XML 문서를 위한 족보 기반 인덱싱 기법
(Genealogy-based Indexing Technique for XML Documents)

이 월 영 † 용 환 승 ††
(Wol-Young Lee) (Hwan-Seung Yong)

요 약 오늘날 인터넷 상의 많은 데이타들은 XML의 여러 장점들로 인하여 XML을 이용하여 표현되고 있다. 이렇게 XML 데이타가 늘어가는 것에 비례하여 XML 문서 상에서 유용한 정보를 검색하기 위하여 다양한 질의를 빠르고 효율적으로 지원할 수 있는 질의 처리 기법이 요구되고 있다. 그러나 현재까지는 XML 데이타를 위한 질의 최적화 연구는 정규 경로 표현을 다루는 방법론에 국한되어 있다. 본 논문은 새로운 족보 기반 인덱싱 기법을 개발하여 정규 경로 표현뿐 아니라, 단순 경로 표현과 다른 엘리먼트를 참조하고 있는 경로 표현과 같은 다양한 질의 처리를 해결하였다. 또한 이 인덱싱 기법을 객체-관계형 모델에 적용하여 여러 종류의 문서와 다양한 질의 종류에 대해 성능을 평가하였고, 다른 저장 기법과 비교하여 성능의 우수성을 입증하였다.

키워드 : 인덱싱 기법, XML, 질의 처리

Abstract Theses days, a number of data over the Internet are represented using XML because of a virtue of XML. In proportion to the increase of XML data, query processing techniques are required that support quickly and efficiently the diverse queries to search the useful information on XML documents. But, up to now, the researches handling queries for XML data are methodologies focusing on how to process regular path expressions. Therefore, we have developed a new genealogy-based indexing technique to solve various queries such as not only regular path expression but also simple path expression, path expression referencing other elements, and so on. Also, we have applied this technique on object-relational model and evaluated the performance for many documents and various query types. The result shows improved performance in comparison with the other storage techniques.

Key words : Indexing technique, XML, Query processing

1. 서 론

XML은 문서 구성 요소들 사이에 계층적인 구조를 가지고는 있으나 그 형태가 일정하게 고정된 스키마를 따를 필요가 없기 때문에 비정규적이고 불완전한 스키마를 갖도록 하는 마크업 언어이다[1]. XML의 이러한 유연한 성질과 환경에 독립적인 특성 때문에 웹 상의 다양한 종류의 수 많은 데이타를 XML로 표현하거나, 서로 다른 기업이나 환경의 데이타들을 XML을 이용하여 교환하고자 하는 표준으로 자리 잡아가고 있다.

이렇게 급속도로 늘어나고 있는 XML 문서를 대상으로 사용자들이 원하는 것을 자유자재로 표현할 수 있을 뿐 아니라 빠르게 검색하여, XML 문서 자체가 가지고 있는 계층적인 정보나 순서 정보와 같은 고유 속성을 잃지 않은 상태로 결과를 반환 받기 위해서는 이를 지원할 수 있는 질의 처리에 대한 연구가 무엇보다 필요하다.

현재까지는 반구조적(semistructured) 데이타를 대상으로 정규 경로 표현(regular path expression)에 의한 질의 처리 기법이 중심이 되는 것으로서, 주로 문서의 스키마를 최적으로 추출하여 계층적인 구조 정보를 찾기 위해 경로를 방문하는 횟수를 최소화하는 방법에 초점이 맞추어져 있다. 그러나 XML은 다른 데이타 모델과는 다르게 엘리먼트 사이의 계층적인 정보뿐 아니라 순서 정보나 애트리뷰트에 의한 참조 정보와 같은 그 문서만이 갖는 고유한 속성이나 의미를 지니고 있기 때문에 이러한 것들을 섬세하게 보존할 수 있는 질의 처리 방법이 요구되고 있다.

† 비 회 원 : 이화여자대학교 컴퓨터학과
wylee@ewha.ac.kr
†† 종신회원 : 이화여자대학교 컴퓨터학과 교수
hsyong@ewha.ac.kr
논문접수 : 2003년 5월 7일
심사완료 : 2003년 10월 15일

본 논문은 새로운 인덱싱 기법인 족보 기반 인덱스를 개발하여 이 기법을 통해 대부분의 XML 질의 언어에서 요구하는 여러 가지 질의 종류를 처리하면서도 XML이 가지고 있는 고유한 속성을 그대로 보존하여 결과를 반환할 수 있도록 하였다. XML 문서나 반구조적 데이타를 대상으로 하는 기존의 인덱싱 기법들이 대부분 정규 경로 표현에 의한 질의 처리를 할 때 문서 전체의 스키마를 어떻게 최적화하여 인덱싱할 것인가라는 한 가지 사항에만 초점을 맞춘 것에 비해, 본 논문에서는 족보 기반 인덱스만으로 XML 문서를 대상으로 하는 여러 가지 질의 처리 문제를 통합적으로 해결하고 있다. 즉, 족보 기반 인덱스는 XML 문서를 사람들의 족보에 비교하여 각 노드의 혈통과 형제 사이에서 순서 등을 알 수 있는 고유의 숫자를 부여함으로써 일반적으로 말하는 정규 경로 표현에 의한 질의는 두 노드간의 조상/후손 관계를 알아내는 것으로 해결하였으며, 간단한 경로 표현(simple path expression)에 의한 질의에 대해서는 두 노드 간의 부모/자식 관계를 밝힘으로써 해결하였다. 또한 XML 문서가 가지고 있는 고유의 속성인 엘리먼트 사이의 순서를 위하여는 족보 기반 인덱스가 가지고 있는 형제를 뜻하는 숫자를 이용하여 같은 엘리먼트 이름을 가지고 있는 것이라 하더라도 정확한 위치에 의한 질의가 가능하도록 하였다.

이와 같이 족보 기반 인덱스를 통해 XML 문서에 대한 고유 속성을 보존하여 XML의 전체적인 질의 종류들을 해결할 수 있을 뿐만 아니라, 성능 또한 기존의 저장 기법에 의한 질의 처리와 비교하여 우수한 성능을 낸다는 평가 결과를 얻었다.

본 논문의 전체적인 구성은 1장의 서론에 이어서 2장에서는 관련 연구, 3장에서는 예제 XML 문서를 모델링하기 위하여 XML 인스턴스 그래프를 정의한다. 4장에서는 XML 문서를 위한 인덱싱 기법에 대해 설명하고 5장에서는 성능 평가를, 6장에서는 결론을 맺는다.

2. 관련 연구

XML 데이타에 대해 질의 처리를 효과적으로 수행하기 위한 인덱싱 기법으로는 문서를 관계형 데이타베이스에 저장하고 관계형 데이타베이스 자체에서 지원하는 인덱싱[2-9]을 사용하는 경우와, 일반적인 반구조적(semistructured) 데이타를 위한 순수한 인덱싱 기법[10-18]으로 나눌 수 있다.

관계형 데이타베이스를 이용하여 XML 문서를 저장하고 인덱싱하는 기법은 관계형 데이타베이스의 안정성 있는 성능을 보장할 수 있다는 장점을 가지고 있다. 그러나 이런 연구들 중 [4]는 XML이 갖고 있는 계층적인 정보를 보존하여 질의하기 위하여 부가적인 데이타들을 너무 많이 저장하기 때문에 비효율적이거나, 인덱싱을 할 때 포함되는 정보가 제한적이어서 SQL-like 질의문이 아주 복잡해지거나[3], 아니면 XML 질의문에 나타나는 경로 사이의 조인(join)을 기반으로 하는 인덱싱 기법이기 때문에 너무 많은 질의 처리 방법들이 생성되는[2] 문제를 가지고 있다. 상업적인 제품으로서 Oracle XML DB[7]이나 SQL 2000 서버[8]는 XML 문서를 사용자가 수동으로 관계형 데이타베이스에 저장하고 이를 SQL의 확장된 구문을 사용하여 질의하고 있기 때문에 효율적인 인덱싱이 어렵고 또한 XML의 유연한 특징을 잘 살릴 수 있는 질의 표현에는 한계가 있다. 또한 DB2 XML Extender[9]는 정보 검색 엔진에서와 같은 질의를 위해 contains와 같은 연산자를 사용하고 있으나 질의의 결과를 반환하기 전에 정보 검색 엔진의 결과와 관계형 데이타베이스 엔진의 결과를 결합하는 형태로 이루어지기 때문에 두 엔진에서 각기 다른 인덱싱을 요구하게 된다.

또한 반구조적인 데이타에 대한 질의 최적화를 위해 연구된 인덱싱 기법들은 대부분이 정규 경로 표현(regular path expression)을 최적화시키기 위한 기법[2,11,12,15-17]에 초점이 맞추어져 있다. 이 정규 경로 표현에 의한 질의는 XML 질의 언어에서 필요로 하는 질의 종류 측면에서 보았을 때 조상/후손의 관계를 이용하는 질의 처리에만 적합한 것으로서 XML을 위한 여러 가지 질의 처리 방법은 만족시키지 못하고 있다. 또한 XML 데이타가 텍스트를 대상으로 한다는 관점에서 정보 검색 시스템에서 지원하는 것과 같은 전체 텍스트 인덱싱을 지원하고 있는 시스템들이 있으나[10,12,18] 이러한 것들 역시 XML에서 고유하게 필요로 하는 여러 가지 다양한 질의 종류를 만족시키지는 못하고 있다.

현재 대부분의 XML 질의 언어들은 XPath[19] 기반의 경로 표현을 허용하도록 하고 있는데, 여기서의 경로 표현은 XML 문서 상에서 각 요소간 부모/자식, 조상/후손, 형제 관계를 사용하고 있다. 이 외에도 참조에 의한 질의, XML 문서 상에서 엘리먼트의 정확한 위치나 순서에 의한 질의 등을 표현할 수 있도록 되어 있다. 그러나 현재까지 이러한 질의를 효율적으로 처리하기 위한 연구는 단지 정규 경로 표현 즉, 조상/후손의 관계를 최적으로 처리할 수 있는 기법만이 나와 있을 뿐이다.

본 논문에서는 이와 같은 문제점들을 극복하기 위해서 새로운 족보 기반 인덱싱 기법을 개발하였다. 이 기법은 XML 문서를 하나의 가계(family line)에 비교하여 XML 문서 각각의 구성 요소가 가지고 있는 계층적 구조 상에서의 자신의 위치와 수평적인 관계에서의 자신의 순서 정보를 정확히 인식할 수 있도록 하였다. 따

라서 XML 질의 언어에서 요구하고 있는 부모/자식, 조상/후손, 형제 관계에 의한 질의, 참조에 의한 질의, 엘리먼트 사이의 위치나 순서 정보를 요구하는 질의 등, 여러 가지 다양한 질의를 전체적으로 지원할 수 있도록 하였고, 성능 또한 다른 질의 처리 기법에 비하여 우수하다는 것을 보여주고 있다.

3. XML 문서 그래프와 예제

우선 XML 문서를 인덱싱하기 위해 필요한 문서의 컴포넌트와 이들 컴포넌트들의 상호 관계를 표현하기 위하여 XML 문서를 모델링하는 그래프를 정의한다.

3.1 XML 문서 그래프

XML 데이타 모델에 대한 기존의 모델링 방식은 반구조적 데이타를 모델링하기 위해 출현된 OEM모델의 변형으로서 간선에 레이블링이 된 그래프(edge labeled graph)로 표현하고 엘리먼트와 애트리뷰트를 같이 취급하고 있다. 그러나 본 논문에서는 XML 문서의 특징을 살려 질의하기 위하여 애트리뷰트와 엘리먼트를 구분하고 참조의 특징과 노드들에 대해서도 보다 섬세하게 구분하여 XML 문서의 각 컴포넌트와 상호간의 관계를 다음과 같이 여섯 가지의 요소를 가진 방향 그래프(directed graph)로 모델링한다.

정의 3.1 XML 문서를 모델링하는 그래프 G = (L, V, E, A, R, O)는 다음에서 정의하는 6가지 속성을 가진 방향 그래프(directed graph)이다.

- **L**은 정점으로서 엘리먼트 이름과 애트리뷰트 이름들을 필드로 하고 있는 레코드로서 사각형으로 나타내고 이 레코드의 첫 번째 필드는 엘리먼트 이름이고 나머지 필드는 임의 개수의 애트리뷰트 이름을 나타낸다.
- **V**는 정점으로서 엘리먼트의 내용이 값인 것과 애트리뷰트의 값을 나타내는 것으로서 원으로 나타낸다.
- **E, A, R**은 각각 방향 그래프에서 간선을 이루는 요소로서 세 가지 의미를 가지고 있는데 첫째는 하나의 엘리먼트가 또 다른 엘리먼트를 포함하고 있는 간선(**E**), 두 번째는 엘리먼트나 애트리뷰트가 값을 가지기 위한 간선(**A**), 세 번째는 IDREF, IDREFS, XLink, URI등에 의해 애트리뷰트가 다른 엘리먼트에 속한 애트리뷰트를 참조하기 위한 간선(**R**)으로 정의한다. 여기서 특별히 참조를 위한 간선 **R**은 점선으로 나타낸다.
- **O**는 XML의 고유 속성 중 하나인 엘리먼트들 사이의 순서를 위해 정의한다. 이 때 애트리뷰트 순서는 XML에서처럼 무시하고 엘리먼트 사이의 순서는 어떤 엘리먼트가 서브엘리먼트로 가지고 있는 엘리먼트들의 순서에 따른다. 또한 IDREFS 타입으로 정의되는 다중 값 애트리뷰트에서 값의 순서는 구현에 달려 있다.

예제 3.1 그림 1은 은행에서 보유하고 있는 계좌와 고객에 대한 정보를 XML 문서로 구축한 예이다. 이것은 애트리뷰트가 갖고 있는 여러 가지 속성 중 데이타를 모델링할 때 꼭 고려하여야 할 것들로서, 참조되는 ID 타입과 참조를 하고 있는 IDREF, 특별히 다중 참조의 속성을 갖는 IDREFS와 같은 링크 타입에 대해 모델링하는 방법을 보여 주기 위한 예로서 선정하였다.

```
<!DOCTYPE banks[
        <!ELEMENT banks (bank+ )>
        <!ELEMENT bank (account+, customer+)>
        <!ATTLIST bank
                id                  ID          #REQUIRED>
        <!ELEMENT account (branch_name, balance)>
        <!ATTLIST account
                account_number      ID          # REQUIRED
                owners              IDREFS      # REQUIRED>
        <!ELEMENT branch_ name (#PCDATA)>
        <!ELEMENT balance (#PCDATA)>
        <!ELEMENT customer (customer_ name, customer_addr)>
        <!ATTLIST customer
                customer_id         ID          # REQUIRED
                accounts            IDREFS      # REQUIRED>
        <!ELEMENT customer_ name (#PCDATA)>
        <!ELEMENT customer_addr (customer_street, customer_city)>
        <!ELEMENT customer_street (#PCDATA)>
        <!ELEMENT customer_city (#PCDATA)>
]>
<banks>
        <bank id="1">
        ...
        </bank>
        <bank id="2">
                <account account_number="A-401" owners="C100 C102">
                        <branch_ name > Downtown </branch_ name >
                        <balance>500 </balance>
                </account>
                <customer customer_id="C100" accounts="A-401">
                        <customer_ name >Elton John</customer_ name >
                        <customer_addr>
                                <customer_street>Monroe</customer_street>
                                <customer_city>Madison</customer_city>
                        </customer_addr>
                </customer>
                <customer customer_id="C102" accounts="A-401 A-402">
                        <customer_ name >John Lennon</customer_ name >
                        <customer_addr>
                                <customer_street>Austin</customer_street>
                                <customer_city>Texas</customer_city>
                        </customer_addr>
                </customer>
        </bank>
        <bank id="3">
...
</banks>
```

그림 1 예제 문서

이 XML 문서를 정의 3.1에 따라 모델링한다면 그림 2와 같이 된다. 본 논문에서 정의하는 데이타 모델은 루트를 가지고 있는 방향 그래프로서 기존 모델이 애트리뷰트를 엘리먼트와 같이 취급하여 애트리뷰트를 포함하고 있는 엘리먼트의 자식으로서 다루었던 것과는 다르게 애트리뷰트를 포함하고 있는 엘리먼트와 같은 레벨에 위치시켜 하나의 레코드 내의 필드를 이루는 데이타로 나타내었다. 따라서 항상 사각형으로 표시되는 엘리먼트를 위한 정점은 가장 처음 필드에 엘리먼트를 나타내고 그 다음부터는 임의 개수의 애트리뷰트를 표현하게 된다. 이렇게 표현함으로써 XML 데이타를 모델링한 그래

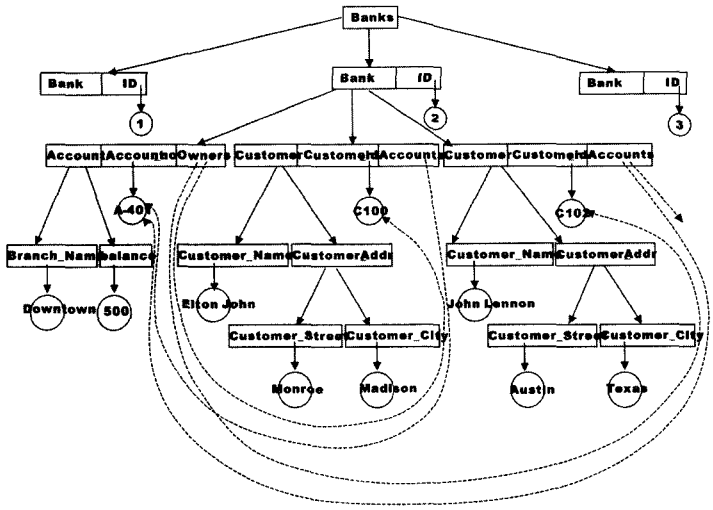

그림 2 XML 문서 그래프

프만 보고서도 엘리먼트와 애트리뷰트를 정확히 구분할 수 있도록 하였고, 애트리뷰트를 담고 있는 엘리먼트와 이 애트리뷰트의 부모와 자식을 구분할 수 있도록 하였다. 또한 IDREF 와 IDREFS 타입과 같이 링크에 의한 참조를 나타내는 간선은 점선으로 나타낸다.

4. 인덱싱과 질의 처리

이 장에서는 족보 기반 인덱싱 기법을 개발하고 이 인덱싱 기법이 어떻게 XML 문서의 특징을 살려 다양한 질의를 지원할 수 있는지 현재 W3C에서 대규모적으로 표준화를 진행하고 있는 XQuery[21]의 경로 표현을 이용하여 여기서 제공하고 있는 여러 가지 질의를 해결하는 것을 보여준다.

4.1 족보 기반 인덱싱 기법

XML 데이타는 그래프로 모델링되면서 보통의 그래프와는 다르게 하나의 루트로부터 시작되며 방향성을 갖는 특징을 가지고 있어 애트리뷰트의 링크 속성에 의해 다른 엘리먼트를 참조하는 의미를 단순히 값으로 표현한다면 트리로 모델링될 수 있다. 즉, 애트리뷰트에서 참조하는 속성을 값의 의미로 해석하고 엘리먼트들 사이의 관계를 보았을 때, 계층적으로는 조상/후손의 관계를 가지며 수평적으로는 형제관계에서 순서가 존재한다. 따라서 XML 데이타를 트리로 모델링하여 사람의 족보를 이용한 인덱싱을 한다면 이러한 관계를 나타내는 경로 표현 질의에 응답하는데 크게 유용하다.

이러한 인덱싱 기법을 위해 먼저 FMFT(First order monadic theory of finite binary trees)이라는 모델[20]을 도입하여 XML 문서를 인덱싱하기에 적당하도록 변형하여 적용하는데, 본 논문는 이 모델에서 혈통관계를 나타내기 위해 상위 노드의 고유 숫자를 하위 노드가 이어 받는 개념과 형제 사이의 순서를 나타내는 숫자를 분리하고 이진으로 표현하였던 고유 숫자를 관리하기 쉽게 십진 숫자로 표현하도록 하였다. 또한 이렇게 부여되는 고유 패턴들 사이에 지닌 관계들을 연산자로 표시하여 새로운 트리 모델 FMFT+를 다음과 같이 정의한다.

정의 4.1 계층적인 속성을 갖는 하나의 트리 모델 t = ({0,⋯,9}*, ⊐, $\sqsupset_d$, >, (P_{a1}, P_{b1}) ⋯, (P_{an}, P_{bn}))를 정의한다. 여기서 {0,...,9}*는 트리의 노드에 부여되는 고유의 패턴인 (P_a, P_b)에서 P_a, P_b를 형성하는 숫자가 0부터 9까지라는 것을 뜻한다. 트리의 노드에 부여되는 고유의 패턴인 (P_a, P_b)에서 P_a, P_b는 {0,⋯,9}에 속하는 수들의 조합으로 이루어지고, P_a는 트리에서의 상하의 계층적인 관계 즉, 혈통관계를 나타내는 숫자이고, P_b는 수평적 관계 즉, 형제에서의 순서를 뜻하는 숫자이다. 한 부모 α의 패턴 값인 두 수 P_a, P_b를 연결(concatenation)한 P_a + P_b가 자식 노드 β의 P_a가 되고 자식 노드의 P_b는 β의 형제 사이의 순서 값을 부여받는다. 이렇게 부여된 P_a와 P_b 두 수의 연결은 트리내에서 그 노드를 결정지을 수 있는 고유의 패턴을 갖게 된다. n은 트리에 있는 노드의 개수를 뜻한다. 또한 이렇게 각 노드에 부여되는 고유 패턴들이 가지고 있는 관계를 다음과 같은 연산자로 정의한다.

- ⊐는 이 트리에 부여된 패턴 값이 갖는 속성으로서 어떤 임의의 패턴 v를 갖는 노드와 또 다른 임의의

패턴 w를 갖는 노드사이의 관계가 상호 조상/후손 관계, 즉 간접 포함 관계를 가지고 있음을 뜻한다.

- $\supset_d$는 이 트리에 부여된 패턴 값이 갖는 두 번째 속성으로서 어떤 임의의 패턴 v를 가진 노드와 또 다른 임의의 패턴 w를 갖는 노드사이의 관계가 상호 부모/자식 관계, 즉 직접 포함 관계를 가지고 있음을 뜻한다.
- >는 이 트리에 부여된 패턴값이 갖는 세 번째 속성으로서 어떤 임의의 패턴 v를 갖는 노드와 또 다른 임의의 패턴 w를 갖는 노드사이의 관계가 상호 형제의 관계, 즉 한 노드의 패턴이 다른 노드의 패턴에 대해 순서상으로 전/후 관계에 있음을 뜻한다.

예제 4.1 그림 3은 FMFT 모델을, 그림 4는 FMFT+ 모델을 보여주고 있다. 그림에서 보는 것처럼 두 모델은 상위 노드에 부여된 고유 값을 하위 노드가 이어 받는 개념은 같으나, FMFT+모델에서는 이진수 대신에 십진수를 사용하여 표현을 간소화하였고, 두 노드 사이의 계층적인 혈통 관계와 형제 사이에서의 순서를 구분할 수 있도록 노드에 부여하는 고유의 숫자를 특수한 패턴으로 변형하여 부여하였다. 즉, FMFT+모델에서는 한 노드에 부여되는 고유 패턴은 혈통을 나타내는 첫 숫자와 형제사이에서의 순서를 나타내는 두 번째 숫자로 이루어지는데 다시 이 두 숫자의 결합이 이 노드의 자식에 부여되는 고유 패턴의 첫번째 숫자가 된다. 또한 고유 패턴들 사이에 존재하는 관계를 연산자 $\supset$, $\supset_d$, >로 표현하여 정의하였다.

FMFT+모델에 부여되는 고유 패턴은 정의 4.1에서 정의하는 연산자의 성질을 만족하고 이러한 성질은 XML 문서가 가지는 고유의 속성을 보존하여 질의할 수 있는 기초가 된다.

예제 4.2 트리내의 한 노드의 패턴을 (P_{a1}, P_{b1})로 부여하고 또 한 노드를 (P_{a2}, P_{b2})라고 했을 때 만일 $P_{a1}+P_{b1}$(두 요소의 연결)=P_{a2}의 관계를 만족한다면 이 두 노드는 부모/자식 관계에 있고 직접 포함($\supset_d$) 관계에 있음을 알 수 있다. 즉, 그림 5에서 B 노드: (P_{a2}=0010, P_{b2}=0), C 노드: (P_{a3}=0010, P_{b3}=9)은 모두 A 노드: (P_{a1}=00, P_{b1}=10)의 연결값($P_{a1}+P_{b1}$)을 P_{a2}, P_{a3}로 가지고 있기 때문에 직접 포함관계에 있고 B,C는 모두 A의 자식이 되고 A는 B의 부모가 된다.

예제 4.3 트리내의 한 노드의 패턴을 (P_{a1}, P_{b1})로 부여하고 또 한 노드를 (P_{a2}, P_{b2})라 했을 때 만일 P_{a1} + P_{b1}(두 요소의 연결)가 P_{a2}의 전위(prefix)가 된다면 이 두 노드는 조상/후손 관계에 있고 간접 포함($\supset$)의 관계에 있음을 알 수 있다. 즉, 그림 5에서 B 노드: (P_{a2}=0010, P_{b2}=0), C 노드: (P_{a3}=0010, P_{b3}=9), D 노드: (P_{a4}=00109, P_{b4}=0)는 모두 A 노드: (P_{a1}=00, P_{b1}=10)의 연결값($P_{a1}+P_{b1}$)을 P_{a2}=0010, P_{a3}=0010, P_{a4}=00109의 전위

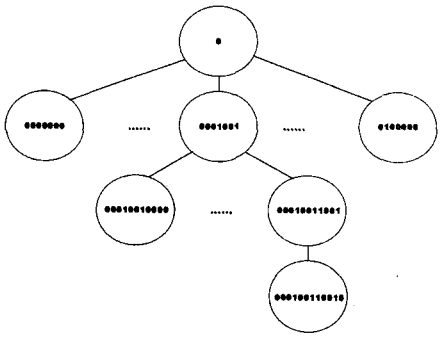

그림 3 FMFT. 모델

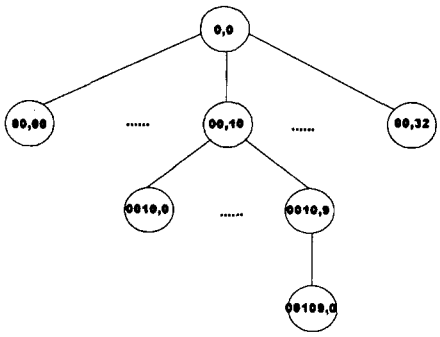

그림 4 FMFT+ 모델

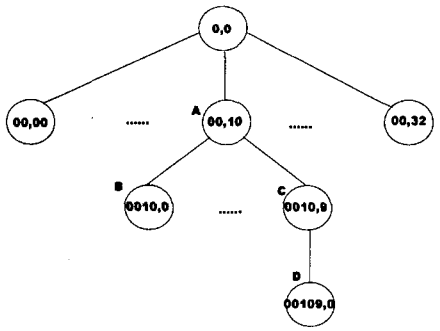

그림 5 고유 패턴 부여

(prefix)로 가지고 있기 때문에 간접 포함관계에 있고 B, C, D는 모두 A의 후손이 되고 A는 조상이 된다. 또한 직접 포함($\supset_d$)관계는 간접 포함($\supset$)관계의 특수한 경우이다.

예제 4.4 트리내에 있는 한 노드의 식별자를 (P_{a1}, P_{b1})로 부여하고 또 한 노드에 대해서는 (P_{a2}, P_{b2})로 하였을 때 $P_{a1}=P_{a2}$이라면 두 노드는 형제 관계(>)에 있으며 $P_{b1}>P_{b2}$이면 P_{b1}은 P_{b2}에 앞서오고 P_{b2}은 P_{b1}에 따라오게 된다. 즉, 그림 5의 B 노드: (P_{a2}=0010, P_{b2}=0), C 노드: (P_{a3}=0010, P_{b3}=9)는 $P_{a2}=P_{a3}$이고 $P_{b2}<P_{b3}$로서 노드 B는 노드 C에 앞에 위치한다.

예제 4.5 트리 내의 두 노드 A와 B가 어떤 한 노드 C의 조상이 된다고 가정하자. 그리고 두 노드 A와 B 노드 사이는 다시 조상/후손의 관계를 갖고 있고 만일 A가 B 노드의 조상라면 이 세 노드는 A ⊃ B ⊃ C 의 관계를 가지고 있다. 즉, 노드 C와 더 가까운 포함관계를 가지고 있는 노드는 B 노드이다. 즉, 그림 5에서 D 노드: (P_{a4}=00109, P_{b4}=0)에 대하여 A 노드: (P_{a1}=00, P_{b1}=10)와 C 노드: (P_{a3}=0010, P_{b3}=9)는 모두 D 노드의 P_a값의 전위를 P_a+P_b로 가지고 있기 때문에 모두 D 노드의 조상이 된다. 또한 다시 C 노드 P_a의 전위가 A 노드의 P_a+P_b이기 때문에 이 세 노드관계는 A⊃B⊃C를 가지고 있다. 이러한 관계를 정의하는 것은 엘리먼트 이름은 같지만 두 엘리먼트가 서로 포함관계를 가지고 있는 경우 어떤 노드를 질의의 결과로 반환할 것인가를 결정할 때 필요하다.

본 논문에서는 FMFT+의 이러한 속성을 이용하여 새로운 인덱싱 기법을 정의하는데 이를 족보 기반 인덱스(genealogy-based index)라 이름하고 이 인덱스를 통하여 XML 데이타가 가지고 있는 고유 속성인 계층적인 관계와 형제 순서와 같은 속성을 보존하여 같은 엘리먼트 이름을 가졌다 하더라도 위치를 지정하여 질의할 수 있고 그 결과를 사용자에게 보존하여 반환할 수 있도록 한다.

정의 4.2 XML 문서에 대한 엘리먼트 이름과 참조 타입이 아닌 애트리뷰트 이름을 위한 인덱스 NI(Name Index) = $Name_1$→((P^1_{a1}, P^1_{b1}),⋯, (P^1_{ak}, P^1_{bk})),⋯, Name $_n$→((P^n_{a1}, P^n_{b1}),⋯, (P^n_{am}, P^n_{bm}))이다. 여기서 (P_{a1}, P_{b1}), ⋯, (P_{an}, P_{bn})은 FMFT+에서 정의한 방법으로 부여되는 노드의 고유의 패턴 값이고 모든 인덱스의 수(총 노드의 수) = k+⋯+m이다.

정의 4.3 XML 문서에 대한 참조 타입의 애트리뷰트 이름에 대한 인덱스 RNI(attribute Name Index for Reference) = $Name_1$→(((P$^1_{a1}$, P$^1_{b1}$),(RP$^1_{a1}$, RP$^1_{b1}$),⋯, (RP$^1_{an}$, RP$^1_{bn}$)),⋯,((P$^1_{ak}$, P$^1_{bk}$),(RP$^1_{ak1}$, RP$^1_{bk1}$),⋯,(RP$^1_{akn}$, RP$^1_{bkn}$))),⋯, $Name_n$→(((P$^n_{a1}$, P$^n_{b1}$), (RP$^n_{a1}$, RP$^n_{b1}$) ,⋯,(RP$^n_{ap}$, RP$^n_{bp}$)),⋯,((P$^n_{am}$, P$^n_{bm}$),(RP$^n_{am1}$, RP$^n_{bm1}$) ,⋯,(RP$^n_{amq}$, RP$^n_{bmq}$)))이다. 여기서 (P_a, P_b)는 FMFT+에서 정의한 방법으로 부여되는 자기 노드의 고유의 패턴 값이고 (RP_a, RP_b)는 자신이 참조하고 있는 노드의 고유 패턴 값이다. 또한 $Name_1$은 애트리뷰트가 문서 상에서 k번 나타나고 있다는 것과 참조하고 있는 엘리먼트의 수는 n개 임을 뜻한다.

정의 4.4 XML 문서에 대한 엘리먼트 내용이 값인 것과 애트리뷰트 값을 위한 인덱스 VI(Value Index)= (Keyword$_1$, frequency$_1$) → (((P$^1_{a1}$, P$^1_{b1}$), wp$^1_1$),⋯, ((P$^1_{am}$, P$^1_{bm}$), wp^1_m)), ⋯, (Keyword $_n$, frequency$_n$) → (((P$^n_{a1}$, P$^n_{b1}$), wp$^n_1$),⋯, ((P$^n_{ak}$, P$^n_{bk}$), wp^n_k))이다. Keyword$_1$은 파서에 의해 토큰으로 분해된 단어들이 발생 빈도 frequency$_1$로 나타남을 뜻하고 frequency$_1$=m의 관계를 만족한다. 또한 wp(word position)는 keyword가 문서상에서 나타나는 단어의 위치를 말한다.

4.2 족보 기반 인덱스를 이용한 질의 처리

본 절에서는 앞의 정의에 따라 예제 문서에 대해 인덱스를 구축하고 이러한 인덱스들이 XML 질의어에서 필요한 질의를 어떻게 처리하는지를 XQuery의 구문을 이용하여 보여줄 것이다.

예제 4.6 본 논문의 3장에서 보여준 그림 2의 은행에 대한 예제 그래프에 대하여 각 노드마다 고유의 패턴 값을 부여한다면 그림 6과 같이 되고, 엘리먼트 이름과 애트리뷰트 이름에 대한 인덱스는 포스팅 파일로 구축한다. 만일 여러 문서를 대상으로 한다면 이러한 패턴 값들에 대해 문서 식별자(Documents Identifier)를 추가하면 된다.

포스팅 파일로 만들어지는 인덱스의 구조는 다음과 같이 엘리먼트 이름과 애트리뷰트 이름에 따라 부여된 고유 패턴을 가르치도록 되어 있다.

```
Banks->(0,0)
Bank->(00,0), (00,1), (00,2)
Customer->(001,1), (001,2)
Customer_name->(0011,0), (0012,0)
...
account_no->(001,0)
customer_id->(001,1), (001,2)
...
owner->((001,0), (001,1), (001,2))
accounts->((001,1), (001,0)), ((001,2), (001,0))
...
```

여기서 owner나 accounts는 IDREFS타입의 참조형이기 때문에 정의 4.3의 정의에 따라 자기 노드에 부여된 고유 식별자와 그 노드가 참조하고 있는 노드의 고유 식별자를 인덱스의 값으로 갖게 된다. 이와 같이 질의문에 사용되는 엘리먼트 이름이나 애트리뷰트 이름을 보고 포스팅 파일에서 검색하여 앞으로 설명할 문서 구성 요소들 사이의 계층적인 관계를 만족하는 고유 식별자를 찾아 원하는 데이타의 위치를 반환하게 된다.

족보기반 인덱스는 질의하고자 하는 대상을 나타낼 때 어떤 노드를 중심으로 조상/후손관계나, 부모/자식관계, 형제 관계 등에 의한 경로 표현을 처리하기 위해 매우 유용하다. 본 논문은 여기서 XQuery 질의문의 문법을 이용하는데 완전한 질의문이 아니라 부분 경로 표현만을 사용하기로 한다.

예제 4.7 만일 customer_street를 포함하는 bank를 찾는다고 한다면 다음과 같이 질의하게 될 것이다.

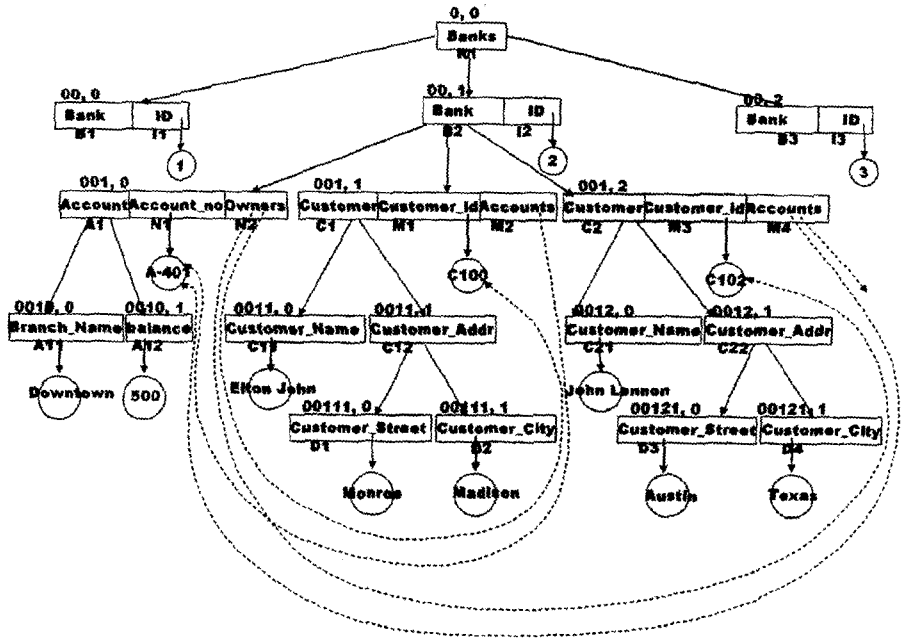

그림 6 FMFT+을 기반으로 고유 패턴 값 할당

```
FOR $b in bank
WHERE $b//customer_street
RETURN $b
```

이 때 족보 기반 인덱스를 사용하여 bank의 고유 패턴 값 중 customer_street의 첫 번째 요소의 전위를 고유 패턴으로 가지고 있는 bank를 찾는다. 즉, bank 노드는 B1, B2, B3가 있지만 customer_street를 포함하고 있는 bank는 B2가 된다. 이것은 customer_street 노드에 부여된 고유 값의 첫 번째 요소인 00111과 00121의 공통 전위 (00,1)를 노드의 값으로 가지고 있는 것으로 쉽게 찾을 수 있다.

예제 4.8 또한 customer_city가 'Texas'인 customer_addr를 모두 찾는다고 한다면 다음과 같이 질의 할 것이다.

```
FOR $c in customer_addr
WHERE $c/customer_city= "Texas"
RETURN $c
```

이것은 customer_city가 'Texas'인 D4 노드의 부모인 customer_addr을 찾으면 되니까 D4 노드의 첫 번째 요소인 00121 값을 패턴 값으로 가지고 있는 customer_addr인 C22가 된다.

또한 족보 기반 인덱스는 형제관계에 있는 엘리먼트를 찾는데도 유용하다. 특히 XML과 같이 순서의 속성이 중요한 경우, 위치를 지정하여 찾을 때 그 속성을 유지하여 질의할 수 있도록 한다.

예제 4.9 만일 customer_street가 'Austin'이고 customer_city가 'Texas'인 customer에 대한 정보를 찾고자 한다면 다음과 같이 질의한다. 이 때 customer_street와 customer_city의 관계는 서로 형제의 관계에 있다.

```
FOR $c in customer
WHERE $c//(customer_street="Austin" AND customer_
city="Texas")
RETURN $c
```

이것은 먼저 customer_street가 'Austin'인 노드 D3를 찾아 이 노드에 부여된 고유 패턴의 첫 번째 요소와 일치하는 값이 첫 번째 요소인 customer_city를 찾으면 D4가 된다. 그 다음 D3와 D4 노드의 첫 번째 요소인 00121의 전위를 패턴의 첫번째 요소로 가지고 있는 customer를 찾으면 C2 노드가 찾아진다.

족보 기반 인덱스는 같은 엘리먼트 이름을 가진 형제들 중에서도 위치를 지정하여 질의할 수 있도록 해준다.

예제 4.10 만일 은행의 고객 중 첫번째로 들어 있는 데이타를 찾고자 한다면 다음과 같이 표현한다.

```
bank/customer[first]
```

이것은 bank의 customer에 할당된 첫번째 요소의 값이 같은 것 중 두 번째 요소의 값이 가장 작은 것을 모두 찾으면 된다.

또한 족보 기반 인덱스는 형제들의 순서를 갖고 있기 때문에 다음과 같이 어떤 엘리먼트에 대해 범위를 지정하는 경로 표현을 가진 질의에도 응답하기 쉽다.

customer[1TO10]//customer_name

이것은 첫번째 고객부터 열번째 고객까지의 이름을 찾기 위해 customer에 할당된 고유 패턴의 첫번째 요소의 값이 같은 것들 중 두번째 요소의 값들을 순서대로 열번째까지 찾아 이들의 첫번째 요소의 값과 두번째 요소의 값을 연결한 값을 customer_name의 고유 패턴의 첫번째 요소의 값으로 가지고 있는 것을 찾는다.

족보 기반 인덱스는 링크를 따라 어떤 엘리먼트를 참조하고 있는 애트리뷰트에 의한 검색도 쉽게 할 수 있다.

예제 4.11 만일 account를 소유하고 있는 customer의 이름을 찾고자 한다면 다음과 같은 경로 표현을 사용하여 질의할 것이다.

Account/@owner=>customer/customer_name

이것은 참조를 위한 인덱스 값을 찾아((001,1), (001,2)) 이 값을 customer의 값으로 가지고 있는 노드 C1, C2를 찾은 다음 owner에 대한 인덱스에서 자식 노드가 되는 customer_name 노드인 C11, C21을 찾는다.

5. 성능 평가

본 논문은 족보 기반 인덱싱 기법의 성능을 평가하기 위해 표 1에서 보는 바와 같은 기준으로 여러 개의 문서들을 선정하였고, 객체 관계형 데이타베이스에 저장하여 원하는 검색을 하기까지 데이타베이스 테이블에 접근하는 횟수를 각 문서마다 계산하여 그들의 평균을 취하였다.

우선 XML 문서의 종류를 트리 형태와 그래프 형태로 분류하였는데, 이것은 애트리뷰트의 타입이 IDREF(S)인 경우 다른 엘리먼트를 참조하게 되는데 이럴 경우 엘리먼트를 공유함으로써 문서의 전체적인 형태가 그래프의 모습을 띄게 된다. 트리 형태의 문서에 대하여서는 높이와 넓이에 대해 균형을 이루는 것과 그렇지 않은 것으로 나누어 실험을 하였다. 트리가 균형을 이루는 경우는 다시 높이가 5이상인 문서와 이하인 문서에 각각 하나씩 실험하였다. 균형을 이루지 않는 문서에 대해서는 높이가 균형을 이루지 않는 것이냐 아니면 넓이가 균형을 이루지 않는 것이냐에 따라 각각의 문서를 선정하였다.

그래프 기반의 문서에 대해서는 경로를 공유하는 경우와 그렇지 않은 경우로 나누어 문서를 선정하고 또 다시 형제 노드 사이에 순서를 구분할 수 있는 문서를 선정하여 실험한 결과를 평가하였다.

또한 이러한 문서들을 대상으로 성능을 평가할 때에 질의의 종류는 표 2와 같이 분류하여 검사하였다. 질의

표 1 성능 평가에 사용된 문서들의 유형

문서 유형	계층화된 문서유형			차수/높이	문서 수
트리	균형 (차수와 높이 둘 다 균형)	단일 레벨			1
		다중 레벨		높이 < 5	1
				높이 >= 5	1
	비균형 (차수나 높이 중 어느 하나라도 비균형	높이 비균형 (차이 > 3)	차수 비균형	차이 > 3	1
			차수 균형	차수 < 5	1
				차수 >= 5	1
		높이 균형	차수 비균형 (차이 > 3)	높이 < 5	1
				높이 >= 5	1
그래프	노드 공유	같은 경로 상에서 공유			1
		다른 경로 상에서 공유			1

표 2 질의 유형

질의 종류		질의
엘리먼트 값 기반 질의	엘리먼트 = 값	Q1
	엘리먼트 > 값	Q2
	엘리먼트1 = 엘리먼트2	Q3
	엘리먼트 BETWEEN 값1 AND 값2	Q4
	엘리먼트 IN (값 리스트)	Q5
	엘리먼트1 = 값1 OR 엘리먼트2 = 값2	Q6
	엘리먼트1 = 값1 AND 엘리먼트2 = 값2	Q7
	NOT (엘리먼트 = 값)	Q8
순회 기반 질의	단순 경로 표현(부모/자식)	Q9
	정규 경로 표현(조상/후손)	Q10
링크 기반 질의	단일 문서 링크	Q11
	다중 문서 링크	Q12

의 종류는 크게 엘리먼트의 값에 따라 질의하는 것과 엘리먼트 사이를 순회하는 것, 링크의 속성에 따라 엘리먼트가 또 다른 엘리먼트를 참조하는 것에 따라 나누었다. 엘리먼트의 값에 따르는 질의는 정확한 그 값만을 찾는 경우와 범위에 의한 검색으로 나누었는데 이 분류는 P.Griffiths Selinger의 "Access Path Selection in a Relational Database Management System"[22]에서 언급되었던 선택요소(Selectivity Factor)의 개념을 기반으로 하여 이를 XML 질의에 적합한 형태로 변형한 것이다. 순회 기반 질의는 두 노드 간의 관계가 직접 포함 관계 즉, 부모/자식 간의 관계에 대한 것과 간접 포함 관계인 조상/후손의 관계로 나누었다. 링크 기반의 질의는 한 문서 안에서의 참조가 일어나는 것과 다른 문서를 참조하는 것으로 나누어 실험하였다.

성능 평가를 위한 자세한 비용 모델은 [23]에 실려있는데, 기본적인 비용을 계산하는 공식은 다음과 같다. 여기서 질의의 유형에 따르는 파라미터에 따라 각기 다른 비용을 갖는데 이것은 표 2의 질의 유형을 분류한 것에 따라 비용 모델을 세웠다.

$$Cost = \sum(T * N_{pt}) + P + \sum(N_{pv})$$

- T : 찾고자 하는 값의 상위 노드 수
- N_{pt} : 찾고자 하는 값을 가진 노드에 도달할 때까지 참조하는 페이지 수
- N_{pv} : 서술어(predicate)를 만족하는 값을 포함하고 있는 페이지 수
- P : 질의의 유형에 따른 파라미터

질의 처리 속도에 대한 비교의 대상으로서는 [19]에서 제안한 8가지 저장 기법에 의한 질의 처리 방법 중 전반적으로 가장 성능이 좋다고 알려진 인라이닝을 가진 애트리뷰트 테이블(attribute table with inlining) 기법과 [21]에서 제안한 세가지 저장 기법 중 조인을 필요로 하는 질의 중 전반적으로 질의 처리 성능이 좋다고 알려진 하이브리드 인라이닝(hybrid inlining) 기법을 비교하였다.

성능 평가 결과, 족보기반 인덱싱 기법의 경우 질의의 형태에 제한을 받지 않았으며 3가지 질의 처리를 위한 저장 기법 중 성능이 가장 우수한 것으로 분석되었다.

애트리뷰트 인라이닝 저장 기법은 질의의 형태에 제한을 받지는 않았으나, 족보 기반 인덱싱보다는 성능이 비효율적으로 평가되었다. 또한, 하이브리드 인라이닝 저장 기법의 경우, 하나의 값이 여러 테이블에 등장하여 데이타 저장의 공간면에서도 중복성으로 인하여 비효율성을 보여주었으며, 이는 XML 문서의 크기가 커질수록 심한 비효율성을 초래한다고 평가되었다. 또한, 질의의 형태에 따라 많은 조인을 유발하여 성능이 더욱 저하되었는데, 특히 Q6과 같은 질의 형태인 "엘리먼트1=값1 OR 엘리먼트2=값2"를 만족하는 경우를 검색하면 여러테이블에서 검색 결과를 반환하기 때문에 다른 기법에 비하여 훨씬 많은 테이블 접근이 발생하게 된다. 만일 검색 결과가 많을 경우에는 데이타 중복이 심한 하이브리드 인라이닝 방법에서는 여러 테이블에 중복되어 나타나기 때문에 더욱 질의 속도의 차이가 벌어지게 된다.

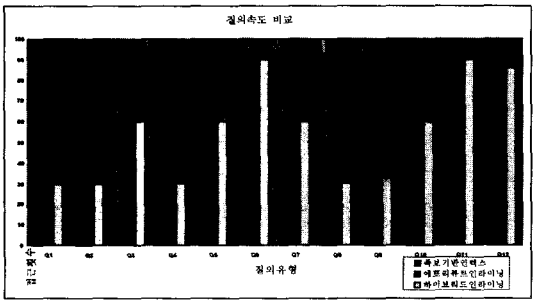

그림 7 성능 평가

6. 결론 및 향후 연구

본 논문는 XML 문서를 가계의 혈통 관계에 비교하여 새로운 인덱싱 기법인 족보 기반 인덱스라는 것을 개발하여 XML 문서를 대상으로 다양한 종류의 질의 처리를 가능하도록 하였다. 즉, XML 문서에서 각 노드의 관계를 족보에 비교하여 XML 질의 언어에서 요구하는 부모/자식, 조상/후손, 형제 관계를 밝힘으로써 쉽게 질의 처리를 할 수 있도록 하였다. 또한 하나의 엘리먼트가 또 다른 엘리먼트를 참조하고 있는 참조에 의한 질의 처리 문제를 해결하였으며 결과를 반환할 때는 XML이 지니고 있는 고유의 의미를 보존하여 반환할 수 있도록 하였다. 이 인덱싱 기법은 질의를 처리하는 속도 면에서도 우수함을 보여 주었다.

본 논문은 앞으로 족보 기반 인덱싱 기법의 연구를 확장하여 XML 문서에 새로운 엘리먼트가 삽입되거나 삭제되는 경우에도 인덱싱하는 데 드는 비용을 최소로 하도록 할 것이다.

참 고 문 헌

[1] W3C Consortium, XML1.0 (Second Edition), W3C Recommendation, 6 Oct. 2000, available at http://www.w3.org.TR/2000/WD-xml-2e-20000814.

[2] Q. Li and B.Moon, Indexing and Querying XML data for Regular Path Expressions, VLDB, 2001.

[3] C. Zhang, J. Naughton, D. DeWitt, Q. Luo, G. Lohman, On Supporting Containment Queries in Relational database Management Systems, SIGMOD, 2001.

[4] D. Florescu and D. Kossmann, Storing and Querying XML data using an RDBMS, IEEE Data Engineering Bulletin, 22(3): 27~34, 1999.

[5] J. Shanmugasundaram, K. Tuffe, G. He, C. Zhang, D. DeWitt, and J. Naughton, Relational databases for Querying XML Documents: Limitations and Opportunities, VLDB, 1999.

[6] C. Zhang, Q. Luo, D. DeWitt, J. Naughton, and F. Tian, On the Use of a Relational database Management System for XML Information Retrieval, 2000.

[7] S. Banerjee, Oracle XML DB, Oracle Corporation Technical White Paper Release 9.2, Jan. 2002.

[8] S. Howlett and D. Jennings, SQL Server 2000 and XML: Developing XML-Enabled data Solutions for the Web, MSDN magazine, Jan. 2002, available at http://msdn.microsoft.com/library/default.asp?url=/msdnmag/issues /0800/sql2000/toc.asp

[9] IBM Corporation, DB2 XML Extender, IBM Corporation, 2000, available at http://www-4.ibm.com/

[10] V.Aguilera, S.Cluet, P.Veltri, D.Vodislav, and F. Wattez, Querying XML Documents in Xyleme, SIGIR, 2000.

[11] C. Chung, J. Min, and K. Shim, APEX: An Adaptive Path Index for XML data, SIGMOD, 2002.
[12] F. Rizzolo and A. Mendlzon, Indexing XML data with ToXin, 4th Int. Workshop on the Web and database, 2001.
[13] B. F. Cooper, N. Sample, M. J. Franklin, G. R. Hialtason, and M. Shadmon, A Fast Index for semistructured data, VLDB, 2001.
[14] B. F.Cooper, N. Sample, and M. Shadmon, A Parallel Index for semistructured data, SAC, 2002.
[15] M. Fernandez and D. Suciu, Optimizing Regular Path Expressions Using Graph Schemas, ICDE, 1998.
[16] R. Goldman and J. Widom, Dataguides: Enabling Query Formulation and Optimization in semistructured databases, VLDB, 1997.
[17] T. Milo and D. Suciu, Index Structures for Path Expressions, ICDT, 1997.
[18] J. McHugh, J. Widom, S. Abitboul, Q. Luo, and A. Rajaraman, Indexing semistructured data, In Stanford Technical Report, Jan. 1998.
[19] W3C Consortium, XML Path language (XPath) Version 1.0, W3C Recommendation 16 Nov. 1999, available at http://www.w3.org/TR/xpath.html
[20] W. Thomas, Automata on Infinite Objects, Handbook of Theoretical Computer Science, Vol. B, 135-191, 1990.
[21] W3C Consortium, Xquery1.0 Formal Semantics, W3C Working Draft 16 Aug. 2002, available at http://www.w3.org/TR/query-semantics.
[22] P. Selinger, M. Astrahan, D. Chamberlin, R. Lorie, and T. Price, Access Path Selection in a Relational Database Management System, Proceedings of the ACM SIGMOD International Conference on Management of Data, Boston, MA, 1979.
[23] J. Kim, W. Lee, K. Lee. "The Cost Model for XML Documents," In Proc. of ACS/IEEE International Conference on Computer Systems and Applications, Beirut, Lebanon, Jun. 2001.

이 월 영

1988년 이화여자대학교 전자계산학과(학사). 2000년 이화여자대학교 컴퓨터학과(석사). 2000년~현재 이화여자대학교 컴퓨터학과 박사과정. 관심분야는 XML,데이타베이스, 정보검색, 인터넷언어설계

용 환 승

1983년 서울대학교 컴퓨터공학과 학사 1985년 서울대 대학원 컴퓨터공학과 공학석사. 1985년~1989년 한국전자통신연구소 연구원. 1994년 서울대 대학원 컴퓨터공학과 공학박사. 2002년 8월~2003년 2월 IBM T.J. Watson 연구소 객원연구원. 1995년~현재 이화여자대학교 컴퓨터학과 부교수 관심분야는 객체-관계 데이타베이스 시스템, 멀티미디어 데이타베이스, OLAP 및 데이타 마이닝, 바이오정보학, 유비쿼터스 컴퓨팅