경로를 표시하지 않는 XML 질의
(XML Queries without Path Expressions)

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

요 약 XML은 수 많은 응용들에서 데이타를 교환하기 위한 표준으로 급속도로 출현되었다. XML 데이타에 대해 효율적인 질의를 지원하기 위하여 많은 질의어들도 설계되었다. 이러한 질의어들은 사용자들이 XML 문서 구조를 알아야 사용할 수 있고 구조에 대한 검색 조건을 명시해야만 한다. 이러한 XML 문서에 대해 경로 기반으로 하는 질의는 XML이 계층적 구조이기 때문에 당연한것이다. 그러나 이러한 현재의 경로 기반의 질의를 보충하기 위하여, 사용자들이 XML 문서에 대하여 경로를 사용하지 않는 질의도 필요하다. 이 논문에서 우리는 XML 문서 구조를 모르고도 질의할 수 있는 질의 표현을 설계하고, 이 경로를 명시하지 않는 질의를 평가하기 위한 질의 처리기를 개발하였다.

키워드 : XML 질의언어, XML 데이타베이스, 질의 처리

Abstract XML has rapidly emerged as the standard for the interchange of data in numerous application areas. To support for efficient queries against XML data, many query languages have been designed. The query languages require the users to know the structure of the XML documents and specify search conditions on the structure. This path-based query against XML documents is a natural consequence of the hierarchical structure of XML. However, it is also desirable to allow the users to formulate no path queries against XML documents, to complement the current path-based queries. In this paper, we design a query expression capable of querying without knowledge about the structure of XML documents, and develop a query processor to evaluate no path queries.

Key words : XML Query language, Query processing, XML Database

1. 서 론

XML[1]은 데이타를 표현할 때 태그라는 것을 사용하여 데이타의 이름을 나타내고 그 값을 시작 태그와 종료 태그 사이에 나타낸다. 또한 시작 태그 안에 애트리뷰트를 이용하여 데이타를 표현할 수도 있다. 태그로 표현하는 데이타는 엘리먼트라 하고 다른 데이타와의 포함 관계 역시 이 엘리먼트를 이용하여 자유롭게 표현함으로써 계층적인 구조를 이루도록 할 수 있다. 이러한 특성 때문에 많은 데이타들이 XML을 이용하여 표현되고 있고, 이렇게 표현된 문서에 대해 사용자가 원하는 것을 질의하기 위해 많은 질의 언어들도 개발되었다[2-5]. 그러나 임의 구조를 허용하는 XML의 특징은 똑같은 내용을 가지고 생성하는 문서라 하더라도 사람마다 생성 목적과 용도에 따라 어떠한 다른 데이타 모델보다도 많은 형태의 문서를 만들 수 있도록 한다. 결국 XML의 유연성은 질의할 때 문서의 구조에 대해 훨씬 많은 지식을 요구하는 요인이 된다.

예를 들어 그림 1과 같이 영화에 대한 정보를 담고 있는 여러 개의 XML 문서가 있다고 하고 사용자는 다음의 질의를 원한다고 하자.

Q1: 장르(genre)는 action이고 제작년도(year)가 1994년인 영화 중, 주연 배우(actor)가 'Jean Reno'인 영화의 제목(title)을 찾아라.

그림 1은 XML에서는 질의에 주어지는 데이타 genre, actor, year, title의 표현방법, 삽입되는 데이타, 포함관계에 따라 여러 가지 모습의 문서들이 존재한다는 것을 보여준다. 대개 기존의 경로(path) 기반 XML 질의 언어에서는 질의 표현에 사용자가 알고 있는 genre, year, actor라는 데이타와 찾고자 하는 이들의 값 외에 이들 사이의 경로를 나타내야 한다. 즉, 그림 1 (a)의 문서에 대해서는 //year="1994" and //genre= "action"

· 본 연구는 한국과학재단 목적기초연구(과제번호 R01-2003-000-10395-0) 지원으로 수행되었음
† 비 회 원 : 이화여자대학교 컴퓨터학과
wylee@ewha.ac.kr
†† 종신회원 : 이화여자대학교 컴퓨터학과 교수
hsyong@ewha.ac.kr
논문접수 : 2004년 7월 6일
심사완료 : 2004년 11월 25일

그림 1 질의에 주어진 데이타 사이에 다양한 구조를 갖는 여러 문서

and //actor="Jean Reno"로 질의 표현을 하면 되지만 그림 1(b)처럼 데이타가 애트리뷰트로 표현되었다면 //@year="1994" and //@genre="action" and //@ actor=" Jean Reno"로 질의해야 한다.

또한 그림 1(c)나 (d)처럼 사용자가 알고 있는 데이타 이름과 그 값 사이에 또 다른 데이타 이름이 삽입되어 있는 경우는 그 사실을 알아야 질의가 가능하고 그것이 엘리먼트로 표현되었는지 애트리뷰트로 표현되었는지에 따라 질의문이 달라진다. 즉, 그림 1(c)에 대해서는 //year/@yyyy="1994" and //genre/@type="action" and //actor/@name="Jean Reno"나 //year/@*="1994" and //genre/@*="action" and //actor/@*="Jean Reno"로 질의 표현을 해야 한다. 이에 반해 그림 1(d)에 대해서는 //year/yyyy="1994" and //genre/type="action" and //actor/name="Jean Reno" 또는 //year/*="1994" and //genre/*="action" and //actor/*="Jean Reno"로 해야 한다. 이것은 삽입된 임의의 데이타 이름은 정확히 모르더라도 그 구조는 알아야 한다는 것을 의미한다.

또한 데이타들 사이의 관계가 의미적으로 포함 관계를 갖지 않을지라도(non-nested relationship) 분류를 목적으로 문서를 구성한 경우에는 그림 1의 (e)나 (f)처럼 포함된 관계(nested relationship)를 갖도록 작성할 수도 있다. 이러한 경우에 문서 (e)에 대해서는 //year="1994"//genre="action"//actor="Jean Reno"로 표현해야 하지만 (f)에 대해서는 year와 genre의 상하관계가 바뀌었기 때문에 역계층의 질의 표현을 해야 한다. 따라서 질의를 하기 위해서는 문서 구조가 어떻게 구성되어 있는지를 가장 먼저 알아야 질의 표현을 할 수 있다.

현재까지 대개의 질의 언어는 XML 문서의 어느 부분을 검색하기 위해 경로를 순회하는 방식의 질의 표현을 사용하도록 되어 있다. XML 문서가 계층적인 구조로 되어 있기 때문에 이러한 접근 방식은 당연한 것이고 섬세한 정보를 아는 경우 경로 기반 질의 표현을 사용하여 정확한 검색 결과를 받을 수 있어야 한다. 그러나 분산 환경 같은 곳에서 사용자는 대개 문서의 구조를 잘 모른다. 또한 로컬 환경이라 하더라도 다른 사람이 만들어 놓은 여러 가지 문서를 임포트하여 사용하는 경우 사용자가 문서의 구조를 자세히 모르더라도 질의할 수 있도록 하기 위해서는 이것을 가능하게 하는 질의 표현이 필요하고 이러한 질의 표현을 지원하는 질의 처리 기법 또한 필요하다. 문서 구조를 모르고도 질의할 수 있는 것을 위해 대개의 질의 언어들은 정규 경로 표현이라는 것을 허용하고 있고 이에 대한 처리 기법도 많이 연구되었으나[6-11] 이것은 단지 우리가 본 논문

에서 제안하고 있는 문서 구조 독립적인 질의를 위해 필요로 하는 6가지 요건 중에 단 한가지 만을 만족 시키는 것으로서 완전한 문서 구조 독립적인 질의를 지원하지는 못한다. 또한 키워드 기반 검색을 통해 문서 구조에 자유로운 질의 표현을 하고자 하는 연구가 있었으나 이것은 질의 표현에 주어지는 정보가 데이타 값뿐이기 때문에 검색 정확도 면에서 한계를 가지고 있다[12,13].

문서 구조를 모르고도 질의할 수 있는 기법에 대한 연구는 하나의 문서를 대상으로 질의할 때도 편리함을 제공하지만 같은 데이타를 가지고 만든 구조가 다른 유사한 종류의 문서들을 대상으로 질의할 때는 더욱 필요하다. 또한 XML 문서 중에는 DTD가 없는 well-formed 문서도 많을 뿐 아니라 DTD가 있는 문서인 경우도 그 구조가 복잡한 경우가 많다. 더구나 분산 환경 같은 곳에서 스키마 정보를 자세히 안다는 것은 매우 어려운 일이다.

따라서 본 연구에서는 다음과 같은 접근 방법을 이용하여 그림 1에서 보여준 다양한 구조의 문서에 대해 그 구조를 고려하지 않고 모든 문서에 대해 하나의 질의 표현을 사용하여 원하는 질의 결과로서 Leon이라는 영화 제목을 검색해 낼 수 있도록 하였다.

- 첫째, 문서 구조에 얽매이지 않는 질의를 위해서는 무엇보다도 질의문을 작성할 때 문서 구조를 명시하지 않는 질의 구문(syntax)이 필요하다. 따라서 단지 사용자가 알고 있는 데이타 이름(names)과 찾고자 하는 값(values)만을 표현하는 X-NPE(No Path Expression for XML)라는 질의 표현(query expression)을 설계하였다. 이 질의 표현은 문서 구조를 모르는 사용자가 정규 경로 표현보다 훨씬 완화되고 키워드 검색보다는 좀 더 정보를 제공하는 중간 형태의 질의 표현을 사용하여 질의할 수 있도록 한다.
- 둘째, X-NPE를 사용하여 질의를 하면 이 질의문에 표현된 데이타 이름들과 그들의 값만을 표현하기 때문에 이들 데이타 이름 사이의 경로는 질의 처리기가 알지 못한다. 본 논문에서는 주어진 데이타 사이의 모든 가능한 경로를 분석하여 이들에 대해 X-NPE를 처리하기 위해 필요한 유형별로 분류하고 이것을 처리할 수 있는 질의 처리 알고리즘을 개발하였다.

우리의 접근 방법은 경로 기반 질의처럼 XML의 고유 속성인 포함 관계나 순서 정보를 보존하여 질의할 수 있을 뿐 아니라 경로 기반 질의 보다 X-NPE를 사용하여 더 빠른 질의 처리를 수행할 수 있다. 또한 단순한 키워드 검색에 비해 데이타들이 문서에서 가질 수 있는 모든 경로들을 고려하여 질의 처리함으로써 검색 정확도를 높일 수 있다. 따라서 본 연구 결과는 다음과 같은 의의를 갖는다.

- 사용자가 질의할 때 그 경로를 어떻게 써야 할까를 먼저 걱정해야 하는 불편함을 없애주고 SQL-like 질의 표현처럼 데이타 이름과 그 값만을 사용하여 기존의 질의 사용자들이 친근하게 사용하도록 하였다. 비록 요즈음 편리한 질의 표현을 위해 시각적 인터페이스가 많이 제공되고 있지만 프로그래밍 환경과 같이 이러한 도구를 이용하지 못하는 경우에 유용하다.
- 분산 환경이나 웹 환경에서 사용자는 여러 사이트에 대해 문서의 구조를 모르는 경우가 많다. 이럴 때 우리의 접근 방법은 데이타 이름과 그 값만을 안다면 여러 사이트의 다양한 문서를 대상으로 쉽게 질의할 수 있도록 한다. 로컬 시스템이라 하더라도 공통의 스키마 없이 만들어진 여러 문서를 임포트하여 사용하는 경우 우리의 연구 결과는 적용될 수 있다.
- X-NPE는 경로를 명시하지 않는 표현식(No path expression)이기 때문에 사용자는 SQL-like 질의처럼 "애트리뷰트=값"의 형태로 질의 표현을 하고 시스템이 이들 사이의 경로를 해결한다. 이러한 사실은 사용자가 질의하는 동안 자연 언어를 사용하는 것처럼 편안함을 느끼도록 해준다.
- 기존 웹 데이타는 HTML로 표현되었지만 현재는 XML이 갖는 여러 가지 장점 때문에 XML로 표현되고 있다. 따라서 기존 웹 데이타를 대상으로 하는 웹 서치 엔진들은 키워드 검색을 기반으로 하고 있지만 XML은 데이타를 표현할 때 태그를 이용하기 때문에 본 연구 결과는 XML 웹 서치 엔진에 이용할 수 있다. 즉, XML이 태그를 이용하여 데이타를 정의한다는 점을 최대한 이용하여 본 연구에서 제안하는 데이타 이름과 값을 명시하는 내용 기반 ad hoc 질의를 XML 웹 서치 엔진에 활용한다면 기존의 키워드 기반 검색보다 훨씬 정확한 결과를 반환 받을 수 있다.

본 논문의 구성은 서론에 이어 2장에서는 관련 연구를 언급한다. 3장에서는 X-NPE를 정의하고 이를 처리하기 위해 필요한 경로들을 분석한다. 4장에서는 X-NPE를 처리할 수 있는 알고리즘을 개발한다. 5장에서는 X-NPE와 경로 기반 질의에 대한 성능 평가를 수행하고 6장에서 결론을 맺는다.

2. 관련 연구

XML 문서를 질의하기 위해 현재까지 많은 질의 언어[2-5]들이 개발되었다. 상업적인 제품으로서 Oracle XML DB[14]은 관계형 데이타베이스와 밀결합(tightly-coupled) 방식으로 XML을 다루고 있어 SQL의 확장된 구문으로 질의하도록 되어 있다. DB2 XML Extender[15]에서의 XML 지원은 정보 검색 엔진에서 질의할

수 있도록 contains와 같은 연산자를 사용하고 있다. 특히 SQL 2000 서버[16]는 URL을 통해 SQL 서버를 접근하고 XPath 기반의 질의 표현을 지원하고 있다. 이들 중 경로 기반의 질의 언어는 주로 XPath[5]를 지원하도록 설계가 되고 있고 사용자들을 문서 구조에 대해 자유롭도록 하기 위해 질의 표현을 완화할 수 있는 정규 경로 표현이라는 것을 허용하고 있다.

또한 반구조적인 데이타와 XML 문서에 대해 효과적인 질의 처리를 위한 많은 인덱싱 기법[6-10,17-22]이 연구되었는데 이 중 [7-8,11,17]은 정규 경로 표현을 위한 인덱싱 방법과 그 처리 기법을 소개하였다. 그러나 이 정규 경로 표현은 문서 구조 독립적인 질의를 위하여 우리가 해결하고자 하는 경로 유형들 6가지 중 단 한가지만을 질의 표현에서 완화하는 것으로서 완전한 문서 구조 독립적인 질의 표현을 지원하지는 못한다. 또한 문서 구조를 알지 못하는 사용자를 위해 XML 문서를 대상으로 키워드 기반 검색[12,23]이 소개되어 최근에는 [13,24-25]와 같은 연구에서 키워드 검색에 구조를 접목하는 기법을 개발함으로써 검색의 정확도를 높이고자 하였다. 그러나 키워드 기반 검색은 질의할 때 제공하는 정보가 데이타 값만을 명시하기 때문에 사용자가 요구하는 정확한 결과를 받는데 한계가 있다.

따라서 XML 문서가 비정규적이고 다양한 구조로 만들어질 수 있는 특징을 가졌다는 것에 비하여, 문서 구조 독립적인 질의가 가능하기 위한 질의 표현이나 처리 기법 등에 대한 연구는 미비한 상태이다.

3. X-NPE에 주어진 데이타 사이의 경로

XML 문서 구조는 태그에 표현하는 데이타 이름 사이의 경로에 의해 이루어진다. 따라서 문서 구조에 독립적인 질의를 위해서는 먼저 문서에서 이러한 경로가 어떻게 구성되는지에 상관없는 질의 표현이 가능해야 한다.

3.1 X-NPE

본 논문에서는 문서 구조에 독립적인 질의를 위해 자연 언어에서와 비슷한 질의 표현(semi-natural query expression)을 제안하고 이를 X-NPE(No Path Expression for XML)이라 명명하기로 한다.

정의 3.1 X-NPE은 문서 구조 독립적인 질의를 가능하게 하는 질의 표현이며 이에 대한 문법을 다음과 같이 정의한다.

NoPathExpr ::= ("not")? Predicate (("and"| "or") ("not")? Predicate)*

Predicate ::= ElemAttrName ("=" | "<" | "<=" | ">" | ">=" | "!=" | "contains") Value

Value ::= numeric | string

ElemAttrName은 찾고자 하는 데이타 이름을 의미하고 value는 그 값을 의미한다. 본 논문에서는 X-NPE를 XQuery[4]문에 적용할 것을 제안한다. 즉, 다양한 형태의 질의를 할 수 있는 XQuery의 확장된 하나의 표현으로서 상대경로표현(regular path expression의 한 형태)과 키워드 검색의 중간 형태인 X-NPE를 제안한다. 좀 더 정확한 질의를 원한다면 경로기반 질의 표현 (path-based query expression)을 사용할 수 있다.

예제 3.1 Q1의 질의를 원한다면 이것을 위한 X-NPE는 **genre = "action" and year = "1994" and actor = "Jean Reno"**이고 이를 XQuery문에 적용한다면 다음과 같이 표현할 수 있다. 조건절에는 X-NPE를 사용하고 결과로 반환하고자 하는 결과 데이타는 결과절에 표현한다.

```
for $t in doc ()//title
where genre = "action" and year = "1994" and
actor = "Jean Reno"
return $t
```

이 질의문은 1994년에 제작한 영화로서 genre는 "action"이면서 주연 배우가 "Jean Reno"인 영화의 제목을 찾기 위한 질의 표현이다. 질의 표현은 이렇게 간단하게 하고 질의 처리기는 질의 처리 시에 genre, year, actor, title 데이타가 상호 간에 가질 수 있는 모든 가능한 경로를 모두 고려하여 처리한다.

3.2 X-NPE의 조건절의 가능한 모든 경로

X-NPE에는 사용자가 알고 있는 데이타 이름과 그 값을 명시하고 이들 사이의 경로는 표현하지 않기 때문에 문서에서 이들이 가질 수 있는 가능한 모든 경로를 알아야만 질의 처리할 수 있다. 따라서 본 논문에서는 이러한 경로를 직관적으로 쉽게 파악하기 위하여 XML 문서에 있는 각 요소를 사각형과 원으로서 나타내는 트리로서 모델링한다. 사각형은 엘리먼트/애트리뷰트 이름을 나타내고 원은 이들의 값을 나타낸다. X-NPE의 조건절에 주어진 데이타들 사이의 가능한 모든 경로를 알아내기 위하여 이 데이타들 사이의 구조를 분석할 수 있는 기준을 세운다.

첫째, 두 데이타 사이의 관계는 비포함 관계(non-nested relationship)와 포함 관계(nested relationship)로 나눌 수 있다. 이러한 구조는 그림 2에서 보여 주는데 X-NPE의 조건절에 주어진 데이타가 genre, actor이고 그들의 값이 "action", "Jean Reno"일 때 그림 2(a)처럼 서로 사이의 관계가 비포함 관계를 이루는 경우와 (b)처럼 포함 관계를 이루는 경우가 있다.

둘째, X-NPE의 조건절에 주어진 데이타 사이에 임의의 데이타가 삽입된다면 이것은 문서 구조에 영향을 주기 때문에 임의의 데이타가 삽입되는 경우를 고려해야 한다. 예를 들어 그림 3(b)처럼 genre와 actor사이에 임

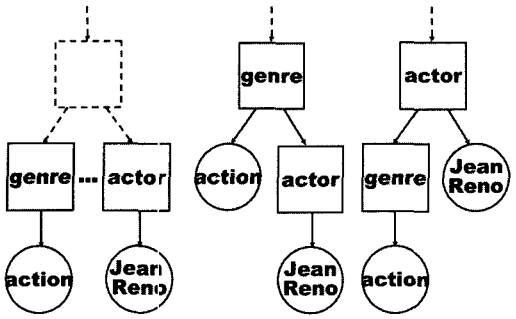
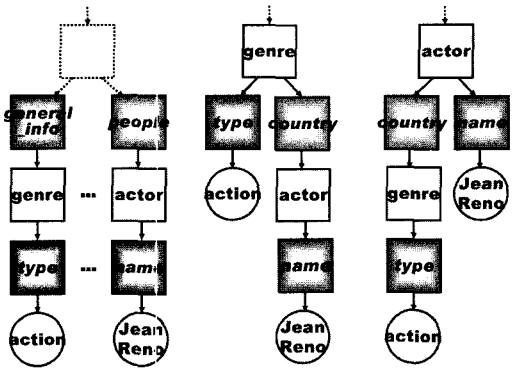


그림 2 주어진 데이타만을 사용하는 경우



(a) 비포함 관계(Non-nested Relationship) (b) 포함 관계(Nested Relationship)

그림 3 임의 데이타를 삽입하는 경우

의의 데이타 country가 삽입된 것처럼 임의의 데이타가 삽입될 수 있다. 또한 그림 3(a)처럼 genre와 "action" 사이에 임의의 데이타 type이 삽입되거나 actor와 "Jean Reno"사이에 name과 같은 임의의 데이타가 삽입될 수 있다.

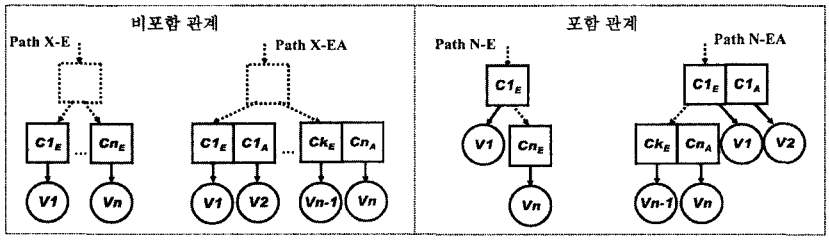
셋째, 데이타의 구조적인 측면만을 생각한다면 모든 가능한 경로를 분석하기 위해 위의 2가지 조건만을 고려하면 된다. 그러나 비록 구조에는 영향을 주지 않지만 XML은 데이타를 표현할 때 엘리먼트 뿐 아니라 애트리뷰트로 표현할 수가 있다. 따라서 X-NPE의 모든 가능한 경로를 분석하기 위해 모든 데이타가 엘리먼트로 표현되었는지 애트리뷰트로 표현되었는지를 고려한다. 이런 의미에서 앞으로 문서의 구조 또는 경로라는 단어를 단지 구조적인 측면만을 고려하는 것이 아니라 데이타를 표현하는 방법까지 포함하는 것으로 사용할 것이다.

이러한 분류 기준에 따라 X-NPE에 주어지는 데이타가 n개 인 경우 이들 사이의 모든 가능한 경로를 분석한다면 그림 4와 같다. 이것은 X-NPE에 주어진 n 개의 데이타가 가질 수 있는 모든 가능한 경로를 모두 보여 주는 것이다. 그 주어진 데이타는 Cn_E 이나 Cn_A로서 표현되고 n 번째 주어진 데이타가 엘리먼트 또는 애트리뷰트가 될 수 있다는 것을 의미한다. 그림에서의 iE (삽입된 엘리먼트)와 iA (삽입된 애트리뷰트)는 삽입되는 데이타를 의미한다. 우리는 이 그림에서 삽입되는 데이타의 값에 대해서는 구조에 영향을 주지 않기 때문에 간단하게 하기 위해 표현하지 않을 것이다. 애트리뷰트는 이것을 포함하는 엘리먼트의 오른쪽에 나타낸다. 경로 이름의 첫 번째 단어는 포함 관계(N) 또는 비포함 관계(X)를 뜻하고, 두 번째 단어는 엘리먼트 표현(E) 또는 애트리뷰트 표현(A)을 나타낸다. 세 번째 단어의 첫 번째 글자는 삽입되는 데이타의 위치로서 상위인 경우(u)와 하위인 경우(d)를 나타내고 두번째 글자는 데이타를 표현하는 방법으로서 엘리먼트(E)와 애트리뷰트(A)를 의미한다.

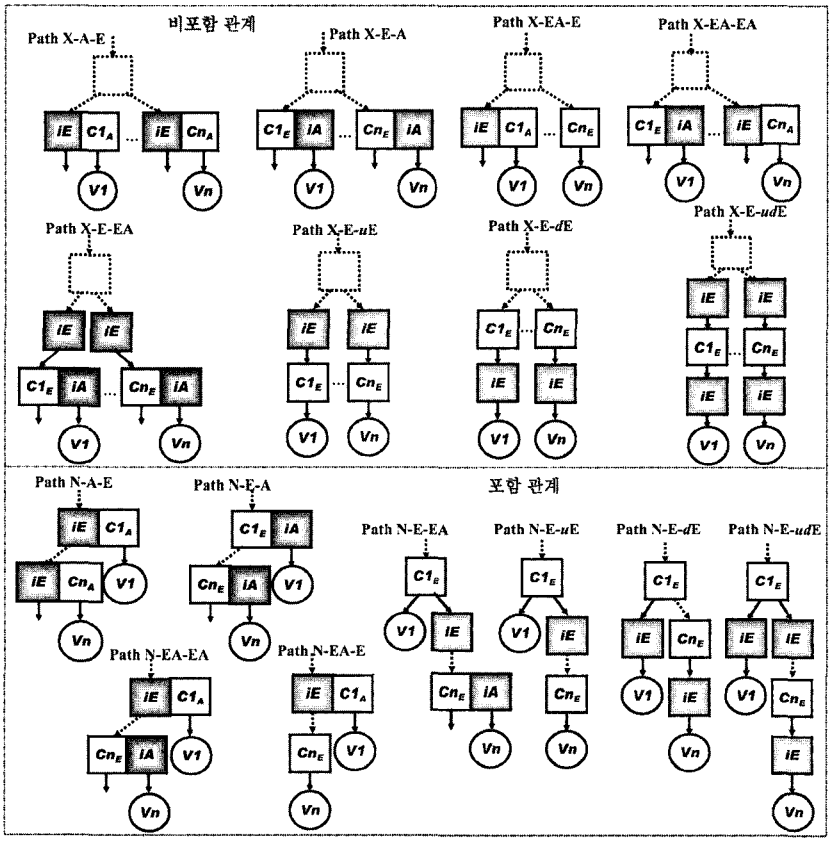
3.3 X-NPE를 처리하기 위해 고려해야 하는 경로 유형

우리는 이러한 가능한 모든 경로 유형 중에서 X-NPE를 이용하여 표현된 질의를 처리할 때 고려해야 하는 경로 유형으로 재 분류하여 다음과 같은 결과를 얻었다.

- 계층적 상하관계 경로 유형 (Type H): X-NPE에 주어진 데이타들에는 계층적인 상하관계를 표현하지 않지만 질의 처리에서는 이를 고려하여야 한다. 그림 1(e)와 (f)는 year와 genre에 대해 서로 계층적인 상하관계가 반대인 경로를 가지고 있다. 그러나 Q1의 질의를 하고자 할 때 예제 3.1의 질의 표현으로 이 두 문서에 대해 모두 질의하여 Leon이라는 결과를 사용자에게 줄 수 있어야 한다(Path N-E, Path N-EA, Path N-A-E, Path N-E-A, Path N-E-EA, Path N-EA-E, Path N-EA-E, Path N-E-uE, Path N-E-dE, Path N-E-udE).
- 주어진 데이타 사이에 임의의 데이타가 삽입된 경로 유형(Type uE): 그림 1(e)와 (f)의 문서에는 genre와 year사이에 모두 country라는 임의의 데이타를 삽입하고 있다. 이렇게 주어진 데이타 사이에 임의의 데이타가 삽입된 것을 X-NPE에 표현하지 않아도 질의 처리를 할 때는 이점을 고려하여야 한다. 이 경로는 기존의 질의 언어에서 정규 경로 표현이라는 것을 통해 해결하고 있다(Path N-E-uE, Path N-E-udE, Path N-E-EA).
- 주어진 데이타 이름과 그 값 사이에 임의의 엘리먼트가 삽입된 경로 유형(Type dE): 그림 1(d)의 문서에서는 year, genre, actor가 모두 그들의 값 사이에 임의의 엘리먼트 이름인 yyyy, type, 그리고name을 삽입하고 있다. X-NPE에 이러한 경로를 표현하지 않아



(a) 조건절에 주어지는 데이타만을 사용하는 경로

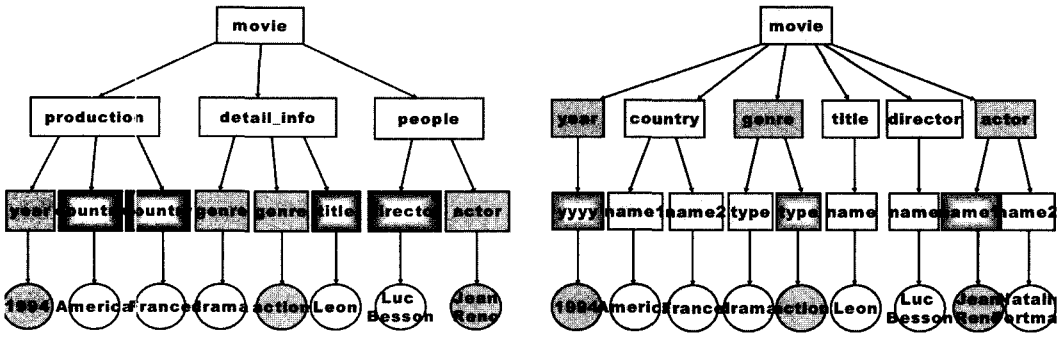


(b) 조건절에 주어지는 데이타 외에 임의로 삽입되는 데이타까지 고려한 경로

그림 4 n 개의 주어진 데이타와 그 값들 사이의 모든 가능한 경로

도 질의 처리를 할 때는 이러한 경로를 고려하여 질의할 수 있어야 한다(Path X-E-*d*E, Path X-E-*ud*E, Path N-E-*d*E, Path N-E-*ud*E).

- 주어진 데이타 이름과 그 값 사이에 임의의 애트리뷰트가 삽입된 경로 유형(Type *i*A): 그림 1(c)의 문서는 year, genre, 그리고 actor와 그들의 값 사이에 모두 임의의 애트리뷰트 이름인 yyyy, type, 그리고 name을 삽입하고 있다. X-NPE에 이러한 경로를 표현하지 않아도 질의 처리를 할 때는 이러한 경로를 고려하여 질의할 수 있어야 한다(Path X-E-A, Path X-EA-EA, Path X-E-EA, Path N-E-A, Path N-E-EA, Path N-EA-EA).
- 애트리뷰트로 표현된 데이타 경로 유형(Type A): 그림 1(b)의 문서에서는 주어진 데이타 year, genre, 그리고 actor가 모두 애트리뷰트로 표현되고 있다. X-NPE에는 엘리먼트와 애트리뷰트의 구분 없이 표현하지만 질의 처리에서는 이점을 고려하여 모두 질의할 수 있도록 하여야 한다(Path X-EA, Path N-



(a) 그림 1(b) 표현 (b) 그림 1(c) 표현

그림 5 애트리뷰트 표현

EA, Path X-A-E, Path X-EA-E, Path X-EA-EA, Path N-A-E, Path N-EA-E, Path N-EA-EA).

- 비포함 관계의 간접 경로 유형(Type N): 그림 1(a)와 같이 year, genre, 그리고 actor 사이의 관계가 비계층 관계의 간접 경로를 가지고 있지만 이들은 Q1 질의에 대해 영화제목 Leon을 검색할 수 있어야 한다 (Path X-E, Path X-EA, Path X-A-E, Path X-E-A, Path X-EA-E, Path X-EA-EA, Path X-E-EA, Path X-E-*u*E, Path X-E-*d*E, Path X-E-*ud*E).

결론적으로 X-NPE는 문서의 계층적인 구조나 데이타 표현 방법, 데이타 사이에 삽입되는 임의의 데이타를 전혀 모르고서도 질의할 수 있도록 하는 것이다. 즉, 경로 기반 질의 언어에서 질의 표현 시 고려해야 하는 6가지 경우의 경로 유형을 모두 일일이 표현하지 않고서도 질의할 수 있도록 하는 것이다. 따라서 기존의 정규 경로 표현이 주어진 데이타 이름 사이에 임의의 데이타 이름을 삽입한 경우(Type *u*E)에 대해 자유로운 표현을 허용하는 것에 비하여 X-NPE는 완전한 문서 구조 독립적인 질의를 가능하게 하는 것이다.

4. X-NPE 질의 처리를 위한 전처리

본 논문에서는 위에서 분류한 유형의 모든 경로에 대해 이러한 구조를 모르고도 질의할 수 있도록 X-NPE를 처리하기 위한 기법을 제안한다. 첫째, 필수 구성 요소가 엘리먼트로 표현되었는지 아니면 애트리뷰트로 표현되었는지의 구분을 두지 않고 애트리뷰트를 포함하고 있는 엘리먼트의 서브엘리먼트로 취급하여 저장한다. 둘째, 필수 구성 요소들 사이의 포함 관계를 알 수 있도록 고유의 식별자(identifier)를 XML 문서에 있는 각 구성 요소들에게 부여한다.

4.1 Types A와 Type iA의 처리

만일 엘리먼트와 애트리뷰트를 같은 방법으로 다룬다면, Type A와 Type *i*A는 자동적으로 해결 가능하다. 예를 들어 Type A를 가정해 보자. 그림 5(a)는 그림 1(b)를 트리로서 표현하는 것인데 애트리뷰트를 엘리먼트와 구분하지 않기 때문에 마치 엘리먼트처럼 똑같이 표현되었다. 이런 원리로 그림 5(b)에서 보는 것처럼 그림 1(c)에서는 year, genre, actor와 그 값 사이에 임의의 애트리뷰트가 삽입되었기 때문에 Type *i*A로 분류되었던 경로가 엘리먼트처럼 취급하여 트리에 표현하니 Type *l*E의 경로처럼 바뀐다.

4.2 노드 식별자(node identifier)

XML 문서를 트리로 모델링하면 각 노드는 XML 문서에서의 구성 요소가 되기 때문에 이 노드들은 엘리먼트 이름, 애트리뷰트 이름, 또는 이들의 값이 된다. 이들 각 구성 요소에 [6,9-10,21] 등에서 사용되었던 텍스트 컴포넌트를 효율적으로 질의할 수 있는 기법으로 알려진 region-numbering scheme이라는 것을 사용한다. 기존 연구들은 이 기법을 대개 정규 경로 표현이라는 것을 위해 사용했지만 우리는 이것을 X-NPE를 처리할 수 있도록 이용한다. 본 논문에서는 문서를 파싱할 때 그림 6과 같이 각 행마다 적당한 간격으로 행번호를 부여한다. 이것을 직관적으로 이해하기 쉽게 트리로 모델링한다면 그림 7과 같이 된다. 이 숫자는 트리의 각 노드를 구분할 수 있고 서로의 관계를 밝힐 수 있는 고유의 식별자가 된다. 또한 XML 문서를 다른 데이타 모델에 적용시킬 때 상호간 구조적인 차이를 보완한다. 즉, XML의 고유 속성인 계층적인 상하관계나 형제 사이의 순서 정보를 나타내는 값을 각 구성 요소에 부여하는 것이기 때문에 XML 문서의 복잡한 계층적 정보를 보존하여 질의할 수 있도록 한다. 또한 이 숫자는 행마다 적당한 간격을 가지고 있기 때문에 나중에 임의 엘리먼트/애트리뷰트를 삽입할 수 있도록 해준다. 이 개념은 초기 BASIC 언어에서의 행번호 의미와 같다. 또한 그

```
10<year yyyy="1994">
30    <country name="America">
50      <movie >
70        <title>Leon</title>
80        <genre type1="drama" type2="action"></genre>
120       <people>
130         <director>Luc Besson</director>
140         <actor>
150           <name>Jean Reno</name>
160           <name>Natalie Portman</name>
170         </actor>
180       </people>190      </movie>
....
500   <country name="France">
    ....
1000</year>
```

그림 6 XML 문서에 행번호 부여 예

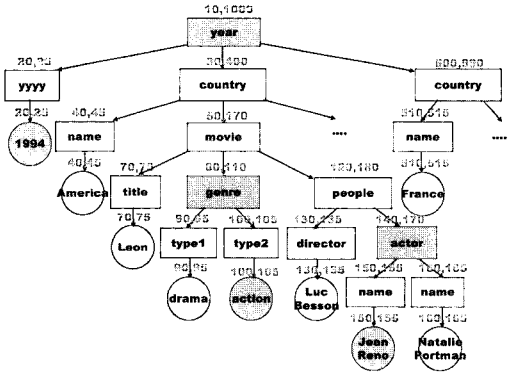


그림 7 그림 6을 트리로 표현

림 7에서 보는 것처럼 엘리먼트/애트리뷰트의 시작과 끝을 나타내는 숫자가 되어 트리에서 한 노드의 영역을 의미하게 된다.

따라서 질의 처리기가 이 노드 식별자의 포함관계를 밝힐 수 있다면 Type H, Type *u*E, Type *l*E, 그리고 Type N에 의한 경로들을 검색할 수 있다. 즉, 그림 7에서 보는 것처럼 genre와 그 값 "action" 사이에 type2라는 애트리뷰트 이름이 삽입되어 있어도(또한 year와 그 값 "1994" 사이에 yyyy) 이것을 엘리먼트와 똑같이 다루어 노드 식별자를 부여한다. 그러면 actor와 그의 값 "Jean Reno" 사이의 경로 유형과 같이 Type *l*E가 된다. 그러면 질의 처리기는 genre의 영역이 "action"의 영역을 포함하는 경우를 검색하면 된다(year의 영역은 "1994"를 포함하고 actor는 "Jean Reno"를 포함하는 영역을 검색). 또한 year와 genre 또는 year와 actor는 Type *u*E의 경로 유형에 해당되는데 이 경우는 genre와 actor의 영역을 포함하는 year의 영역을 검색하도록 한다. X-NPE에 주어진 데이타 사이의 계층적인 상하 관계가 반대인 경우까지 고려하는 Type H의 경로 유형을 검색하기 위해 역시 주어진 두 데이타 사이의 포함 관계를 이용한다. 마지막으로 그림 1(a)처럼 주어진 데이타 genre, year, actor사이의 관계가 서로 비포함 관계를 가지고 있지만 Q1의 질의를 처리하기 위해서는 이들 모두를 만족하는 공통 영역을 계산하여 이것의 후손이 되는 title을 검색하면 된다.

5. 질의 처리

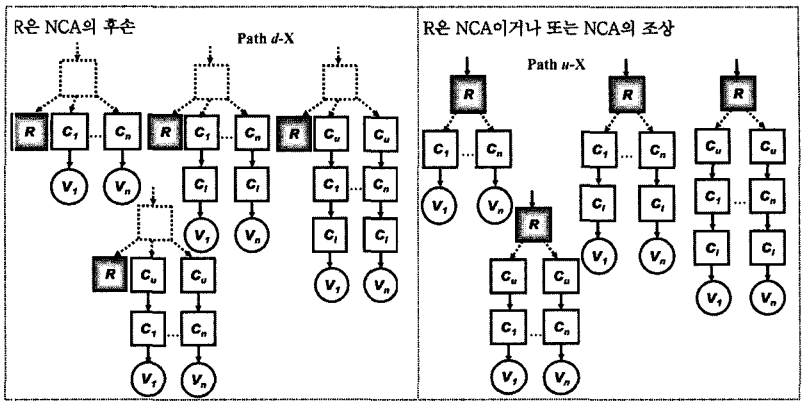
예를 들어 C_1, C_2,…, C_n이 그들의 값 V_1, V_2,…, V_n을 만족하는 R 을 찾고자 한다면 확장된 XQuery 문으로 WHERE C1 = "V1" AND C2 = "V2" AND…Cn = "Vn" RETURN R로 표현할 수 있다. 이것은 조건절에 표현되는 C_1, C_2,…, C_n와 이들의 값 V_1, V_2,…, V_n 사이의 관계 뿐 아니라 결과절에 표현되는 R 사이의 경로도 고려하여 검색해야 함을 의미한다.

5.1 조건절과 결과절 사이의 경로

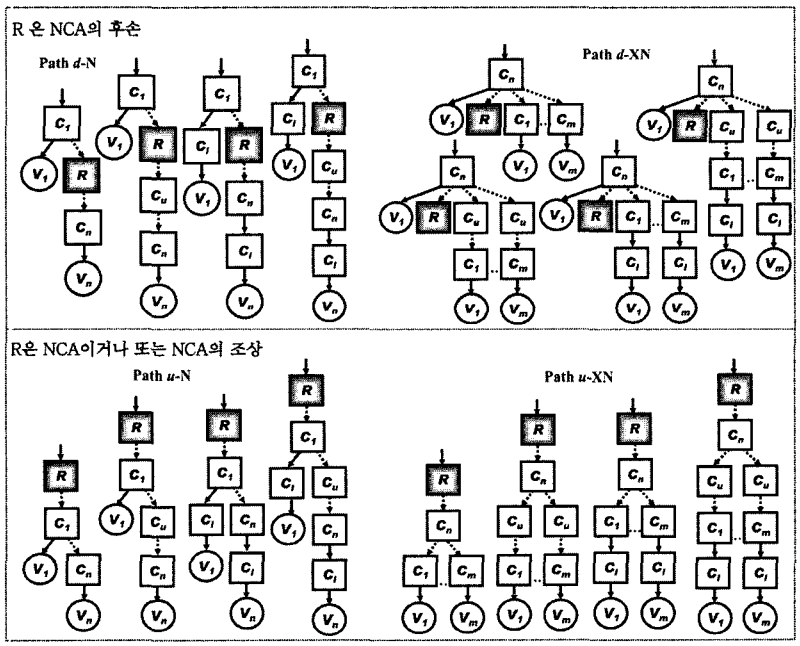
조건절과 결과절 사이의 경로는 그림 8과 같이 분석할 수 있다. 분류 기준은 조건절에 주어진 데이타 구조와 결과절에 주어진 데이타의 위치로 이루어진다. 조건절에 주어진 데이타 구조는 경로 이름의 두 번째 단어로 표현하였고 의미는 포함 관계(N)또는 비포함 관계(X)이다. 결과절에 주어진 데이타 R의 위치는 경로 이름의 첫 번째 단어로 표현하였고 이것의 의미는 가장 가까운 공통 조상(Nearest Common Ancestor)의 상(*u*p) 또는 하(*d*own)이다. 그림 8에서 C_1, C_2,…,C_n은 조건절에 주어진 데이타를 의미하고 R은 결과절에 주어진 데이타를 의미한다.

예를 들어, 그림 6에서 genre가 "action"이고 actor가 "Jean Reno"인 title을 구하고자 한다면 이것은 Path *d*-X이다. 왜냐하면 조건절에 주어지는 genre와 actor는 비포함 관계를 갖고 다시 결과절에 주어지는 title은 조건절과 또 다시 비포함 관계를 갖기 때문이다. 그러나 만일 그림 6의 문서에서 같은 질의 조건에서 결과는 country를 원한다고 하면 이것은 Path *u*-X이다. 왜냐하면 조건절의 데이타들은 비포함 관계이지만 조건절과 결과절의 관계는 포함 관계를 갖기 때문이다. 또한 year가 "1994" 이고 actor는 "Jean Reno"인 country 정보를 검색하고자 한다면 이 경우는 Path *d*-N이다. 왜냐하면 year와 actor는 포함 관계를 갖고 country는 이들의 공통 조상의 후손이기 때문이다. 그리고 country가 "America"이고 actor는 "Jean Reno"인 year 정보를 검색하고 싶다고 한다면 이 경우는 Path *u*-N에 해당하는 유형이다. 왜냐하면 country와 actor는 포함 관계를 갖고 year는 이들의 공통 조상이 되기 때문이다.

5.2 질의 처리 알고리즘



(a) 조건절에 주어진 데이타 사이의 관계 : 비포함 관계



(b) 조건절에 주어진 데이타 사이의 관계 : 포함 관계

그림 8 조건절과 결과절 사이의 가능한 모든 경로

우리는 X-NPE기반 질의를 처리하기 위해 이것이 포함하고 있는 모든 경로를 검색할 수 있는 알고리즘을 정의한다. 기본적인 원리는 앞에서 분석한 X-NPE에 포함되어 있는 모든 경로를 이 질의문에 주어진 데이타 사이의 포함관계를 밝힘으로써 해결한다. 우선 알고리즘 1에서는 X-NPE의 조건절에 주어진 데이타 사이의 포함관계는 그림 2처럼 부여된 노드 식별자 사이의 포함관계를 계산하여 모든 경로 유형들을 검색한다.

우선 알고리즘 1에서는 주어지는 데이타 값에 할당된 영역과 이를 만족하는 데이타 이름의 영역을 먼저 검색한 후, 조건절에 주어지는 두 데이타가 포함 관계를 이룬다면 이 중 상위에 위치하는 데이타의 영역을 계산하고 만일 서로가 비포함 관계이면 두 데이타의 영역을 모두 계산한다.

알고리즘 2에서는 조건절과 결과절 사이의 모든 경로를 만족하는 결과 R을 구한다. 조건절에 주어진 데이타들이 비포함 관계인 경우 알고리즘 1에서 구한 C_1, $C_2, \cdots, C_n$의 영역을 만족하는 가장 가까운 공통 조상(NCA)[26]을 계산한 후 그러한 조상이 결과 R의 영역과 같다면 이것을 결과로 반환한다. 그렇지 않다면

알고리즘 1 조건절의 처리 (WHERE)

```
입력: 질의문에 주어진 n 개의 주어진 데이타 이름(C1, C2,…,Cn)과 그 값(V1, V2,…,Vn)
While (n 개의 값) {
  Vi의 영역을 검색하고
  이 영역 개수 계산하여 nv에 저장
}
While (n 개의 주어진 데이타 이름){
  Ci영역을 검색하고
  이 영역 개수를 계산하여 nc에 저장
}
For ( i=1; i < nv; i++)
  For ( j=1; j < nc; j++)
    {If (Vi 영역을 포함하거나 같은 Cj 가 존재)
      만족하는 Cj 를 저장하고
      그 개수를 계산하여 ncv에 저장
    }
For ( i=1; i < ncv - 1; i++)
  For ( j=i+1; j < ncv; j++)
    { Ci 와 Cj 의 포함관계를 검사
      Ci 와 Cj 공통 영역을 계산
      If (Ci 와 Cj 사이에 포함관계가 존재)
        서로를 포함하는 Ci or Cj 의 영역을 Cr에 저장
      Else
        모든 Ci 영역을 Cr에 저장
    }
```

알고리즘 2 결과절의 처리(RETURN)

```
입력: 알고리즘 1에 의해 계산된 조건절을 만족하는 영역 Cr과 결과절에 주어진 데이타 R
//조건절에 주어진 데이타가 비포함 관계인 경우
  Cr의 가장 가까운 공통 조상(NCA) 계산
  If (NCA가 Cr 중 하나라면) {
      //조건절에 주어진 데이타가 비포함 관계인 경우
        If (NCA 영역이 R영역을 포함한다면) //Path d-X
            R을 만족하는 후손 영역을 결과로서 반환
    If (NCA영역이 R 영역과 같거나 R영역에 포함)
        이러한 R 영역을 반환 //Path u-X
}
  Else {
      //조건절에 주어진 데이타가 포함 관계인 경우
    If (NCA 영역이 R 영역을 포함) //Path u-N, Path u-XN
        계산된 영역 내부에 포함되는 R 영역 반환
    If (NCA영역이 R 영역과 같거나 R영역에 포함)
        이러한 R 영역을 반환 // Path d-N, Path d-XN
  }
```

NCA의 후손에 해당하는 R의 영역을 반환한다. 만일 조건절에 주어진 데이타가 포함 관계를 갖는 경우, R 영역이 이들의 NCA의 후손이라면 조건절에서 계산한 영역에 포함되는 R을 결과로서 반환한다. 만일 R의 영역이 NCA의 조상이라면 조건절에서 계산한 모든 영역을 포함하는 R의 영역을 반환한다. 여기서 NCA의 계산은 [26]에서 제안하는 방법을 사용하며 따라서 질의 처리 시간은 평균적으로 O(n)만큼 걸린다.

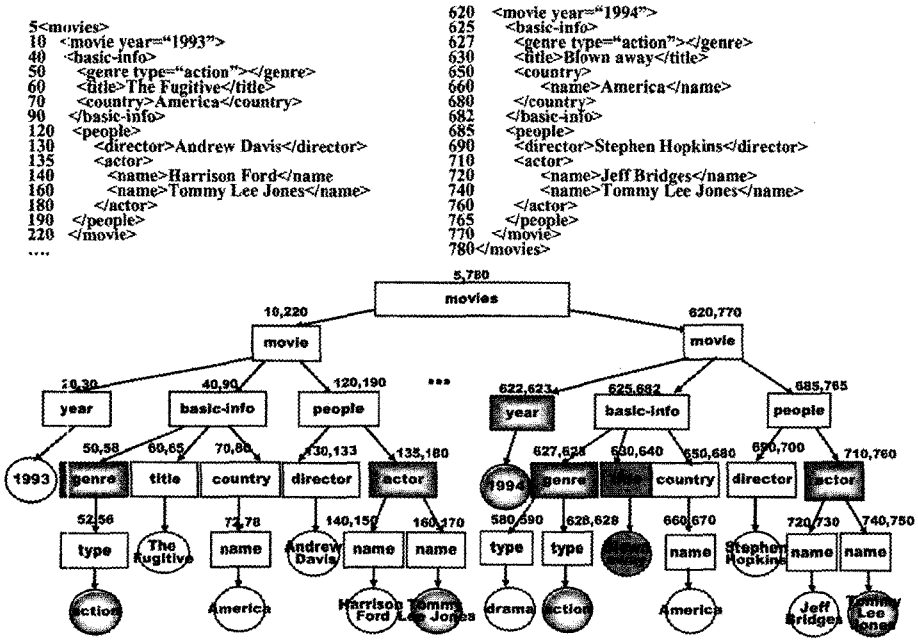
예를 들어 그림 9(a)는 Path d X 문서이고 (b)는 Path d-XN의 문서이다. 이러한 문서들에 대해서, "1994"에 제작된 영화로서 장르는 "action"이며 주연 배우는 "Tommy Lee Jones"인 영화의 제목을 알고 싶다고 하자. 그러면 먼저 질의 처리기는 (a)문서에서 데이타 값이 "Tommy Lee Jones"인 것을 검색(160-170, 740-750)하고, 이 영역을 만족하는 actor의 영역을 검색한다(135 180, 710-760). Year와 genre에 대해서도 같은 방법으로 처리한다. 문서 (b)에 대해서도 같은 방법으로 수행한다.

그 다음으로 검색한 세 영역을 만족하는 영역을 계산한다. (a) 문서의 경우 주어진 데이타 사이에 비포함 관계를 가지고 있기 때문에 이러한 영역이 없다. 따라서 세 데이타의 NCA 영역을 계산한다(620 770). 그리고 결과절에 주어진 title의 영역이 NCA의 후손이기 때문에 이 title의 영역을 결과로서 반환한다(630-640). 그러나 (b)문서의 경우는 주어진 데이타들이 포함 관계를 가지기 때문에 세 영역을 만족하는 공통의 영역을 계산한다(220-790). 그리고 결과절에 주어진 title이 세 데이타를 만족하는 영역을 결과로서 반환한다(630-640).

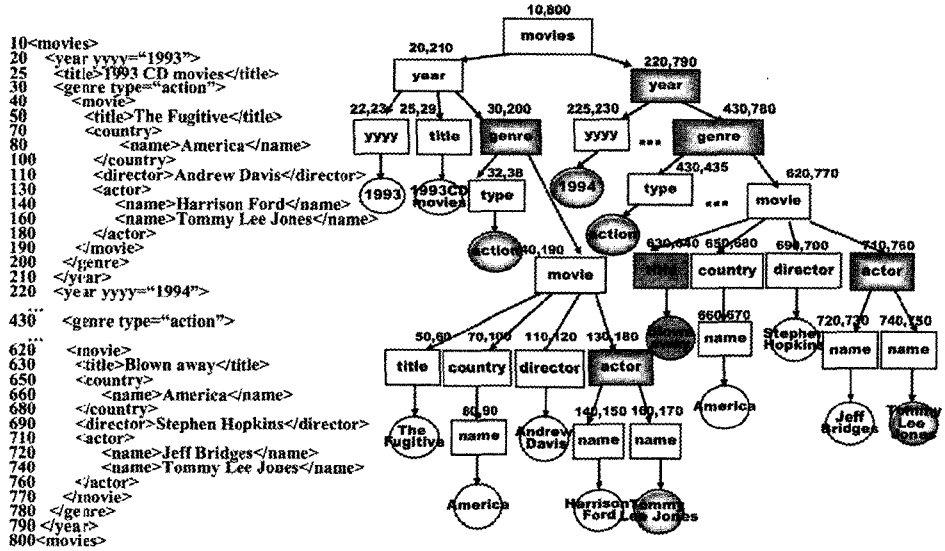
6. 실 험

시스템 사양은 Windows 2000 서버, Microsoft-SQL 서버 2000 데이타베이스, 주 기억 장치 520MB, CPU 속도1.9GHz인 컴퓨터를 사용한다. X-NPE나 경로 기반 질의 처리는 같은 조건 하에서 비교를 해야 공정하기 때문에 같은 저장 기법과 같은 처리 알고리즘을 사용한다. 따라서 두 기법의 속도 차이는 질의 표현이 다르기 때문에 발생한다. 본 논문에서 제안한 트리의 각 노드에 부여되는 노드 식별자와 이것을 처리하기 위한 질의 처리 알고리즘은 (객체)관계형 데이타베이스나 순수 XML 서버 시스템 등, 어떠한 저장 시스템에도 적용할 수 있지만 본 논문에서는 현재 가장 많이 사용되고 있는 관계형 데이타베이스를 하부에 저장 시스템으로 사용하였다. 관계형 데이타베이스 테이블에 저장되는 구조는 다음과 같다.

Element_attribute_name(Doc_ID, Start_region, End_region, Name)



(a) 주어진 데이터 사이에 비포함 관계를 가진 문서

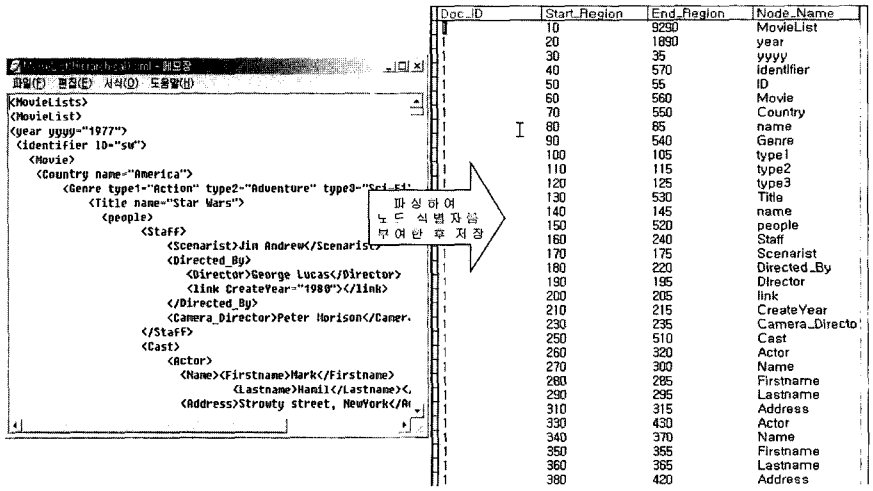


(b) 주어진 데이타 사이에 포함 관계를 가진 문서

그림 9 두 가지 종류의 문서

Element_attribute_value(Doc_ID, Start_region, End_region, Value)

엘리먼트/애트리뷰트 이름을 위한 테이블은 문서 식별자(document identifier), 노드 식별자의 시작 영역(start region) 및 끝 영역(end region), 엘리먼트/애트리뷰트 이름(name)으로 이루어지고 엘리먼트/애트리뷰트 값(value)을 저장하기 위한 테이블은 이름 대신에 그들의 값을 저장하는 필드로 구성된다. 그림 10은 예제 문서를 파싱하여 각 구성 요소에 노드 식별자를 부여한 후 엘리먼트 이름과 애트리뷰트 이름은 노드 테이블에 저장하고 이들의 값은 모두 값 테이블에 저장한 모습을 보여 주고 있다. 문서의 각 행마다 임의의 행번호를 부



Doc_ID	Start_Region	End_Region	Node_Name
1	10	9290	MovieList
1	20	1890	year
1	30	35	yyyy
1	40	570	identifier
1	50	55	ID
1	60	560	Movie
1	70	550	Country
1	80	85	name
1	90	540	Genre
1	100	105	type1
1	110	115	type2
1	120	125	type3
1	130	530	Title
1	140	145	name
1	150	520	people
1	160	240	Staff
1	170	175	Scenarist
1	180	220	Directed_By
1	190	195	Director
1	200	205	link
1	210	215	CreateYear
1	230	235	Camera_Directo
1	250	510	Cast
1	260	320	Actor
1	270	300	Name
1	280	285	Firstname
1	290	295	Lastname
1	310	315	Address
1	330	430	Actor
1	340	370	Name
1	350	355	Firstname
1	360	365	Lastname
1	380	420	Address

그림 10 예제 문서를 파싱하여 노드 테이블에 저장한 모습

여하여 이것이 각 엘리먼트의 노드 식별자가 되도록 한다. 애트리뷰트에 대해서는 애트리뷰트를 포함하고 있는 엘리먼트의 서브엘리먼트가 되도록 노드 식별자를 부여한다. 엘리먼트와 애트리뷰트의 값은 엘리먼트와 애트리뷰트 이름에 부여된 노드 식별자와 같은 값을 할당 받는다. 이 때 할당하는 노드 식별자는 X-NPE에 주어지는 데이타 사이의 포함관계를 한번에 계산할 수 있도록 하는 숫자이기 때문에 X-NPE에 의해 질의하는 모든 경로를 한번에 검색할 수 있도록 한다.

그림 10의 문서는 영화 제작 년도와 identifier에 의해 정렬을 했기 때문에 질의문에 주어지는 데이타 사이에 포함 관계를 갖는 문서이다. 숫자는 10에서 시작하여 한 엘리먼트마다의 간격을 10으로 할당하였고 엘리먼트와 그 값 사이는 5씩 간격을 잡았다. 따라서 여러 서브엘리먼트를 포함하고 있는 엘리먼트는 시작 영역과 끝 영역 사이의 간격이 넓고 값을 가지고 있는 단말 노드의 엘리먼트는 시작 영역과 끝 영역이 5가 차이가 나게 된다.

질의문을 처리하는 단계는 다음과 같다.

- XML문서->파서->노드 식별자 생성기->관계형 데이타베이스
- X-NPE 질의문->파서->SQL변환기->질의처리기->결과 생성기

실험 모듈은 XML 문서를 파싱하여 저장하는 부분과 질의문을 파싱하여 처리하는 부분으로 나누어진다. 파싱하는 동안 XML문서는 노드와 값으로 나누어져 고유 식별자를 부여하고 각각의 테이블에 저장한다. 또한 빠른 처리를 위해 인덱스 테이블을 유지한다. X-NPE 기반 질의는 파서에 의해 조건절에 주어진 데이타와 결과절에 주어진 데이타로 나누어 저장되고 이들은 SQL문으로 변환된다. 그러면 질의 처리기에서 주어진 데이타 사이의 포함관계를 이용하여 결과 영역을 계산한다. 결과 생성기는 반환 받은 결과 영역을 가지고 XML 문서 형식으로 사용자에게 반환한다.

6.1 실험에 사용되는 문서와 질의문

우리는 영화 정보를 포함하고 있는 비계층과 포함 관계가 혼합된 문서에 대해 질의 속도를 평가한다. 예제 문서는 표 1처럼 3장에서 언급한 X-NPE의 모든 경로 유형들을 포함하고 있다. 문서에는 하나의 영화 정보가 최대 32 너비와 15의 높이를 가지고 있다. 총 영화 수, 주어진 데이타 수, 문서 크기는 10개의 문서 전체를 뜻한다.

X-NPE에 주어지는 데이타는 하나인 경우와 두 개인 경우에 대해 평가한다. X-NPE로 genre="action"(XQ1)로 표현되는 질의문은 경로 기반 질의 표현으로 //genre="action"나 //@genre="action"로 해석될 수 있다. 또한 데이타 이름과 그 값 사이에 임의의 데이타 type이 엘

표 1 Movie의 실험 데이타

질의문에 주어진 데이타 사이의 관계	문서에 포함되는 경로 유형	너비	높이	문서 수	총 영화 수	엘리먼트/애트리뷰트 수	문서 크기
비포함 + 포함	Type *i*A, Type *i*E, Type A, Type N, Type H, Type *u*E	32	15	10	10,784	523,885	10.8MB

리먼트로 삽입되는 경우라면 //genre/type="action"로 해석될 수 있고 주어진 데이타와 그 값 사이에 임의의 데이타 type이 애트리뷰트로 삽입되었다면 //genre/@type ="action"처럼 해석될 수 있다. 따라서 XQ1과 다음과 같이 경로 기반 질의 표현으로 가능한 모든 경우를 비교하여 성능 평가를 수행한다.

PQ1: //genre="action"
PQ2: //genre/type="action"
PQ3: //@genre="action"
PQ4: //genre/@type ="action"

또한 계층 또는 비포함 관계의 문서가 혼합된 경우, 두 개의 데이타를 가지고 X-NPE로 genre="action" and actor="Jean Reno" (XQ2)로 표현된 질의문을 경로 기반 질의문으로 표현한다면 가장 간단한 형태는 //genre="action" and //actor="Jean Reno" or //genre="action" //actor="Jean Reno"와 같이 나타낼 수 있다. 그러나 만일 두 개의 데이타가 문서에서 여러 가지 구조와 여러 가지 방법으로 표현이 되었다면 사용자는 문서 구조를 알 수 없기 때문에 PQ5처럼 예측 가능한 모든 경로를 다 포함하도록 복잡한 질의 표현을 사용해야 한다.

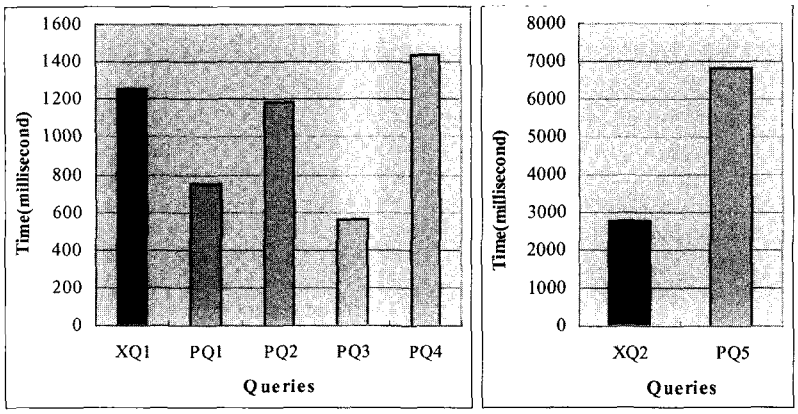
이것은 다양한 구조의 여러 문서에 대해 질의하는 경우 대개가 그 구조를 정확히 모르기 때문이다. 따라서, 만일 사용자가 경로 기반 질의문으로 X-NPE와 같이 이 질의문에 포함된 모든 경로를 만족하는 결과를 얻고자 한다면 비포함 관계와 포함 관계의 문서 모두를 고려하여 질의 표현을 하여야 한다. 또한 포함 관계의 문서를 위하여 주어진 데이타의 위치도 고려하여야 한다. 그러나 X-NPE를 사용한다면, XQ2처럼 그 질의 표현은 문서 구조나 계층적인 관계에 상관없이 간단한 표현을 사용하면 된다.

6.2 성능 평가 결과

질의 시간은 주로 I/O 시간에 영향을 받는다. 그림 11은 X-NPE 질의 속도가 경로 기반 질의 표현으로 가장 간단한 형태보다 약간 느리고 복잡한 구조의 문서에 대해 질의하는 경우보다는 훨씬 빠르다는 것을 보여준다. 즉, 경로 기반 질의는 복잡한 구조의 문서에 대해 여러 가지 경우를 고려하는 경로를 모두 질의문에 표현하기 때문에 데이타를 읽어 들이는 횟수가 많아진다. 따라서 질의 시간이 많이 느려진다는 것을 알 수 있다.

그림 11(a)는 X-NPE에 주어진 데이타가 하나인 경우에 대한 것이고 (b)는 두 개인 경우이다. X-NPE와 경로 기반 질의 모두 질의 시간은 주어진 데이타의 개수와 문서의 크기에만 크게 영향 받고 검색한 데이타를 비교하는 시간이나 문서 구조에는 크게 영향 받지 않는다. 그림 11(a)에서 PQ1, PQ3는 질의 표현에서 보는 것처럼 검색하는 데이타 이름의 수가 하나이지만 PQ2, PQ4는 검색해야 하는 데이타 이름이 두 개이다. 따라서 PQ2, PQ4는 검색하는 속도가 PQ1, PQ3에 비해 현저하게 느리게 된다. 이것은 경로를 모두 알고 질의하는 경로 기반 질의 표현과 비교하여 X-NPE를 사용하는 질의 처리 기법이 크게 나쁘지 않다는 것을 보여 주기

PQ5: //genre="action" and (//actor="Jean Reno" *or* //@actor="Jean Reno" or //actor/*="Jean Reno" or //actor/@*="Jean Reno") or
//@genre="action" and (//actor="Jean Reno" or //@actor="Jean Reno" or //actor/*="Jean Reno" or //actor/@*="Jean Reno") or
//genre//*="action" and (//actor="Jean Reno" or //@actor="Jean Reno" or //actor/*="Jean Reno" or //actor/@*="Jean Reno") or
//genre//@*="action" and (//actor="Jean Reno" or //@actor="Jean Reno" or //actor/*="Jean Reno" or //actor/@*="Jean Reno") or
(//genre="action"//actor="Jean Reno" or //actor="Jean Reno"//genre="action") or
(//genre="action"//@actor="Jean Reno" or //@actor="Jean Reno"//genre="action") or
(//genre="action"//actor//*="Jean Reno" or //actor//*="Jean Reno"//genre="action") or
(//genre="action"//actor//@*="Jean Reno" or //actor//@*="Jean Reno"//genre="action") or
(//genre//*="action"//actor="Jean Reno" or //actor="Jean Reno"//genre//*="action") or
(//genre//*="action"//@actor="Jean Reno" or //@actor="Jean Reno" //genre/*="action") or
(//genre//*="action"//actor//*="Jean Reno" or //actor//*="Jean Reno" //genre//*="action") or
(//genre//*="action"//actor//@*="Jean Reno" or //actor//@*="Jean Reno" //genre//*="action") or
(//@genre="action"//actor="Jean Reno" or //actor="Jean Reno" //@genre="action") or
(//@genre="action"//@actor="Jean Reno" or //@actor="Jean Reno" //@genre="action") or
(//@genre="action"//actor//*="Jean Reno" or //actor//*="Jean Reno" //@genre="action") or
(//@genre="action"//actor//@*="Jean Reno" or //actor/@*="Jean Reno"//@genre="action") or
(//genre//@*="action"//actor="Jean Reno" or //actor="Jean Reno"//genre//@*="action") or
(//genre//@*="action"//@actor="Jean Reno" or //@actor="Jean Reno"//genre//@*="action") or
(//genre//@*="action" //actor//*="Jean Reno" or //actor//*="Jean Reno" //genre//@*="action") or
(//genre//@*="action"//actor//@*="Jean Reno" or //actor//@*="Jean Reno"//genre//@*="action")



(a) 주어진 데이타가 한 개인 경우 (b) 주어진 데이타가 두 개인 경우

그림 11 성능 평가

위한 것이다. 만일 정확한 경로를 모르는 사용자가 경로 기반 질의를 하고자 한다면 PQ1+PQ2+PQ3+PQ4의 질의 표현을 사용해야 하고 질의 처리 시간도 이 모두를 합한 처리 시간이 걸리게 된다. 그러나 X-NPE를 사용하면 이를 위한 질의 처리 시간은 genre와 "action"이라는 데이타를 검색하는 시간과 이들의 관계를 비교하는 시간만이 추가 되기 때문에 PQ1의 시간에 비교 시간만이 더 소요되게 된다. 그림 11(b)는 XQ2가 정확한 경로를 모르는 사용자가 X-NPE를 사용하여 질의한 경우의 질의 처리 시간이라면 PQ5는 정확한 경로를 모르는 사용자가 경로 기반 질의 표현을 사용하는 경우의 질의 처리 시간이다.

X-NPE나 경로 기반 질의에 대한 성능 평가는 비교 시간이나 검색 속도 등, 처리 시간 최적화를 위해 인덱싱 기법 등을 통해 아직 개선할 수 있는 여지가 남아있다고 생각한다.

7. 결론 및 향후 연구

XML 데이타 모델은 비정규적인 문서 구조를 허용하기 때문에 다양한 구조의 여러 가지 문서를 만들 수가 있다. 이것은 사용자가 질의할 때 문서 구조에 대해 보다 많은 지식을 요구하는 원인이 된다. 본 연구에서는 XML을 위하여 문서 구조에 독립적인 질의를 지원할 수 있는 기법을 개발하였다. 이를 위해 먼저 X-NPE라는 질의 표현을 고안하였고 이 질의 표현을 처리하기 위해 데이타들 사이에 가능한 모든 경로를 분류하고 이를 검색할 수 있는 기법을 개발하였다. 향후에 우리는 이 질의 처리 기법에 대한 최적화된 방법을 연구하고자 한다. 또한 새로운 데이타 삽입이나 삭제와 같이 문서를 수정하는 경우에 대해서도 X-NPE 질의 표현에 대한 처리 기법의 성능 평가를 수행하고 이것의 개선 방향을 모색할 것이다.

참 고 문 헌

[1] W3C Consortium, XML1.0(Second Edition), W3C Recommendation 06 Oct. 2000, available at http://www.w3.org/TR/REC-xml

[2] S. Adler, A. Berglund, J. Caruso, S. Deach, T. Graham, P. Grosso, E. Gutentag, A. Milowski, S. Parnell, J. Richman, and S. Zilles, Extensible Stylesheet Language (XSL) Version 1.0, W3C Proposed Recommendation Aug. 2001, available at http://www.w3.org/TR/xsl/

[3] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. L. Wiener, The Lorel Query Language for Semistructured Data, International Journal on Digital Libraries, Apr. 1997.

[4] S. Boag, D. Chamberlin, M. F. Fernandez, D. Florescu, J. Robie, and J. Siméon, XQuery 1.0: An XML Query Language, W3C Working Draft 16 Aug. 2002, available at http://www.w3.org/TR/xquery/

[5] W3C Consortium, XML Path Language(XPath) Version 1.0, W3C Recommendation 16 Nov. 1999, available at http://www.w3.org/TR/xpath.html

[6] Q. Chen, A. Lim, and K. W. Ong, D(k)-Index: An Adaptive Structural Summary for Graph-Structured Data, Proc. of the ACM SIGMOD Int. Conf. on Management of Data, 2003.

[7] Q. Li and B.Moon, Indexing and Querying XML data for Regular path expressions, Proc. of the VLDB Conf, 2001.

[8] M. Fernandez and D. Suciu, Optimizing Regular path expressions Using Graph Schemas, Proc. of Int. Conf. on Data Engineering, 1998.

[9] Haifeng Jian, Honggjun Lu, and Wei Wang, XR-Tree: Indexing XML data for efficient struc-

tural joins, Proc. of Int. Conf. on Data Engineering, 2003.
[10] Shu-Yao Chien, Zografoula Vagena, Donghui Zhang, Vassillis J. Tsotras, and Carlo Zaniolo, Efficient structural joins on indexed XML documents, Proc. of the VLDB Conf, 2002.
[11] Elisa Bertino and Won Kim, Indexing techniques for queries on nested objects, IEEE Transactions on knowledge and data engineering, Vol. 1, No. 2, June 1989.
[12] D. Florescu, D. Kossmann, and I. Manolescu, Integrating keyword search into XML query processing, Proc. of the 9th int. World Wide Web Conf. on Computer networks, 1999.
[13] D. Carmel, Y. S. Maarek, M. Mandelbrod, Y. Mass, and A. Soffer, Searching XML documents via XML fragments, Proc. of the 26th Int. ACM SIGIR Conf., 2003.
[14] S. Banerjee, Oracle XML DB, Oracle Corporation Technical White Paper Release 9.2, Jan. 2002.
[15] IBM Corporation, DB2 XML Extender, IBM Corporation, 2000, available at http://www-4.ibm.com/
[16] 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
[17] Elisa Bertino and Won Kim, Indexing techniques for queries on nested objects, IEEE Transactions on knowledge and data engineering, Vol. 1, No. 2, June 1989.
[18] Milo and D. Suciu, Index Structures for path expressions Proc. of the IEEE Int. Conf. on Data Theory, 1997.
[19] H. Wang, S. Park, W. Fan, and P. S. Yu, Vist: A Dynamic Index Method for Querying XML Data by Tree Structures, Proc. of the ACM SIGMOD Int. Conf. on Management of Data, 2003.
[20] S. Al-Khalifa, C. Yu, and H. V. Jagadish, Querying Structured Text in an XML Database, Proc. of the ACM SIGMOD Int. Conf. on Management of Data, 2003.
[21] M. P. Consens and T. Milo, Algebra for querying text regions: expressive power and optimization, Journal of computer and system sciences, 1998.
[22] Masatoshi Yoshikawa, Toshiyuki Amagasa, Takeyuki Shimura, and Shunsuke Uemura, XRel: A path-based approach to storage and retrieval of XML documents using relational databases, ACM Transaction on Internet Technology, Vol. 1, No. 1, Aug.2001, Pages 110-141.
[23] V. Hristidis, Y Papakonstantinou, and A. Balmin, Keyword proximity search on XML graph, Proc. of Int. Conf. on Data Engineering, 2003.
[24] L. Guo, F. Shao, C. Botev, J. Shanmugasundaram, XRANK: Ranked keyword search over XML documents, Proc. of the ACM SIGMOD Int. Conf. on Management of Data, 2003.
[25] S. Cohen, J. Mamou, Y. Kanza, and Y. Sagiv, XSEarch: A semantic search engine for XML, Proc. of the VLDB Conf., 2003.
[26] Dov Harel and Robert Endre Tarjan, Fast algorithms for finding nearest common ancestors, SIAM Journal on Computing, v.13 n.2, p.338-355, May 1984.

이 월 영
1988년 2월 이화여자대학교 전산학과 졸업. 2000년 2월 이화여자대학교 대학원 컴퓨터학과 석사학위 취득. 2003년~현재 이화여자대학교 대학원 컴퓨터학과 박사학위과정 중. 관심분야는 Internet/Web 기반 데이타 처리

용 환 승
1983년 2월 서울대학교 컴퓨터공학과 졸업. 1985년 2월 서울대학교 대학원 컴퓨터공학과 석사학위 취득. 1994년 2월 서울대학교 대학원 컴퓨터공학과 박사학위 취득. 1995년~현재 이화여자대학교 컴퓨터학과 교수로 재직 중. 관심분야는 Internet/Web 기반 멀티미디어 데이타베이스 시스템