XQuery 기반 XML 검색시스템의 구조적인 질의 검색 성능 평가

Performance Evaluation of an XQuery-based XML Retrieval System for the Structured Queries

정영미, 김희섭, 신동현, 경북대학교 대학원 문헌정보학과
양중식, (주)하이엘리더스 투모로우

Youngmi Jung, Heesop Kim, Donghyun Shin,
Graduate School of Library & Information Science, Kyungpook National University
Jungshik, Yang, Hiel Leaders Tomorrow Co., Ltd.

XQuery는 W3C에서 가장 최근에 발표한 XML 질의 언어 표준 초안으로 다양한 형태의 XML 데이터 소스에 폭넓게 적용할 수 있도록 설계되어 있다. 또한 XQuery는 데이터 내용뿐만 아니라 구조 검색에 대해 경로 질의를 이용하여 쉽고 간단하게 처리할 수 있는 특징이 있다. 본 연구에서는 XQuery를 지원하는 XML 검색시스템을 설계 및 구현하고, 개발된 시스템(Litch Search Server)을 INEX 2004를 통해 구조적인 질의에 대한 성능을 평가하여 그 개략적인 결과에 대하여 기술하고 있다.

1. 서론

웹의 광범위한 확산은 다양한 형태의 전자정보를 급속하게 증대시킴과 동시에 이기종 시스템에 구축되어 있는 다양한 문서들을 통합하고 교환해야하는 문제를 발생시켰다. 이에 XML(eXtensible Markup Language) 표준은 확장성과 호환성이 좋으며 정보를 구조화하기에 용이한 특징을 갖고 있기 때문에 차세대 웹 문서 표준으로 각광받고 있다. 이러한 XML 문서표준의 장점은 다양한 분야의 다양한 속성을 지닌 XML 문서의 확장으로 이어지면서 효과적인 XML 문서를 색인하고 검색하기 위해서는 새로운 형태의 데이터 관리 기술의 개발이 불가피하게 되었다. XML 문서의 색인과 검색에 관한 새로운 기술들이 세계의 많은 연구자들에 의해 활발하게 연구 개발되고 차세대 웹 환경을 위해 표준화되고 있다.

XML 문서의 색인과 검색에 관한 연구 분야의 일부분이 XML 문서 검색을 위한 질의 언어 준비에 할당되어왔다. 기존의 이 분야 연구들은 XML 문서들을 관계형 데이터베이스에 저장하고 XPath, XML-QL, XQL, Quilt, LOREL등의 질의 언어를 사용하여 검색하는 방법에 관한 것이 다수였고, 이들 중에서 XPath와 같은 질의 언어는 이미 W3C에서

XML 문서를 위한 표준 질의 언어로 지정되어 있다.

가장 최근에 발표된 질의 언어는 XQuery로 W3C의 XML Query Working Group에 의해 작성되었고 현재 표준화를 위해 노력중이다. XQuery는 처음에는 Quilt의 유용한 기능과 단점을 보완할 질의 언어로 개발되었으나 이후에 다른 질의 언어들이 가지고 있는 몇 가지 유용한 기능들을 추가하여 다양한 형태의 XML 데이터 소스에 폭넓게 적용할 수 있도록 설계되어 있을 뿐만 아니라 XPath의 경로 표현식을 지원하고 있다.

XQuery를 이용한 XML 검색은 문서의 내용뿐 아니라 XML 문서의 구조도 경로 질의를 이용하여 쉽게 처리할 수 있는 표준 질의 인터페이스를 제공함으로써 XML 문서를 위한 유용하고 효율적인 질의언어로 평가되고 있다 (Graaumans. 2004). XQuery에 있어 무엇보다 중요한 것은 점차 많은 기능들을 포함하기 위해 개발 과정 중에 있기 때문에 계속 발전하고 있고 표준화를 위한 노력 또한 계속 진행 중에 있다는 것이다. 이런 노력과 더불어 XML 문서 검색을 위한 질의언어로써 XQuery 유용성과 효율성에 관한 소수의 연구 노력들이 있다.

하지만 실질적인 XQuery를 지원하는 XML 색인 및 검색시스템 개발과 그 성능 평가에 관한 연구는 여전히 미비하다.

이에 본 연구에서는 XQuery를 지원하는 XML 검색시스템을 설계 및 구현하여 세계적인 XML 검색 연구그룹인 INEX(INitiative for the Evaluation of XML retrieval)를 통해 구조적인 질의(CAS: Content-and-Structure)에 대한 성능을 평가하였다.

2. 관련 연구 및 기술

2.1 관련 연구

XQuery가 최근에 작성된 것이고, 여전히 표준화 과정 중에 있기 때문에, 이 분야 연구의 대부분은 XQuery의 새로운 기능 추가 및 XML 문서를 검색하기 위한 질의 언어로써의 유용성과 효율성에 관한 것이다.

특히 이종 데이터베이스 검색에서 XQuery 질의 처리기를 사용하면 데이터베이스 내에 저장된 다양한 XML 문서의 다양한 객체 유형을 일괄적으로 질의할 수 있도록 해준다.

Wiegand의 연구는 전통적인 DBMSs에서 이종 XML 문서들을 저장하고 검색하는데 있어서 XQuery를 사용하는 것이 편리하고 유용하다는 것과 또한 웹기반 정보시스템내의 이종 XML 문서를 저장하고 질의하는 것에도 유용하다고 제시하고 있다(Wiegand, 2002). 기존의 DBMS 질의 언어들은 이종 데이터 베이스 내의 서로 다른 구조를 가지는 XML 문서를 일괄적으로 질의하는 것이 복잡하고 어려웠기 때문에 XML 문서 구조가 다르거나 변화가 생길 때마다 XML 문서를 재 색인해야할 필요가 있었다. 그러나 내용과 특정적인 구조 검색이 편리한 XQuery 질의 처리기를 이용하면 XML 문서내의 객체들의 유형이 변화할 때마다 새롭게 저장할 필요없이 이종 XML 문서내의 다양한 객체 유형을 일괄적으로 질의하는 것이 가능하다.

또한 최근의 XQuery, SQL/XML 그리고 XSLT 세 개의 XML 질의 언어들간 유용성에 대한 질적인 비교 실험에서 XQuery를 사용한 것이 다른 질의 언어보다 저장과 검색에서 가장 성능이 우수한 것으로 나타났고, XSLT가 SQL/XML보다 성능이 우수한 것으로 나타났다(Graaumans, 2004).

XQuery는 표준화 과정 중에 있고 계속적인 개선의 여지가 있지만, 앞에서 언급된 그 유용성은 실제로 XQuery 질의 처리기를 사용한 XML 문서의 저장 및 검색시스템의 개발 및 성능 평가에 관한 연구들을 통해 더욱 명백해

질 수 있다. 기존의 XML 문서 질의 언어들이 주로 관계형 데이터베이스(RDBMS: Relational Database Management System)에서의 사용이 논의되었던 것처럼, 최근 이 분야 연구에서도 RDBMS에서 XQuery 질의 처리기를 이용한 XML 저장 및 검색시스템에 관한 논의들이 있다. RDBMS를 이용하여 XML 문서를 효과적으로 저장하고 검색하기 위해 XQuery 질의를 채택하는 것은 XQuery 질의를 SQL 질의로 변환하여 얻은 SQL 질의의 결과를 XML 문서로 재구성하는 방법을 사용할 수 있다. 이런 시스템 구성은 XML 질의 처리에 필요한 시간을 단축시키는 것으로 나타났고(김희섭, 2004; 박명제 외, 2002) XML/EDI 저장 시스템을 위한 RDBMS에서 XQuery 사용하는 것 또한 성능을 향상시키는 것으로 나타났다(김지선, 홍의경, 2004).

2.2 관련 기술

① XPath 개요

XPath는 문서에서 노드의 하위 집합을 선택하기 위해 단순한 구문을 사용하는 XML 질의 언어이다. 현재 W3C에서는 XML 문서 검색을 위한 질의 언어로 XPath 1.0을 권고하고 있다. XPath는 URL 경로 표기법을 사용하여 XML 문서의 계층적인 구조를 논리적으로 검색하는데, 문서의 논리적인 구조를 위한 경로뿐만 아니라 조건을 지정함으로써 노드의 하위 집합을 검색할 수 있다. 하지만 XPath는 XQuery가 순서가 있는 노드 시퀀스를 반환하는 것과 달리 하나의 노드 세트를 반환한다. 그래서 XQuery 1.0은 XPath 2.0의 표현식을 개선하기 위해 XPath 2.0의 모든 표현식을 지원한다.

② XQuery 개요

XQuery 1.0는 2001년 2월에 XML Query Working Group에서 실무 초안을 작성하여 발표했으며 현재도 이 작업 초안은 계속 개정 중에 있다. XQuery는 다양한 구조와 외형을 가진 몇 개의 표현이 지원되고, XQuery를 사용한 검색식은 하나 이상의 표현식으로 구성된다. XPath 2.0에 근거한 XQuery는 Path 표현과 FLWOR(For, Let, Where, Order by, Return; <그림 1> 참조)로 표현된다. 여기에서 'For'는 선택한 노드에 대해 바인딩을 작성하고 'Let'은 단일 바인딩을 작성한다. 'For'는 중첩이 가능한데 이런 특징은 XQuery가 결과 시퀀스를 통해 반복될 때 루프(roof) 작성시 유용하게 사용될 수 있다. 'Where'는 필터링을 위한 구문이고 정렬은 'Order by'에 의해 가능하다. 'Return'은 돌려줄 결과물에 관한 것을 지정한다.

```
for $d in fn:doc("depts.xml")/depts/deptno
let $e := fn:doc("emps.xml")/emps/emp[deptno = $d]
where fn:count($e) >= 10
order by fn:avg($e/salary) descending
return
    <big-dept>
      {
      $d,
      <headcount>{fn:count($e)}</headcount>,
      <avgsal>{fn:avg($e/salary)}</avgsal>
      }
    </big-dept>
```

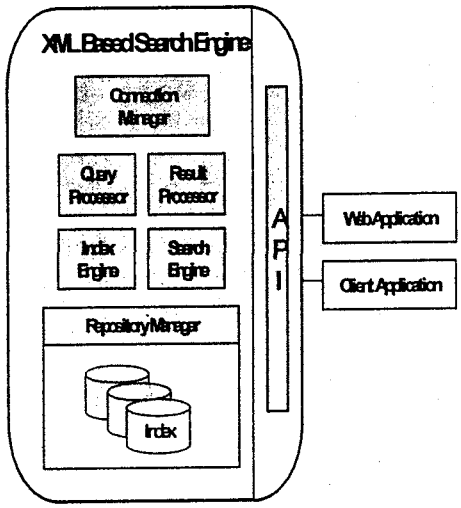
<그림 1> FLWOR 표현

3. 시스템 개요

3.1 시스템 구조

XML을 저장 및 검색하기 위한 개발한 검색시스템(Litch Serach Server라 명함)의 전체 구조는 <그림 2>와 같이 API, Connection Manager, Query Processor, Index Engine,

Search Engine, Result Processor, Repository Manger 그리고 Index File system 등으로 구성되어 있다.



<그림 2>전체 시스템 구조도

각 시스템의 구조별로 기능을 살펴보면 다음과 같다.

① Connection Manager

Connection Manager는 외부 시스템 또는 애플리케이션이 검색 엔진에 색인/검색 요청을 할 수 있도록 접속관리를 담당하는 관리자로 검색 엔진이 시스템에 적하(load)될 때 실행된다. 또한 이것은 다수에 의한 동시 접속을 빠르고 안정되게 처리할 수 있도록 설계되었으며 검색엔진에 요청되는 명령들을 전달하고 처리된 결과를 전달해주는 역할을 한다.

② Query Processor

Query Processor는 Connection Manager를 통해 전달된 질의를 분석하여 그에 따른 요청을 Index Engine 또는 Search Engine에 전달해주는 역할을 담당한다. Search Engine에 전달되는 검색 요청 질의 언어는 XQuery로 XML문서에 구조적 질의를 간결하고 이해하기 쉽도록 기술할 수 있다. Query Processor는 검색 질의어로 사용되는 XQuery와 색인 또는 기타 정보를 요청할 수 있도록 확장 질의를 처리할 수 있다. 또한 이것은 XQuery Parser를 내장하고 있어 XQuery에 들어 있는 의미를 분석하여 Search Engine이 정확한 검색을 수행하도록 도와주고 계속적인 검색시스템 개선을 위해 색인 정보의 추가/수정/삭제 등에 대한 명령을 분석하여 Index Engine에 전달해준다.

③ Index Engine

Index Engine은 검색 대상이 되는 XML 문서를 처리하여 검색이 가능하도록 구조적으로 저장하는 역할을 담당한다. XML Parser를 이용해 구조적인 정보를 추출하는데 그 정보들 중에는 Repository에 저장되는 Term, Element, Attribute 등의 XML 문서의 노드 정보들을 추출한다.

이러한 정보들은 최상의 검색 속도와 검색된 문서의 빠른 출력을 위해 적절한 조합을 거친 후 저장된다.

④ Search Engine

Search Engine은 Query Processor를 통하여 전달된 XQuery에 대한 내용을 단계별로 실행하는 역할을 담당한다. XQuery는 'For', 'Where', 'Order by', 'Return' 형식의 구문을 처리하는데 우선 'For' 구문에 속해있는 Bases에 해당하는 $값을 구한 후 그 Bases값을 기준으로 'Where' 구문을 수행하여 결과를 출력 받는다. 검색 결과에 대하여 'Order by'를 처리한 후 검색 결과에 'Return' 값을 생성하기 위하여 검색 결과 노드 리스트를 기준으로 Return 될 노드에 대한 부모-자식 관계를 계산하고 기존의 XML 문서의 일부를 추출하여 결과 세트를 재구성한다.

Term에 대한 검색은 Inverted Index 구조를 이용하여 검색하고 XPath가 들어간 검색에 대해서는 Term이 발생된 Node와 검색하고자 하는 구조적인 경로에 대한 연산을 실행하여 최종 검색 결과 리스트를 생성한다.

⑤ Result Processor

Result Processor는 Query Processor를 통하여 각각의 엔진 또는 Processor에 전달되어 생성된 결과 정보를 Connection Manager에 보낼 수 있도록 결과 정보에 대한 재구성을 담당하는 역할을 한다. 검색의 경우에 검색 결과에 대한 페이징 처리 또는 검색 결과에 대한 로그 처리 기능도 지원한다.

⑥ XML Storage Manager

Storage Manager는 Index Engine에서 처리된 XML 구조 정보를 Storage에 저장시키고 불러오는 작업을 담당한다. XML Storage는 XML 문서의 구조적 정보와 검색 Term을 저장하고 있다. 저장된 구조적인 정보는 B+ Tree 기반이며 이것은 Path와 DOM Tree를 통해 문서의 구조적인 검색을 가능하게 한다.

검색 단어에 대한 접근은 Inverted Index를 통해 해당하는 문서 또는 노드에 접근하는 것이 가능하다.

3.2 시스템의 특징 및 장점

XQuery를 지원하는 본 시스템은 어떤 유형의 XML 문서가 들어와도 재 색인과정이 필요없이 구조적인 검색이 가능하도록 되어 있는 것이 특징이다. 기존의 XML 검색시스템들은 검색하고자 하는 엘리먼트를 설정한 후 검색하고자 하는 엘리먼트에 대해서만 색인하는 방식을 취했고 검색 엘리먼트가 변할 경우에는 검색 엔진의 일부를 수정해야 하는 단점이 있었다. 본 시스템은 초기 색인시 문서의 모든 검색 정보를 담고 있기 때문에 검색 질의 변경만으로 검색이 가능하며 검색 결과에 대한 자유로운 출력도 가능한 장점이 있다. 검색의 성능향상을 위해 스태밍과 불용어사전을 통해 정확하고 불필요한 정보를 배제 하여 빠르고 정확한 검색을 유도하였다.

4. XML 문서의 저장 및 질의 처리

4.1 XML 문서의 저장

본 시스템은 INEX 2004에서 제공하는 총 12,107개의 논문기사로 된 XML 문서의 집합을 저장하고 색인하였다. 사용된 색인작성 방법과 저장구조는 아래와 같다.

① 색인작성 방법

색인작성은 SAX 기반 Parser를 사용하였고 문서의 구조정보를 PreOrder 순서로 순회하면서 각각의 'Element'와 'Attribute' 값을 읽어서 각각의 테이블에 저장하였다. XPath는 Path Table에, 'Element'와 'Attribute'의 이름 및 PreOrder에 해당하는 순서 정보는 Dom Index Table에, 그리고 'Attribute'와 'Element'의 값은 스태밍과 불용어를 처리하여 저장하였다.

② 저장 구조

색인처리가 끝난 후 처리된 문서의 정보는 색인기반 테이블과 B+ Tree의 두 가지 형태로 저장된다. 색인기반 테이블에서는 주로 문서

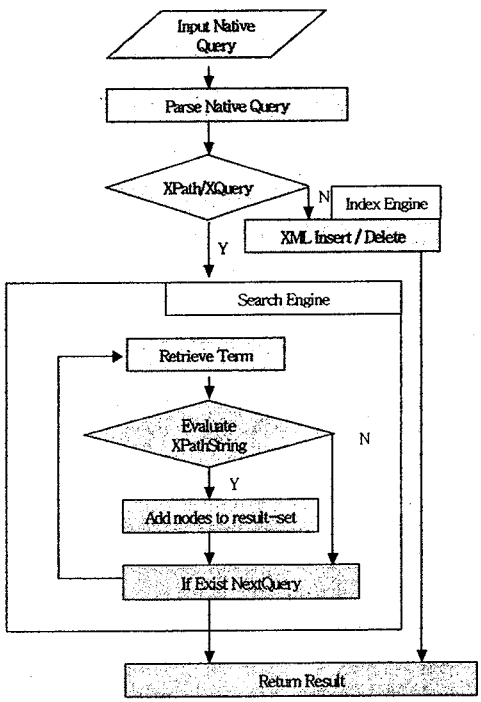
<표 1> 색인 테이블과 저장 유형

Term Name	Storage Type
Document Table PathIndex Table Term Table	Index Table
TermIdex Table	B+ Tree
DOMTable	B+ Tree

의 ID를 관리하는 테이블이 저장되어 있으며 빠른 저장 및 검색을 필요로 하는 테이블은 B+ Tree에 저장하였다. 색인기반 테이블은 문서의 고유한 ID를 생성하여 저장하는 Document Table, XML 문서의 Path 정보를 ID별로 관리하는 PathIndex Table, 그리고 검색 단어를 ID로 저장하여 관리하는 Term Table이 있다. 그리고 Term에 대한 위치와 노드 정보들이 들어 있는 TermIndex Table과 문서의 구조 정보가 저장되어 있는 DOMTable은 B+ Tree에 저장되었다. <표 1>은 본 시스템의 실험을 위해 구축된 색인 테이블과 저장 유형이다.

4.2 질의 처리

구현된 검색 시스템에서의 질의 처리에 따른 검색과정을 살펴보면 <그림 3>과 같다.



<그림 3> 질의에 따른 검색시스템 흐름도

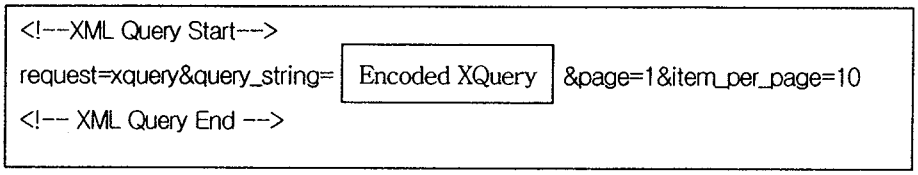
웹 브라우저를 통해 입력되는 질의는 XQuery 기반으로 <그림 4>와 같이 표현된다.

```
for $t in /books/*/article
where ( $t/body/*/#TEXT = [ Computer ] )
order by $t/fno/#TEXT
return
  <FM> { $t/fno  } </FM>
```

<그림 4>XQuery의 한 예

<그림 4>와 같은 XQuery가 웹브라우저를 통해 검색엔진에 전달되기 위해서는 <그림 5>와 같이 XQuery에 부가적인 정보가 추가되어 검색엔진에 전달된다. 이때 XQuery가 인코딩되어 전달되는데 이는 웹을 통해 질의가 전달될 때 XQuery에 들어있는 연산자 '&, =' 등과 중첩되어 오류가 발생될 것을 방지하기 위한 것이다. 질의어 중 'request' 뒤에 나오는 'xquery' 구분자는 검색을 위한 식별자이며 다음의 페이지 정보는 검색결과의 출력에 관한 페이지 지정이다. 'request'의 유형이 'request=doc_insert'인 것은 색인될 XML 문서의 추가 명령이고, "request=doc_delete"는 색인된 XML 문서를 삭제하라는 명령이다.

검색 요청 구문이 입력되었을 때 Search Engine은 <그림 4>와 같은 XQuery의 'For' 구문에 있는 $t의 값인 /books/*/article을 Basis로, XPathString을 Key 값으로 DOM Table을 통해 XPathString에 속하는 Node List를 가지고 온다. 'Where'에 있는 복합 질의문을 단일 질의문으로 나눈 후 단일 질의문을 하나씩 실행하면서 연산을 수행한다. 단일 질의문에 있는 XPath 값을 이용하여 검색 Term Table에 해당하는 Node List를 얻어온 후 'for'에서 계산한 Node List와 비교하여 서로 일치되는 값들만 골라낸다. 이런 과정을 통해 'Where' 구문만으로도 최종 검색 정보를 얻을 수 있다.



<그림 5> Native 질의의 한 예

하지만 'Order by'는 검색 결과 정렬에 관한 정보가 추가된 경우로 'order by $t/fno/#TEXT'는 '/books/*/article/fno/#TEXT'의 경로에 fno, element, vlaue의 순서대로 정렬한다.

정렬을 위해서는 정렬하기 위한 XPath을 Key로 하여 DOM Table에서 Node List를 받아온 후 'Where'에서 나온 결과와 비교하여 순서를 재배열하여 결과 리스트를 생성한다. 'Return'에 표현된 결과 문서의 재구성은 검색 결과로 생성된 Node List와 'Return'에 있는 XPath 정보를 이용하여, 'Return'에 표현된 형태로 XML 문서 형태를 표현해 준다.

5. 실험 및 성능 평가

5.1 실험 환경

구현된 XML 검색시스템을 평가하기 위해 XML 문서로 구성된 대규모 테스트 컬렉션을 제공하는 INEX 2004에 참여하여 실험하였다.

INEX는 2002년부터 DELOS(A Network of Excellence for Digital Libraries)에 의해 조직되었고 INEX 2002, INEX 2003 그리고 INEX 2004가 진행되었다. INEX의 목적은 세계 각국의 XML 검색 시스템 연구팀들이 자발적으로 참여하여 그 연구팀들이 개발한 XML 검색 시스템의 효과성을 통일된 적합성 평가 기준 및 방법을 통해 평가하고 그 결과들을 비교할 수 있도록 대규모의 XML 테스트 컬렉션을 구축하고 제공하기 위한 것이다. INEX 2004에서 제공한 테스트 컬렉션은 XML 문서들의 집합, 토픽들 그리고 적합성 평가로 이루어진다.

① XML 문서들의 집합

XML 문서들의 집합은 XML로 마크업 된 IEEE Computer Society의 1995-2002년의 12개 학술잡지에 실린 12,107개의 논문기사로 구성되어 있다. 각 각의 논문기사들은 복잡한 XML 구조를 가지고 있으며 191개의 다른 DTD로 구성되어 있고 한 논문 기사당 평균 1,532개의 XML 노드를 가지고 있으며 평균 노드 깊이 6.9이다. 전체 문서 집합의 크기는 494Mbyte이다.

② 토픽

XML 문서에 대한 실제 이용자의 요구를 표현한 토픽들이 참여하는 연구팀들에 의해 수집되고 선택된다. INEX 2004에서 선택된 토픽은 내용 검색만을 요구하는 CO(Content-Only) 토픽 40개와 내용과 구조 검색을 요구하는 CAS(Content-And-Structure) 토픽 35개이다. 토픽을 기반으로 한 질의 생성방법은 수작업과 자동추출 둘 다가 가능하다. 본 연구에서는 XML 문서의 내용과 구조 모두를 포함한 CAS 토픽만을 실험대상으로 하였으며 질의 생성방법은 수작업으로 이루어졌고 질의 언어는 XQuery를 사용하였다. <그림 6>은 CAS 토픽의 한 예이다. 'topic_id=132'는 초록에 'classification'을 포함하는 논문기사 중 'experiment compare'를 포함하고 있는 섹션을 요구하는 토픽이다. 각 각의 토픽은 검색을 위해 <그림 4>와 같이 XQuery를 사용하여 내용과 구조를 모두 포함한 질의어로 표현된다.

③ 적합성 평가

참여 연구팀들이 제출한 결과 세트들은 토픽별로 풀링(pooling)되고, INEX 온라인 평가 시스템을 사용하여 참여 연구팀들은 각각 할당받은 토픽의 검색결과들에 대해 적합성을 평가한다. INEX 2004의 토픽에 대한 결과 문서들의 적합성 평가 기준은 <그림 7>과 같이 망라성(e: exhaustive)과 특정성(s: specificity)의 두 가지 측면에서 이루어진다.

적합성 평가를 토대로 XML 검색시스템의 성능평가 결과는 재현률-정확률 그래프로 제시되는데 재현률-정확률 그래프를 위한 INEX 2004의 수량화는 다음의 $f_{strict}(e,s)$와 $f_{gen}(e,s)$에 의해 계산된다. 재현률-정확률 그래프의 'generalised'는 부분적으로 망라적이고 특정적인 적합 문서들에 대한 보간을 위해 계산되었다.

$$f_{strict}(e,s) := \begin{cases} 1 & \text{if } e=3 \text{ and } s=3, \\ 0 & otherwise \end{cases}$$

$$f_{gen}(e,s) := \begin{cases} 1 & \text{if } (e,s)=(3,3), \\ 0.75 & \text{if } (e,s) \in \{(2,3),(3,\{2,1\})\}, \\ 0.5 & \text{if } (e,s) \in \{(1,3),(2,\{2,1\})\}, \\ 0.25 & \text{if } (e,s) \in \{(1.2),(1,1)\}, \\ 0 & \text{if } (e,s)=(0,0). \end{cases}$$

5.2 성능 평가

INEX 2004에서 제시된 CAS 토픽의 수는 총 35건이었고 각각의 토픽에 대해 수작업으로 XQuery를 작성하여 검색하였다. 검색 결과 검색시 유효건수 5건 이상 1,500건 이하를 전제조건으로 하고 조건을 불만족 시키는 2개의 토픽을 제외하고 33개의 토픽에 대한 평균 검색건수는 76.90개가 나타났다.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE inex_topic(View Source for full doctype...)>
<inex_topic topic_id="132" query_type="CAS" ct_no="41">
<title>//article[about(.//abs,classification)]//sec[about(.,experiment compare)]
</title>
<description>We are interested in finding the sections about experiment design
and experiment results of different classification methods. We hope that the
abstract should mention the classification methods.</description>
<narrative>To be relevant, the returned component needs to be a section. It
different classification methods. Sections about comparing different
classification methods are also preferred.</narrative>
<keywords>classify, experiment, compare, result</keywords>
</inex_topic>
```

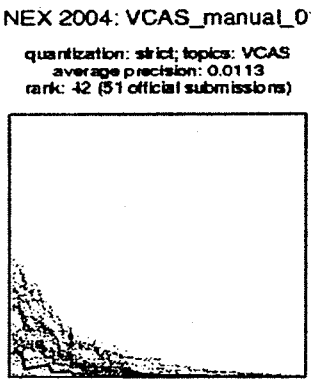
<그림 6> INEX 2004 CAS 토픽의 한 예

Exhaustivity / Specificity	Highly exhaustivity(E3)	Fairly exhaustivity(E2)	Marginally exhaustivity(E1)
Highly specific(S3)	●	●	○
Fairly specific(S2)	◕	◕	◎
Marginally specific(S1)	◕	◕	◎

<그림 7> 적합성 평가 기준

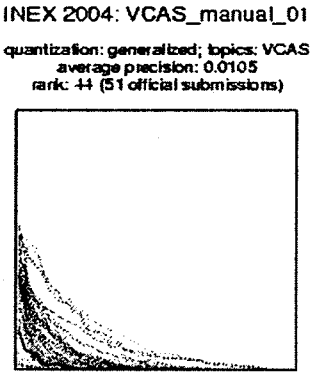
본 연구에서 구현된 XML 검색시스템의 CAS 토픽에 대한 'Strict'한 평균 정확률은 0.0113으로 나타났고 재현률-정확률 그래프는 <그림 8>와 같이 나타났다.

I



<그림 8> CAS 검색결과_Strict

'Generalized'한 계산에 의한 평균 정확률은 0.0105로 나타났으며 이것의 재현률-정확률 그래프는 <그림 9>와 같이 상대적으로 타 연구팀들의 검색 성능보다 낮은 결과를 보였다.



<그림 9>CAS 검색결과_Generalized

6. 결론

최근 웹 상에는 XML 형태로 저장된 자료가 급증하고 있고 이와 더불어 XML이 가지는 유연성으로 인해 이질적인 XML 데이터 소스들 또한 더욱 다양해지고 있다. 이런 환경에서 XML 문서를 효과적으로 저장하고 검색하기 위해 XML 질의 언어에 대한 연구들이 활발하게 진행되어왔고 그 중 가장 최근의 결실은 XQuery이다. 본 연구에서 구현한 XQuery 질의 처리기를 사용한 XML 검색 시스템은 어떤 유형의 XML 데이터 소스가 들어와도 재 색인할 필요없이 구조적인 검색이 가능하도록 되어 있다. 또한 XQuery를 사용하여 내용과 뿐만 아니라 구조적인 제한을 포함하는 질의 생성을 비교적 간단하고 쉽게 사용할 수 있기 때문에 검색이 용이하다. 하지만 INEX 2004를 통한 검색시스템의 성능을 평가한 결과, 그 결과값이 다소 만족스럽지는 않았다. 이런 결과는 현재 평가된 검색 시스템이 적합성 정도에 따라 검색된 결과들을 순위화하는 알고리즘을 포함하지 않았고, 검색문에서 가장 기본적인 불리언 연산자의 사용조차 허락하지 않았으며 또한 엄격한 경로설정에 의해 XQuery의 유용성을 제대로 활용하지 못했다는 제약점들을 지니고 있었기 때문인 것으로 판단된다. 계속적인 연구를 통해 이런 제약점들이 극복된다면 XQuery를 지원하는 XML 검색시스템은 저장과 검색 측면 둘 다에서 효과적일 것이다. 뿐만 아니라 XQuery 자체가 지니는 장점들로 인해 앞으로 XML 문서가 계속 증가하고 점차 서로 다른 데이터 소스간 경계가 모호해짐에 따라 XQuery를 지원하는 XML 검색 시스템은 점점 유용해 질 것이다.

참 고 문 헌

김지선, 홍의경. 2004. 「관계 데이터베이스에서 XQUERY 질의 처리기를 이용한 XML/EDI 구축 시스템의 설계 및 구현.」 *Journal of the Institute of Industrial Technology*, vol.12 : 89-95.

김희섭. 2004. 「Retrieval Performance of XML documents Using Object-Relational Database.」 정보관리학회지, 22(2): 189-210.

박명제 외. 2002. 「관계형 데이터베이스와 XQuery를 이용한 XML 문서의 저장 및 검색 시스템.」 한국데이터베이스 학술대회 논문집, 283-290.

Graaumans, Joris. 2004. "A Qualitative Study to the Usability of Three XML Query Language." *Proceedings of the Conference on Dutch Directions in HCI, Amsterdam.*

INitiative for the Evaluation of XML Retrieval 2004 homepage. [cited 2004. 11. 21] <http://inex.is.informatik.uni-duisburg.de:2004/index.html>

Robert W. P. et al. 2002. "A Survey in Indexing and Searching XML Document." *Journal of The American Society for Information Science and Technology*, 53(6): 415-437.

Wiegand, Nancy. 2002. "Investigating XQuery for Querying Across Database Object Types." *SIGMOD Record*, 31(2): 28-33.

W3C XQuery 1.0: An XML Query Language. [cited 2005. 7. 14] <http://www.w3.org/TR/xquery/>

XML Path Language(XPath) Version 1.0. [cited 2005. 7. 12] <http://www.w3.org/TR/xpath>

XQuery Tutorial. [cited 2005. 7. 10] <http://www.w3schools.com/xquery/>

XQuery 1.0 and XPath 2.0 Formal Semantics. [cited 2005. 8. 6] <http://www.w3.org/TR/xquery-semantics/>