XQL 기반의 XML 문서 구조 검색

박종훈[1)]

XML Document Structure Search based on XQL

Park, Jong Hoon

요약

사실 표준 XQL(XML Query Language) 기반의 문서 질의를 이용하여 XML 문서에 대한 구조 검색과 내용 검색을 위한 문서검색 시스템을 구성하였다. 즉 XML 문서를 파징하여 내용 및 구조 정보를 트리 구조로 구성하고, 표준 XQL을 이용하여 트리 구조 정보에 질의함으로서 구조 및 내용 검색을 하도록 구현하였다. 이를 이용한 응용분야로는 향후 방대한 크기의 XML문서 검색과 문서 관리, 전자 상거래 등 다양한 분야에 활용될 수 있는 요소기술로서 기대된다.

keyword : XML, XQL, 문서 검색

1) 중부대학교 정보공학부

1. 서론

현재 XML(eXtensible Markup Language)에 대한 관심과 연구가 한창 진행되고 있으며, 점차적으로 핫 이슈가 될것으로 예상된다. 따라서 사용자나 개발자가 쉽게 접근할 수 있도록 설계 기법과 단순 기법을 채택하고 있으며, 이에 대한 응용은 다양하다 할 것이다. 차세대 웹 문서 포맷으로 부각하는 XML은 W3C에서 제안된 국제 표준의 전자문서 메타 언어이다[3][4][5]. XML 은 웹에서 구조화된 문서를 전송 가능하도록 설계된 표준화된 텍스트 형식으로, 문서를 구성하는 각 요소들의 독립성을 보장하게 함으로서 문서의 호환성, 내용의 독립성, 요소변경의 용이성 등의 특성을 제공한다. 이에 따라 HTML의 새로운 대안으로 떠오르고 있는 XML이 인터넷 관련 업계에 주요 관심사가 되고 있다. 현재 인터넷 웹 문서뿐만 아니라 전자도서관, CSCW(Computer Supported Cooperative Work) 그리고 CALS(Commerce At the Light Speed)를 포함한 다양한 분야에서 XML을 활용하고자 폭 넓은 연구를 하고 있으며, 수학 분야의 MathML(Mathemetical Markup Laguage), 채널 기술의 CDF(Channel Definition Format), 이동통신에서의 HDML(Handheld Device Markup Language) 등 최근에 구체화되는 응용이 부쩍 많아지고 있다[4][6][7].

이와 같이 전세계적으로 XML에 대한 관심이 고조되고 실제 많은 분야에서 활용되고 있기 때문에 향후 XML 응용분야의 진행 발전 단계에서 정보의 생성, 재사용성, 처리 및 지속성, 이식성 등과 같은 XML 사용 분야가 점차 증대될 것으로 예상된다. 이에 따른 XML 문서에 대한 구조 및 내용 검색에 대한 중요성이 점차 중요해 지고 있는 실정이다[7].

본 논문에서는 사실 표준 XQL 기반의 문서 질의를 이용하여 XML 문서에 대한 구조 검색과 내용 검색에 대하여 구성 하였다. 즉 XML 문서를 파징하여 구조정보를 트리구조로 구성하고, 이를 이용하여 표준 XQL을 이용하여 질의함으로서 구조 및 내용 검색을 하도록 구현하였다. 이를 이용한 응용분야로는 향후 XML문서의 검색과 문서 관리 등 다양한 분야에 활용될 수 있는 요소 기술로서 기대된다.

2. 관련 연구

XML은 HTML과 SGML(Standard Generalized Markup Languag)과 함께 일반 마크업 언어의 일종으로, HTML에서의 간결한 데이터 표현 형식과 웹 브라우저에서의 뛰어난 표현 능력뿐만 아니라 SGML에서의 뛰어난 데이터 표현 능력을 동시에 갖도록 설계되었다. XML 문서는 태그(tag), 엘리먼트(element), 속성(attribute), 처리명령, 주석문 등으로 구성된다. 태그는 문맥을 나타내는 것으로 문서의 의미를 표현할 수 있게 해준다. 엘리먼트는 태그와 그 사이의 텍스트를 말하며, 태그는 속성을 가질 수 있다.처리명령은 파싱을 위하여 파서에 전달되는 정보이며, 주석문은 문서에 대한 설명을 나타낸다. XML 문서의 형태는 엘리먼트로 구성되어 있으며, 트리 형태의 구조적 표현이 가능하다[3][4][8].

기존의 XML 문서 저장 및 검색 시스템에 대한 연구는 객체지향 DBMS를 사용하는 eXcelon, POET XML Repository 등이 있으며, 관계형 DBMS를 사용하는 XDMS, Oracle8i 등이 있다. eXcelon은 단순성과 유연성과 같은 XML 장점을 살릴 수 있는 응용 프로그램을 개발하도록 하는 XML 데이터 서버이다. 이는 W3C의 DOM(Document Object Model)을 지원하며, XQL를 XML 문서의 질의어로 사용한다. 그러나 eXcelon은 모든 데이터를 XML 이라는 하나의 논리적관점에서 취급한다. 또한 XML 검색 엔진으로 Milner가 개발한 SCOOBS가 있다. 이것은 각 용어들을 엘리먼트들과 연결시켜 나타내고, 역 인덱스 파일 구조를 가진다[8][9].

3. 사실 표준 XQL

W3C에서 현재 공식적인 XML 질의어에 대한 표준을 진행중에 있으며, 이러한 작업활동에 매우 중요한 영향을 미칠것으로 예상되는 XML 질의어(XQL)을 기반으로 설명을 하며, 또한 본 논문에서도 XQL 기반으로 문서검색 질의어를 구성하였다.

XQL은 1998년 9월 XSL 워킹그룹에 제안되었으며, 현재 이러한 기능들이 마이크로소프트사 IE5.0에서도 일부 구현되고 있다. XQL은 XML 문서의 엘리먼트와 텍스트를 필터링하기 위한 표기법이다. XQL은 XSL(eXtensible Stylesheet Language) 패턴 문법으로도 확장 가능하며, 특정 노드와 엘리먼트를 검색하고, 지정하기 위한 간결하고 쉬운 표기법이다. XQL의 설계목표로는 XQL 문장이 간결하여야 하며, XQL은 이해하기 쉬어야 하며,XQL 문법은 단순하여야 한다. 또한 XQL은 쉽게 파싱될 수 있어야 하며, XML 문서 내에서 노드들을 식별할 수 있어야 한다. 그리고 XQL 질의는 여러개의 결과를 나타낼 수 있어야 하며, 기타 여러 가지의 설계목표를 가지고 만들어 졌다. XQL은 문서내에서의 정보를 검색하기 위한 표기법이다. 정보는 특정 연관된 노드들의 집합이다. 이 표준에서는 출력에 대한 형식을 지정하고 있지 않으며, 질의에 대한 결과는 한개의 노드, 노드 목록, XML 문서, 배열, 또는 다른 구조가 될 수 있다. 일부 구현에서는 질의에 대한 결과가 XML 문서 또는 트리구조가 될 수도 있다. XQL의 특성을 알기 위해 다음 4가지 환경에 대한 기본적인 비교를 SQL과 비교하면 <표 1>과 같다[1][2].

<표 1> SQL과 XQL 비교

SQL	XQL
데이터베이스는 테이블들의 집합이다.	데이터베이스는 XML 문서 집합이다.
SQL 질의어 기본모델로서 테이블을 이용한 언어이다. FROM 절은 질의어에 의해 검색되는 테이블들을 결정한다.	XQL 질의어 기본모델로서 XML 구조를 이용한 언어이다. 질의어는 여러 문서들의 입력노드들에 적용되며, 이들 노드와 후손 노드들을 검색한다.
질의어에 대한 결과는 여러 행들의 집합을 구성하는 테이블이다.	질의어의 결과는 XML 문서 노드들의 집합이며, well formed XML 문서를 생성시킨다.

XQL은 크게 XML 패턴과 XQL 확장으로 나누어 지며 다음과 같다.

3.1 XML패턴

XQL의 핵심적인 표기법을 기술한다. 이러한 기능은 XQL의 구현에 달려 있으며, 다른 기술들에 적용할 경우 기본이 되는 요소로서 역할을 할 수 있다.

- 집합(Collections) : 태그 이름을 갖고 있는 모든 엘리먼트들은 태그 이름을 사용하여 특정 엘리먼트를 표현할 수 있다.
예) 모든 first-name 엘리먼트를 검색할 경우
-> 질의어) ./first-name은 first-name

- 자식과 후손 노드에 대한 선택 : 특정 엘리먼트에 대한 집합은 경로 연산자('/'또는 '//')를 사용하여 지정할 수 있다. 이 연산자는 엘리먼트의 왼쪽에, 그리고 선택되어질 엘리먼트는 오른쪽에 나타낸다. 자식 연산자('/')는 왼쪽 엘리먼트 집합의 다음 자식으로 선택되는 반면에 후손 연산자('//')는 왼쪽 엘리먼트 집합의 임의의 후손 엘리먼트들을 선택한다. 효율적으로 '//'은 구조 계층의 여러 하위 레벨을 반복하여 추적한다.
예1) author 엘리먼트내의 자식노드 중에서 모든 first-name 엘리먼트들을 검색할 경우
->질의어) author/first-name
예2) bookstore 엘리먼트 내에서 title을 갖는 후손 엘리먼트들을 검색할 경우
->질의어) bookstore//title

- 자식 엘리먼트 집합 : 엘리먼트는 '*'을 이용하여 이름을 사용하지 않고 참조할 수 있다. '*' 집합은 현재 노드의 자식을 모두 리턴한다.
예) author 엘리먼트의 모든 자식 엘리먼트를 검색할 경우
->질의어) author/*

- 속성 검색 : 속성 이름이 '@'이 선행되고, XQL은 엘리먼트 내의 속성을 처리하도록 설계되었다.
예1) 현재 엘리먼트의 style 속성을 검색할 경우
->질의어) @style
예2) 현재 엘리먼트인 price내에 exchange속성들을 검색할 경우
->질의어) price/@exchange

- 필터 : 필터절은 '[ ]'으로 표기하며, 이것은 SQL의 WHERE 절과 유사하다. 이 필터는 서브 질의를 포함하며, 여기에 대한 조건을 만족하는 집합이 존재할 경우 불리안 값으로 TRUE를 발생한다.
예1) 적어도 한 개 이상의 abstract 엘리먼트를 포함하는 모든 book을 검색할 경우
->질의어) book[abstract]
예2) 적어도 한 개 이상의 abstract 엘리먼트를 포함하는 book의 title을 검색할 경우
-> 질의어) book[abstract]/title

- 불리안 대수 표현 : 이것은 서브 질의내에서 사용되며, 연산자로는 and, or, not을 이용하여 조건에 맞는 결과를 취한다.
예1) 적어도 degree와 award이 동시에 한 개 이상을 포함하는 모든 author 엘리먼트들을 검색할 경우
->질의어) author[degree and award]
예2) degree이 한 개 이상을 포함하고, 그중에서 publication 엘리먼트를 포함하고 있지 않은 모든 author 엘리먼트들을 검색할 경우
->질의어) author[degree and not publication]

- 등호 : '='은 eq, '!='은 ne와 동일하며, 비교연산이 가능하다.
예) last-name 엘리먼트가 Bob인 모든 author 엘리먼트를 검색할 경우
->질의어) author[last-name = 'Bob']

- 매소드 : XQL은 집합을 처리하는 함수를 제공한다. 이 메소드는 {메소드}(변수)와 같은 형태를 갖는다.지원 되는 정보 메소드로는 text(), value(), nodeType(), nodeName()이 있다. 특히 인덱스 함수인 index()은 인덱스 번호를 리턴하며 인덱스 0는 첫 번째 엘리먼트이다.
예1) Bob을 포함하는 모든 author들을 검색할 경우
->질의어) author[text() = 'Bob'] : 현재노드의 텍스트를 리턴하여, 비교한다.
예2) 처음 3개의 degree를 검색할 경우
->질의어) degree[index() lt 3]

- 집합 인덱스 : XQL은 노드 집합으로 부터 특정 노드를 검색할 수 있다.
예1) 첫 번째 author 엘리먼트를 검색할 경우
->질의어) author[0]
예2) first-name 엘리먼트를 갖고 있는 세 번째 author를 검색 할 경우
->질의어) author[first-name][2]
예3) 집합중에서 마지막 엘리먼트를 검색할 경우
->질의어) book[end()]

3.2 XQL 확장

XQL은 필요한 최소한의 집합을 제공하고 있으나, 여기 XQL확장은 XQL의 기능을 확장한 것이다.

- 이름공간 : 질의어 내에서 이름공간 정보를 인식할 수 있어야 한다. XQL 처리기는 이러한 정보를 이용하여 효율적으로 처리 되어야 한다. 이름 공간 메소드는 이름공간 정보를 리턴하기 위해 노드에 적용할 수 있어야 한다. 즉 접두사를 제외한 노드의 이름 부분을 리턴할 수 있는 baseName()과 노드의 이름 공간을 위한 URI를 리턴할 수 있는 namespace(), 노드의 접두어를 리턴할 수 있는 prefix()가 있다.
예) author 엘리먼트를 갖는 모든 my:book 엘리먼트를 검색할 경우
->질의어) my:book[author]

- 속성 집합에 대한 검색 : 엘리먼트의 모든 속

성들은 @*을 사용하여 검색할 수 있으며, 이것은 레코드의 필드처럼 속성을 처리할 수있다.
예1) 현재 엘리먼트의 모든 속성을 검색할 경우
->질의어) @*
예2) 이름 공간을 갖고있는 style 속성을 검색할 경우
->질의어) @*:style

- 비교 : 이진 비교 연산자는 숫자와 스트링을 비교하여, 불리안 결과를 리턴한다. le(less than), le(les than or equal), gt(graeter than), ge(graeter than or equal)을 사용하며, <, <=, >, >=을 사용할 수도 있다.
예) price가 50보다 크고, last-name이 Bob인 모든 author 엘리먼트를 검색할 경우, author[last-name = 'Bob and price gt 50]

<표 2> 비교 연산자

문자형태	비교	예
String	text(lvalue) op text(rvalue)	a < 'foo'
Integer	(long)lvalue op (long) rvalue	a < 3
Real	(double) lvalue op (double) rvalue	a < 3.1

- 합 또는 공통집합 : 합 연산자는 여러개 엘리먼트의 합 집합을 리턴하며, 공통 집합 연산자는 두 개의 집합에서 공통된 엘리먼트만을 리턴한다.
예) 모든 first-name과 last-name을 검색할 경우
->질의어) first-name $union$ last-name

- 집합 메소드 : 집합 메소드는 문서내의 다양한 형태 접근을 제공한다.

< 표 3> 집합 메소드

textNode()	텍스트 노드의 집합
comment()	주석 노드의 집합
pi()	pi 노드의 집합
element(['name'])	모든 엘리먼트 노드의 집합. 텍스트 변수가 제공된다면, 해당 이름과 일치된 엘리먼트들만 리턴
attribute(['name'])	모든 속성 노드의 집합. 텍스트 변수가 제공된다면, 특정 이름과 일치되는 속성 들만 리턴
node()	비 속성 노드의 집합

예) p 엘리먼트의 두 번째 텍스트 노드를 검색할 경우
->질의어) p/textNode()[1]

- 집계 메소드 : 집합에서 노드들의 총수를 리턴한다. count()

- 추가 메소드 : 노드의 타입을 스트링으로 리턴한다. nodeTypeString() : 다음 값중에 하나를 리턴한다. ('document', 'element', 'atribute', 'processing_instruction', 'comment', 'text')

- 조상 : 질의와 일치하는 가장 가까운 조상 노드를 리턴한다. ancester(query)
예) author 엘리먼트를 포함하고 있는 book 엘리먼트중에서 가장 가까운 조상을 검색할 경우
->질의어) ancester(book/author)

- 첨자 연산자
엘리먼트의 범위를 지정할 수 있도록 하며, [,] 사이에 설정할 수 있다.
예1) 첫번째와 네 번째 엘리먼트를 리턴 할 경우
->질의어) author[0,3]
예2) 첫번째부터 네 번째까지 얼리먼트를 리턴할 경우
->질의어) author[0 to 3]

<표 4> 첨자 속성

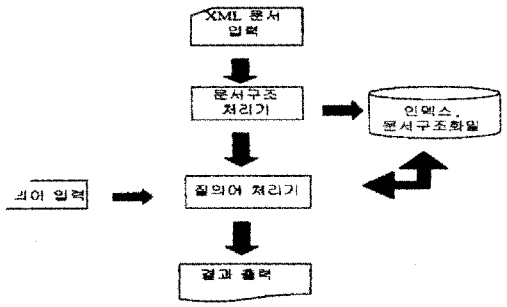
n	n+1 번째 엘리먼트 리턴
-n	마지막 엘리먼트로부터 n번째 엘리먼트를 리턴. -1일 경우 마지막 엘리먼트를 리턴
m to n	m부터 n 까지 엘리먼트 리턴

4. XQL에 기반한 XML 문서검색 시스템

본 논문의 문서 검색 시스템은 자바언어로 구성하였고, JDK1.2, SUN 사의 DOM과 SAX를 지원하는 파서와 사실 표준인 XQL을 기반으로 설계하였다. XQL 문법을 따르는 문서 검색 시스템에서는 XML 문서의 엘리먼트, 속성과 내용 전문에 대하여 검색할 수 있도록 시스템을 구성하였으며, 결과를 XML 문서로 출력할 수 있도록 하였다. Well- formed XML 문서를 입력으로 취하고, 이를 파싱하면서 인덱스 구조정보를 생성시킨다. 이러한 인덱스 구조정보를 질의어 처리기에서 이용하여 검색한다. XQL 확장 환경 중에서 합 집합, 공통집합, 불리안 표현 등 구현이 진행중에 있으며, 단일 유저 또는 소규모 크기의 XML 문서 검색에 적용가능하다.

4.1 XML 문서검색 시스템 구성

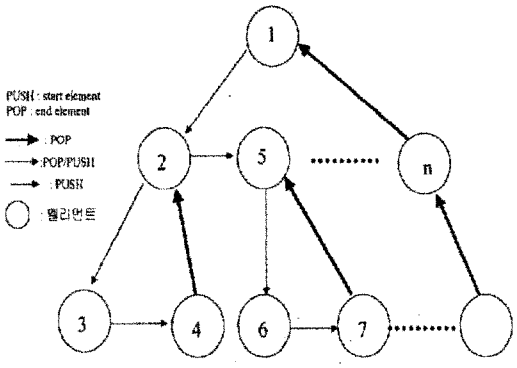
여러종류의 XML 문서를 입력하여 문서구조 처리기에 의해 파싱한 후에 내부 데이터 구조인 인덱스와 문서구조 정보를 구성한 후, 차후에 XML 질의어 처리기의 문서 검색을 위해 필요하다. 문서가 입력될 경우 문서 단위로 구분하여 인덱스 구조를 등록하고, 각각의 인덱스 구조를 생성한 후에 유지 관리하게 된다. 검색된 결과에 대한 구조 정보는 인덱스 정보와 함께 스택에서 유지되어 지며, 결과에 대한 질의를 계속하여 처리할 수 있도록 인덱스 정보를 계속 유지한다.



<그림 1> 문서 검색 시스템 구성도

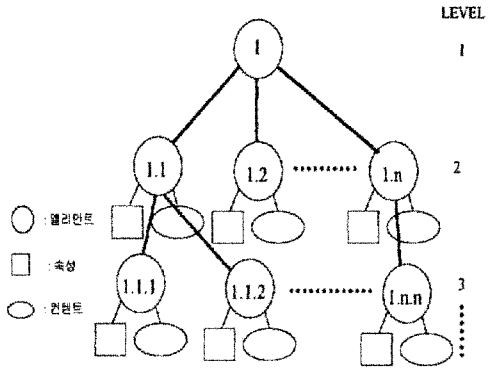
4.2 문서 구조 처리기

이 처리기는 XML 문서를 파싱하면서 인덱스 구조정보를 생성시킨다. Well formed XML 문서를 입력 받아서 스택과 구조정보 및 텍스트 내용정보 등에 관하여 초기화 한 후, start document, end document, start element, end element, 텍스트(내용)들을 분류하여, XML 문서를 인덱스된 데이터 구조로 변환시킨다. 각 엘리먼트를 노드로 하며, 트리 진행 순서는 입력들어 오는 문서의 순서에 따라 진행하고,동시에 스택을 운영한다. 그림 2에서 트리 구조는 XML 문서의 구조를 의미하며, 노드안의 번호는 트리진행 순서이다. 이것은 XML 문서의 순서와 일치한다. 화살표는 세종류로서 PUSH, POP/PUSH, POP으로 구성되며, 이것은 스택을 운영할 때의 상태를 의미한다.



<그림 2> 스택에 의한 XML 문서 구조 생성 순서

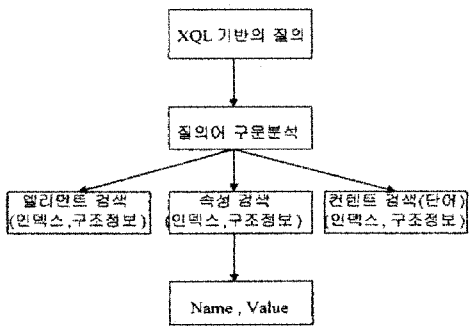
다음 그림 3은 각 노드에 따라 구조화 정보를 결정하며, 인덱스 및 구조정보를 가진 데이터 구조를 생성한다. 노드는 엘리먼트를 나타내며, 각 노드의 숫자는 레벨과 부모노드, 자식노드, 형제노드와 순서의 의미를 동시에 갖고 있다. 또한 각 엘리먼트는 자신의 엘리먼트 이름과 여러개의 속성과 컨텐트(내용)인 텍스트를 가질 수 있다. 속성 검색을 위해 속성은 내부 인덱스 정보를 구성하고 있으며, 텍스트 검색을 위해 텍스트도 단어 단위의 인덱스 정보를 갖고 있다.



<그림 3> 노드의 인덱스 정보와 구조정보

4.3 XQL 질의어 처리기

사용자로부터 질의어 입력을 받고, 질의어에 대한 구문분석을 처리한 후 엘리먼트 검색, 속성검색, 컨텐트(텍스트)검색에 대하여 분류하여, 기존의 인덱스정보와 구조 정보를 구성하였던 인덱스 문서구조 파일을 참조한다. 속성검색의 경우 속성 이름과 속성값에 대하여 인덱스정보를 검색하며, 컨텐트 검색의 경우 단어 단위로 인덱스된 정보를 검색한다.



<그림 4> XQL 질의어 처리기

4.4 XQL 질의 예

1) 입력 XML 문서

다음은 xdoc 엘리먼트가 124개인 XML 문서를 이용하여 XQL 질의어에 대한 검색을 한다. 여기서는 간단한 예로 2가지만을 보여주고 있다.

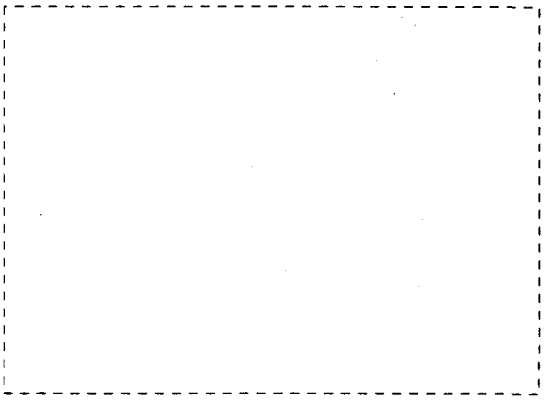
```
<xdocs>
  <xdoc pgnum="690">
      <fname>rfree10.xml</fname>
      <date>16-Feb-00</date>
      <fsize>347k</fsize>
      <docname>Proposed Roads to Freedom </docname>
      <docauth>
            <first>Bertrand</first>
            <middle></middle>
            <last>Russell</last>
      </docauth>
      <marker>Elias Hantzakos</mar
      <email show="no">ilias@em
      <dtd>gutbook1.dtd</dtd>
      <ss>darwin.css</ss>
      <note></note>
  </xdoc>
```

```
<xdoc pgnum="1197">
    <fname>taras10.xml</fname>
    <date>15-Feb-00</date>
    <fsize>647k</fsize>
    <docname>Taras Bulba and Other Tales</docname>
    <docauth>
        <first>Nikolai</first>
        <middle>Vasilievich</middle>
        <last>Gogol</last>
    </docauth>
    <marker>Lewis Overton</marker>
    <email show="no">lewy_o@yahoo.com</email>
    <dtd>gutbook1.dtd</dtd>
    <ss>anne11.css</ss>
    <note></note>
</xdoc>
.....
<xdocs>
```

2) 질의어 예와 출력 결과 문서

- XQL 질의어 입력 : //xdoc/email[@show = 'no'] 이것은 후손 엘리먼들 중에서 xdoc 엘리먼트 바로다음 자식 email 엘리먼트를 찾고, 그 중에서 속성 이름이 show이면서 속성값이 no인 email 엘리먼트들을 찾을 경우이다.

- 질의어 출력 : XML 문장으로 구성되어 있으며, 루트 엘리먼트로 XQL 태그를 사용하고 있고, 속성에는 질의어 입력문장, hitcount와 엘리먼트에 대한 정보가 출력된다. 다음 엘리먼트부터 실제 검색하고자 하는 email 엘리먼트들이 출력된다. 여기서는 총 65개의 hit를 보여주고 있다.



<그림 5> XQL 질의 처리기 실행화면

4. 결론

현재 XML을 이용한 응용분야는 더욱더 확산 되리라 예상되며, 이에대한 XML 문서에 대한 관리와 검색은 점차적으로 중요해 지고 있는 실정이다. 특히 XML 문서는 구조화되어 있는 문서이기 때문에 문서 검색과 문서 관리에 대단한 장점을 지니고 있다. 이에 대한 연구와 문서 관리 및 검색에 대한 응용분야 또한 다양해 질 것이다 [4][7].

따라서 현재 XML 분야의 자체 기술 축적을 위해 본 논문에서는 이러한 문서에 대한 구조와 표준 질의로 검색할 수 있는 소규모의 시스템을 구현하였다. XML 문서구조 처리기에 의해 XML 문서를 구조 분석하고, XQL 기반의 질의어 처리기에 의해 문서의 엘리먼트, 속성과 내용을 검색할 수 있도록 구성하였다. 본 문서 검색 시스템은 XML 검색엔진 또는 XML 기반 데이터 베이스 응용분야에 광범위하게 적용될 수 있는 기술이며, 특히 전자 상거래 분야의 상품 카탈로그 관리 및 검색에 응용이 가능할 것으로 예상된다. 또한 XML에 대한 전반적인 기술 축적을 위해 필요한 기술이다. 향후 과제로는 온라인 상에서 대규모의 XML 데이터를 이용하여, 멀티유저용으로 검색 관리할 수 있는 환경으로 확장시킬 필요가 있다.

참고문헌

[1] Jonathan Robie, Joe Lapp and David S chach, "XML Query Language(XQL)", http://www.w3.org/Style/XSL/Group/1998/09/XQL-propose.html, Sep., 1998.

[2] Jonathan Robie, "The Design of XQL", http://www.texcel.no/whitepapers/xql-design.html.

[3] Extensible Markup Language(XML)1.0, W3C Recmmendation, http://www.w3.org/TR/1998/REC-xml-19980210, 1998.

[4] Elliotte Rusty Harold, XML Bible, IDG BOOKS, 1999.

[5] Steven holzner, XML complete, Mcgraw-Hill, 1999.

[6] Edd Dumbill, "The state of XML", http://www.gca.org/papers/xmleurope2000/papers/s37-02.html

[7] Jaideep Roy, Anupama Ramanujan, "Building an XML application", http://www.gca.org/papers/xmleurope2000/papers/s37-02.html

[8] 박서영,신영길,우치수,"XML 컴포넌트 명세서 기반의 컴포넌트 검색기법",정보과학회 논문지,제 27권 제2호, 2000년 2월

[9] 나중찬 외 4명,"XML 문서관리 시스템",정보처리 논문지, 제7권 제2S호, 2000년 2월