XML 질의의 효율적인 전처리를 위한 시그너처 방법

(A Signature Method for Efficient Preprocessing of XML Queries)

정 연 돈 † 김 종 욱 †† 김 명 호 †††
(Yon Dohn Chung) (Jong Wook Kim) (Myoung Ho Kim)

요 약 본 논문은 대량의 XML 문서들이 존재하는 정보 검색 시스템에서, XML 질의의 효과적인 처리를 위한 선 처리 방법을 제안한다. 선 처리를 위해 시그너처 기반의 접근 방식을 사용한다. 기존의 (평면적인 문서를 사용하는) 정보 검색 시스템에서는, 대부분 사용자 질의들이 키워드와 부울 연산자로 구성되고, 따라서 시그너처 역시 평면적인 형태로 구성하고 있다. 하지만, XML 기반의 정보 검색 시스템에서는 사용자 질의가 경로 질의의 형태를 띄게 된다. 따라서, 평면적인 시그너처는 XML 문서에 대하여 효과적이지 못하다. 본 논문에서는 XML 문서를 위한 구조화된 시그너처 방법을 제안한다. 실험을 통해 제안하는 방법의 성능을 평가한다.

키워드 : XML, 시그너처, 정보 검색

Abstract The paper proposes a pre-processing method for efficient processing of XML queries in information retrieval systems with a large amount of XML documents. For the pre-processing, we use a signature-based approach. In the conventional (flat document-based) information retrieval systems, user queries consist of keywords and boolean operators, and thus signatures are structured in a flat manner. However, in XML-based information retrieval systems, the user queries have the form of path query. Therefore, the flat signature cannot be effective for XML documents. In the paper, we propose a structured signature for XML documents. Through experiments, we evaluate the performance of the proposed method.

Key words : XML, Signature, Information Retrieval

1. 서 론

XML(eXtensible Markup Language)은 최근 인터넷(internet) 상에서의 자료 교환 및 표현을 위한 표준으로 자리잡고 있으며, 웹(web) 상에서 동작하는 다양한 응용들에서 사용하는준 구조적(semi-structured) 데이타를 표현하는 대표적인 방법이다[1]. XML은 사용자들이 정의하는 태그(tag)들을 사용하여 정의되며, DOM[2]이라 불리는 트리 형태의 구조로 구성된다. 아래의 그림 1은 간단한 XML 문서와 그 문서의 DOM 트리를 나타낸 것이다.

· 본 연구는 2003년도 동국대학교 신임교원연구비 지원으로 이루어졌음
† 비 회 원 : 동국대학교 컴퓨터멀티미디어공학과 교수
ydchung@dgu.edu
†† 비 회 원 : LG전자 디지털미디어연구소 연구원
jwkim@dbserver.kaist.ac.kr
††† 종신회원 : 한국과학기술원 전산학과 교수
mhkim@dbserver.kaist.ac.kr
논문접수 : 2002년 4월 23일
심사완료 : 2003년 6월 18일

XML 문서에 대한 정보 검색을 위하여 지금까지 Lorel[3], XML-QL[4], XQuery[5] 등의 다양한 질의어들이 제시되어 오고 있다. 이들 질의어들은 모두 경로 질의(path query)라는 특성을 지니고 있으며, '정규 표현식(regular expression)'으로 표현 가능하다[3,6]. 표 1은 정규 표현식에서 사용되는 일반적인 기호들이다. 예를 들어, 그림 1의 문서에서 정규 표현식 "invoice/_/[address='Seoul']"으로 나타낸 질의는 구매자 혹은 판매자의 주소가 '서울'인 모든 레코드를 검색하는 것이다.

XML 문서의 저장 방법에 대하여, 기존의 객체형 데이타베이스를 사용하는 방법과 일반적인 파일 시스템을 사용하는 방법들에 대한 연구들이 이루어져 오고 있다[6,7]. 본 논문은 파일 시스템을 사용하는 환경을 가정하고 있으며, 특히 도서관의 서지 정보와 같이 작은 크기의 XML 문서 파일들이 많이 존재하는 정보 시스템을 가정하고 있다. XML에 대한 기존의 연구[6,9]들이 하나의 대규모 XML 문서에 대한 효율적인 색인 및 질의처리 방법에 대한 연구들이었던 점에서, 본 논문에서 제

```
<invoice>
  <seller>
    <name> ABC Shopping Inc. </name>
    <address> Seoul </address>
  </seller>
  <buyer>
    <name> Hong Gil Dong </name>
    <address> Busan </address>
  </buyer>
  <item count=3> toy </item>
  <item count=2> shoe </item>
</invoice>
```

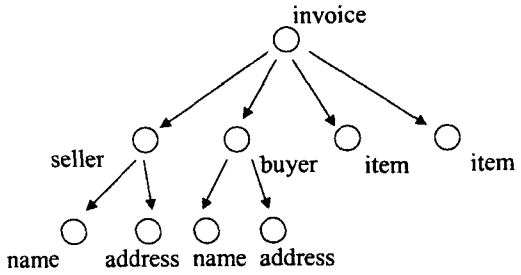


그림 1 XML 문서와 트리 표현의 예

표 1 경로 질의를 구성하는 정규 표현식의 기호

기호	의미
_	임의의 문자열을 나타낸다
|	두 문자열의 논리 합을 나타낸다
?	0번 또는 1번 반복됨을 나타낸다
+	1번 이상 반복됨을 나타낸다
*	0번 이상 반복됨을 나타낸다
[ ]	프레디키트(predicate)를 나타낸다
@	속성을 나타낸다
( )	선행 관계를 표시한다
/	수준을 구분한다

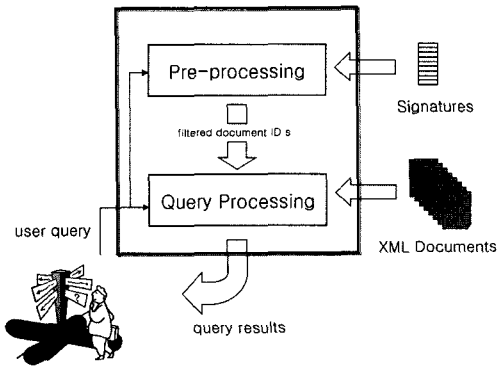


그림 2 시그너처를 이용한 XML 질의 선 처리 구조

안하는 방법은 실제 질의 처리를 시행할 문서의 개수를 줄이는 선 처리 방법이라는 차이점을 갖는다.

그림 2는 본 논문에서 제안하는 질의 처리를 위한 선 처리 (pre-processing)의 개념을 소개하고 있다. 다량의 XML 문서들이 존재하는 시스템 환경이며, 각 문서에서 시그너처[8,9](signature) 형태의 요약 정보를 미리 추출한다. 사용자의 질의를 처리할 때, 요약된 정보를 먼저 접근하여, 해당 문서가 입력된 사용자 질의에 부합할 가능성이 있는지 여부를 사전 검사한다. 이러한 선 처리를 통해 불필요한 문서의 접근을 줄임으로써, 질의 성능을 개선하는 것이다.

논문의 구성은 다음과 같다. 2장에서는 기존의 정보 검색(IR: Information Retrieval)분야에서 이루어졌던 시그너처 개념에 대하여 알아본다. 기존의 시그너처 방법이 XML 문서 검색에서 적절하지 못한 이유도 함께 살펴본다. 3장에서는 XML 문서 검색을 위해, 본 논문에서 제안하는 새로운 시그너처의 구조와 이 시그너처를 사용하는 질의 선 처리 방법에 대하여 기술한다. 4장에서 성능 실험 결과를 설명한 후, 5장에서 결론을 맺는다.

2. 배경 및 동기

기존의 텍스트 기반 정보 검색 분야에서, 사용자의 질의 문자열을 포함하는 문서를 찾는 검색을 위하여 시그너처를 이용한 선 처리 기법에 대하여 많은 연구들이 있었다[8]. 시그너처란 문서에 나타나는 각 단어들을 하나의 비트 열로 해시한 후, 각 해시 비트열 들을 연결 혹은 OR한 결과 비트열을 말한다. 이 시그너처를 통해 각 문서에 어떠한 단어들이 존재하는지 알 수 있게되어, 문서 검색 시 모든 문서에 대하여 검색할 필요 없이, 시그너처를 통해 검색 질의어가 있을 가능성이 있는 문서만을 검색하게 된다. 그림 1의 XML 문서에 대하여, 표 1과 같은 해시 함수 H를 사용하여 기존의 시그너처 기법을 적용할 경우, 이 문서의 시그너처는 이들 해시 열을 OR한 결과인 "011101101"이 된다.

하지만, 기존의 텍스트를 기반으로 하는 정보 검색에서는 '키워드'와 같은 문자열들의 논리식(AND 및 OR)으로 질의를 생성한 반면, XML 문서들의 경우에는 경로 식으로 주어지는 사용자 질의들을 처리하여야 한다. 예를 들어, 'invoice'와 'seller'라는 문자열이 포함되어 있는 문서를 찾는 것이 아니라, invoice 와 seller가 부모와 자식의 관계, 즉 '/invoice/seller'의 관계로 존재하는 문서를 찾아야 하는 것이다. 이 경우, 평면적으로 각 문자열의 시그너처들을 합하여(OR 연산) 문서 시그너처를 생성하는 일반적인 방법은 그 효용성에 문제를 지니게 된다. 따라서, 본 논문에서는 경로식 형태로 주어지는 사용자 질의에 대하여, 효과적으로 대응할 수 있는 시그너처 방법을 제안한다.

표 2 해시 함수 H를 이용한 문자열들의 해시 결과 예

H(invoice)	010000100
H(seller)	001001000
H(buyer)	000100000
H(item)	001000001
H(name)	000001000
H(address)	000100100

3. 시그너처를 이용한 선 처리 방법

본 논문에서 제안하는 XML 문서를 위한 시그너처 구조는 '수준 시그너처'와 '경로 시그너처'의 두 가지로 구성된다. 두 가지 시그너처를 설명한 후, 시그너처를 이용한 선 처리 방법에 대하여 살펴본다.

먼저, 논문에서 사용할 용어들에 대하여 정의하기로 한다. 하나의 XML 문서는 하나의 DOM 트리 형태로 표현되며, 트리의 원소(element)와 속성(attribute) 문자열에 대하여 해시 비트 열을 생성하는 해시 함수 H가 존재한다고 가정한다. 용어 설명의 편의상 트리는 완전히 균형잡힌 (full-balanced) n-ary 트리라고 가정한다. (실제로 논문에서 제시하는 방법은 트리의 형태에 대하여 아무런 제약을 두지 않는다. 그림 3은 완전히 균형잡힌 트리를 사용하지 않은 예이다) 트리의 높이를 'h'라고 정의하면, 루트 노드(root node)는 1번째 수준(level) 노드이며, 최종 단말 노드(leaf node)는 'h'번째 수준의 노드가 된다. 그리고, 문서 D를 나타내는 트리에서, $N_D(a,b)$를 'a'번째 수준의 'b'번째 노드로 정의한다. 따라서, 트리에서 가장 우측 최종 단말 노드는 $N_D(h, n^{(h-1)})$이 된다. $N_D(a,b)$ 노드에 대한 시그너처 $H(N_D(h, n^{(h-1)}))$는 $S_D(a,b)$로 나타내기로 한다. 노드가 포함하는 원소 이름과 속성 값들을 모두 포함하여 시그너처를 생성한다.

3.1 수준 시그너처와 경로 시그너처의 구조

수준 시그너처 (Level Signature: LS)는 XML 문서를 DOM 트리 형태로 구성하였을 때, 트리의 각 수준(level)별로 노드들에 대한 해시 값들을 OR한 시그너처로서 다음과 같이 정의된다. 그리고, 각 수준별 시그너처들을 연결한 경로 배열을 그 문서의 수준 시그너처 경로 (LS Path: LSP)라고 한다.

정의 1. XML 문서 D의 i번째 수준 시그너처 $LS_D(i)$는 다음과 같이 정의된다. (여기에서 '|'는 비트열 OR 연산자이다.)

$$LS_D(i) = S_d(i,1) \mid S_d(i,2) \mid \cdots \mid S_d(i, n^{(h-1)})$$

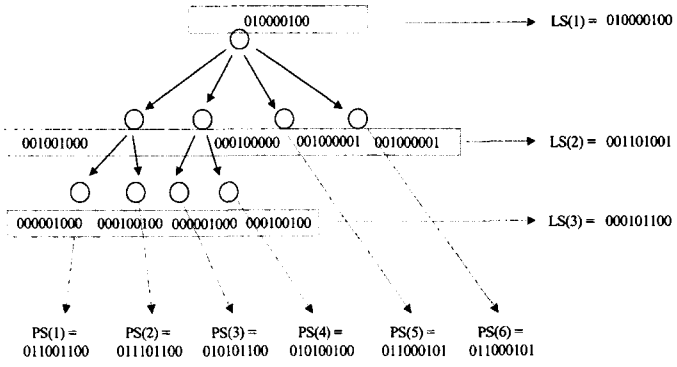
정의 2. XML 문서 D의 수준 시그너처 경로 LSP_D는 "$LS_D(1)/LS_D(2)/ \ldots /LS_D(h)$"이다. □

경로 시그너처 (Path Signature: PS)는 XML 문서를 DOM 트리 형태로 구성하였을 때, 트리의 각 경로(path)별로 노드들에 대한 시그너처의 OR 값으로 정의된다. 그리고 문서의 모든 경로들에 대한 경로 시그너처들의 집합을 경로 시그너처 집합 (PS Set: PSS)라고 부른다.

정의 3. XML 문서 D의 k 번째 경로, 즉 루트 노드에서 $N_D(h,k)$ 노드에 이르는 경로에 대한 $PS_D(k)$는 $\sum_{j=1}^{h} S_D(j, w)$이다. 이때 w는 이 경로를 이루는 노드들의 위치들을 나타내며, $\sum$는 OR 연산의 합을 나타낸다.

정의 4. XML 문서 D의 경로 시그너처 집합 PSS_D는 $\{PS_D(k) \mid k=1, \ldots, n^{(h-1)}\}$이다. □

그림 3은 그림 1에서 예를 든 문서에 대하여 본 논문



LSP = 010000100/001101001/000101100
PSS = {011001100, 011101100, 010100100, 010101100, 011000101}

그림 3 그림 1의 문서 예에 대한 LS 및 PS 적용 예

에서 제안하는 시그너처(LS 및 PS) 방법을 적용한 예를 나타내고 있다. LSP는 각 수준별 시그너처들을 경로 형태로 연결한 것이며, PSS는 각 경로별 시그너처들의 집합이다. (그림에서는 설명의 편의를 위해, XML 문서의 원소에 대해서만 시그너처를 생성하고 있지만, 실제는 원소뿐만 아니라 문자열로 나타나는 모든 내용을 시그너처로 생성한다.)

사용자의 질의가 입력되면, 먼저 LSP와 PSS를 검색한 후, 해당 질의가 두 가지의 시그너처를 만족하는지 확인하게 된다. 시그너처 확인에서 통과되면 해당 문서를 읽어서 질의 처리를 실행하게 되며, 그렇지 않을 경우에는 이 문서를 디스크로부터 읽을 필요가 없는 것이다. 시그너처 정보는 메모리 상에 존재하게 되며, 실제 XML 문서들은 디스크 상에 존재하기 때문에, 작은 선처리 비용으로 상당수의 디스크 검색을 줄일 수 있기 때문에, 효과적인 질의 검색이 가능하게 된다. 이제부터 두 가지 시그너처를 사용하여 XML 질의를 선 처리하는 방법에 대하여 살펴보자.

3.2 선 처리 알고리즘

사용자의 질의는 정규 표현식(regular expression)으로 표현되는 문자열로 주어진다. 사용자의 경로 질의 문자열 PQ에서 '/'로 구분되는 각 문자열들을 PQ_i라고 하며, PQ_i들의 개수를 PQ의 크기라고 부르며, 't'이라고 표기한다. 즉, "PQ = $PQ_1/PQ_2/ \ldots / PQ_t$"이 된다. 따라서, 경로 질의 PQ가 "invoice/_*/name"일 때, PQ_1 = 'invoice', PQ_2 = '_*', PQ_3 = 'name'이다. get_first( ) 함수는 경로 질의에서 맨 처음 나타나는 문자열 PQ1을 돌려주는 함수이며, move_next( ) 함수는 첫 번째 문자열을 제거하고, 두 번째 문자열부터 시작하는 경로 질의를 돌려주는 함수이다. (경로 구분자인 '/'도 생략된다.) 예를 들어 질의 PQ가 "invoice/ _*/name"일 때, get_first(PQ)는 'invoice'이며, move_next(PQ)는 "_*/name"이 된다.

이제 LSP와 PSS를 이용한 질의 선 처리 알고리즘에 대하여 알아보자. 먼저 PSS를 이용한 선 처리 방법을 설명한다. PSS를 이용한 질의 선 처리는 PSS를 구성하는 각 PS들을 사용자 질의 PQ와 비교하는 것이다. 그림 4는 PSS를 이용한 질의 선 처리 알고리즘이다. 먼저 사용자 질의 PQ를 구성하는 각 문자열들을 추출하여 그들의 해시 비트열을 모두 OR하여 질의 시그너처(Q-Sig)를 만든 후, 이 질의 시그너처를 각 경로 시그너처(PS)와 부합하는지를 비교하게 된다[1]. 문서의 모든 PS들이 질의 시그너처와 부합하지 않는 경우, 이 문서

```
function PreProcess-PSS (PSP, PQ)
/* INPUT: PSP (the set of PS for a document) , PQ (the path query) */
/* OUTPUT: TRUE (matched) or FALSE (not matched) */
{
    remove '?', '+', and '*' characters from PQ;
    Q-Sig = '000...0';
    while (PQ != NULL) {
        String = get_first(PQ);
        if (String starts with '_') { /* do nothing */ }
        else { Q-Sig = Q-Sig | H(String); /* bit-wise OR operation */ }
        PQ = move_next(PQ);
    }
    foreach PS in PSS {
        if ((PS & Q-Sig) == Q-Sig) { /* bit-wise AND operation */
            return TRUE;        /* no more check */
        }
    }
    return FALSE;
}
```

그림 4 PSS를 이용한 질의 선 처리 알고리즘

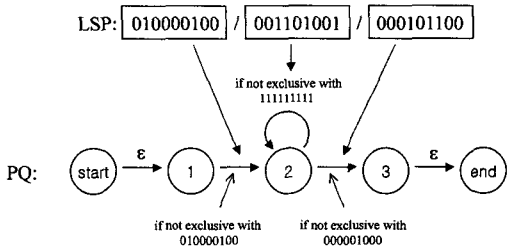


그림 5 LSP를 이용한 질의 선 처리 NFA의 예

는 선 처리에서 제거되어 본 처리를 하지 않게 된다.

LSP를 사용한 선처리는 문서의 시그너처 경로(LSP)와 사용자가 입력한 경로 질의를 비교하는 것이다. 정규 표현식의 합치 여부를 판단하는 방법이므로, 기본적으로 NFA와 동일하다. 단, 질의의 각 문자열을 문서의 시그너처를 생성할 때 사용한 해시 함수를 사용하여 비트열로 변환시키는 과정을 필요로 하고, 문자열의 일치대신 질의의 각 해시 비트열과 LSP의 각 비트열 사이의 포함관계를 판단하여 NFA를 수행하게 된다. 이를 위해, 먼저 사용자가 입력한 경로 질의에서 '/'로 구분되는 각 문자열들을 해시 함수 H를 통해 비트 열로 변환시킨다. '_'는 모든 문자열을 나타낼 수 있으므로 '111..1' 형태의 비트 열로 변환한다. 반복을 나타내는 '?', '+', '*' 등은 본래 NFA에서 처리 가능하므로 그대로 유지된다. 가령, 사용자 질의가 "invoice/_*/name"일 경우, 해시 변환 결과는 "010000100/111111111*/ 000001000"이 된다. 이 질의와 그림 1의 문서 LSP인 "010000100/001101001/000101100"를 NFA를 통해 선 처리하는 예를 그림 5에 나타내고 있다.

두 가지의 선 처리 과정을 통해 해당 문서가 사용자

1) 이 알고리즘은 사용자 질의에 논리 합 ('|') 기호가 들어있지 않은 경우를 가정한다. 사용자 질의에 논리 합 기호가 있는 경우에는, 여러 개의 질의로 분리하여 처리한다.

질의에 부합되지 않는다고 결정될 경우, 이 문서는 본(main) 질의 처리 과정을 할 필요가 없게 된다. 두 선처리 방법 모두를 통과한 경우만 문서를 읽어서 질의 처리를 하면 된다. 본 질의 처리 과정에서는 기존에 연구된 각종 색인 기법들이나 질의 처리 기법들을 사용할 수 있다[6,9].

4. 성능 실험

4.1 실험 개요

실험 환경은 Linux(mandrake 8.1), Pentium3 600 Mhz, 256MB 메모리 환경이며, g++ 플랫폼(platform)을 기반으로 구현하였다. XML문서를 파싱하기 위한 파서는 Xerces toolkit(c++ version 1.5.2)를 사용했다. Xerces toolkit[10]은 XML 문서를 순차적으로 읽으면서 시작 태그와 종료 태그를 만날 때마다 이벤트를 발생시켜 주는 역할을 한다.

XML 문서를 만들기 위해서 DocBook DTD [11]를 사용했다. DocBook DTD는 컴퓨터 하드웨어나 소프트웨어와 관련된 논문과 책을 작성하기 위해 고안된 DTD이다.

XML 문서는 IBM의 XML 문서 생성기[12]를 이용하여 만들었다. IBM XML 문서 생성기는 임의의 DTD를 입력받아서, XML 문서를 합성해주는 역할을 한다. 이때, XML 문서는 정규분포(normal distribution)를 갖도록 생성된다. 즉, 각 수준(level)에서 원소(element)들의 선택은 균일(uniform)하게 이루어진다. 본 실험에서 사용된 문서 생성 패러미터 값들은 최대 깊이(max_depth) 10, 각 노드별 최대 자식 노도의 개수 (max_repeat) 5이다. 따라서, 높이 10 이하, 각 노드별 자식노드의 개수가 5 이하인 다양한 형태의 트리 구조를 갖는 문서들이 생성된다.

실험 방법은 다음과 같다. 문서 생성기를 이용하여 실험에 사용될 문서를 생성한 후, Xerces toolkit을 이용하여 파싱하여 각 문서의 시그너처를 메모리에 저장했다. 그리고 사용자로부터 임의의 질의를 입력받아 선 처리 과정을 거쳐 필터링 되는 문서의 비율을 측정하였다. (즉, 선 처리 과정을 통과하여 실제 본 질의 처리를 해야하는 문서의 비율을 말하며, 이 값이 작을수록 질의 처리 비용을 줄일 수 있다.) 실험 대상은 (1) LSP만 사용하는 선 처리 방법, (2) PSS만을 사용하는 선 처리 방법, (3) LSP와 PSS를 함께 사용하는 선 처리 방법, 그리고 (4) 기존의 문서 검색에서 사용되던 일반적인 시그너처 방법(OLD)이다. 단위 시그너처의 크기는 32bit와 64bit, 두 가지를 대상으로 하였다. 실험에 사용한 질의 유형들은 다음과 같다.

- DTD의 루트에 가까운 노드들을 지닌 질의들
  - Q1: /set/book/title
  - Q2: /set/book/_*/title
  - Q3: /set/book/_/title
  - Q4: /set/book/_/_/title
- DTD에 존재하지 않는 노드를 지닌 질의들('body'는 DTD에서 정의되지 않은 노드임.)
  - Q5:/set/book/body
  - Q6:/set/book/_*/body
  - Q7: /set/book/_/body
- DTD의 루트에 가까운 노드들과 중간 이후의 노드들로 구성된 질의들
  - Q8: /set/_*/title/procedure
  - Q9: /set/_*/title/_/procedure
  - Q10: /set/_*/title/_/_/procedure
  - Q11: /set/_*/title/_*/procedure
- DTD의 중간 이후의 노드들로 구성된 질의들
  - Q12: /_*/partintro
  - Q13: /_*/partintro/sect1
  - Q14: /_*/partintro/_*/sect1
  - Q15: /_*/partintro/_/sect1
  - Q16: /_*/partintro/_/_/sect1
- DTD와 노드 순서가 다른 질의들
  - Q17: /_*/title/book
  - Q18: /_*/title/_*/book

4.2 실험 결과

두 가지 크기의 단위 시그너처를 사용하여 질의 선처리를 한 결과를 표 3과 4에 나타내었다. 문서 100개에 대해 각 질의를 선 처리한 후, 본 질의 처리를 하여야 하는 문서의 수를 나타낸다. 즉, 질의 선 처리를 통해 필터링되고 남은 문서의 비율(%)로 이해할 수 있다(이 값이 '0'인 경우는 선 처리를 통해, 해당 질의를 만족하는 문서가 하나도 없다는 사실을 발견한 것이다. 따라서 본 질의 처리를 할 필요가 없다.).

기존의 시그너처 방법(OLD)은 질의의 구조를 고려하지 않기 때문에, 각 질의 그룹별로 동일한 결과를 보이지만, 제안한 방법들은 동일 질의 그룹에서도 실제 질의 유형에 따라 다른 값을 보인다. 질의의 깊이가 낮은 경우에는, PSS 방법보다 LSP 방법이 더 나은 방법을 보인다는 점을 알 수 있다. PSS 방법의 경우, 경로를 구성하는 각 노드들에 대한 시그너처들을 모두 OR하는 개념이기 때문에, 기존 방법과 유사해 지는 경향이 많기 때문이다. 하지만, 질의의 깊이가 깊어질 경우, PSS의 필터링 효과가 높아지게 된다.

표 5는 각 방법들을 사용할 때 필요한 메모리 사용량을 나타내고 있다. 문서 100개에 대한 값으로, 문서의 개수가 증가하면 단조 증가하게 된다. 선 처리에 필요한

표 3 선 처리 과정을 통과하는 문서 비율 (%) - 시그너처 크기 32비트

Query No.	OLD	LSP	PSS	LSP+PSS
1	96	73	85	69
2	96	82	85	74
3	96	57	85	54
4	96	61	85	56
5	42	1	31	0
6	42	1	31	0
7	42	1	31	0
8	66	38	21	18
9	66	29	21	18
10	66	30	21	18
11	66	37	21	20
12	80	44	70	43
13	72	21	33	17
14	72	26	33	17
15	72	22	33	16
16	72	9	33	5
17	96	37	85	36
18	96	57	85	35

표 4 선 처리 과정을 통과하는 문서 비율 (%) - 시그너처 크기 64비트

Query No.	OLD	LSP	PSS	LSP+PSS
1	8	7	8	7
2	8	8	8	8
3	8	8	8	8
4	8	6	8	6
5	6	3	6	3
6	6	6	6	6
7	6	1	6	1
8	6	2	5	2
9	6	3	5	2
10	6	1	5	1
11	6	5	5	4
12	6	0	6	0
13	6	0	6	0
14	6	0	6	0
15	6	0	6	0
16	6	0	6	0
17	6	0	4	0
18	6	0	4	0

표 5 각 방법별 메모리 사용량

	OLD	LSP	PSS	LSP+PSS
32 비트 단위 시그너처 사용시	400 bytes	3.5 KB	8.5 KB	12 KB
64 비트 단위 시그너처 사용시	800 bytes	7 KB	17 KB	24 KB

표 6 문서 깊이가 5일 때의 선처리 결과

Query No.	OLD	LSP	PSS	LSP+PSS
1	93	72	76	62
2	93	67	76	63
3	93	47	76	47
4	93	36	76	35
5	41	1	13	0
6	41	1	13	0
7	41	1	13	0
8	40	24	16	13
9	40	18	16	7
10	40	21	16	10
11	40	35	16	14
12	36	11	19	6
13	33	7	19	3
14	33	4	19	2
15	33	3	19	1
16	33	1	19	1
17	93	19	76	18
18	93	35	76	33

표 7 문서 깊이가 2일 때의 선처리 결과

Query No.	OLD	LSP	PSS	LSP+PSS
1	8	6	8	6
2	8	8	8	8
3	8	8	8	8
4	8	6	8	6
5	4	2	4	2
6	4	4	4	4
7	4	1	4	1
8	5	2	4	2
9	5	3	4	2
10	5	1	4	1
11	5	4	5	4
12	6	0	6	0
13	6	0	6	0
14	6	0	6	0
15	6	0	6	0
16	6	0	6	0
17	6	0	4	0
18	6	0	4	0

수행 시간은 64비트 시그너처를 사용한 경우 1ms의 시간이 소모되었으며, 문서의 개수를 100,000개로 늘인 경우, 1초 가량의 선 처리 시간이 소모되었다. 따라서, 본 논문에서 제안한 선 처리 방법은 아주 효과적으로 XML 질의 처리 성능을 향상시킬 수 있는 방법이라고 판단된다.

표 6과 7은 XML 문서의 깊이에 따른 실험 결과들이다. 64비트의 시그너처를 사용하였으며, 사용된 문서의 수는 10개이다. (XML 문서 생성기에서는 주어진 DTD를 기반으로 이에 부합하는 문서를 생성하기 때문에, 정확한 트리의 깊이 혹은 원소의 개수를 보장하지 못하게 된다. 가령, DTD에 따라 깊이가 6인 문서는 생성될 수 있지만 7인 문서는 생성이 안 되는 경우가 발생할 수 있다. 표 6, 7의 실험을 위해 문서 생성기를 통해 문서를 생성한 다음, 생성된 문서들을 읽어서 입력한 깊이 패러미터 값과 가깝게 생성된 문서들을 수작업으로 골라서 시행하였다. 그래서, 실험에 사용된 문서의 개수가 크지 못하다.) 따라서, 실험 결과 값의 통계적 신뢰도가 높다고 할 수는 없다. 하지만, 문서의 깊이와 같은 요인이 제안하는 선처리 방법에 영향이 있는지를 보이는 데

에는 도움이 된다고 판단된다.

실험 결과에 따르면, 문서의 깊이가 커질수록 시그너처의 효과가 조금 떨어지는 모습을 나타내고 있다. 문서의 깊이가 길어지면서 경로 시그너처의 경우 경로의 길이가 길어지기 때문이며, 수준별 시그너처 역시 깊이가 증가하면서 한 레벨에 나타나는 노드의 수가 커지기 때문에 당연한 결과라 할 수 있다. 기존의 일반적인 시그너처 방법 역시, 깊이가 커지면서 노드의 개수가 증가하기 때문에 시그너처의 효과가 떨어지게 된다.

5. 결 론

본 논문에서는 XML 질의 처리 성능 향상을 위하여, 시그너처 개념을 사용한 질의 선 처리 방법은 제안하였다. 기존의 정보 검색에서 사용되던 일반적인 시그너처 방법은 경로 질의를 사용하는 XML 문서에서는 비 효과적이기 때문에, 본 논문에서는 경로 질의의 구조적 특징을 반영할 수 있는 두 가지 시그너처 구조를 제안하였다. 수준별 시그너처 (LSP)는 XML 문서의 각 수준별로 노드들의 시그너처를 OR한 시그너처들로 구성한 시그너처의 경로 배열이며, 경로 시그너처 (PSS)는 XML 문서내에 존재하는 루트에서 각 단말 노드로의 모든 경로에 대한 시그너처들의 집합이다. 즉, XML 문서에 대한 수직적 시그너처와 수평적인 시그너처의 조합으로, 경로 질의로 주어지는 XML 질의 선 처리에 효과적으로 사용될 수 있다. 실험을 통하여 제안하는 방법들의 성능을 보였으며, 이러한 선 처리를 통하여 전체적인 질의 비용을 개선할 수 있다는 점을 알 수 있었다. 본 논문에서 제안한 XML 질의 선처리 개념은 아직까지 거의 연구된 바가 없으며, 기존의 연구들과 같이 사용될 수 있다는 점에서 의미를 지닌다. 향후 연구로서, XML 질의 선 처리에서 이루어지는 작업의 결과를 사용하여, 본 질의 처리를 빨리 수행하는 방안에 대한 연구가 가능할 것으로 판단된다.

참 고 문 헌

[1] Tim Bray, et. al., Extensible markup language (XML) 1.0 second edition W3C recommendation. Technical Report REC-xml-20001006, World Wide Web Consortium, 2000.

[2] W3C. Document Object Model (DOM), http://222.w3.org, Feb. 2000.

[3] S. Abiteboul, et. al., "The Lorel query language for semistructured data," International Journal on Digital Libraries, Vol. 1, No. 1, pp. 68-88, 1997.

[4] Stefano Ceri, et. al., "XML-GL: A graphical language forquerying and restructuring XML documents," In proceedings of the 8th International World Wide Web Conference, pp. 93-109, 1999.

[5] Don Chamberlin, et. al., XQuery: A Query Language for XML, W3C working draft. Technical Report WD-query-20010215, World Wide Web Consortium, 2001.

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

[7] Daniel Florescu and Donald Kossmann, A performance evaluation of alternative mapping schemes for storing XML data in a relational database. Technical Report 3680, INRIA, Rocquencourt, France, 1999.

[8] Christos Faloutsos, "Signature files: Design and performance comparison of some signature extraction methods," In proceedings of ACM SIGMOD, pp. 63-82, 1985.

[9] Sangwon Park and Hyung-Joo Kim, "A New Query Processing Technique for XML Based on Signature," In proceedings of DASFAA, pp. 22-31, 2001.

[10] "Apach XML Project," http://xml.apache.org

[11] "DocBook DTD," http://www.docbook.com

[12] A. L. Diaz, and D. Lovell, "XML Generator," http://www.alphaworks.ibm.com/tech /xmlgenerator

정 연 돈

1994년 2월 고려대학교 전산과학과 학사 1996년 2월 한국과학기술원 전산학과 석사. 2000년 8월 한국과학기술원 전자전산학과 전산학전공 박사. 2000년 9월~2001년 8월 한국과학기술원 정보전자연구소 Post-Doc. 연구원. 2001년 9월~2003년 2월 한국과학기술원 전자전산학과 전산학전공 연구교수. 2003년 3월~현재 동국대학교 컴퓨터 멀티미디어 공학과 교수. 관심분야는 XML, Spatio-temporal Databases, Stream Data Processing, Mobile Distributed Systems, Database Systems 등

김 종 욱

2000년 2월 고려대학교 전산과학과 학사 2002년 8월 한국과학기술원 전산학과(석사). 2003년 4월~현재 LG전자 디지털미디어 연구소 연구원. 관심분야는 XML, Semi-structured data, Database System 등

김 명 호

정보과학회논문지 : 데이타베이스
제 30 권 제 1 호 참조