XML 데이타베이스에서 경로-지향 질의처리를 위한 병렬 매치 방법

(A Parallel Match Method for Path-oriented Query Processing in XML Databases)

박 희 숙 † 조 우 현 ††
(Hee-Sook Park) (Woo-Hyun Cho)

요 약 XML은 인터넷상에서 데이타를 표현하고 교환하기 위한 새로운 표준이다. 본 논문에서는, XML문서에 대한 경로-지향 질의어의 평가를 위한 새로운 접근법에 대하여 기술한다. 본 논문의 접근법에서는, 경로-지향 질의어의 평가속도를 개선하기 위해 경로서명을 이용하는 병렬 매치 인덱싱 구조의 제안과 함께 데이타베이스 안에 저장된 엘리먼트들의 경로서명들과 입력된 질의어의 경로서명 사이에 매치 작업을 수행하기 위한 병렬 매치 알고리즘을 설계한다. 먼저, 병렬 매치 구조를 형성하기 위해서는 XML 문서상의 모든 경로서명들에 대한 이진 트라이를 구성한 다음 이들을 병렬 매치 인덱싱 구조로 변환한다. 경로-지향 질의어의 검색 연산을 수행하기 위해 병렬 매치 인덱싱 구조와 병렬 매치 알고리즘을 사용한다. 본 논문에서 제안한 방법에서 알고리즘의 시간 복잡도는 XML 문서내의 경로서명의 수에 대하여 로그 값에 비례한다.

키워드 : XML, 경로서명, 병렬 매치 인덱싱 구조, 이진 트라이

Abstract The XML is the new standard for data representation and exchange on the Internet. In this paper, we describe a new approach for evaluating a path-oriented query against XML document. In our approach, we propose the Parallel Match Indexing Fabric to speed up evaluation of path-oriented query using path signature and design the parallel match algorithm to perform a match process between a path signature of input query and path signatures of elements stored in the database. To construct a structure of the parallel match indexing, we first make the binary trie for all path signatures on an XML document and then which trie is transformed to the Parallel Match Indexing Fabric. Also we use the Parallel Match Indexing Fabric and a parallel match algorithm for executing a search operation of a path-oriented query. In our proposed approach, Time complexity of the algorithm is proportional to the logarithm of the number of path signatures in the XML document.

Key words : XML, path signature, Parallel Match Indexing Fabric, binary trie

1. 서 론

XML은 네트워크상에서 문서를 교환하고 조작하기 위해 사용되는 반-구조적(semi-structured)으로 데이타를 표현하는 인기 있는 문법이다. XML의 잠재성은 무한하며 XML을 이용한 많은 애플리케이션들이 현재 개발되고 있다[1-4]. 대규모 관계형 데이타베이스에 저장된 XML문서에 대하여 이슈된 경로-지향 질의(Path-Oriented Query)를 평가하는 문제는 오늘날 중요한 이슈가 되고 있다. 따라서 XML문서를 저장하고 있는 데이타베이스로부터 질의를 수행할 때 질의 평가 성능을 높이기 위한 효과적이고 개선된 인덱싱 알고리즘에 대한 연구가 필요하다.

XML문서에 대한 경로-지향 질의어 평가를 위해서는 XML문서 내의 모든 엘리먼트(Element)들의 경로와 검색을 위해 입력되는 경로-지향 질의어의 경로들 사이에 매치(Match)를 수행하기 위해 매치 알고리즘을 사용한다. 이것은 인덱싱 시스템의 성능을 측정하는데 있어서 중요한 핵심요소이다. 지금까지 경로-지향 질의어에 대한 평가 성능을 향상하기 위한 몇 가지 순차적 매치 방

† 학생회원 : 부경대학교 전자컴퓨터정보통신공학부
bg007@edunet4u.net
†† 정 회 원 : 부경대학교 전자컴퓨터정보통신공학부 교수
whcho@pknu.ac.kr
논문접수 : 2005년 1월 24일
심사완료 : 2005년 6월 9일

법들이 제안되었다. 그 대표적인 예로는 관련이 없는 엘리먼트들에 대한 접근을 최대한 회피하기 위한 방법으로 각 경로-지향 질의어의 경로 서명(Path Signature)이 스캐닝될 때 각 비트 값에 따라서 패트리샤 트리(Patricia Tree) 기술과 결합하는 것으로 성능을 향상하는 인덱싱 기술이 Chen에 의해 제안되었다[8]. 또 다른 접근법으로는 엘리먼트들에 대한 넘버링 스키마를 기반으로 한 XML문서 저장과 인덱싱을 위한 기법이 소개되었다[10].

본 논문에서는 경로서명을 이용하여 경로-지향 질의어의 평가 처리 속도를 개선하기 위해 병렬 매치 방법을 사용한다. 병렬 매치 방법에 관한 기존의 연구는 주로 텍스트 문서 검색을 위한 방법들이 주류였으며, 그들 중에서 가장 대표적인 것으로는 SIMD 컴퓨터 구조와 수평적 서명 분할법을 사용하는 방법이 Stanfill과 Kahle에 의해 이미 연구된 바 있다. 이것은 본 논문에서 제안하는 방법과 유사한 방법을 사용하고 있다[14]. 그러나 본 논문에서는 텍스트가 아닌 경로에 기반을 둔 XML문서를 검색대상으로 하며, 평가를 하고자하는 질의어 형태는 Xpath등과 같은 경로-지향 질의를 평가 대상으로 한다. 또한, 각 엘리먼트들에 대한 경로 정보를 고려한 경로서명 파일을 생성하며, 관계형 데이타베이스에 저장된 XML문서에 대한 평가를 위해 경로-지향 질의어가 입력될 때 이들의 검색 속도를 개선하기 위해 이 경로서명 파일을 이용하여 병렬 매치 인덱싱 구조를 구성한다.

본 논문의 병렬 매치 인덱싱 접근법은 다음 같은 순서로 실행한다. 첫째, 생성된 경로서명 파일을 이용하여 이진 트라이 구조를 구성한다. 둘째, 구성된 이진 트라이 구조를 프로세스 엘리먼트(Process Element: PE)들로 구성된 병렬 매치 인덱싱 구조(Parallel Match Indexing Fabric: PMIF)로 변환한다. 셋째, 평가를 위해 입력된 경로-지향 질의에 포함된 엘리먼트들의 경로서명 값을 생성하고 연속적인 이진 비트 패턴(binary bit pattern)들로 이루어진 이들의 경로서명 값을 비트단위로 루트에서부터 아래쪽으로 연결된 각 PE들로 입력된 값을 순차적으로 전달한다. 넷째, 병렬 매치를 수행하기 위해 루트 PE에 매치 처리 수행 시작 신호를 입력한다. 본 논문에서 제안한 병렬 매치 알고리즘의 시간 복잡도는 O(h)이다.

본 논문은 다음과 같이 구성된다. 2장에서는 관련연구와 기존 방법의 소개 및 문제점을 제시하고 3장에서 병렬 매치 시스템의 설계와 실험 그리고 실험 결과 분석에 대한 내용을 기술한다. 마지막 4장에서 결론 및 향후 과제에 대하여 논의한다.

2. 관련연구

본 장에서는 XML문서의 간단한 예와 문서를 구성하고 있는 엘리먼트, 애트리뷰트, 텍스트의 표현방법 및 경로서명(Path Signature)을 생성하는 방법들에 대하여 기술한다.

2.1 XML 기술들

XML은 SGML에서 유도되었으며, 단순하고 매우 유연한 텍스트 형식을 가진다. XML문서는 시작태그와 끝태그로 둘러싸여 있으며, 태그는 대소문자를 구별한다. 또한, XML문서는 DTD(Document Type Descriptor)에 의해 미리 기술된 구조와 태그 규칙을 따라서 구성된다. 문법에 맞게 작성된 XML문서를 "적정 문서(well-formed document)"라 하며, XML문서가 문법에 맞으면서 DTD에 맞게 작성된 문서를 "유효한 문서(valid document)"라 하고 이것을 체크하기 위한 프로그램을 파서(parser)라 한다[11,12]. XML문서는 DOM(Document Object Model) 파서를 이용하여 트리(tree)와 같이 표현될 수 있다. 트리에서 노드의 형태는 각각 엘리먼트(Element)와 애트리뷰트(Attribute) 그리고 텍스트(Text)중에 한가지이다[8,11]. XML문서의 간단한 예는 그림 1과 같다.

```
<library filecode="c1023">
  <creation>
    <book>
      <title> Object Database </title>
      <author> Lee Soon-Sin</author>
      <publisher> ABC Co. </publisher>
      <date> July 2nd 2002 </date>
    </book>
  </creation>
  <report>
    <subject> Dynamic Hashing </subject>
    <writer> Kim Yoo-Sin </writer>
    <date> June 5th 2004 </date>
  </report>
</library>
```

그림 1 간단한 XML 문서의 예

2.2 경로-지향 언어와 경로서명

XQL과 XML-QL 같은 몇 가지 경로-지향 질의 언어들이 XML문서들의 교차 참조(cross-reference) 및 애트리뷰트들과 엘리먼트들을 트리와 같은 구조로 다루기 위해 제안되었다. 다음은 그림 1의 XML문서에 대한 간단한 경로-지향 질의어 예이다.

/library/report [writer$contains&'Kim Yoo-Sin']

여기서 '/library/report'는 경로이고, [writer$contains

&'Kim Yoo-Sin']는 술어이며, 엘리먼트 writer가 단어 'Kim Yoo-Sin'을 포함하는가 여부를 질의한다. 입력된 경로-지향 질의에 대해 데이타베이스 관리 시스템은 질의어 안에 포함된 엘리먼트에 대한 경로를 평가해야만 한다. 즉, 데이타베이스 관리시스템은 질의어 내부의 엘리먼트들이 데이타베이스 내에 저장되어 있는지 아닌지를 알기 위해 매치 작업을 수행해야 하며 매치된 결과를 이용하여 그들이 저장된 데이타베이스의 위치를 검색한다.

경로서명 파일(Path Signature File)은 부정확한 여과장치(Inexact Filter)의 개념을 기반으로 하고 있다. 따라서 그들은 많은 부적합한 값들을 버리는 빠른 테스트를 제공한다. 그러나 적합한 값들은 확실히 테스트를 통과한다. 그러나 몇 개의 값들은 실제로는 검색 요구 조건을 만족하지 않는다 할지라도 우연히 여과장치를 통과하게 되는 경우가 발생할 수 있으며, 이들을 허위드롭(False Drop)이라 한다. 허위드롭의 가능성을 최소화하기 위해서는 경로서명 값을 생성할 때 다음의 공식들을 사용한다.

$$Fd=2^{-m} \quad (1)$$

$$Fln2=mD \quad (2)$$

경로서명은 1로 설정된 n개의 비트를 가진 길이 h인 해시-코드화된 이진비트 패턴들이다. 경로서명을 생성하기 위한 방법으로는 전형적으로 중첩기호법(Superimposed ORing)을 사용한다. 어떤 경로-지향 질의어가 도착했을 때 먼저 질의어 안에 포함된 경로에 대한 경로서명 값을 생성한 후 이것을 포함하고 있는 모든 엘리먼트들을 경로서명 파일로부터 검색한다. 이 과정에서 많은 부적합한 엘리먼트들은 버려지게 되고 적합한 엘리먼트들은 허위드롭을 제거하기 위해 체크되어지거나 사용자에게 검색결과로서 되돌려 준다[8,13].

경로서명 파일의 사용 목적은 대부분의 부적합한 엘리먼트들을 가려내는 것이다. 본 논문의 실험에 사용되는 각 엘리먼트의 경로서명 값은 XML문서의 파싱과정에서 각 엘리먼트들의 경로 정보를 고려하여 생성한다. 즉, 루트 엘리먼트에서 각 엘리먼트에 이르는 경로 상에 존재하는 모든 엘리먼트들의 첫 번째 머리글자만을 추출하고 이들을 결합(concatenation)하여 각 엘리먼트의 경로서명 값을 생성한다. 우리는 이것을 단축-경로라 부르고 이들을 경로서명 파일에 저장한다[15]. 여기서 만들어진 경로서명 파일은 우리의 병렬 매치 인덱싱 구조를 형성하는 데 사용된다.

예를 들어 그림 1의 예제 문서에서 엘리먼트 title의 경로는 library-creation-book-title이다. 따라서 이 엘리먼트의 경로서명은 "lcbt"가 되며, 경로서명의 길이를 16비트라 가정한다면 실제 병렬 매치 인덱싱 구조를 형성할 때에는 이것의 해시 코드값 "0000001101101110"을 사용한다.

2.3 기존 방법의 소개 및 문제점

기존의 병렬 검색 방법에 관한 연구는 Stanfill과 Kahle[14]에 의해 이미 연구된 바 있으나 그것은 텍스트 문서를 검색 대상으로 하며, 주로 불리언 질의를 평가하기 위한 방법이다. 그렇기 때문에 본 논문에는 우리와 동일하게 XML문서에 대한 한 경로-지향 질의어 평가를 위해 연구된 Chen의 방법[8]을 성능비교 대상으로 삼는다.

Chen의 방법은 경로-지향 질의에 대한 평가 성능을 향상시키기 위하여 경로서명 파일과 패트리샤 트리(Patricia Tree: PT)를 결합한 방법을 제안하고 있다.

패트리샤 트리는 이진 디지털 트리(Binary Digital Tree)이며, 내부노드에는 분기를 위한 각 비트의 위치를 결정하는 값을 저장하고 각 외부노드에 한 개의 정보만을 저장한다[8]. 기존의 논문[8]에서 정보검색 연산은 패트리샤 트리를 이용하여 테스트를 수행하고 있으며, 경로서명에서 각 위치 비트의 값이 1이면 오른쪽 자식노드를 따라가고 0이면 왼쪽 자식노드를 모두 따라가게 된다. 따라서 기존의 방법은 입력되는 질의어의 경로서명 해시-코드가 포함하는 1의 비트수와 경로서명 파일에 저장된 경로서명의 엔트리 수에 따라서 검색성능에 많은 영향을 받는다. 만약 1의 비트수가 많아지면 검색을 위한 비교횟수는 적어지고 1의 비트의 수가 적어진다면 비교횟수는 상대적으로 많아진다. 또한, 입력되는 인덱스 키의 수가 많아짐에 따라 패트리샤 트리의 구조 확장이 요구된다. 기존[8]의 방법에서는 최악의 경우에 있어서 검색을 위한 키의 최대 비교횟수는 $O(N/2^{l})$이다. 여기서 N은 경로서명 파일이 포함하는 경로서명의 수이며 l은 입력되는 질의어의 경로서명 값에서 이진수 1로 설정된 비트들의 개수이다. 따라서 N의 수가 커지면 키의 비교 횟수도 증가하게 된다. 본 논문에서는 많은 적합한 엘리먼트들을 검색하는데 있어서 경로서명 매치 작업을 수행할 때 성능을 개선하기 위한 새로운 접근법으로 병렬 매치 인덱싱 구조와 병렬 매치 알고리즘의 사용을 제안한다.

3. 병렬 매치 인덱싱 구조 설계 및 실험

본 장에서는 XML문서가 파싱되어 데이타베이스 내부에 저장되는 형태와 경로서명 파일을 이용하여 병렬 매치 인덱싱 구조를 구성하는 방법 그리고 이들에 대한 병렬 매치 알고리즘 수행에 관하여 기술한다.

3.1 전체 시스템의 구성

본 논문의 전체 시스템 구성도는 그림 2와 같다. 모든

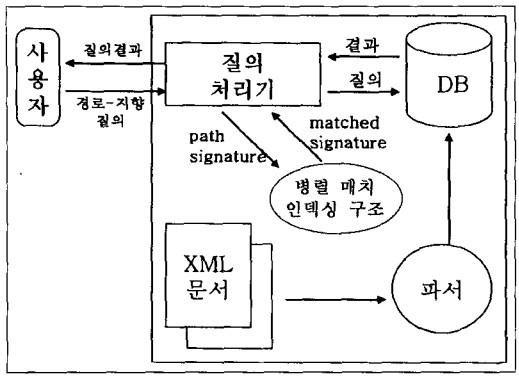


그림 2 전체 시스템 구성도

XML문서는 파서에 의해 파싱되어 관계형 데이타베이스에 엘리먼트, 텍스트, 애트리뷰트 테이블로 각각 분리 저장된다. 또한, XML문서내의 모든 엘리먼트들은 파싱과정에서 트리구조를 따라 엘리먼트들의 경로를 탐색한 다음 이들의 경로서명을 생성하여 엘리먼트 테이블에 함께 저장한다. 엘리먼트 테이블은 {문서ID(숫자), 엘리먼트ID(숫자), 엘리먼트명(문자열), 부모ID(숫자), 경로서명(문자열)}로 구성되며, 텍스트 테이블은 {문서ID(숫자), 부모ID(숫자), 텍스트값(문자열)}로 구성된다. 그리고 애트리뷰트 테이블은 {문서ID(숫자), 부모ID(숫자), 애트리뷰트명(문자열), 애트리뷰트값(문자열)}로 구성된다.

전체 시스템 구성에서 질의처리기(Query Processor)는 다음과 같은 기능들을 수행한다. 첫째, 사용자가 입력한 경로-지향 질의어에 포함된 경로들의 경로서명을 생성한 다음 병렬 매치 인덱싱 구조로 전달한다.

둘째, 병렬 매치 작업에 의해 여과된 결과를 이용하여 경로-지향 질의어를 관계형 질의어 형태로 변환하고 이를 이용하여 XML문서가 저장된 관계형 데이타베이스에 질의를 요청한다.

셋째, 질의 수행 결과를 적당한 출력양식으로 구성하여 사용자에 전달하는 역할을 수행한다.

3.2 병렬 매치 인덱싱 구조의 생성

본 논문에서는 경로-지향적 질의어의 평가 성능을 개선하기 위해 병렬 매치 인덱싱 구조를 사용한다. 병렬 매치 인덱싱 구조는 기존의 이진 트라이(Binary Trie) 구조와 유사하다. 이진 트라이는 이진 비트 패턴들을 이용하여 트라이를 구성하며 중간노드인 분기노드(Branch Node)와 외부노드인 정보노드(Information Node)로 구성되어 있다. 모든 정보는 외부노드에만 저장되기 때문에 분기노드는 정보를 가지지 않고 링크만을 가진다. 이진 트라이에서 최대 레벨은 경로서명의 길이(h)와 같다. 본 논문은 XML문서의 모든 엘리먼트들에 대한 경로서명들을 이진 트라이로 구성하는 알고리즘 설계로부터 시작한다. 예를 들어 하나의 XML문서내의 모든 엘리먼트들에 대한 경로서명 파일의 내용이 다음과 같다고 하자.

경로서명 파일 = {0011, 0111, 0101, 1100, 1010, 1101, 1110}

위의 경로서명 파일을 이진 트라이 구조로 구성하면 그림 3과 같다.

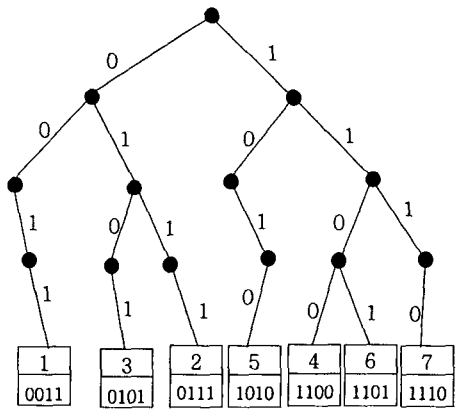
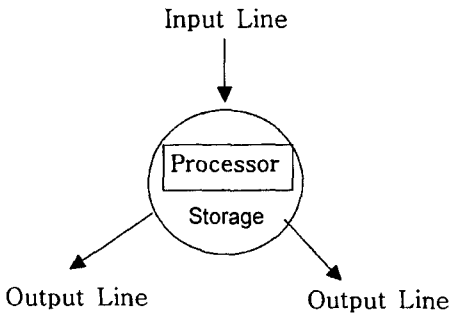


그림 3 경로서명파일의 이진 트라이 구조

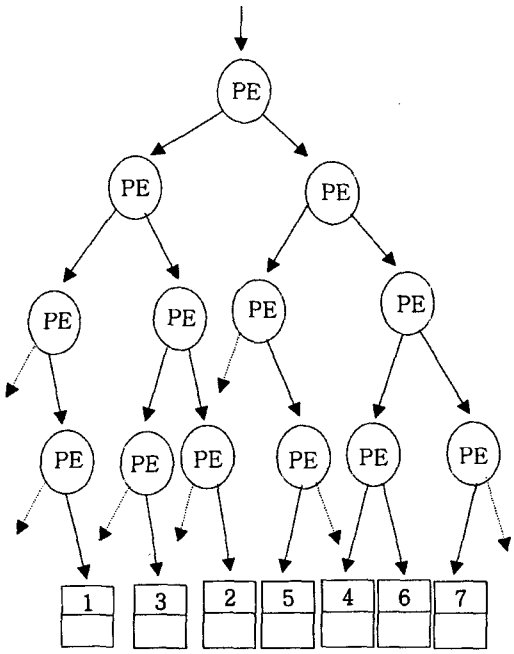
만약 경로-지향 질의어에 포함된 엘리먼트들에 대한 경로서명이 0110이라고 한다면 경로서명 파일에서 적합한 엘리먼트들에 대한 매치 결과로 나타나는 경로서명은 0111, 1110이다.

본 논문에서는 적합한 엘리먼트들을 검색하는데 있어서 검색 성능을 향상시키기 위해 그림 3의 이진 트라이 구조를 병렬 매치 구조로 변환한다. 병렬 매치 구조로 변환하는 작업은 그림 3의 이진 트라이 구조에서 각 중간노드인 분기노드를 PE로 변환하고 정보노드에는 경로서명 파일내의 경로서명들에 대한 정보를 저장한다. 병렬 매치 인덱싱 구조에서 각 PE는 그림 4의 (a)에서 보는 바와 같이 한 개의 입력선(Input Line)과 2개의 출력선(Output Line) 그리고 약간의 저장 공간(Storage)과 단순 비교기(Comparator)로서의 역할하는 프로세서로 구성되어 있다. 외부노드인 정보노드는 경로서명 파일내의 각 경로서명들의 주소와 매치여부를 알려주는 매치 표시기(Match Indicator)로 구성된다. 각 PE는 부모 PE로부터 하나의 신호를 입력받아 처리 한 후 2개의 출력선을 통하여 왼쪽, 오른쪽 자식 PE로 신호를 전달한다. 각 PE들은 모두 동등하며 병렬로 동작한다. 위의 예제 경로서명 파일을 사용하여 구성한 병렬 매치 구조는 그림 4의 (b)와 같다.

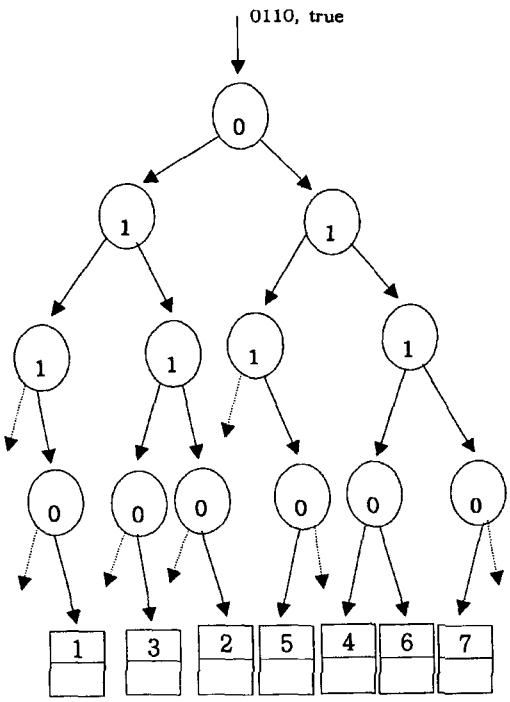
본 논문의 병렬 매치 인덱싱 구조에 사용되는 알고리즘은 세 가지 단계로 동작을 한다. 첫 번째 단계는 검색할 경로서명 비트 값들을 병렬 매치구조의 각 PE들에



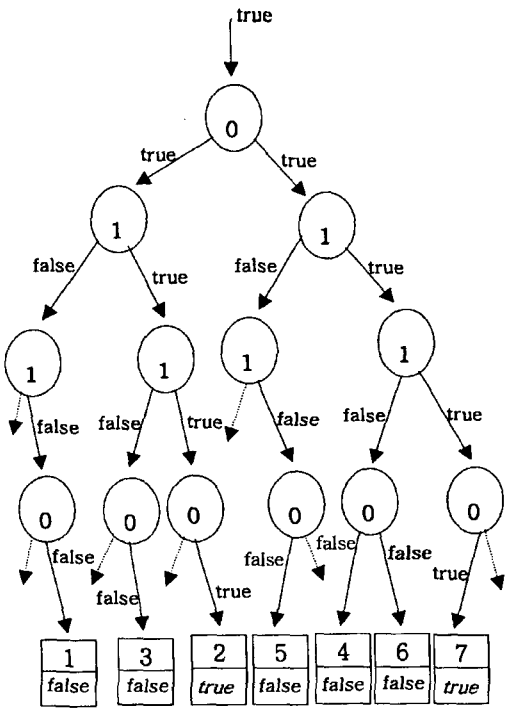
(a) PE



(b) 초기상태



(c) h시간이후



(d) 2*h시간이후

그림 4 병렬 매치 인덱싱 구조와 매치 알고리즘 수행과정

저장하는 단계이고, 두 번째 단계는 매치 또는 노-매치 신호(match/no-match signal)를 전달하여 매치 작업을 병렬로 수행하는 단계이다. 마지막 세 번째 단계는 병렬 매치 수행 결과 값을 수집하는 단계이다.

첫 번째 단계는 먼저 검색을 위해 입력된 경로-지향 질의어가 포함하고 있는 엘리먼트들에 대한 경로서명 값을 생성하고 이것을 가장 오른쪽 비트부터 역순으로 클럭당 한 비트씩 하위 레벨에 연결된 자식 PE들로 전달한다. 각 PE는 입력선으로부터 입력받은 비트 데이타를 자신의 기억장소에 저장하고 자신이 보유하고 있던 비트 데이타를 왼쪽과 오른쪽 자식 PE들에게 전달한다. 경로서명의 모든 비트들은 경로서명 길이와 같은 h클록이 경과하면 경로서명의 가장 오른쪽 비트는 가장 아래쪽 단에 연결된 PE들에 도착하게 되고 가장 왼쪽 비트는 최상위 루트 PE에 도착하게 되어 검색을 원하는 경로서명의 모든 비트 값들을 PE에 저장하는 작업은 완료된다. 두 번째 단계는 전 단계에 이어서 루트 PE에 매치 신호를 입력한다. 각 PE들은 입력단에 신호가 입력되면 입력된 신호가 매치 신호인가, 노-매치 신호인가를 판단하여 매치신호가 입력되었다면 각 PE는 자신이 보유한 비트 값이 '0'이면 왼쪽과 오른쪽 자식 PE로 매치 신호를 전달하고 저장된 값이 '1'이면 왼쪽 자식 PE에 노-매치 신호를 오른쪽 자식 PE에 매치-신호를 전달한다. 그러나 만일 입력선으로 노-매치 신호가 전달되었다면 왼쪽과 오른쪽 자식 PE 모두로 노-매치 신호를 전달한다. 여기서 우리는 매치신호로 True를 노-매치 신호로는 False를 입력한다. 그림 4의 (c)와 (d)는 경로서명 0110에 대한 병렬 매치 인덱싱 처리 과정을 보여주고 있다. 마지막 세 번째 단계는 이전 단계 수행결과 매치 표시기가 true로 되어 있는 경로서명들의 정보를 결과 값으로 수집하는 단계이다.

3.3 병렬 매치 알고리즘

본 논문의 병렬 매치알고리즘이 수행되기에 앞서 모든 XML문서는 파싱 과정을 거쳐서 관계형 데이타베이스내에 미리 저장되어 있어야하며, 이 파싱과정에서 XML 문서내에 포함된 각 엘리먼트들에 대한 경로서명 값이 생성되고 이들은 모두 경로서명 파일에 저장된다. XML문서 파싱과 경로서명 파일 생성을 위한 알고리즘은 그림 5와 같다.

병렬 매치 알고리즘은 크게 두 가지로 구별한다. 그 첫 번째는 병렬 매치 인덱싱 구조를 생성하기 위한 알고리즘이며 두 번째는 검색을 위해 입력되는 경로서명을 병렬로 매치하는 작업을 수행하는 알고리즘이다. 병렬 매치 인덱싱 구조를 생성하는 알고리즘은 그림 6과 같다.

두 번째는 병렬 매치 인덱싱 구조를 이용하여 검색하

```
input : DOM파서에 의해 파싱된 XML문서의 각노드, 트리에서 노드의 레벨
output: 경로서명파일, 엘리먼트테이블, 에트리뷰트테이블,텍스트테이블

XML문서파싱함수(XML문서의 각 노드, 트리에서 노드의 레벨) {
    현재 노드의 노드타입을 구함;
  switch(노드타입) {
    case 문서노드:
        문서객체 생성;
        XML문서파싱함수를 재귀적으로 호출하고 파라메터로
        문서의 루트 엘리먼트와 루트 엘리먼트 노드의 레벨을 전달;
        break;
    case 엘리먼트노드:
        부모노드ID 값으로 현재 엘리먼트 노드의 ID값을 저장;
        현재 엘리먼트 노드의 ID값을 1증가;

        엘리먼트 경로추적을 위해 엘리먼트ID를 배열 IDTrace에 저장;
        단축-경로 생성에 사용될 경로상의 엘리먼트명을 배열 PathTrace에 저장;

        엘리먼트명로 루트레벨에서 현재 엘리먼트노드의 레벨까지
        loop를 돌면서 배열 PathTrace를 사용하여 단축-경로를 생성;

        문서ID, 엘리먼트ID, 엘리먼트명,부모노드ID,단축-경로를
        엘리먼트 테이블에 삽입;

        생성된 단축-경로를 경로서명 파일에 저장;

        현재 엘리먼트 노드가 포함하고 있는 에트리뷰트 개수만큼
        loop를 돌면서 XML문서파싱함수를 재귀적으로 호출하고
        파라메터로 에트리뷰트 노드와 엘리먼트노드의 레벨을 전달;

        현재 엘리먼트노드의 자식노드가 존재한다면 자식노드의
        개수만큼 loop를 돌면서 XML문서파싱함수를 재귀적으로
        호출하고 파라메터값으로 자식노드와 자식노드의 레벨을
        전달;
        현재 노드의 부모노드ID 값을 추적하여 부모노드ID 값으로 저장;
        break;
    case 에트리뷰트노드:
        부모노드ID 값으로 현재 엘리먼트 노드의 ID값을 저장하고
        문서ID, 부모노드ID, 에트리뷰트명, 에트리뷰트값을
        에트리뷰트 테이블에 삽입;
        break;
    case 파싱 되지 않은 문자:
    case 텍스트노드:
        만약 텍스트노드 또는 파싱되지 않은 문자의 길이가 0이 아니면
          부모노드ID 값으로 현재 엘리먼트 노드의 ID값을 저장하고
          문서ID, 부모노드ID, 텍스트값을 텍스트 테이블에 삽입한다
        그렇지 않다면 텍스트 테이블에 삽입하지 않는다
  }
}
```

그림 5 XML문서 파싱과 경로서명 파일 생성 알고리즘

```
input : 인덱스 키의 경로서명 값, 루트 PE 노드의 주소, 경로서명의 길이

병렬_매치_인덱스구조_생성함수(인덱스 키의 경로서명 값, 루트 PE 노드의 주소, 경로서명의
길이)
{
 루트 PE 노드의 주소를 현재노드의 주소를 가리키는 변수의 값으로 지정;
for(i=1 to 경로서명의 길이-1)
 {
   if (인덱스 키의 경로서명 값 i-번째 비트가 0이면서 왼쪽 자식 PE 노드가 없다면){
     새로운 왼쪽 자식 PE 노드를 생성하고 부모 PE 노드와 연결;
     현재노드의 주소를 가리키는 변수의 값으로 새 PE 노드의 주소값을 지정;
     }

   if (인덱스 키의 경로서명 값 i-번째 비트가 1이면서 오른쪽 자식 PE 노드가 없다면){
     새로운 오른쪽 자식 PE 노드를 생성하고 부모 PE 노드와 연결;
     현재노드의 주소를 가리키는 변수의 값으로 새로운 PE 노드의 주소값을 지정;
    }

   if (인덱스 키의 경로서명 값 i-번째 비트가 0이면서 왼쪽 자식 PE노드가 존재한다면){
     현재노드의 주소를 가리키는 변수의 값으로 왼쪽 자식 PE의 주소값을 지정;
     }

   if (인덱스 키의 경로서명 값 i-번째 비트가 1이면서 오른쪽 자식 PE 노드가 존재한다면){
     현재노드의 주소를 가리키는 변수의 값으로 오른쪽 자식 PE 노드의 주소값을 지정;
   }
}
  if (인덱스 키의 경로서명 값의 가장 오른쪽 비트 값이 0이라면){
    새로운 정보노드를 생성하고 현재 노드의 왼쪽 자식 노드로 연결;
   }
  if (인덱스 키의 경로서명 값의 가장 오른쪽 비트 값이 1이라면){
    새로운 정보노드를 생성하고 현재 노드의 오른쪽 자식노드로 연결;
   }
}
```

그림 6 병렬 매치 인덱싱 구조 생성 알고리즘

고자 하는 경로-지향 질의어의 경로서명 값과 경로서명 파일에서 매치 되는 경로서명들을 찾아내는 과정을 수행하는 알고리즘이다. 이 알고리즘은 입력된 경로-지향 질의어의 경로서명에 대한 각 비트 값들을 전달하기 위한 알고리즘과 매치/노-매치 신호를 병렬 매치 인덱싱

```
input : 검색대상 질의어의 경로서명 값, 경로서명의 길이, 루트 PE노드와 주소
output : match된 경로서명 파일내의 경로서명들의 주소

경로서명_매칭함수(검색대상 질의어의 경로서명 값, 경로서명의 길이, 루트 PE노드의 주소)
{
  병렬 매치 인덱싱 구조의 모든 정보노드의 매치 표시기의 값을 false로 리셋;

 for(clock=1 to 경로서명의 길이)
  {
    검색대상 질의어의 경로서명값을 오른쪽부터 한 비트씩 취하여
    병렬 매치 인덱싱 구조로 전달;
  }
    병렬매치 처리 시작을 위한 match signal을 병렬 매치 인덱싱 구조로 전달;

    정보노드의 매치 표시기에 저장된 값이 true인 모든 경로서명들의 경로를 선택;

}
```

그림 7 병렬 매치 인덱싱 구조를 이용한 병렬 매치 알고리즘

```
input : 역순으로 검색할 질의어의 경로서명 비트값('0' or '1'),
        비트값의 위치, 루트 PE 노드의 주소

경로서명비트값_병렬인덱싱구조_전달함수(입력비트, 레벨, 베이스 PE 노드)
 {
   임시변수에 현재 베이스PE노드가 기억하는 비트값을 저장;
   새로 입력된 비트값을 베이스 PE 노드에 저장;

   if (레벨이 1과 같다면)
      return;
  else
   {
     if (루트 PE 노드의 왼쪽 자식노드가 존재한다면)
      함수 자신을 순환 호출하고 파라메터값으로
      임시변수의 값 ,레벨-1,베이스 PE노드의
      왼쪽 자식노드의 주소를 전달;

     if (루트 PE 노드의 오른쪽 자식노드가 존재한다면)
      함수 자신을 순환 호출하고 파라메터값으로
      임시변수의 값,레벨-1,베이스 PE노드의
      오른쪽 자식노드의 주소를 전달;
   }
```

그림 8 병렬 매치 인덱싱 구조로 경로서명 비트 값을 전달하는 알고리즘

구조로 전달하기 위한 알고리즘들을 결합하여 구성하며 그림 7과 같다. 여기서 모든 연산 동작은 병렬로 수행된다.

입력된 경로-지향 질의어의 경로서명 값을 비트단위로 병렬 매치 인덱싱 구조로 전달하는 과정을 수행하는 알고리즘은 그림 8과 같다. 그림 9는 병렬 매치 작업수행을 위해 매치 또는 노-매치 신호를 병렬 매치 인덱싱 구조로 전달하는 알고리즘이다.

정리 1. 병렬 매치 인덱싱 구조를 이용한 병렬 매치 방법은 정확하다. 즉, 매치된 결과로서 이진 트라이에서 각 단말 노드는 정확한 매치 신호와 경로서명 파일내에서 유효한 엘리먼트들의 경로서명 값을 출력한다.

증명. 이진 트라이는 경로서명 파일을 위한 표현이다. 이 트라이에서 루트 노드에서부터 단말 노드까지 이르는 비트 스트링은 경로서명과 동일하다. 본 논문의 병렬 매치 알고리즘은 경로서명 매치 작업 수행을 위해 2가지 단계로 구성된다. 첫 번째 단계 수행 후 트라이에서 레벨 1의 루트 노드는 매치하려는 경로서명의 가장 왼쪽 비트 값을 가지며 레벨 i의 노드들은 그들의 i번째 비트 값을 가진다.

예를 들어 경로-지향 질의어로 /library/report[writer$contains&'Kim Yoo-Sin']가 입력된다면 위의 병렬 매치 알고리즘 수행 후 그 결과로 얻어진 경로서

```
input : match  또는 no-match 신호(true or false),
        경로서명의 길이, 루트 PE 노드의 주소
output : match  또는 no-match 신호(true or false)

매치신호처리함수(match  또는 no-match 신호, 레벨, 베이스 PE 노드)
{
  if (레벨값이 1이라면)
   {
    if (베이스 PE노드의 왼쪽 자식노드가 존재한다면) {
       마지막 단의 PE 노드들이 전달한 signal(true or false)을
       왼쪽 자식노드로 연결된정보 노드의 매치 표시기에 저장;
     }

    if (베이스 PE노드의 오른쪽 자식노드가 존재한다면) {
       마지막 단의 PE 노드들이 전달한 signal(true or false)을
       오른쪽 자식노드로 연결된 정보 노드의 매치 표시기에 저장;
       }

   return;
}

 if (신호 true가 입력되었다면){
      if (베이스 PE노드에 저장된 값이 0이라면) {
        함수 자신을 순환 호출하고 파라메터값으로
          true, 레벨-1, 베이스 PE노드의 왼쪽 자식 노드의 주소를 전달;
        함수 자신을 순환 호출하고 파라메터값으로
          true, 레벨-1, 베이스 PE노드의 오른쪽 자식 노드의 주소를 전달;
        }
     else {

          함수 자신을 순환 호출하고 파라메터값으로
            false, 레벨-1, 베이스 PE노드의 왼쪽 자식 노드의 주소를 전달;
          함수 자신을 순환 호출하고 파라메터값으로
            true, 레벨-1, 베이스 PE노드의 오른쪽 자식 노드의 주소를 전달;
        }
      }
 else {
       함수 자신을 순환 호출하고 파라메터값으로
         false, 레벨-1, 베이스 PE노드의 왼쪽 자식 노드의 주소를 전달;
       함수 자신을 순환 호출하고 파라메터값으로
         false, 레벨-1, 베이스 PE노드의 오른쪽 자식 노드의 주소를 전달;
  }
}
```

그림 9 병렬 매치 인덱싱 구조로 매치 또는 노-매치신호 전달 알고리즘

명 값들은 관계형 데이타베이스에 대한 질의를 수행할 때 사용된다. 따라서 질의 처리기에 의해 생성된 질의어는 다음과 같다.

```
SELECT * FROM ElementTable a, TextTable b
WHERE a.엘리먼트명='writer' and a.경로서명 IN
(MatchedKeys)
and a.문서ID=b.문서ID and a.엘리먼트ID=b.부모ID
and b.텍스트값='Kim Yoo-Sin';
```

여기서 MatchedKeys는 병렬 매치 알고리즘 수행 결과로 얻어진 경로서명들의 집합을 의미하며, 병렬 매치 알고리즘 수행으로 얻어진 결과들에서 허위드롭된 결과값이 포함되어 있다면 이를 제거하기 위한 추가 작업이 요구된다.

3.4 실험결과 및 성능 비교 분석

본 논문의 시스템 성능 실험은 다음과 같은 시스템 환경에서 이루어졌다. 하드웨어는 윈도우즈 2000 운영체제를 탑재한 펜티엄 4 시스템으로 메모리 1GB, CPU속도 1.7GHz, HDD 80GB 그리고 데이타베이스 시스템으로는 Oracle 9i를 사용하여 수행하였다. 또한, 실험을 위한 시스템의 구현은 C++언어와 JAVA를 사용하여 소

프트웨어적으로 구현하였다.

시스템의 성능 실험을 위해 입력되는 경로-지향 질의어의 경로는 모두 절대위치경로(absolute location path) 형태로 입력하였으며, 경로-지향 질의 유형은 단순경로들만 입력하는 형태와 술어를 함께 포함하는 형태 2가지로 실험하였다. 실험에 사용한 데이타들은 표 1과 같다.

실험결과, 병렬 매치 알고리즘을 사용한 본 논문의 방법에 대한 시간 복잡도는 O(h)이다. 왜냐하면 h-클럭 이후 입력 스트링의 첫 번째 비트패턴 값(가장 오른쪽 비트 패턴)이 리프 노드에 도착하고 마지막 매치 신호는 h-클럭 이후 루트 노드에 입력되기 때문이며, 2*h 클럭 이후에 매치 또는 노-매치 신호가 리프 노드에 도착하기 때문이다. 본 논문에서 사용한 방법은 매치를 위해 2*h 클럭을 요구한다. 따라서 시간 복잡도는 O(h)이다. 경로서명 파일의 수(N)의 상한 경계는 2^h이다. 즉, 이 방법은 $\log_2 N$과 유사한 크기 순서를 가진다. 결과적으로, 병렬 매치 알고리즘은 경로서명 파일의 개수(N)에는 영향을 받지 않으며 단지 경로서명의 길이 h에 의해만 영향을 받는다. 다음의 그림 10은 패트리샤 트리를 이용하는 기존의 Chen[8]의 알고리즘과 본 논문의 병렬 매치 인덱싱 구조를 사용하는 병렬 매치 알고리즘의 수행결과에 대한 키의 최대 비교 횟수를 비교분석한

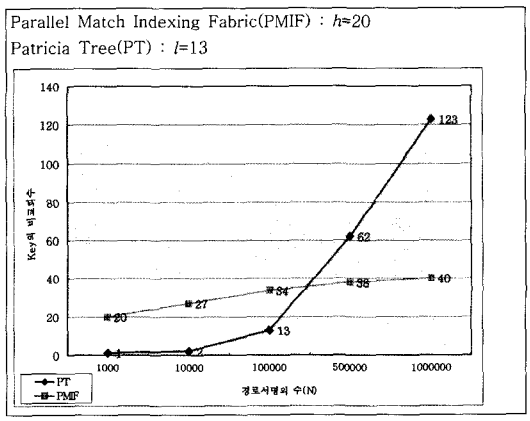


그림 10 패트리샤 트리(PT)와 병렬 매치 인덱싱 구조(PMIF)의 검색횟수 비교 분석

것이다.

그림 10에서 보는 바와 같이 경로서명 파일내의 경로서명의 개수(N)가 적을 때에는 기존 방법이 성능면에서 우수함을 나타내지만 파일내의 인덱스의 수가 점차 증가함에 따라 제안된 병렬 매치 알고리즘이 더 나은 성능을 보임을 알 수 있다. 또한, 기존의 방법은 1로 설정된 비트의 수가 많고 적음 또는 경로서명 파일내의 경로서명의 엔트리 수에 따라서 수행 성능에 많은 차이를 보인다. 그러나 제안된 방법은 경로서명 파일내의 경로서명의 수가 많고 적음이나 경로서명 값에서 1로 설정된 비트의 수가 많고 적음에는 영향을 받지 않으며 오직 경로서명의 길이(h)에만 영향을 받는다. 따라서 중간 규모이하의 데이타베이스에 대한 검색에는 기존의 방법이 더 나은 성능을 나타내고 있으며 정적인 대규모의 XML문서 데이타베이스에서는 제안된 방법이 더 우수한 성능을 나타낸다. 그러나 본 논문에서 제안한 병렬 매치 인덱싱 구조에서 PE들을 구현함에 있어서 실제 하드웨어로 구현한다면 성능개선과 비용문제 측면에서 tradeoff가 일어 날 수 있다. 또한, 병렬 시스템을 구현함에 있어서 필연적으로 PE들 사이에 통신오버헤드 문제가 발생할 것이나 본 논문에서는 통신오버헤드 문제는 고려하지 않았다. 또 다른 문제는 각 엘리먼트에 대한 경로서명을 생성할 때 단축-경로 구성상의 단순함 때문에 서로 다른 경로에 대해 동일한 경로서명 값이 생성될 수 있으며 이것으로 인해 발생하는 허위드롭의 수를 줄이기 위한 효율적인 방안에 대한 연구가 필요하다.

4. 결론 및 향후과제

XML문서들에 대한 경로-지향 질의어가 이슈될 때 데이타베이스 시스템은 데이타베이스 내에 저장된 위치를 찾기 위해 질의어가 포함하고 있는 경로들에 대한 질의 평가를 수행 하여야만 한다. 본 논문에서 우리는 관계형 데이타베이스에 저장된 XML문서에 대한 경로-지향 질의어 처리 평가 속도를 개선하기 위해 병렬 매치 인덱싱 구조와 병렬 매치 알고리즘에 대한 설계 및 구현과 그에 대한 성능 실험을 수행하였다. 이진 트라이는 이진 비트 패턴내에 연속적인 비트 값들을 이용하여

표 1 실험에 사용한 데이타들

경로서명의 수	XML문서의 최대 깊이	해시코드 길이(bit)	평균 엘리먼트 수/문서	XML문서 파일의 수	경로-지향 질의어 유형
1,000	4	20	53	19	/library/creator /book/title
10,000	4	20	210	48	
100,000	4	20	300	334	/library/report [writer$contains&'Kim Yoo-Sin']
500,000	4	20	400	1,250	
1,000,000	4	20	500	2,000	

루트에서부터 아래쪽으로 향하는 트라이의 경로를 결정한다. 본 논문에 제안한 접근법에서는 XML문서에 포함된 각 엘리먼트들의 경로 정보를 저장하기 위해 단축-경로를 생성하고, 이들을 경로서명 파일에 저장한 후 이진 트라이 구조를 형성하는데 활용하였으며, 경로-지향 질의어에 대한 평가속도를 개선하기 위해 이진 트라이 구조를 병렬 매치 인덱싱 구조로 변환하였다. 또한, 제안된 구조를 이용하여 병렬 매치 처리를 수행하기 위한 병렬 매치 알고리즘을 설계하였다. 본 논문에서 제안한 방법을 이용한 성능 실험 분석 결과에 따르면 제안된 알고리즘은 XML문서 파일에 대한 경로서명 엔트리 수에 대해 $\log_2 N$과 유사한 크기 순서를 가진다. 따라서 경로서명 파일의 개수에는 영향을 받지 않기 때문에 결과적으로 알고리즘의 시간 복잡도는 O(h)이다.

향후 연구과제로 XML문서에 포함된 각 엘리먼트들의 경로서명 값을 생성할 때 그 방법상의 단순성으로 인하여 허위적중의 비율이 증가하는 요인이 되었다. 따라서 이에 대한 문제를 해결하기 위한 효율적인 방안에 대한 연구가 필요하며, 본 논문에서 제안한 병렬 매치 인덱싱 구조의 PE들을 실제 하드웨어로 구현하고 이에 대한 실험 및 결과 분석에 따른 정확한 성능 평가에 대한 연구가 필요하다.

참 고 문 헌

[1] T. Bray and E.maler and J. Paoli and C.M. SPErberg McQueen, "Extensible Markup Language (XML) 1.0," http://www.w3.org/TR/2000/REC-xml-20001006. Oct. 2000.

[2] A. Deutch and M. Fernandez and D. Forescu and A. Levy and D. Suciu, "XML-QL : A Query Language for XML," http://www.w3.org/TR/NOTE-xml-ql, Aug. 1998.

[3] S. Boag and D. Chanberlin and M. Fernandez and D. Florescu and D. Florescu and J. Robie and J. Simeon, "XQuery 1.0 : An XML Query Language," http://www.w3.org/TR/2002/WD-xquery-20020816, Aug. 2002.

[4] D. Eastlake and J. Reagle and D. Solo and W3C Recommendation, "XML-Signature Syntax and Processing," http://www.w3.org/TR/2002/REC-xmldsig-ore-20020212, Feb. 2002.

[5] C. Faloutsos and ACM Computing Surveys, "Access Methods for Text," Vol.17, No.1, pp.48-74, March 1985.

[6] Carlo Zaniolo et al., "Advanced Database Systems," Morgan Kaufman Publishers, 1997.

[7] J. D. Ullman, "Computational Aspects of VLSI," Computer Science Press, Maryland, 1984.

[8] Yangjun Chen and Gerald Huck, "Path Signature : A Way to Speed up Evaluation of Path-oriented Queries in Document Database," Proceedings of WISE 2000 Conference, pp. 240-244, Jun. 2000.

[9] Brian F. Cooper et al., "A Fast Index and Querying XML Data for Regular Path Expression," Proceeding of the 27th VLDB Conference, 2001.

[10] Q. Li and B. Moon, "Indexing and Querying XML Data for Regular Path Expression," Proceeding of the 27th VLDB Conference, 2001.

[11] World Wide Web Consortium, "Document Object Model (DOM)," http://www.w3.org/DOM/, 2002.

[12] Harvey M. Deitel et al., "XML How TO PROGRAM," Prentice Hall, pp. 134-155, pp. 192-216, pp. 298-310, 2000.

[13] William B. Frakes and Ricardo Baeza-Yates, "Information Retrieval : Data Structures and Algorithms," Prentice Hall PTR; Facsimile edition, pp. 44-80, 1992.

[14] S. Stanfill and B. Kahle, "Parallel free-text search on the connection machine system," Communication of ACM, Vol.29, No.12, pp. 1229-1239, 1986.

[15] 박희숙, 조우현, "단축-경로와 확장성 해싱 기법을 이용한 경로-지향 질의의 평가속도 개선 방법," 정보처리학회논문지, 제11-D권, 제7호, pp. 1409-1416, 2004.

박 희 숙

1998년 경남대학교 교육대학원 전자계산교육전공(교육학석사). 2004년 8월 부경대학교 대학원 컴퓨터공학과(박사수료) 관심분야는 객체지향 데이타베이스, 인덱싱 성능개선 문제, 멀티미디어 데이타베이스

조 우 현

1985년 경북대학교 전산공학전공 졸업(학사). 1988년 경북대학교 대학원 전산공학전공(공학석사). 1998년 경북대학교 대학원 전산공학전공(공학박사). 1989년~현재 부경대학교 공과대학 전자컴퓨터정보통신공학부 교수. 1988년~1989년 한국전자통신연구소 지능망연구실 연구원. 2002년~2003년 텍사스주립대-산마르코스 방문연구. 관심분야는 지식의 표현과 병렬처리, 멀티미디어 데이타베이스 관리시스템, 객체지향 데이타베이스