XML 문서에서의 레이블 경로 발생 빈도수에 따른 스키마 추출 방법

The Schema Extraction Method using the Frequency of Label Path in XML documents

김 성 림* 윤 용 익**
Sung-Rim Kim Yong-Ik Yoon

요 약

인터넷상에서 데이터를 표현하고 교환하는 새로운 표준으로 등장하는 XML 문서는 정해진 스키마를 가지고 있지 않다. XML 문서를 기존의 관계형 데이터베이스나 객체 지향 데이터베이스 질의어에 바로 적용하기에는 부적합하여 이러한 XML 문서에 대해 스키마를 추출하는 방법과 질의어에 대한 연구가 활발히 진행되고 있다. 본 논문에서는 XML문서의 레이블 경로 발생 빈도수에 따른 여러 단계의 스키마를 추출하는 방법을 제시하고, 이를 실험하여 그 효율성을 보인다.

Abstract

XML documents found over internet are generally fairly irregular and have no fixed schema. The SQL and OQL are not suitable for query processing in XML documents. So, there are many researches about schema extraction and query language for XML documents. We propose a schema extraction method using the frequency of label path in XML documents. Our proposed method produces multi-level schemas and those are useful for query processing.

1. 서 론

XML(eXtended Markup Language)는 인터넷상에서 데이터를 표현하고 교환하는 새로운 표준으로 등장하고 있다[1]. HTML과 마찬가지로 XML은 SGML의 부분집합이지만 HTML 태그가 데이터 표현에 중점을 둔 것이라면, XML 태그는 데이터 자체를 기술한다. 따라서 XML은 자기 서술적인 특징(self-describing)을 바탕으로 XML 문서를 여러 형태로 보여줄 수 있고, 내용을 기반으로 데이터를 필터링하거나 어플리케이션의 목적에 맞게 재구성이 가능하다.

XML 문서는 데이터 구조를 나타내는 중첩된 태그 엘리먼트의 집합으로 구성되어있기 때문에 기존의 데이터베이스에서처럼 정해진 스키마를 갖고 있지 않지만 문서마다 어떤 구조(Document Type Definition : DTD)를 갖고 있다고 볼 수 있다. XML의 데이터 모델은 구조상 기존의 데이터베이스와 많은 차이점이 있고, 또한 SQL이나 OQL을 바로 적용하기에 부적합하다. 따라서 이러한 XML 문서들에 대해 스키마를 추출하는 방법과 질의어에 대한 연구가 활발히 진행되고 있다[5,8,9,11,21].

본 논문에서는 XML 문서에 대해 스키마를 추출하고, 추출된 스키마를 바탕으로 각 XML 문서에서의 레이블 경로의 발생 빈도 수를 구하고, 어떤 임계치에 따라 새로운 여러 단계의 스키마를 추출하는 방법을 제시하고자 한다. 이러한 스키마 추출 방법은 사용자 질의에 대해 질의 수행 결과가 너무 적거나 많을 때 여러 단계의 스키마에 적용해서 질의를 수행함으로써 질의 범위를 축소

* 정회원 : 숙명여자대학교 정보학부 컴퓨터학과 박사과정
srkim@cs.sookmyung.ac.kr
** 정회원 : 숙명여자대학교 정보학부 멀티미디어학과 교수
yiyoon@sookmyung.ac.kr

혹은 확장을 가능하도록 하기 때문에 사용자의 요구를 효율적으로 반영할 수 있게 한다.

본 논문의 구성은 다음과 같다. 2장에서는 스키마 추출에 대한 관련 연구를 살펴보고, 3장에서는 본 논문의 배경이 되는 이론들을 서술한다. 4장에서는 본 논문에서 제안하고자 하는 스키마를 추출하는 알고리즘과 예제를 보이고, 5장에서 스키마 추출을 위해 제안한 방법을 실험한 내용을 설명하고, 6장에서 결론을 맺는다.

2. 관련 연구

본 장에서는 발생 빈도에 따라 스키마를 추출하는 기존의 방법에 대해서 살펴본 뒤, 기존 연구의 문제점을 설명한다.

2.1 트리 표현식의 발생 빈도수에 따른 스키마 추출

트리 표현식들의 발생 빈도에 따라 최대의 트리 표현식으로 공통적인 스키마를 추출하는 방법이 있다 [12,13].

트리 표현식에서의 스키마 추출은 다음과 같이 이루어진다. 트리 표현식(*te* : tree expression) 에서 *te*의 지지도(support : MINISUP)는 도큐먼트 *d* 보다 표현식이 약한 *te*를 갖는 도큐먼트의 개수이다. 사용자가 정의한 최소 지지도 MINISUP 에 대해, *te*의 지지도가 크다면, *te*가 빈도수가 높다고(frequent) 볼 수 있다. *te*의 빈도수가 높고, 다른 어떤 트리 표현식보다 빈도수가 높다면, 이러한 *te*는 최대 발생 빈도수(maximally frequent)를 갖는다고 한다.

자주 발생되는 비슷한 질의에 대해서 트리 표현식을 만들고, 이를 바탕으로 스키마를 추출함으로써 유사한 질의에 대해 효율적으로 실행할 수 있다는 장점이 있지만 문서 전체에 대한 스키마를 찾기 힘들다는 제약이 있다.

2.2 발생 빈도 패턴 트리

발생 빈도 패턴을 찾는 방법은 트랜잭션 데이터베이스, 시계열 데이터베이스등 많은 데이터베이스 분야에서 연구되어 왔다. 여러 방법 중에서 발생 빈도 패턴 트리(FP-tree : Frequent Pattern Tree)를 구축하여 최대 패턴을 구하는 방법이 제시되었다 [3].

FP-tree에서는 발생 빈도 패턴에 대한 정보를 저장하고, 이를 이용하여 조건 FP-tree를 형성함으로써 빈도 패턴을 찾아내는데 보다 효율성을 증가시켰다.

사용자 정의 발생 빈도 수를 바탕으로 스키마가 다양하게 추출될 수 있다는 장점은 있지만 전체 스키마보다는 특정 패턴에 대한 스키마가 추출이 되는 제약과 특정 패턴과 발생 빈도 수에 대한 사용자 정의 값에 대해 매번 스키마 추출 단계를 반복해야 한다는 제약점이 있다.

2.3 Lore 시스템의 DataGuide

Lore는 스탠포드대학에서 개발한 XML을 위한 데이터베이스 관리 시스템이다[17,18]. 1995년 처음 개발될 때는 반구조적 데이터를 관리하기 위한 시스템이었지만 이후 XML 개념이 도입되면서 XML을 위한 시스템으로 발전되었다.

DataGuide는 XML 데이터베이스에 대해 정확하고, 동적으로 정리된 구조를 표현해줌으로써 데이터베이스 스키마나 DTD 역할을 수행하게 된다. 사용자는 DataGuide를 통해 데이터베이스의 전체적인 구조를 파악하여 질의를 만들 수 있게 된다.

DataGuide는 모든 문서에 대한 전체적인 스키마를 파악할 수 있다는 장점이 있지만 모든 문서에 있는 데이터가 표현됨으로써 생성되는 스키마의 범위가 최대가 되고, 따라서 질의 수행을 위한 검색 범위가 넓어질 수 있다는 제약점이 있다.

3. 모델링

본 장에서는 본 논문에서 제안하는 스키마 추출 방법에 필요한 기본 개념을 살펴보고, 몇 가지 정의를 설명한다.

3.1 XML 문서의 특징

인터넷상에서 데이터를 교환하는 표준이 되어 가고 있는 XML은 W3C에서 제안한 XML 규격 1.0 문법 형식을 따른다[19,20]. HTML은 사용이 간단하고 쉽기는 하지만 확장성, 구조성, 데이터의 검사기능에 있어서 한계를 가지고 있다. 이러한 HTML의 단점을 극복하고 문서의 표준화와 이의 대중화에 적합한 형태로 SGML을 새롭게 정의하고 간결화시킨 것이 바로 XML이다[1]. HTML과는 달리, XML은 데이터를 표현하는 방법은 제공하지 않고, 데이터를 사용자에게 표현하는 기능은 스타일시트에서 따로하게 된다. XSL(XML Stylesheet Language)이라는 명세어에 있는 스타일시트는 XML 데이터를 HTML 문서로 변환하는데 사용된다. 이렇게 만들어진 HTML 문서는 일반적인 웹 브라우저에서 볼 수 있다. 이렇듯 XML은 단지 데이터를 전달하기 위한 문법이고, 웹 상에서 데이터를 교환하는 표준이 되고 있다.

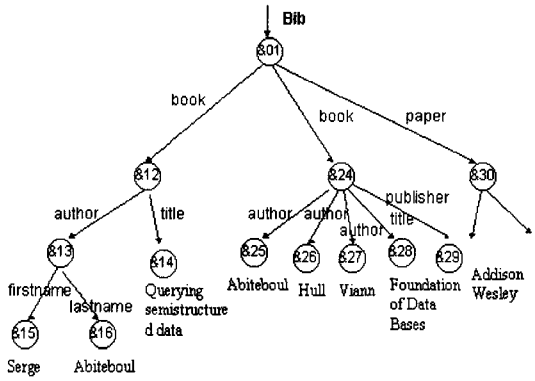
3.2 Edge Labeled Graph

XML 문서를 반구조적 데이터(semistructured data)처럼 방향성있는 edge-labeled graph로 표현할 수 있다[2,21]. 다음은 XML 문서와 이를 그래프로 표현한 것이다.

Edge-labeled graph에서 엘리먼트는 객체(노드)로 표현되고, 각 객체는 &01같은 객체 식별자 oid (object identifier)를 갖는다.

Edge-labeled graph에서는 두 개의 객체-단순 객체와 복합 객체-를 정의한다. 단순 객체는 정수, 실수,

```
<Bib>
  <paper id="o12" references="o24">
    <author>
        <firstname>Serge</firstname>
        <lastname>Abiteboul</lastname>
    </author>
    <title>Querying semistructured data</title>
  </paper>
  <book id="o24">
    <author>Abiteboul</author>
      <author>Hull</author>
    <author>Viann</author>
    <title>Foundation of Data Bases</title>
    <publisher>Addison Wesley</publisher>
  </book>
  <paper id="o30" references="o12, o24">
                .....
  </paper>
</Bib>
```



(그림 1) Edge Labeled Graph

이미지, html 문서등의 타입을 갖는 데이터를 갖는 객체로써 더 이상 그 객체로부터 나가는 간선을 갖지 않는 객체이다. 복합 객체는 단순 객체 또는 복합 객체로 구성된 객체로써 그 객체로부터 나가는 간선을 갖는 객체를 뜻한다. 예를 들어 그림 1에서 &14는 단순 객체이고, &13은 복합 객체이다.

Edge-labeled graph에서는 객체들간에 간선이 존재하고, 각 간선마다 엘리먼트 이름으로 레이블이

있고, 서브 엘리먼트를 표현하는 방향성을 갖는다.

3.3 스키마 추출을 위한 그래프

스키마 추출을 위하여 두 개의 그래프를 다음과 같이 정의한다.

• 정의 1 : 데이터 그래프(Data Graph)

XML 문서의 모든 데이터가 표현되는 edge labeled directed graph를 '데이터 그래프(Data Graph)'라고 정의한다. 루트 노드로부터 하위노드로 방향성 있는 간선이 만들어지고, 간선의 레이블은 엘리먼트의 이름이 된다. 그리고 각 노드는 oid를 갖고, 노드는 단순 객체 또는 복합 객체의 형태를 갖는다.

• 정의 2 : 스키마 그래프(Schema Graph)

XML 문서에 대한 데이터 그래프에서 깊이 우선 탐색 기법을 바탕으로 모든 경로가 단 한번만 표현될 수 있도록 만들어진 그래프를 '스키마 그래프(Schema Graph)'라고 정의한다. Lore 시스템[6]의 DataGuide[2]처럼 스키마 그래프에서는 모든 레이블 경로가 유일하고(concise), XML 문서에 있는 모든 데이터는 표현되어야 하고(accuracy), 각 노드의 구성이 어떻게 되어있는지(convenience) 알 수 있도록 한다.

3.4 비트맵 인덱싱을 이용한 레이블 경로 인덱싱

비트맵 인덱싱의 기본 개념은 튜플에 있는 애트리뷰트가 어떤 특별한 값을 갖고 있느냐 없느냐를 나타내기 위해 0 혹은 1의 비트로 표현하는 것이다[14].

비트맵 인덱스 기법은 Oracle, Informix, IBM등의 몇몇 상용화 DBMS에서 구현되어 있다. 비트맵 인덱싱의 장점은 bitwise-AND, OR, NOT 연산이 하드웨어적으로 가능함으로써 수행속도가 빨라질 수 있다는 것이다. 이러한 장점으로 의사결정 시스템(Decision Support System), 데이터웨어하우징에서 많이 사용되고 있다. 그리고 적은 카디날리티를 갖는 경우 적은 공간을 차지함으로써 효율적이 될 수 있다[4,7,10].

본 논문에서는 아래와 같이 정의한 XML 문서에서의 레이블 경로를 비트맵 인덱싱하여 보다 유동적인 스키마 그래프를 생성한다.

• 정의 3 : 레이블 경로(Label Path)

데이터 그래프 혹은 스키마 그래프에서 한 노드에서 어떤 하위 노드로의 경로를 '레이블 경로(Label Path)' 라고 정의한다. 그래프에서 노드와 노드사이에는 간선이 존재하고, 간선은 엘리먼트 이름으로 레이블이 존재한다. 그리고 루트 노드에서 리프 노드까지의 경로에서 나타나는 중간 노드는 . (dot)을 사용하여 표현한다.

레이블 경로는 스키마 그래프에서 루트 노드에서 리프 노드까지 깊이 우선 탐색 기법을 이용하여 단 하나씩 구하게 된다. 이러한 비트맵 인덱스 기법을 스키마 그래프에서 레이블 경로에 적용하였다[16]. 스키마 그래프에서 구해진 레이블 경로를 각 XML 문서에 적용하여 레이블 경로의 존재 여부를 비트로 표현하게 된다. 즉, 레이블 경로가 해당 XML 문서에 존재하면 1의 값을, 존재하지 않으면 0의 값을 갖는다. 이렇게 모든 XML 문서들에 의해 구해진 비트맵 스트링으로 각 비트의 빈도 수를 구하게 된다. 이는 레이블 경로의 발생 빈도 수에 따라 여러 단계의 스키마 그래프를 구하기 위함이다. 모든 XML 문서의 비트맵 스트링에 대해 bitwise-OR 연산을 수행하게 되면 모든 레이블 경로를 포함하게 되는 스키마 그래프를 구하게 될 것이다. 만약 bitwise-AND를 하게 되면 모든 XML 문서에 공통적으로만 존재하는 레이블 경로로만 구성된 스키마 그래프를 구하게 될 것이다. Bitwise-OR를 통해 구하게 된 스키마 그래프의 경우는 모든 XML 문서를 표현할 수 있기 때문에 질의 수행시 질의를 수행할게 될 XML 문서의 범위를 가장 넓게 되고, bitwise-AND를 이용

한 스키마 그래프의 경우는 그 질의 범위가 최소화 될 것이다.

4. 스키마 추출

본 장에서는 3장에서 설명한 기본 개념을 바탕으로 영화에 대한 정보를 표현하는 DTD와 XML 문서를 가지고 예를 들어 설명한다.

4.1 예제 DTD와 XML 문서

영화 정보에 대한 내용이 있는 http://www.imdb.com/에서 표 1과 같은 DTD와 이를 바탕으로 생성한 XML문서를 가정한다[22].

영화에 대한 정보는 제목, 제작년도, 감독, 시나리오 작가, 장르, 배역, 언어, 나라에 대한 정보로 구성되어 있다. 감독, 시나리오 작가, 장르, 배역은 한 사람이거나 그 이상이 될 수 있고(+), 언어나 나라에 대한 정보는 없거나 있으면 하나의 정보를 갖는다(*). 그리고 감독과 시나리오 작가의 이름은 성, 이름이 따로 표현되거나 아니면 한꺼번에 표현된다(|). 그리고 배역의 경우 배우의 이름과 극중 인물의 정보를 표현한다.

표 1의 DTD를 바탕으로 몇 개의 XML 문서를 만들어 보면 그림 2와 같다.

(표 1) DTD 예제 : movie.dtd

```
<!ELEMENT movie  (title, year, director+,
                  writer+, genre+, cast+,
                  language*, country* ) >
<!ELEMENT title (#PCDATA) >
<!ELEMENT year (#PCDATA) >
<!ELEMENT director ((lastname, firstname) |
                    fullname)>
<!ELEMENT writer ((lastname, firstname) |
                  fullname)>
<!ELEMENT lastname (#PCDATA) >
<!ELEMENT firstname (#PCDATA) >
<!ELEMENT fullname (#PCDATA) >
<!ELEMENT genre (#PCDATA) >
<!ELEMENT cast (name,role) >
<!ELEMENT name (#PCDATA) >
<!ELEMENT role (#PCDATA) >
<!ELEMENT language (#PCDATA) >
<!ELEMENT country (#PCDATA) >
```

- XML 문서 : d1.xml

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE MOVIE SYSTEM "movie.dtd">
<movie>
  <title>SixthSense</title><year>1999</year><director>M.Night Shyamalan</director>
  <writer>M.Night Shyamalan</writer>
  <genre>Thriller</genre><genre>Drama</genre><genre>Horror</genre>
  <cast>
    <name>Bruce Willis</name><role>Malcolm Crowe</role>
    <name>Haley Joel Osment</name><role>Cole Sear</role>
  </cast>
  <language>English</language><language>Spanish</language><language>Latin</language>
</movie>
```

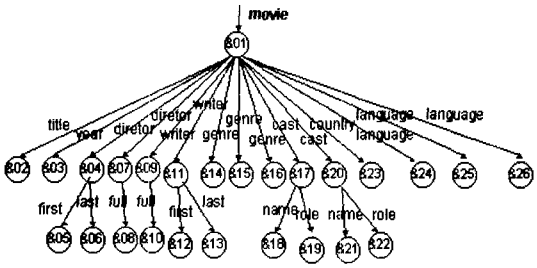
- XML 문서 : d2.xml

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE MOVIE SYSTEM "movie.dtd">
<movie>
  <title>Matrix</title><year>1999</year><director>Andy Wachowski </director>
  <director>Larry Wachowski </director>
  <writer>Andy Wachowski</writer><writer>Larry Wachowski</writer>
  <genre>Action</genre><genre>Thriller</genre><genre>Sci-Fi</genre>
  <cast><name>Keanu Reeves</name><role>Thomas A. Anderson/Neo</role></cast>
  <language>English</language><country>USA</country>
</movie>
```

(그림 2) 영화 정보에 대한 예제 XML문서 -1

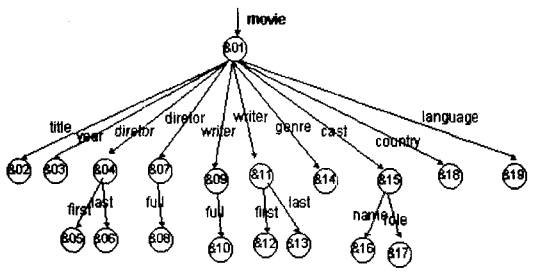
4.2 데이터 그래프와 스키마 그래프

그림 2의 예제 XML 문서를 데이터 그래프로 표현하면 그림 3과 같다. 3.3에서 정의한바와 같이 그림 3의 데이터 그래프는 예제 XML 문서에 있는 모든 엘리먼트가 표현되어 있다.



(그림 3) XML 문서 (d1,d2)에 대한 데이터 그래프

그림 2의 예제 XML 문서를 스키마 그래프로 표현하면 그림 4와 같다. 3.3에서 정의한바와 같이 그림 4의 스키마 그래프는 모든 레이블 경로가 단 한번씩만 표현되게 된다. 데이터 그래프에서 깊이 우선 탐색 기법을 이용하여 모든 레이블 경로가 단 한번만 존재하게 하고, XML 문서의 엘리먼트는 반드시 나타날 수 있도록 하였다.



(그림 4) XML 문서(d1,d2)에 대한 스키마 그래프

4.3 비트맵 인덱싱을 이용한 스키마 추출

XML문서를 바탕으로 3장에서 정의한 스키마 그래프를 생성한 후 레이블 경로를 구하고, 각 XML 문서에서의 레이블 경로의 존재여부를 비트 벡트로 표현한 후 그 빈도 수에 따라 다양한 스키마 그래프를 재생성 할 수 있다.

4.3.1 레이블 경로

스키마 그래프에 대해 레이블 경로는 루트 노드부터 리프 노드까지 깊이 우선 탐색 기법을 바탕으로 중간노드의 레이블이 레이블 경로에 추가되는데 그 방법은 알고리즘 1과 같다. 그림 4의 스키마 그래프에 대해 레이블 경로 구하는 알고리즘을 적용한 결과는 표 2와 같다.

비트맵 인덱스 기법을 그림 4의 스키마 그래프에서 레이블 경로에 적용하면 각 XML 문서에서의 레이블 경로가 표 2에서 구해진 레이블 경로에 존재하면 1의 값을, 존재하지 않으면 0의 값을 갖는다. 이는 알고리즘 2와 같고, 그림 4의 스키마 그래프에 대해 레이블 경로와 비트 벡터는 표 3과 같다. *pi*는 레이블 경로이고, *di*는 XML 문서이다.

알고리즘 1. 레이블 경로 구하기

```
MakeLabelPath (treeNode u)
 begin
    path = u.label
    while (u isNot leafNode)
     begin
        v = u.childNode
        path = path + v.label
       u = v
     end
 end
```

(표 2) 그림 4에 대한 레이블 경로

레이블 경로명	레이블 경로
p1	movie.title
p2	movie.year
p3	movie.director.firstname
p4	movie.director.lastname
p5	movie.fullname
p6	movie.writer.firstname
p7	movie.writer.lastname
p8	movie.writer.fullname
p9	movie.genre
p10	movie.cast.name
p11	movie.cast.role
p12	movie.country
p13	movie.language

알고리즘 2. 레이블 경로 인덱싱

```
CalcBitVector(document d)
 begin
    while (pi isNot EOF)
          if (pi in d)
              d(pi) = 1
          else
              d(pi) = 0
          endif
 end
```

(표 3) 스키마 그래프에 대한 레이블 경로와 비트 벡터

	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	p13
d1	1	1	0	0	1	0	0	1	1	1	1	0	1
d2	1	1	0	0	1	0	0	1	1	1	1	1	1
d3	1	1	1	1	0	0	0	1	1	1	1	0	0
d4	1	1	1	1	0	1	1	0	1	1	1	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...

4.3.2 레이블 경로 빈도 수를 이용한 스키마 추출

모든 XML 문서에 대한 비트맵 인덱스에 대해 Bitwise-OR 연산을 수행하면 모든 레이블 경로를 포함되는 스키마 그래프가 생성되고, 이 그래프는 데이터 그래프에서 생성되는 스키마 그래프와 동일하다. 그 알고리즘은 3과 같다.

알고리즘 3. 레이블 경로에 대한 bitwise-OR 연산

```
bitwiseORLabelPath()
begin
   for each i in path P
      new_path(i) = bitwise_OR(∑djPi)
end
```

(표 4) XML 문서의 bitwise-OR 연산

	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	p13
d1	1	1	0	0	1	0	0	1	1	1	1	0	1
d2	1	1	0	0	1	0	0	1	1	1	1	1	1
d3	1	1	1	1	0	0	0	1	1	1	1	0	0
d4	1	1	1	1	0	1	1	0	1	1	1	1	1
Bitwise-OR													
	1	1	1	1	1	1	1	1	1	1	1	1	1

레이블 경로에 대한 비트맵 인덱스를 구한 결과, 예제 XML 문서 *dl*은 1100100111101, *d2*는 11001001111111의 비트 스트링을 얻게 된다. 모든 문서에 대해 *p1*은 1111, *p2*는 1111, *p3*는 0011등의 값을 갖는다.

각 레이블 경로의 발생 빈도 수를 이용하여 여러 단계의 스키마 그래프를 생성한다. 유동적인 스키마 그래프는 사용자 질의를 효과적으로 수행하기 위해 질의 수행을 위한 XML 문서의 범위를 유동적으로 축소/확장하기 위해서이다. 모든 XML 문서에서 각 레이블 경로에 대한 비트맵을 bitwise-OR 연산을 수행하면 모든 레이블 경로가 표현되는 광범위한 스키마 그래프를 구할수 있고, bitwise-AND 연산을 수행하면 모든 XML 문서에서 공통적으로만 표현되는 레이블 경로로만 구성된 스키마 그래프가 생성되어 질의 범위를 축소할 수 있다. 또한 레이블 경로 발생 빈도수를 조절하여 스키마 그래프를 생성함으로써 그 질의 범위를 유동적으로 할 수 있다. 여러 단계의 스키마를 추출하기 위해 레이블 경로의 발생 빈도 수를 알고리즘 4에 의해 계산한다. 그리고 표 3의 비트벡터를 알고리즘에 적용하면 그 결과는 표 5와 같다.

알고리즘 4. 레이블 경로 빈도수 계산

```
countLabelPath()
 begin
    for each i in path P
        Freq(i) = count(∑dj)
 end
```

(표 5) 레이블 경로의 발생 빈도수

	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	p13
d1	1	1	0	0	1	0	0	1	1	1	1	0	1
d2	1	1	0	0	1	0	0	1	1	1	1	1	1
d3	1	1	1	1	0	0	0	1	1	1	1	0	0
d4	1	1	1	1	0	1	1	0	1	1	1	1	1
Frequency													
Fi	4	4	2	2	2	1	1	3	4	4	4	2	3

표 5와 같이 계산되어진 레이블 경로의 빈도수를 어떤 임계치를 기준으로 여러 단계의 스키마를 추출할 수가 있다. 주어진 임계치보다 빈도수가 적을 레이블 경로는 0의 값을, 빈도수가 많은 레이블 경로는 1의 값을 갖는다. 1의 값을 갖는 레이블 경로로만 이루어진 스키마 그래프를 생성하는 것이다.

예를 들어 표 5의 레이블 경로 발생 빈도수에서 빈도수가 3보다 적은 레이블 경로에 대해서는 비트의 값을 0으로 하고, 빈도수가 3보다 큰 경우는 1의 값을 준다. 그 결과는 표 6과 같다. 따라서 그림 4의 스키마 그래프에서 빈도수가 3 이상인 레이블 경로에 대해서만 스키마를 추출한다면 빈도수가 3보다 작은 레이블 경로 *p3*, *p4*, *p5*, *p6*, *p7*, *p12*는 새로 생성되는 스키마에서 제외되고 그림 5와 같이 나머지 경로들에 대해서만 그림 5와 같은 새로운 스키마가 생성된다.

스키마 그래프 그림 4와 그림 5를 비교해 보면, 그림 5는 그림 4에 비해 스키마 그래프가 비교적 간단하게 표현됨을 알 수 있다. 결국 레이블 경로가 3보다 적은 레이블 경로에 대해서는 질의 수행범위에 포함하지 않겠다는 것이다. 이렇게 함으로써 질의 수행 범위를 질의 처리전에 줄일 수가 있다.

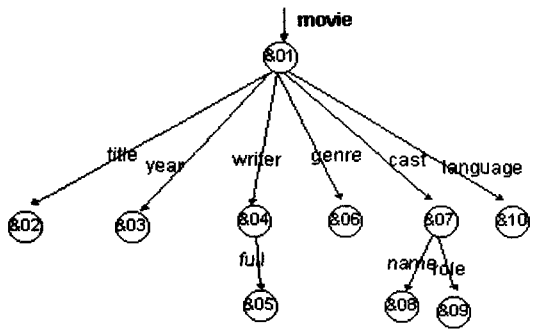
마찬가지 방법으로 레이블 경로의 발생 빈도수가 2이상인 경우의 비트맵 인덱스와 스키마 그래프를 구하면 표 7, 그림 6과 같다. 이 경우는 그림 4의 스키마 그래프보다는 적은 레이블 경로로 구성되지만 그림 5의 임계치 3인 경우의 스키마 그래프보다는 많은 레이블 경로로 구성된 스키마 그래프가 생성됨을 알 수 있다.

알고리즘 5. 임계치에 따른 레이블 경로 비트맵

```
FilterFreq(threshold t)
 begin
    while (Freq(i) isNot EOF)
        if (Freq(i) >= t)
            New_Freq(i) = 1
        else
            New_Freq(i) = 0
        end if
 end
```

(표 6) 빈도수가 3 이상인 레이블 경로

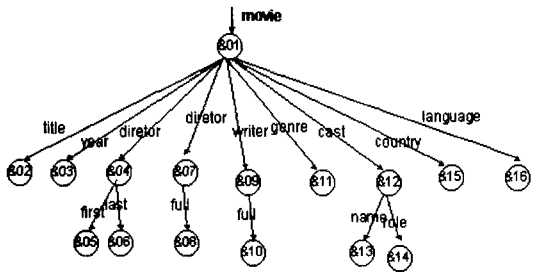
	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	p13
d1	1	1	0	0	1	0	0	1	1	1	1	0	1
d2	1	1	0	0	1	0	0	1	1	1	1	1	1
d3	1	1	1	1	0	0	0	1	1	1	1	0	0
d4	1	1	1	1	0	1	1	0	1	1	1	1	1
Frequency >= 3													
F>=3													



(그림 5) 빈도수 ≥ 3 인 스키마 그래프

(표 7) 빈도수≥ 2인 레이블 경로

	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	p13
d1	1	1	0	0	1	0	0	1	1	1	1	0	1
d2	1	1	0	0	1	0	0	1	1	1	1	1	1
d3	1	1	1	1	0	0	0	1	1	1	1	0	0
d4	1	1	1	1	0	1	1	0	1	1	1	1	1
Frequency >= 2													
F>=2													



(그림 6) 빈도수 ≥ 2인 스키마 그래프

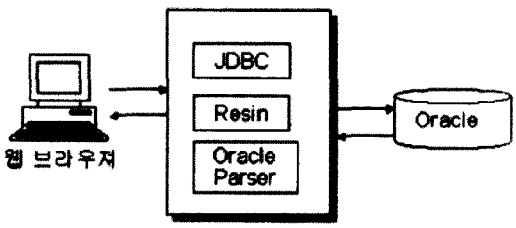
레이블 경로의 빈도 수에 따라 스키마를 다양하게 추출하게 되면 사용자 질의결과에 따라 빈도수가 많은 혹은 적은 스키마에 재적용해 볼 수 있다. 그러면 보다 효율적인 질의 수행이 가능해진다.

5. 실험

이 장에서는 본 논문에서 제시하는 레이블 경로의 발생 빈도수를 바탕으로 한 스키마 추출방법에 대한 구현 및 실험을 기술한다. 그리고 구현을 위한 환경, 구현 모델 및 구현 예를 보인다.

5.1 실험 환경

본 논문에서 제시한 레이블 경로의 발생 빈도수에 따른 스키마 추출방법의 구현 환경은 다음과 같다. 운영체제는 Windows2000 이고, 데이터베이스는 Oracle8i를 사용하였다. 구현 언어는 JDK 1.3.1과 JSP를 사용하였고, 오라클과의 연동을 위하여 Oracle JDBC thin driver를 사용하였다. JSP 엔진으로는 resin 2.0.1을 사용하고 XML 파서로는 Oracle Parser(버전 2.0.1.0)를 사용하였다.



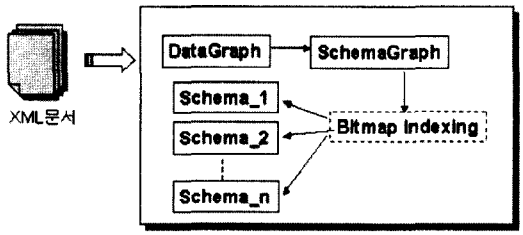
(그림 7) 실험 환경

5.2 실험 모델

본 논문에서 사용된 XML 문서는 영화에 관한 정보가 있는 http://www.imdb.com 의 텍스트 데이터를 바탕으로 표 7과 같이 생성하였다[22]. 그리고

(표 7) XML 문서의 예

```
<?xml version="1.0" standalone="no"?>
<movie>
<title>Amadeus</title>
<year>1984</year>
<director>Milos Forman</director>
<writer>Peter Shaffer </writer>
<genre>Drama</genre>
<cast>
  <name>F. Murray Abraham</name>
  <role>Antonio Salieri </role>
  <name>Tom Hulce</name>
  <role>Wolfgang Amadeus Mozart </role>
<name>Elizabeth Berridge</name>
<role>Constanze Mozart </role>
</cast>
<language>English</language>
<country>USA</country>
</movie>
```



(그림 8) 스키마 추출 과정

본 논문에서 제안하는 스키마를 추출하는 과정은 그림 8과 같다.

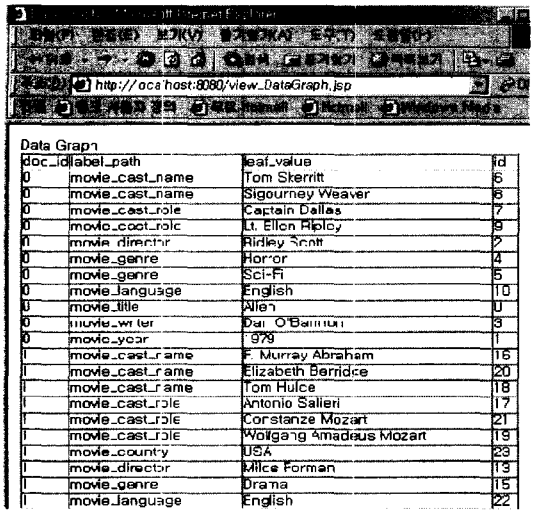
5.3 실험 예

5.3.1 데이터 그래프 생성 예

그림 9는 입력받은 XML 문서들에 대해 파싱 작업을 하고 생성된 데이터 그래프가 데이터베이스에 저장된 예이다.

5.3.2 스키마 그래프 생성 예

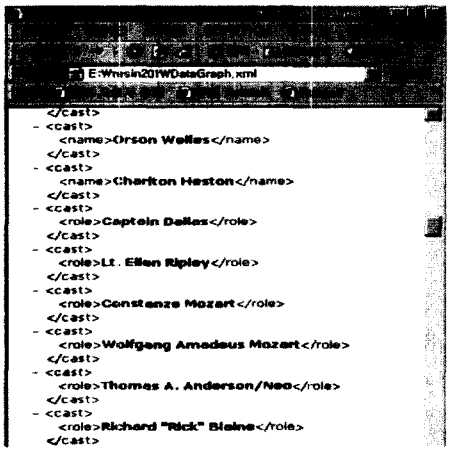
그림 11은 데이터베이스에 저장된 스키마 그래프의 예이다. 그리고 그림 11의 스키마 그래프는 그림 9의 데이터 그래프에 비해 중복된 레이블 경로없이 간단한 형태로 저장됨을 알 수 있다.



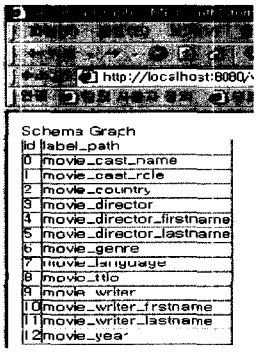
Data Graph

doc_id	label_path	leaf_value	id
0	movie_cast_name	Tom Skerritt	6
0	movie_cast_name	Sigourney Weaver	8
0	movie_cast_role	Captain Dallas	7
0	movie_cast_role	Lt. Ellen Ripley	9
0	movie_director	Ridley Scott	2
0	movie_genre	Horror	4
0	movie_genre	Sci-Fi	5
0	movie_language	English	10
0	movie_title	Alien	0
0	movie_writer	Dan O'Bannon	3
0	movie_year	1979	1
1	movie_cast_name	F. Murray Abraham	16
1	movie_cast_name	Elizabeth Berridge	20
1	movie_cast_name	Tom Hulce	18
1	movie_cast_role	Antonio Salieri	17
1	movie_cast_role	Constanze Mozart	21
1	movie_cast_role	Wolfgang Amadeus Mozart	19
1	movie_country	USA	23
1	movie_director	Milos Forman	13
1	movie_genre	Drama	15
1	movie_language	English	22

(그림 9) 데이터베이스에 저장된 데이터 그래프



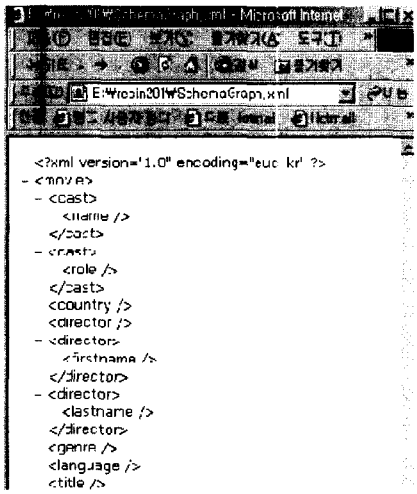
(그림 10) XML문서 형태의 데이터 그래프



Schema Graph

id	label_path
0	movie_cast_name
1	movie_cast_role
2	movie_country
3	movie_director
4	movie_director_firstname
5	movie_director_lastname
6	movie_genre
7	movie_language
8	movie_title
9	movie_writer
10	movie_writer_frstname
11	movie_writer_lastname
12	movie_year

(그림 11) 데이터베이스에 저장된 스키마 그래프



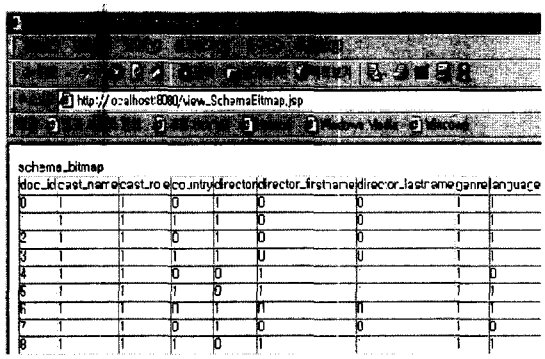
(그림 12) XML 문서 형태의 스키마 그래프

데이터베이스에 저장된 스키마 그래프를 XML 문서의 형태로 보면 그림 12와 같다. 스키마 그래프에는 레이블 경로에 대한 정보만 저장되기 때문에 XML 문서로 표현했을 때 각 엘리먼트는 빈 엘리먼트로 표현되었다.

5.3.3 레이블 경로 발생 빈도수

그림 13은 스키마 그래프와 데이터 그래프를 이용하여 각 XML 문서의 비트맵 인덱싱을 구한 예이다.

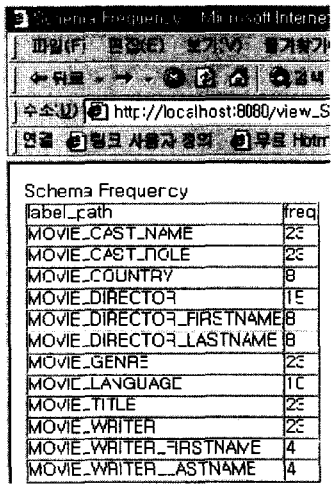
그림 14는 그림 13에서와 같이 각 문서에 대해 레이블 경로의 존재여부를 비트맵 인덱싱으로 나타낸 것에 대해 레이블 경로의 총 발생 빈도수를



schema_bitmap

doc_id	cast_name	cast_role	country	director	director_firstname	director_lastname	genre	language
0	1	1	0	1	0	0	1	1
1	1	1	1	1	0	0	1	1
2	1	1	0	1	0	0	1	1
3	1	1	1	1	0	0	1	1
4	1	1	0	0	1		1	0
5	1	1	1	0	1		1	1
6	1	1	1	1	1	1	1	1
7	1	1	0	1	0	0	1	0
8	1	1	1	0	1		1	1

(그림 13) 비트맵 인덱싱 예



(그림 14) 레이블 경로 발생 빈도수

(그림 15) 임계치 입력 예

계산한 것이다. 그림 14의 결과를 보면 cast, genre, title, writer에 대한 레이블 경로가 24개로 가장 많이 나타난걸 알 수 있고, writer의 레이블 경로가 firstname과 lastname으로 구별되어 최소의 발생 빈도수로 4개라는 것을 알 수 있다.

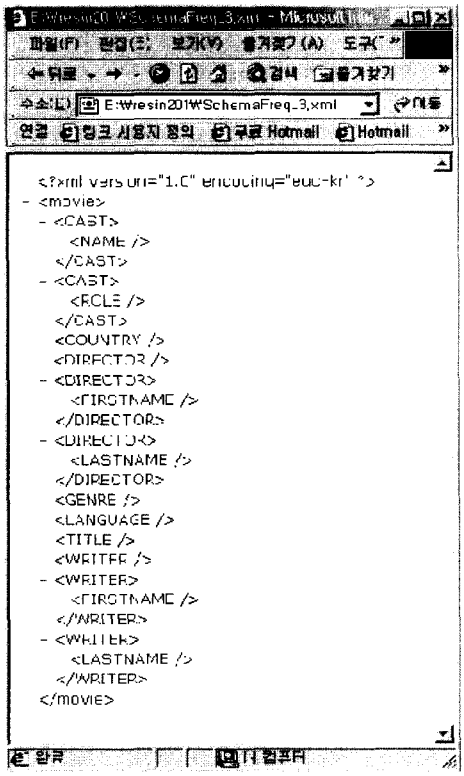
5.3.4 레이블 경로의 발생 빈도수에 따른 스키마 추출 예

레이블 경로 발생 빈도수에 대한 정보를 파악한 후 임계치를 부여하여 질의 처리를 하기 전에 그 질의 범위를 질의 처리 전에 축소 혹은 확대할 수 있다.

그림 15는 레이블 경로 발생 빈도수에 대한 임계치를 입력하는 예이다.

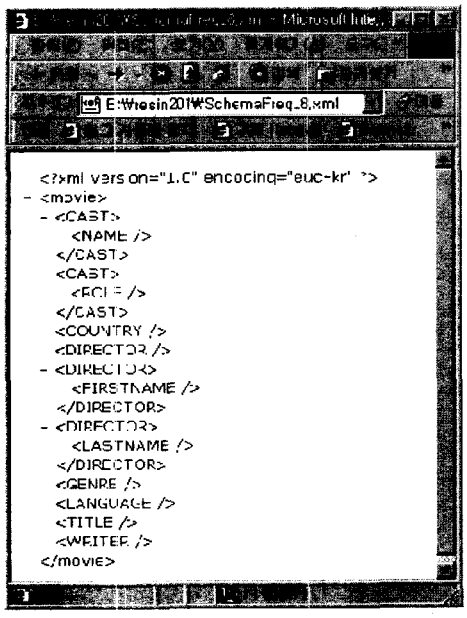
레이블 경로 발생 빈도수에 대한 임계치를 각각 3과 8로 입력했을 때 추출되는 스키마 그래프를 XML 문서 형태로 표현하면 그림 16, 그림 17과 같다

임계치가 3인 경우와 8인 경우에 추출되는 스키마의 형태를 비교해보면 8인 경우는 writer 에 대한 정보가 firstname과 lastname으로 구별되어 있는 정보에 대해서는 포함되어 있지 않다는 것을 알 수 있다. 즉, 임계치를 8로 설정하여 추출



(그림 16) 임계치가 3인 경우에 추출된 스키마

된 스키마에 대해서는 writer의 정보를 구별하여 질의를 수행할 수 없음을 알 수 있다.



(그림 17) 임계치가 8인 경우에 추출된 스키마

사용자 질의에 대해 writer 정보를 firstname과 lastname 정보까지 필요한 경우라면 임계치가 3이상인 스키마에 대해서 질의 처리를 하고, 그렇지 않을 경우에는 임계치가 8 이상인 경우에 추출된 스키마에 대해 질의 처리를 하면 질의 처리 범위를 줄일수가 있다.

이렇게 질의 처리전에 임계치를 다양하게 주어 여러 단계의 스키마 추출이 가능하게 됨으로써 보다 효율적인 질의 처리가 가능해 질 것이다.

6. 결 론

XML이 인터넷상에서 데이터를 표현하고 교환하는 새로운 표준으로 등장하고 있다. XML은 미리 정의된 스키마가 없고, 문서 자체에 데이터와 데이터 구조를 갖고 있기 때문에 기존의 관계형 데이터베이스나 객체 지향 데이터베이스에서 사용되는 SQL이나 OQL을 바로 적용하기가 어렵다. 따라서 이러한 XML에 대해 새로운 질의어와 질의 처리를 위한 스키마 추출에 대한 많은 연구가 이루어지고 있다.

본 논문에서는 XML 문서에 대해 레이블 경로의 발생 빈도수에 따른 스키마 추출 방법을 제안하였다. 스키마 추출 방법은 일단 같은 DTD를 갖는 XML 문서들에 대해 스키마 그래프를 생성하여 XML 문서에 있는 모든 엘리먼트의 정보가 모두 표현되면서 단 한번만 표현될 수 있도록 한다. 그리고 스키마 그래프를 바탕으로 입력 XML 문서의 레이블 경로 존재 여부를 비트 벡터로 표현하고, XML 문서에서 레이블 경로의 발생 빈도 수를 계산하였다. 그리고 어떤 임계치에 따라 여러 단계의 스키마 추출을 가능하게 함으로써 사용자 질의에 대해 보다 효율적으로 처리할 수 있도록 하였다.

본 논문에서 제안하는 방법은 XML 문서에 나타나는 레이블 경로의 발생 빈도수에 따라 여러 단계의 스키마 추출을 가능하게 함으로써 사용자 질의에 대해 보다 유동적으로 적용시킬 수 있음을 알 수 있다. 사용자 질의에 대해 너무 적은 혹은 너무 많은 질의 결과가 나왔다면 임계치를 두어 그 질의 범위를 축소 혹은 확장하여 사용자 질의에 보다 적합한 결과를 보여줄 수 있다.

향후 연구과제로는 메타 데이터를 이용한 스키마 추출방법을 고려하여 데이터의 형태뿐만 아니라 의미적으로도 분석이 가능해 보다 효율적인 스키마를 추출하는 방법에 대한 연구가 필요하다.

참 고 문 헌

[1] Jon Bosak, "XML, Java, and the Future of the Web", http://webreview.com/wr/pub/97/12/19/xml/index.html.

[2] Roy Goldman, Jennifer Widom, "DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases", In Proceedings of VLDB, 1997

[3] Jiawei Han, Jian Pei, Yiwen Yin, "Mining Frequent Patterns without Candidate Generation",

Proceedings of the 2000 ACM SIGMOD on Management of data, 2000, pp. 1-12.

[4] Theodore Johnson, "Performance Measurements of Compressed Bitmap Indices", VLDB 1999, pp. 278-289.

[5] Alon Levy, "More on Data Management for XML", University of Washington, May 9th, 1999. http://www.cs.washington. edu/homes/alon/widom-response.html.

[6] J. McHugh, S. Abiteboul, R. Goldman, D. Quass, J. Widom, "Lore : A Database Management System for Semistructured Data", SIGMOD Record, 26(3), pp. 54-66, September 1997.

[7] Patrick O'Neil, "Improved Query Performance with Variant Indexes", Proceedings of ACM SIGMOD 1997, pp. 38-49.

[8] Jayavel Shanmugasundaran, Kristin Tufte, Gang He, Chun Zhang, David DeWit, Jeffrey Naughton, "Relational Databases for Querying XML Documents: Limitations and Opportunities", Proceedings of the 25th VLDB Conference 1999.

[9] Dan Suciu, "Semistructured Data and XML", In Proceedings of International Conference on Foundation of Data Organization, 1998.

[10] M.C. Wu, A.P. Buchmann, "Encoded Bitmap Indexing for Data Warehouses", Proc. ICDE '98, 220-230.

[11] Jennifer Widom, "Data Management for XML", Working Document, initial draft appeared April 1999, Also IEEE Data Engineering Bulletin, Special Issue on XML, 22(3):44-52, September 1999.

[12] Ke Wang, Huiqing Liu, "Schema Discovery from Semistructured Data", International Conference on Knowledge Discovery and Data Mining, August 1997, pp.271-274.

[13] Ke Wang, Huiqing Liu, "Discovering Typical Structures of Documents : A Road Map Approach", The ACM SIGR conference on Research and Development in Information Retrieval, August 1998, pp. 146-154.

[14] Ming-Chuan Wu,, "Query optimization for selections using bitmaps", Proceedings of the 1999 ACM SIGMOD international conference on Management of data, pp. 227-238.

[15] "XML Path Language(XPath) Version 1.0", W3C Recommendation 16 November 1999, http://www.w3.org/TR/xpath.

[16] J. Yoon, S. Kim, "Schema Extraction for Multimedia XML Document Retrieval", in Proc. of International Database Symposium on Mobile, XML and Post-Relational Databases, Hong Kong, June 2000. Also to appear in Journal of Applied Systems Studies, Cambridge International Science Publishing, Cambridge, UK, 2001.

[17] S. Abiteboul, D. Quass, J. McHugh, J. Widom and J. Wiener, "The Lorel Query Language for Semistructred Data", International Journal of Digital Libraries, 1(1):66-88, 1997.

[18] Roy Goldman, Jason McHugh, Jennifer Widom, "Lore: A Database Management System for XML", Dr. Dobb's Journal, 25(4):76-80. April 2000, http://www.ddj.com/articles/2000/0004/0004i/0004i.htm ?topic=xml.

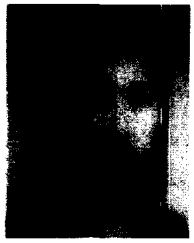
[19] Serge Abiteboul, Peter Buneman, Dan Suciu, "Data on the Web : From Relations to Semistructured Data and XML", Morgan Kaufmann, 2000.

[20] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, "Extensible Markup Language(XML)1.0", http://www.w3.org/TR/REC-xml#dt-xml-doc, 1998.

[21] Serge Abiteboul, "Querying Semi-Structured Data", In IDCT 1997.

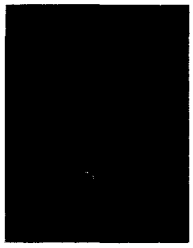
[22] http://us.imdb.com/top_250_films

◐ 저 자 소 개 ◑



김 성 림

1994년 숙명여자대학교 전산학과 졸업 (이학사)
1997년 숙명여자대학교 대학원 전산학과 졸업(석사)
1997년~현재 : 숙명여자대학교 대학원 컴퓨터학과 박사과정
2001년~현재 : 동덕여자대학교 정보학부 컴퓨터학전공 강의 전임
관심분야 : XML, 멀티미디어 데이터베이스, 질의 처리
E-mail : srkim@cs.sookmyung.ac.kr



윤 용 익

1983년 동국대학교 통계학과 졸업 (이학사)
1985년 한국과학기술원 전산학과 졸업 (공학 석사)
1994년 한국과학기술원 전산학과 졸업 (공학 박사)
1997년~현재 : 숙명여자대학교 정보학부 멀티미디어학과 교수
관심분야 : 정보통신, 멀티미디어 통신, 분산 시스템, 실시간 처리 시스템, 분산 미들웨어 시스템, 분산 데이터베이스 시스템, 실시간 OS/ DBMS
E-mail : yiyoon@sookmyung.ac.kr