XML 질의 처리를 위한 효율적인 시퀀스 매칭 기법

서동민*0 송석일** 유재수*
충북대학교 정보통신공학과 충주대학교 컴퓨터공학과 충북대학교 정보통신공학과
dmseo@chungbuk.ac.kr sisong@chungju.ac.kr yjs@chungbuk.ac.kr

An Efficient Sequence Matching Method for XML Query Processing

Dong-Min Seo*0 Seok-Il Song** Jae-Soo Yoo*
*Department of Computer and Communication Engineering, Chungbuk National University
**Department of Computer Engineering, Chungju National University

1. 서 론

XML 문서에 대한 XML 질의를 효율적으로 처리하기 위한 경로 색인 그래프 기반의 XML 질의 처리 기법의 문제를 해결하기 위해 부모-자식 또는 조상-후손 관계를 효율적으로 처리할 수 있는 구조 색인과 번호 부여 기법(*numbering scheme*)에 대한 연구가 많이 이루어졌다. 하지만 구조 조인 기법은 단일-경로 질의를 효율적으로 처리 하지만 다중-경로 질의는 항상 두 개 이상의 부 질의(*sub query*)로 분해한 뒤, 각각의 부 질의에 대한 단일-경로 질의를 통해 처리하고 최종 결과는 많은 비용이 요구되는 부 질의 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 [1]. Rao *et al.*은 XML 문서와 XML 질의를 *LPS*(*Labeled Prüfer Sequence*)와 *NPS*(*Numbered Prüfer Sequence*)로 변환하는 PRIX를 제안하였다 [2]. ViST와 PRIX는 다중-경로 질의를 처리하기 위해, 다중-경로 질의를 부 질의로 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.

2. 제안하는 시퀀스 매칭 기법

본 논문 제안하는 색인 구조는 ViST의 번호 부여 기법 문제를 해결하기 위해 *Durable* 번호 부여 기법[3]을 사용한다. 그림 1은 본 논문에서 제안하는 색인 구조를 보여준다. 그림 1의 XML *Doc1*, *Doc2*와 같은 문서들에 대해 *Durable* 번호인 <*order*, *size*>을 부여하고, ViST와 유사하게 시퀀스 색인을 위한 D-Ancestor B+트리와 시퀀스 노드들에 부여된 번호 색인을 위한 S-Ancestor B+트리를 구축한다.

또한, 시퀀스 노드들의 *prefix* 정보를 활용하면 질의 처리 성능을 크게 향상시킬 수 있다. 그래서 본 논문에서는 질의 처리 시, 시퀀스 노드들의 *prefix* 정보를 활용하는 상향식 질의 처리 기법을 제안한다. 단일-경로 질의의 경우, 제안하는 상향식 질의 처리 기법은 질의를 구성하는 마지막 시퀀스 노드에 대해서만 제안하는 색인 구조를 통해 시퀀스 매칭을 수행한다. 또한, 예제 1에서 볼 수 있듯이 와일드-카드가 포함된 질의의 경우, ViST에 비해 제안하는 색인 구조가 질의 처리 시 적은 노드를 접근하기 때문에 질의 처리 성능이 향상됨을 볼 수 있다.

예제 1 *와일드-카드가 포함된 단일-경로 질의 /P/*/L/v1을 고려하자. 질의의 시퀀스는 (P, ε)(L, P*)(v1, P*L)이고 질의의 마지막 시퀀스 노드는 (v1, P*L)이다. 그러므로 제안하는 상향식 질의 처리 기법은 그림 5의 D-Ancestor B+트리에 대해 (v1, P*L)에 대한 범위 질의를 수행한다. (v1, P*L)의 범위 질의 결과로 (v1, PSL)을 얻는다. 그리고 (v1, PSL)을 포함하는 문서들을 찾기 위해 그림 1의 S-Ancestor B+트리에 대해 (v1, PSL)에 대한 범위 질의를 수행한다.*

알고리즘 1은 제안하는 상향식 질의 처리 기법을 통한 다중-경로 질의 처리 과정을 보여준다. 기본 처리 방법은 단일-경로 질의 처리와 같다. 즉, 다중-경로를 구성하는 각 경로의 마지막 노드들에 대한 시퀀스 노드 매칭을 수행한다. 또한, *false alarms*을 해결하기 위해 다중-경로 질의의 연결 노드에 대한 범위 질의와 연결 노드들과 마지막 노드들에 대한 구조 관계를 검사한다. 예제 2에서 볼 수 있듯이 제안하는 상향식 질의 처리 기법은 *Durable* 번호인 <*order*, *size*>을 통해 범위 질의를 수행하기 때문에 *false alarms*문제가 해결되는 것을 볼 수 있다.

이 논문은 2007년도 정부(교육인적자원부)의 재원으로 한국학술진흥재단(지방연구중심대학 육성사업 / 충북BIT연구중심대학 육성사업단)과 한국과학재단 특정기초 연구(과제번호R01-2006-000-10809-0)의 지원에 의하여 연구되었음

예제 2 *다중-경로 질의 /P/S[L/v11]/L/v2을 고려하자. 질의의 시퀀스는 (P, ε)(S, P)(L, PS)(v1, PSL)(L, PS)(v2, PSL)이다. 질의를 구성하는 각 경로의 마지막 노드는 (v1, PSL)과 (v2, PSL)이다. 그리고 다중-경로 질의의 연결 노드는 (S, P)이다. 제안하는 상향식 질의 처리 기법은 먼저, 그림 1의 D-Ancestor B+트리에 대해 (v1, PSL), (v2, PSL), (S, P)에 대한 범위 질의를 수행한다. 그리고 범위 질의 결과로 얻은 (v1, PSL), (v2, PSL), (S, P)의 각 S-Ancestor B+트리에 대한 범위 질의를 수행한다. 위 범위 질의를 통해 (v1, PSL)에 부여된 <30, 0>, (v2, PSL)에 부여된 <41, 0>, (S, P)에 부여된 <10, 20>과 <31, 10>을 얻는다. 마지막으로, 획득한 연결 노드의 번호들과 마지막 노드의 번호들에 대한 구조 관계를 검사한다. ㅋ(연결노드.order$_l$) < ㅋ(각 마지막 노드.order$_l$) ≤ ㅋ(연결노드.order$_l$+연결노드.size$_l$)를 통해 연결 노드의 한 번호가 각 마지막 노드들의 번호를 포함한다면 구조 관계가 성립됨을 알 수 있다. 예를 들어, 위 범위 질의를 통해 획득한 번호들에 대해서는 {10<30, 41≤10+20}과 {31<30, 41≤31+10}을 검사한다. 결과적으로, 위 계산의 모든 결과가 거짓이기 때문에 Doc1과 Doc2에서는 질의를 만족하는 서브 시퀀스가 존재하지 않음을 알 수 있다.*

알고리즘 1 : BranchingQuery Algorithm

```
Input : QS          // QS is a query sequence
Output : (D, S) // D is a set of document identifiers
                // S denotes the positions of subsequence match
procedure BranchingQuery(QS)
  // retrieve connection nodes and last nodes from QS
  (CN, LN) = SequencePaser(QS)
  // retrieve the S-Ancestor B+Trees from D-Ancestor B+Tree
  (CS, LS) = FindDAncestor(CN, LN)
  // retrieve all numbering values from S-Ancestor B+Trees
  (CV, LV) = FindSAncestor(CS, LS)
  // determine the structural relationship for the ancestor-descendant
  // or parent-child among nodes from CV and LV
  (D, S) = DetermineRelationship(CV, LV)
  output(D, S)
end
```

3. 결 론

그림 2에서 볼 수 있듯이, 제안하는 색인 구조의 상향식 질의 처리 기법이 질의 처리 시 요구되는 중간 노드들에 대한 접근을 확연히 줄이기 때문에 ViST와 PRIX에 비해 우수한 성능을 나타냄을 확인할 수 있다. 또한, ViST에서는 *false alarms* 문제로 인해 질의에 대한 잘못된 결과가 나타나는데 반해, 제안하는 색인 구조는 *Durable* 번호 부여 기법을 통해 질의를 처리하기 때문에 *false alarms* 문제가 발생하지 않는 것을 확인할 수 있었다.

참고문헌

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

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

[3] Q. Li and B. Moon, "Indexing and Querying XML data for regular path expressions", *In VLDB*, pp.361-370, 2001.

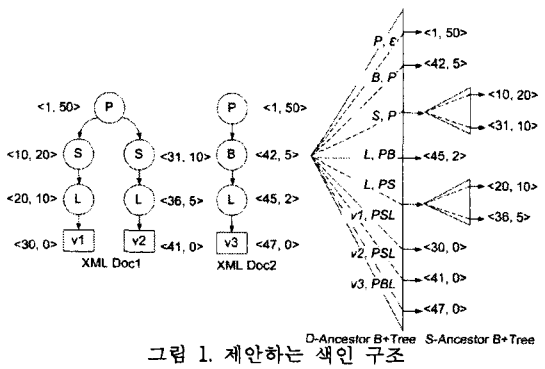

그림 1. 제안하는 색인 구조

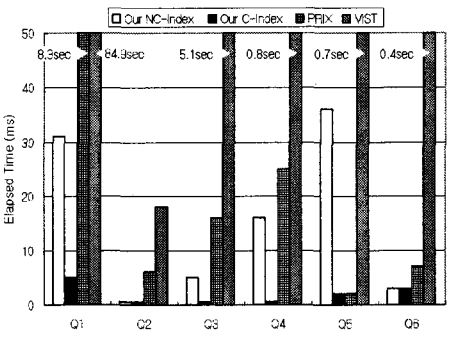

그림 2. 질의 처리 시간