}^{i-1}(1-p(y_k|x)) \quad (3)$$

그러므로 [l, r)가 부여된 부모 노드 x에 대해 자식 노드 y_i에 부여되는 번호는 식 4로 나타낼 수 있다.

$$l_i = l+1+(r-l-1)\sum_{j=1}^{i}P_x(y_j) \quad (4)$$
$$r_i = l_i + s_i$$

반면에, 동적 *Durable* 번호 부여 기법은 노드들의 발생 확률 값을 사용하지 않고 번호를 부여하는 방법으로, 부모 노드에 부여된 번호의 범위 값에 대해 자식 노드에 부여될 수 있는 번호 범위 값의 비율을 λ라고 한다면 [l, r)가 부여된 부모 노드 x에 대해 자식 노드 y_k에 부여될 수 있는 번호의 범위 값은 식 (5)로 나타낼 수 있다.

$$s_k = (r-l-1)(\lambda-1)^{k-1}/\lambda^k \quad (5)$$

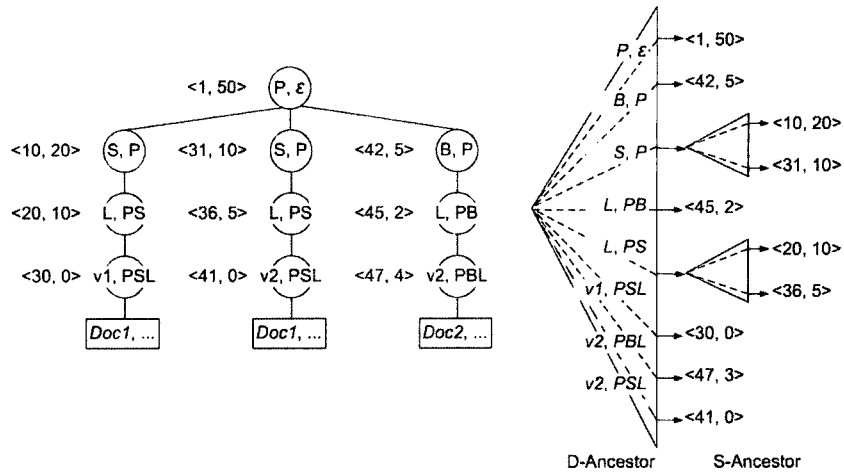
그러므로 [l, r)가 부여된 부모 노드 x에 대해 자식 노드 y_k에 부여되는 번호는 식 (6)로 나타낼 수 있다.

$$l_k = l+1+(r-l-1)(1-(\lambda-1)^{k-1}/\lambda^{k-1}) \quad (6)$$
$$r_k = l_k + s_k$$

본 논문에서는 문서 내 노드들에 대한 발생 확률 값을 얻을 수 있다면 통계적 *Durable* 번호 부여 기법을 통해 번호를 부여하고 아니면 동적 *Durable* 번호 부여 기법을 통해 노드들에 대한 번호를 부여한다. 그림 5는 본 논문에서 제안하는 색인 구조를 보여준다. 그림 1의 XML *Doc1*와 *Doc2*의 문서들에 대해 임의의 발생 확률 값에 대한 통계적 *Durable* 번호인 <*order*, *size*>을 부여하고, ViST와 유사하게 시퀀스 색인을 위한 D-Ancestor B+-트리와 시퀀스 노드들에 부여된 번호 색인을 위한 S-Ancestor B+-트리가 구축된 것을 보여준다.

4.2 제안하는 상향식 질의 처리 기법

시퀀스 노드들의 접두사 정보를 활용하면 질의 처리 성능을 크게 향상시킬 수 있다. 그래서 본 논문에서는 질의 처리 시, 시퀀스 노드들의 접두사 정보를 활용하는 상향식 질의 처리 기법을 제안한다. 그림 6과 같이, 선형 경로 질의의 경우 제안하는 상향식 질의 처리 기법은 질의를 구성하는 마지막 시퀀스 노드에 대해서만 제안하는 색인 구조를 통해 시퀀스 매칭을 수행한다. 또



(a) *Durable* 번호가 부여된 suffix tree (b) (a)에 대한 색인 구조

그림 5 제안하는 색인 구조

```
Input : Q : q1, ... , qk, a query sequence
        D-Ancestor B+-tree, index of (symbol, prefix) pairs
        S-Ancestor B+-tree, index of <order, size> labels
        DocID R-tree, mapping between the order values in node labels
        and document IDs
Output : all occurrences of Q in the XML data

LinearQuery(Q, k) /* k is the number of a query sequence's nodes */

Function LinearQuery(Q, k)
1 : while not found T do
2 :     T ← retrieve the S-Ancestor B+-tree that represents qk
3 :         from the D-Ancestor B+-tree;
4 :     N ← retrieve all nodes with <order, order+size> from T;
5 :     for each node c ∈ N do
6 :         Perform a query with key(order) on the DocID R-tree
7 :         to output all document IDs in that key;
8 :     end
9 : end
```

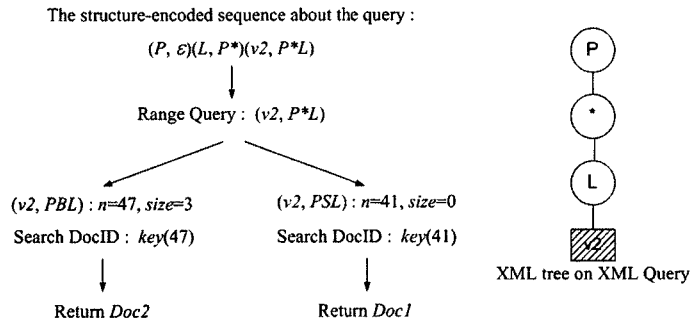
그림 6 선형 경로 질의에 대해 제안하는 상향식 질의 처리 알고리즘

한, 예제 3에서 볼 수 있듯이 와일드-카드가 포함된 질의의 경우도 ViST에 비해 제안하는 색인 구조가 질의 처리 시 적은 노드를 접근하기 때문에 질의 처리 성능이 향상됨을 볼 수 있다.

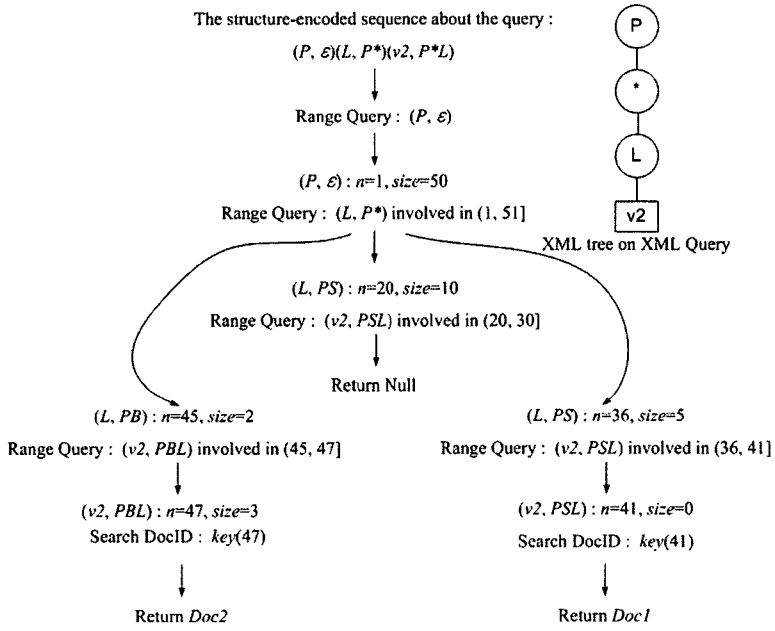
예제 3. 그림 7은 ViST와 제안하는 시퀀스 매칭 기법을 통해 와일드-카드가 포함된 선형 경로 질의 */P/*/L/v2*을 처리하는 과정을 보여준다. 질의의 시퀀스는 (*P*, *ε*)(*L*, *P**)(*v2*, *P*L*)이고 질의의 마지막 시퀀스 노드는 (*v2*, *P*L*)이다. 그러므로 그림 7의 (a)와 같이 제안하는 상향식 질의 처리 기법은 그림 5의 D-Ancestor B+-트리에 대해 (*v2*, *P*L*)에 대한 범위 질의를 수행한다. (*v2*, *P*L*)의 범의 질의 결과로 (*v2*, *PBL*)과 (*v2*, *PSL*)를 얻는다. 그리고 (*v2*, *PBL*)과 (*v2*, *PSL*)를 포함하는 문서들을 찾기 위해 그림 5의 S-Ancestor B+ 트리에 대해 (*v2*, *PBL*)와 (*v2*, *PSL*)에 대한 범위 질의만을 수행해서 결과를 얻는다. 하지만, ViST는 그림 7의 (b)와 같이 하향식 질의 처리 과정을 통해 질의를 처리하기 때문에 제안하는 상향식 질의 처리 기법에 비해 많은 범위 질의 연산이 요구된다.

본 논문에서 제안하는 선형 경로 질의 처리를 위한 상향식 질의 처리 기법을 통해 가지 경로 질의를 처리할 경우, 가지 경로를 구성하는 선형 경로들에 대한 구조 관계를 판단하지 않기 때문에 *false alarms* 문제가 발생한다. 그래서 가지 경로 질의 처리에 있어서 제안하는 상향식 질의 처리 기법은 가지 경로를 구성하는 선형 경로들에 대한 효율적인 구조 관계 판단 기법을 제공한다. 가지 경로 질의 처리를 위해 제안하는 상향식 질의 처리 기법은 ViST나 PRIX와 같이 질의를 구성하는 모든 시퀀스 노드들에 대한 구조 관계를 판단하지 않고, 가지 경로 질의를 구성하는 단말 노드들과 정점(vertex) 노드들에 대한 구조 관계 판단만으로도 가지 경로 질의를 구성하는 선형 경로 질의들의 구조 관계를 정확하게 판단할 수 있다.

예제 4. 그림 8은 제안하는 상향식 질의 처리 기법을 통해 가지 경로 질의 */P/S[L/v1]/L/v2*를 처리하는 과정을 보여준다. 질의의 시퀀스는 (*P*, *ε*)(*S*, *P*)(*L*, *PS*)(*v1*, *PSL*)(*L*, *PS*)(*v2*, *PSL*)이다. 질의를 구성하는 각 선형 경로의 마지막 노드는 (*v1*, *PSL*)과 (*v2*, *PSL*)이다. 그리고 가지 경로 질의의 정점 노드는 (*S*, *P*)이다. 제안하는 상향식 질의 처리 기법은 단말 노드들에 대해 부여된 번호를 D-Ancestor B+-트리와 S-Ancestor



(a) 제안하는 상향식 질의 처리를 통한 과정



(b) ViST의 하향식 질의 처리를 통한 과정

그림 7 ViST와 제안하는 상향식 질의 처리 기법을 통한 선형 경로 질의 처리

B+-트리를 통해 획득한다. 그림 5에서는 (*v2, PSL*)에 부여된 번호 <*41, 0*>과 (*v1, PSL*)에 부여된 번호 <*30, 0*>을 얻는다. 이후 단말 노드들이 질의와 같은 구조 관계를 갖는지 판단하기 위해서 단말 노드들에 부여된 번호와 정점 노드만을 가지고 구조 관계를 판단한다. 즉, 단말 노드 (*v1, PSL*)와 (*v2, PSL*)는 정점 노드 (*S, P*)를 중심으로 분기된 노드들이기 때문에 식 7을 만족하는 정점 노드 (*S, P*)를 찾고 이 노드에 부여된 번호를 가지고 질의를 만족하는 문서를 찾을 수 있다. 그림 8에서는 (*v2, PSL*)에 대한 (41, 41]$\in order(S, P)$와 (*v1, PSL*)에 대한 (30, 30]$\in order(S, P)$를 만족하는 (*S, P*)가 존재하지 않기 때문에 질의를 만족하는 XML 문서가 없음을 알 수 있다.

$$(order_{(v1,\ PSL)},\ order_{(v1,\ PSL)}+size_{(v1,\ PSL)}]\in order_{(S,\ P)} \quad (7)$$

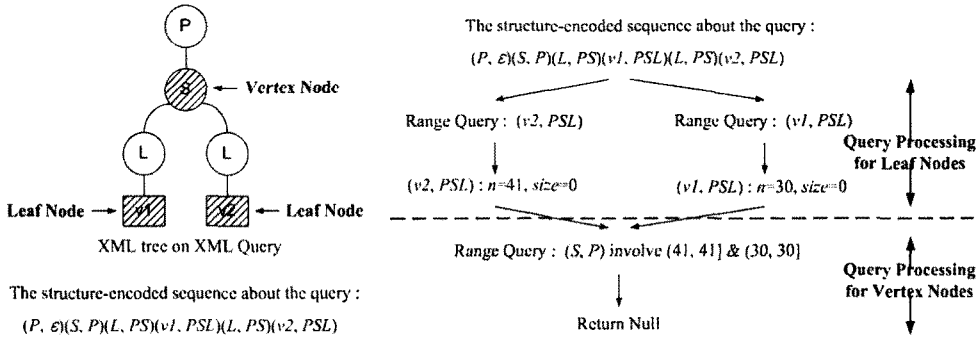
&

$$(order_{(v2,\ PSL)},\ order_{(v2,\ PSL)}+size_{(v2,\ PSL)}]\in order_{(S,\ P)}$$

5. 실험 및 성능 평가

5.1 실험 환경

제안하는 색인 구조, ViST 그리고 PRIX는 C 언어로 구현되었고 각 색인 구조에서 사용된 B+-트리는 GiST [12]를 사용하였다. ViST 알고리즘은 *Wang et al.*로부터 제공받았고 PRIX 알고리즘은 [13]의 웹사이트에서 취득했다. 성능 평가에 사용된 시스템은 펜티엄Ⅳ 2GHz 프로세서에 1GBytes의 메모리를 가지며, 운영체제는



(a) 정점노드와 단말노드 획득 (b) 가지 경로 질의 처리 과정

그림 8 제안하는 상향식 질의 처리 기법을 통한 가지 경로 질의 처리

Red Hat 리눅스 2.6.11을 사용하였다. 코드 컴파일은 GNU gcc 컴파일러로 ver.3.4.4를 사용하였다. 그리고 버퍼의 크기는 8KByte를 사용하였고 각 색인 구조의 노드 레이블(label)을 위해서 8Byte의 숫자 범위를 사용하였다.

표 1은 성능 평가에 사용된 데이타 셋의 특성을 보여준다. 사용된 데이타 셋은 워싱턴대학교의 XML 데이타베이스[14]로부터 얻었다. 성능 평가에는 각각 다른 특성을 가지는 세 개의 데이타 셋을 사용하였다. SWISS-PROT 데이타 셋은 적은 요소와 적은 깊이의 문서 높이를 가지지만 많은 속성을 가진다. TREEBANK 데이타 셋은 적은 요소와 하나의 속성을 가지지만 깊은 문서 높이와 조상 노드가 후손 노드로 다시 사용된다. DBLP 데이타 셋은 많은 요소를 가지지만 문서의 높이가 깊지 않고 문서의 전체적인 구조가 유사하다.

표 1 성능평가에 사용된 데이타 셋

Dataset Name	SWISSPROT	TREEBANK	DBLP
Size in MB	115	86	134
# of Elements	2977031	2437666	3332130
# of Attributes	2189859	1	404276
Max-depth	5	36	6
# of Sequences	50000	56385	328858

표 2는 성능 평가에 사용된 XPath 질의 셋을 보여준다. 사용된 질의는 PRIX에서 성능 평가를 위해 사용된 동일한 질의 셋으로 와일드-카드 '*' 및 '//'을 포함하는 다양한 구조의 선형 경로 질의와 가지 경로 질의를 사용하였다.

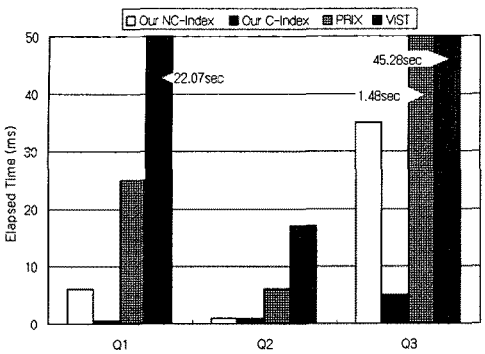
5.2 성능 평가

제안하는 색인 구조는 *NC*-색인과 *C*-색인으로 분류하여 성능을 비교하였다. *C*-색인은 ViST와 같은 구조로 XML 문서 안의 텍스트 노드를 D-Ancestor B+-트리와 S-Ancestor B+-트리에 색인한 구조이다. 반면에, *NC*-색인은 PRIX와 같은 구조로 XML 문서 안의 텍스트 노드를 D-Ancestor B+-트리와 S-Ancestor B+-트리에 색인하지 않고 데이타베이스에 저장한 구조이다. *NC*-색인은 데이타베이스에 저장된 텍스트 노드와 텍스트 노드에 대한 부모 요소 노드의 포인트 정보를 부모 요소 노드의 S-Ancestor B+-트리 저장한다. 그리고 *NC*-색인은 텍스트 노드에 대한 질의 요청 시, 텍스트 노드의 부모 요소 노드를 단말 노드로 보고 제안하는 상향식 질의 처리 기법을 통해 질의를 처리한 뒤 결과로 획득된 요소와 데이타베이스에 저장된 텍스트 노드의 구조 관계를 판단해 최종 결과를 산출한다.

그림 9는 DBLP 데이타 셋에서의 성능 평가를 보여

표 2 성능평가에 사용된 질의 셋

No.	Query	Dataset
Q_1	//article/author="E. F. Codd"	DBLP
Q_2	//phdthesis[/year][/number]	DBLP
Q_3	//inproceedings[/author="Jim Gray"][/year="1990"]	DBLP
Q_4	//Ref/Author="Moss J"	SWISSPROT
Q_5	//Entry[/Org="Piroplasmida"][/Ref/Author="Kemp D.J"]	SWISSPROT
Q_6	//Entry[/PFAM[@prim_id=PF00304][/SIGNAL/Descr]	SWISSPROT
Q_7	//S//NP/SYM	TREEBANK
Q_8	//NP[/RBR_OR_JJR]/PP	TREEBANK
Q_9	//NP/PP/NP[/NNS_OR_NN][/NN]	TREEBANK



(a) 질의 처리 시간

	NC-Index	*C*-Index	PRIX	ViST
Q_1	8 pages	1 pages	23 pages	2,280 pages
Q_2	2 pages	2 pages	8 pages	17 pages
Q_3	39 pages	7 pages	185 pages	3,543 pages

(b) 페이지 접근 횟수

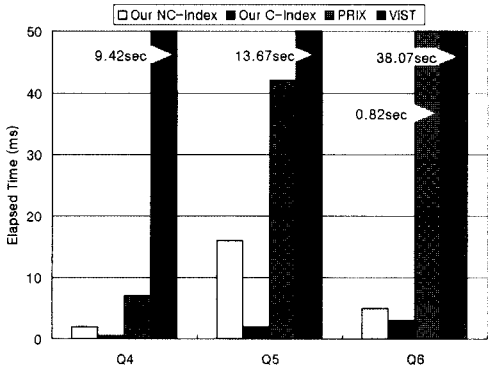
그림 9 Q_1, Q_2 그리고 Q_3에 대한 성능 평가

준다. Q_1은 요소 노드와 텍스트 노드로 구성된 선형 경로 질의이다. ViST와 PRIX는 하향식으로 질의를 처리하기 때문에 질의를 구성하는 모든 요소 노드들에 대한 범위 질의를 수행한다. 하지만, 제안하는 색인 구조는 상향식으로 질의를 처리하기 때문에 질의를 구성하는 단말 노드에 대한 범위 질의만을 수행한다. 그래서 ViST와 PRIX가 제안하는 색인 구조에 비해 많은 CPU 비용 및 페이지 접근 비용이 요구된다. 그리고 PRIX와 *NC*-색인은 텍스트 노드를 색인 구조에 색인하지 않고 데이타베이스에 저장하기 때문에 요소들에 대한 질의 처리 후 데이타베이스 내 텍스트 노드와 구조 관계를 판단하기 위한 비용이 추가적으로 요구된다. 반면에, *C*-색인은 D-Ancestor B+-트리와 S-Ancestror B+-트리에 텍스트 노드를 색인하고 상향식 질의 처리를 통해 텍스트 노드의 범위 질의만을 통해 질의를 처리할 수 있다. 그래서 *NC*-색인이 *C*-색인에 비해 많은 CPU 비용 및 페이지 접근 비용이 요구된다. Q_2는 요소 노드만으로 구성된 가지 경로 질의이고 Q_3은 요소 노드와 텍스트 노드로 구성된 가지 경로 질의이다. 가지 경로 질의는 가지 경로를 구성하는 선형 경로 질의들에 대한 구조 관계를 판단해야 하기 때문에 다른 질의들에 비해 많은 CPU 비용 및 페이지 접근 비용이 요구된다. ViST와 PRIX는 질의를 구성하는 모든 요소 노드들에 대한 조인 연산을 수행한다. 하지만, 제안하는 색인 구조는 단말 노드들과 정점 노드들에 대한 조인 연산만을 수행한다. 그래서 제안하는 색인 구조가 ViST와 PRIX에 비해 우수한 성능을 가진다. 그리고 텍스트 노드를 가지지 않는 질의에 대해 *NC*-색인과 *C*-색인은 같은 방법으로 질의를 처리하기 때문에 Q_2에 대한 *NC*-색인과 *C*-색인의 질의 처리 성능은 같다. 또한, 문서 내에 'phdthesis' 요소를 부모로 가지는 'year'와 'number' 요소가 'article' 요소를 부모로 가지는 'author' 요소보다 매우 적은 빈도수를 가지기 때문에 가지 경로 질의인 Q_2가 선형 경로 질의인 Q_1에 비해 적은 질의 처리 비용을 가진다. Q_3에 대한 *NC*-색인의 질의 처리 비용이 다른 질의들에 비해 많은 이유도 'inproceedings' 요소를 부모로 가지는 'author'와 'year' 요소가 문서 내에 많이 등장하며 'Jim Gray'와 '1990'에 대한 텍스트 노드들 별도로 읽어 많은 조인 연산을 수행하기 때문이다.

DBLP 데이타 셋에 대한 Q_1, Q_2 그리고 Q_3에 대한 성능 비교를 통해 텍스트 노드를 포함하는 가지 경로 질의 Q_3은 가지 경로 질의를 구성하고 있는 선형 경로 질의들에 대한 구조 관계 및 텍스트 값을 비교하기 때문에 텍스트 값을 포함하지 않는 가지 경로 질의 Q_2와 텍스트를 포함하는 선형 경로 질의 Q_1에 비해 많은 CPU 비용 및 페이지 접근 비용이 요구되는 것을 확인할 수 있다. 그리고 빈도가 높은 요소에 대한 Q_1과 Q_3은 질의 처리 시 많은 요소들을 대상으로 질의를 처리하기 때문에 빈도수가 적은 요소에 대한 Q_2에 비해 많은 CPU 비용 및 페이지 접근 비용이 요구되는 것을 확인할 수 있다.

그림 10은 SWISSPROT 데이타 셋에서의 성능 평가를 보여준다. Q_4는 Q_1과 유사한 질의 구조로 요소 노드와 텍스트 노드로 구성된 선형 경로 질의이다. Q_1은 요소 노드의 수가 많은 문서에서 수행되었고 Q_4는 요소 노드의 수가 적은 문서에서 수행되었기 때문에 Q_1이 Q_4에 비해 많은 CPU 비용 및 페이지 접근 비용이 요구된다. 그리고 Q_1과 Q_4의 텍스트 노드 수가 비슷하기 때문에 *C*-색인은 두 질의에 대해서 비슷한 성능 평가 결과를 가진다.

Q_5는 Q_3과 유사한 질의 구조로 요소 노드와 텍스트 노드로 구성된 가지 경로 질의이다. Q_1과 Q_4의 성능 비교에서 볼 수 있듯, 요소가 많은 문서에서 수행된 Q_3이 요소가 적은 문서에서 수행된 Q_5에 비해 많은 CPU 비용 및 페이지 접근 비용이 요구된다. 그리고 Q_5는 두 개의 선형 경로로 구성된 가지 경로 질의이기 때문에 선형 경로들에 대한 구조 관계를 판단해야하기 때문에 선형 경로 질의 Q_4에 비해 많은 CPU 비용 및 페이지 접근 비용이 요구된다. Q_6은 요소 노드와 속성 노드 그리고 와일드-카드 '//'로 구성된 가지 경로 질의이다. Q_6은 속성을 많이 포함하는 XML 문서에 대해 속성을 찾는 질의이고 와일드-카드 '//'에 대한 많은 범위 질의와 가지 경로를 구성하는 선형 경로들에 대한 구조 관계

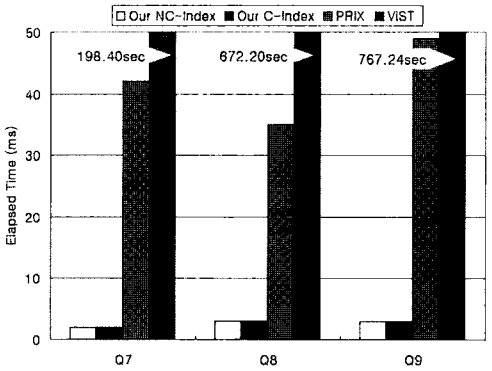


(a) 질의 처리 시간

	NC-Index	*C*-Index	PRIX	ViST
Q_4	2 pages	1 pages	9 pages	1,657 pages
Q_5	18 pages	2 pages	49 pages	1,885 pages
Q_6	7 pages	4 pages	83 pages	4,367 pages

(b) 페이지 접근 횟수

그림 10 Q_4, Q_5 그리고 Q_6에 대한 성능 평가



(a) 질의 처리 시간

	NC-Index	*C*-Index	PRIX	ViST
Q_7	3 pages	3 pages	46 pages	40,827 pages
Q_8	4 pages	4 pages	35 pages	94,505 pages
Q_9	4 pages	4 pages	55 pages	121,928 pages

(b) 페이지 접근 횟수

그림 11 Q_7, Q_8 그리고 Q_9에 대한 성능 평가

판단을 위한 많은 질의 처리 비용이 요구된다. 그래서 Q_6가 Q_4와 Q_5에 비해 많은 CPU 비용 및 페이지 접근 비용이 요구된다. 또한, *NC*-색인의 경우 Q_6에 비해 Q_5를 수행하는데 더 많은 CPU 비용 및 페이지 접근 비용이 요구된다. 이는 상향식 질의 처리 시 요구되는 질의 Q_6의 단말 노드 '@prim_id'와 'Descr' 수보다 Q_5의 단말 노드 'Org'와 'Author' 수가 많기 때문이다.

SWISSPROT 데이타 집합에 대한 Q_4, Q_5 그리고 Q_6에 대한 성능 비교를 통해 속성을 많이 포함한 문서의 경우 요소에 대한 질의 보다는 속성에 대한 질의 처리 비용이 더 많이 요구되고 와일드-카드가 포함되지 않은 질의 처리보다는 와일드-카드가 포함된 질의 처리 비용이 더 많이 요구되는 것을 확인할 수 있다. 또한, 와일드-카드가 포함된 가지 경로 질의 처리의 경우 다른 질의들에 비해 매우 많은 질의 처리 비용이 요구되는 것을 확인할 수 있다.

그림 11은 TREEBANK 데이타 셋에서의 성능 평가를 보여준다. Q_7은 두 개의 와일드-카드 '//'로 구성된 선형 경로 질의이다. ViST는 Q_7을 수행하기 위해서 D-Ancestror B+-트리에 대해 (*S*, *//*)와 (*NP*, *//S//*)에 대한 범위 질의를 수행한다. 이와 같은 범위 질의는 무수히 많은 검색 결과를 산출하고 최종 결과를 얻기 위해서 이들에 대한 구조 관계를 판단한다. 이는 매우 많은 CPU 비용 및 페이지 접근 비용이 요구된다. PRIX는 ViST와 같이 (*S*, *//*)와 (*NP*, *//S//*)에 대한 범위 질의를 수행하는 것이 아니라, "S"와 "NP"에 부여된 *LPS*와 *NPS*에 대한 범위 질의와 이들에 대한 구조 관계 판단을 통해 질의를 처리하기 때문에 ViST에 비해 좋은 성능을 가진다. 그리고 제안하는 색인 구조는 상향식 질의 처리 기법을 사용해 (*SYM*, *//S//NP*)에 대한 범위 질의만으로 질의를 처리하기 때문에 ViST와 PRIX에 비해 가장 좋은 성능을 가진다. 또한, TREEBANK 데이타 집합은 텍스트 노드를 가지고 있지 않다. 그래서 TREEBANK 데이타 집합에 대한 *CN*-색인과 *N*-색인은 같은 구조와 같은 성능 평가 결과를 가진다. Q_8은 한 개의 와일드-카드 '//'로 구성된 가지 경로 질의이다. ViST는 (*NP*, *//*)에 대한 범위 질의 수행 후 얻은 많은 결과를 가지고 가지 경로를 구성하는 두 개의 선형 경로들에 대한 구조 관계를 판단하기 때문에 Q_7에 비해 많은 CPU 비용 및 페이지 접근 비용이 요구된다. PRIX는 'NP'에 부여된 *LPS*와 *NPS*에 대한 범위 질의와 가지 경로를 구성하는 두 개의 간단한 선형 경로들에 대한 구조 관계를 판단하기 때문에 Q_7에 비해 좋은 성능을 가진다. 이를 통해, PRIX의 질의 처리 성능은 질의에 포함된 와일드-카드의 수가 아니라 구조 관계를 판단하는데 참여하는 노드의 수임을 확인할 수 있다. 또한, 제안하는 색인 구조는 상향식 질의 처리 기법을 사용하기 때문에 Q_8에 대해 ViST 및 PRIX에 비해 좋은 성능을 가진다. 하지만, 제안하는 색인 구조는 가지 경로를 구성하는 선형 경로들의 구조 관계를 판단하기 위해 단말 노드와 정점 노드에 대한 구조 관계를 판단하므로 Q_7에 비해 많은 CPU 비용 및 페이지 접근 비용이 요구된다. Q_9는 Q_8과 유사한 질의로 한 개의 와일드-카드 '//'로 구성된 가지 경로 질의이다. 하지만, Q_9는

Q_8에 비해 더 많은 요소들을 포함하고 와일드-카드를 부모로 가졌던 'NP'가 자식으로 반복해서 나타나는 구조를 가졌다. ViST와 PRIX 모두 Q_9가 Q_8에 비해 구조 관계를 판단하기 위한 더 많은 요소를 포함하고 있기 때문에 Q_9가 Q_8에 비해 많은 CPU 비용 및 페이지 접근 비용이 요구된다. 반면에, 제안하는 색인 구조는 상향식 질의 처리를 통해 같은 비슷한 수의 단말 노드와 정점 노드에 대한 구조 관계를 판단하므로 Q_9와 Q_8에 대해 비슷한 성능 평가 결과를 가진다.

TREEBANK 데이타 집합에 대한 Q_7, Q_8 그리고 Q_9에 대한 성능 비교를 통해 문서 높이가 깊고 한 요소의 자식 혹은 후손으로 동일 요소가 반복해서 나타나는 문서의 경우, ViST는 질의에 포함된 와일드-카드 수와 구조 관계 판단을 위한 요소들의 수가 많을수록 질의 처리 비용이 더 많이 요구되는 것을 확인할 수 있다. 그리고 PRIX는 질의에 포함된 와일드-카드의 수에는 크게 영향을 받지 않지만 구조 관계 판단을 위한 요소들의 수가 많을수록 질의 처리 비용이 더 많이 요구되는 것을 확인할 수 있다. 또한, 제안하는 색인 구조도 질의에 포함된 와일드-카드의 수에는 크게 영향을 받지 않지만 상향식 질의 처리를 위해 구조 관계를 판단하기 위한 단말 노드와 정점 노드의 수가 많을수록 질의 처리 비용이 더 많이 요구되는 것을 확인할 수 있다.

그림 12는 ViST, PRIX 그리고 제안하는 색인 구조의 *false alarms* 문제에 대한 성능 평가를 보여준다. PRIX는 ViST의 *false alarms* 문제를 해결하기 위해 *LPS*와 *NPS*를 통한 시퀀스 매칭 시 *gap consistent*와 *frequency consistent*를 검사를 수행한다. 그리고 제안하는 색인 구조는 *Durable* 번호 부여 기법을 통한 시퀀스 매칭을 수행해 ViST의 *false alarms* 문제를 해결한다. 그림 12에서 볼 수 있듯, 모든 질의들에 대해 PRIX와 제안하는 색인 구조는 문서 내에 포함된 정확한 결과를 획득한다. 하지만, ViST는 문서 내에서 조상-후손 또는 부모-자식 관계가 아닌 노드들이 ViST의 suffix 트리 내에서는 조상-후손 또는 부모-자식 관계로 표현될 수 있기 때문에 실제 결과보다 많은 결과가 획득된다. 예를 들어, Q_2에 대해 DBLP 데이타 셋에서 'phdthesis' 요소의 수는 72개이고 'phdthesis' 요소를 부모로 가지는 'year'과 'number' 요소의 수는 각각 72개와 3개이다. PRIX와 제안하는 색인 구조는 'year'과 'number' 요소를 자식으로 가지는 'phdthesis' 요소에 대한 결과 3개를 정확히 획득한다. 하지만, ViST는 문서 내에서 'year'과 'number' 요소가 같은 부모 'phdthesis' 요소에 없어도 ViST의 suffix 트리에서 'phdthesis' 요소의 자식 또는 후손 노드로만 포함되면 결과로 보기 때문에 4602개의 결과를 획득한다. 즉, 질의를 구성하는 노드의 수가 많고 문서 내에서 해당 노드들의 빈도수가 클수록 ViST의 *false alarms* 문제는 증가한다.

6. 결론 및 향후 연구

본 논문에서는 ViST에서 사용하는 번호 부여 기법과 시퀀스 노드의 접두사 정보를 활용하지 않는 질의 처리로 인해 질의 처리 성능이 저하되는 문제를 분석했다. 그리고 ViST의 분석된 문제와 *false alarms* 문제를 해결하고 PRIX보다 질의 처리 성능이 향상된 새로운 시퀀스 매칭 기법을 제안했다. 제안하는 시퀀스 매칭 기법은 ViST을 기반으로 하며, *Durable* 번호 부여 기법을 사용해 *false alarms* 문제를 해결했다. 또한, 시퀀스 노드의 접두사 정보를 활용한 상향식 질의 처리 기법을 제안함으로써 질의 처리 시 요구되는 중간 노드 접근 수를 현저하게 줄여 질의 처리 성능을 향상시켰다. 그리

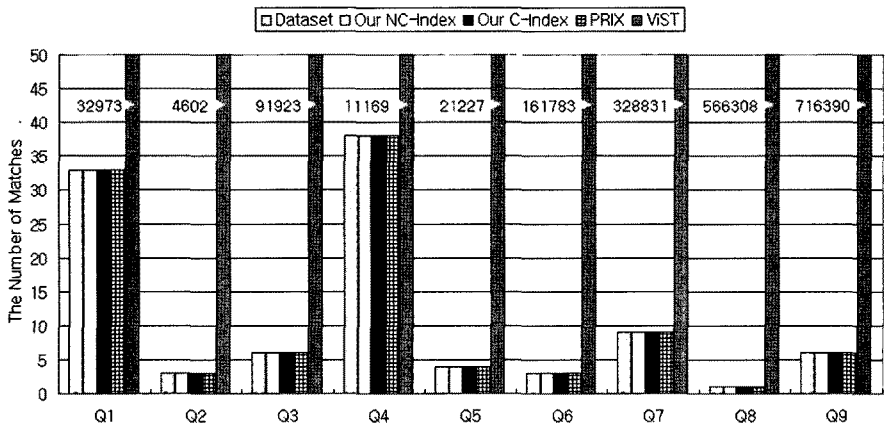


그림 12 *false alarms*에 대한 성능 평가

고 성능 평가를 통해 제안하는 시퀀스 매칭 기법의 질의 처리 시간이 ViST, PRIX보다 확연히 감소함을 보임으로써 질의 처리 성능이 우수함을 증명하였다.

본 논문에서 제안한 구조-인코드 시퀀스는 요소, 속성 그리고 텍스트 노드의 수가 증가할수록 그 크기가 증가한다. 구조-인코드 시퀀스의 크기가 증가하는 것은 색인 구조의 크기가 증가하는 문제를 가져온다. 그래서 향후 연구에서는 구조-인코드 시퀀스를 압축하는 방법과 압축된 구조-인코드 시퀀스를 통해 질의 처리하는 기법에 대한 연구를 수행해 제안된 색인 구조에 비해 향상된 질의 처리 성능을 제공할 것이다.

참 고 문 헌

[1] World Wide Web Consortium, "Extensible Markup Language (XML) 1.0," http://www.w3.org/TR/REC-xml, 2006.

[2] World Wide Web Consortium, "XML Path Language (XPath) Version 2.0," http://www.w3.org/TR/xpath20, 2007.

[3] World Wide Web Consortium, "XQuery 1.0: An XML Query Language," http://www.w3.org/TR/xquery, 2007.

[4] M. Fernandez and D. Sucju, "Optimizing Regular Path Expressions using Graph Schema," *In 1998 ICDE*, pp.14-23, 1998.

[5] C. W. Chung, J. K. Min, K. S. Shim, "APEX: An Adaptive Path Index for XML Data," *In Proceedings of the 2002 ACM SIGMOD Conference*, pp.121-132, 2002.

[6] Q. Li and B. Moon, "Indexing and Querying XML Data for Regular Path Expressions," *In Proceedings of the 27VLDB*, pp.361-370, 2001.

[7] S. Al-Khalifa, H. V. Jagadish, N. Koudas, J. M. Patel, D. Srivastava, and Y. Wu, "Structural Joins: A Primitive for Efficient XML Query Pattern Matching," *In Proceedings of the 18th IEEE International Conference on Data Engineering*, pp.141-152, 2002.

[8] S. -Y. Chien, Z. Vagena, D. Zhang, V. Tsotras, and C. Zaniolo, "Efficient Structural Joins on Indexed XML Documents," *In Proceedings of the 28th VLDB Conference*, pp.263-274, 2002.

[9] H. Wang, S. Park, W. Fan, and P. S. Yu, "ViST: A Dynamic Index Method for Qu- erying XML Data by Tree Structures," *In Proceedings of the 2003 ACM SIGMOD Conference*, pp.110-121, 2003.

[10] P. Rao and B. Moon, "Sequencing XML Data and Query Twig for Fast Pattern M- atching," *ACM Transactions on Database Systems(TODS)*, pp.299-345, 2006.

[11] P. F. Dietz, "Maintaining Order in a Linked List," *In Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, pp.122-127, 1982.

[12] J. M. Hellerstein, J. F. Naughton, and A. Pfeffer, "Generalized Search Trees for Database Systems," *In Proceedings of the 21th VLDB Conference*, pp.562-573, 1995.

[13] P. Rao and B. Moon, "PRIX Project," http://www.cs.arizona.edu, 2006.

[14] G. Miklau, "UW XML Repository," http://www.cs.washington.edu/research/xmldata- sets, 2006.

서 동 민

2002년 충북대학교 정보통신공학과(공학사). 2004년 충북대학교 정보통신공학과(공학석사). 2008년 충북대학교 정보통신공학과(공학박사). 2008년 3월~현재 한국과학기술원 전산학과 연수연구원. 관심분야는 데이타베이스 시스템, 에이전트 시스템, XML, 이동 객체 데이타베이스, 시공간 색인 구조, 센서 네트워크 등

송 석 일

1998년 충북대학교 정보통신공학과(공학사). 2000년 충북대학교 정보통신공학과(공학석사). 2003년 충북대학교 정보통신공학과(공학박사). 2003년 8월~현재 충주대학교 컴퓨터공학과 교수. 관심분야는 데이타베이스 시스템, 센서 네트워크, 이동 객체 데이타베이스, 저장 관리 시스템, XML 등

유 재 수

정보과학회논문지 : 데이타베이스
제 35 권 제 2 호 참조