관계형 모델에서 XML 변경과 전문 검색을 지원하기 위한 역 인덱스 구축 기법

천 윤 우† · 홍 동 권††

요 약

최근 산업체를 중심으로 XML 전문 검색과 XML 문서의 변경 기능에 대한 표준의 시도가 활발히 이루어지고 있다. XML 질의어에서의 전문 검색 기능은 매우 중요한 부분을 차지한다. XML 문서는 관계형 테이블과는 달리 문서의 구조가 복잡하며 때로는 매우 불규칙하다. 이런 상황에서의 검색은 부분적인 정보를 최대한 활용해야 하는 전문 검색이 일반적인 구조적 검색보다 매우 중요한 역할을 한다. 본 논문은 XML 데이터를 관리하기 위하여 관계형 모델을 사용하는 환경에서 XML 문서의 변경과 다양한 형태의 전문 검색을 동시에 지원하기 위한 방안으로 효율적인 역 인덱스 구축 기법을 제안한다. 본 논문에서 제안한 방법은 인덱스 크기의 큰 변화 없이 역 인덱스를 구축하며, 대용량의 XML 문서의 다양한 전문 검색 기능을 성능의 저하 없이 지원한다. 또 XML 문서의 부분적인 변화에 역 인덱스의 변경이 기존의 방법들에 비해서 급격히 줄어든 좋은 성능을 보인다.

Inverted Indexes for XML Updates and Full-Text Retrievals in Relational Model

Yun-Woo Cheon† · Dong-Kweon Hong††

ABSTRACT

Recently there has been some efforts to add XML full-text retrievals and XML updates into new standardization of XML queries. XML full-text retrievals plays an important role in XML query languages. Not like tables in relational model an XML document has complex and unstructured natures. We believe that when we try to get some information from unstructured XML documents a full-text retrieval query is much more convenient approach than a regular structured query. XML update is another core function that an XML query have to have. In this paper we propose an inverted index to support XML updates and XML full-text queries in relational environment. Performance comparisons exhibit that our approach maintains a comparable size of inverted indexes and it supports many full-text retrieval functions very well. It also shows very stable retrieval performance especially for large size of XML documents. Foremost our approach handles XML updates efficiently by removing cascading effects.

키워드 : XML 변경(XML Update), XML 전문 검색(XML Full-Text Retrieval), 관계형 모델(Relational Model), 역 인덱스(Inverted Index)

1. 서 론

XML(Extensible Markup Language)은 인터넷상에서 디지털 정보를 표현하고 교환하기 위한 표준으로 확고한 자리를 굳혔다[1]. 기존의 표현 중심의 HTML(HyperText Markup Language)과는 달리 XML은 데이터의 의미를 표현한다는 점에서 프로그램을 사용하여 데이터를 분석할 수 있는 장점을 가지고 있다. 이는 인터넷상의 데이터를 다양한 형태의 프로그램으로 자동 처리할 수 있는 새로운 컴퓨팅 패러다임을 제공하는 획기적인 것으로 평가되고 있으며 이러한 변화를 바탕으로 현재 XML 데이터를 여러 응용 분야에서 활용하려는 노력이 활발히 이루어지고 있다. 그 예로 XML은 자동차 회사들이 모든 자동차에 대한 정보를 제공하는데 사용되기도 하고 도서관에서 도서 목록을 만들거나 전자상거래(E-Commerce) 웹 사이트의 다양한 제품을 소개하는데 사용되고 있다. 이렇듯 다양한 분야에서의 사용으로 인해 XML 데이터의 구조정보 표현 기법 및 저장 시스템에 대한 연구가 필요하게 되었고 그 결과 Lorel, XML-QL, XPath, Quilt, XQuery 등과 같은 XML 질의어에 대한 표준도 제안되고 있다[2-6].

기존 데이터베이스 기술에서 살펴본 XML은 많은 새로운 어려운 문제점을 만들어내고 있다. 최근 20~30년 동안

※ 본 연구는 한국과학재단 목적기초연구(R01-2003-000-10001-0) 지원으로 수행되었음.
† 준 회 원 : (주)데이타뱅크 시스템즈
†† 종신회원 : 계명대학교 컴퓨터공학 교수
논문접수 : 2003년 12월 5일, 심사완료 : 2004년 4월 20일

기술적, 상업적으로 급속한 발전을 이룬 관계형 데이터베이스의 테이블과는 다르게 XML은 반구조적(semi-structured) 문서로 표시된다. 따라서 XML 문서에 대한 질의는 관계형 데이터베이스에서의 질의와는 달리 훨씬 더 다양하고 자유스러운 형식을 가지게 되어 기존의 관계형 모델의 질의어인 SQL의 기술을 확장하여 사용할 것인지, 아니면 새로운 질의어를 만들어 사용할 것인지에 대한 많은 연구가 진행되고 있다. 특히 반구조적인 XML문서의 특성에 따른 새로운 질의 기능의 추가에서 전문 검색(full-text search) 기능은 W3C(World Wide Web Consortium)에서 최근 몇 년간의 요구 사항을 검토하였고 2003년부터 본격적인 표준화 작업을 시작하여 2003년 2월에 2건의 "Working Draft"를 발표하였다.

본 논문에서는 최근 W3C에서 채택하고 있는 XQuery의 전문 검색 기능을 알아보고, 그 기능을 기존의 관계형 기술을 활용하는 방식을 채택하여 구현하는 기법을 연구하고, 연구 결과를 활용하여 관계형 모델을 활용한 전문 검색 기능의 구현 기법들에 대한 장단점을 알아본다. 또한, 현재까지의 연구들에서 거의 고려되지 않았던 XML 문서의 변경을 지원할 수 있는 기법에 대해서도 알아본다. 본 논문의 구성은 다음과 같다. 2장에서는 본 논문과 관련된 연구들에 대해 언급하고 3장에서는 본 논문에서 제안하는 인덱스 구조의 설계와 구현에 대해서 언급하고 4장에서는 구현된 시스템에 대한 성능 평가를 위한 모델을 제시한다. 5장에서는 제시된 성능 평가 모델을 바탕으로 실험 결과를 분석하고 마지막으로 6장에서는 결론을 제시한다.

2. 관련 연구

XML 모델과 관계형 모델은 많은 차이점이 존재하며 이에 따라 데이터의 표현, 질의어의 형식 및 기능이 매우 다르다. 그렇지만 이미 성숙 단계에 이른 관계형 기술을 최대한 활용하여 관계형 모델을 사용한 XML 문서의 저장 기법, XML 질의를 SQL로 변환하는 방법에 대한 연구가 진행되어 왔다. 하지만 XML 질의에서의 전문 검색 기능은 2003년 2월에 처음으로 "Working Draft"가 발표된 만큼 그 기능과 구현 방법에 대한 연구는 아직 많이 진행되지는 않고 있다.

2.1 XML 질의

데이터를 그래프 형태로 보는 XML 모델과 데이터를 테이블의 형태로 보는 관계형 모델사이에는 많은 차이점이 있다. 따라서 관계형 모델의 질의어로 사용되는 SQL을 XML 데이터를 위한 질의어로 그대로 사용하는 것은 매우 어렵다. 이로 인해 새로운 질의어가 W3C에서 XQuery 표준화로 진행되고 있으며 XQuery 구문 형태는 XML 스키마, XSLT, XPath의 내용들에 의해서 특히 IBM의 Quilt 연구 내용을 많이 수용하고 있다[4-8]. XQuery는 초기에는 데이터의 검색 부분만 강조했으나 점차 전문 검색의 기능도 고려하고 있으며 2003년 2월에 "XQuery and XPath Full-Text Requirement"와 "XQuery and XPath Full-Text Use Cases" 2건의 Working Draft가 발표되었다[9, 10]. 이 2가지의 문서는 향후 XML Query에 추가될 전문 검색(full-text retrieval) 기능에 대한 부분이 포함되어져 있다.

2.2 관련 기법 분석

최근 XML의 구조적 특성을 사용한 전문 검색에 대한 연구가 활발히 시작되고 있다[11-13]. 특히 관계형 데이터베이스를 활용한 XML Query 전문 검색에 관한 관련 연구[11]에서는 각 엘리먼트마다 서로 다른 테이블을 형성하는 방식을 사용함으로써 같은 단어에 대해서 루트 엘리먼트에서부터 실제 그 단어가 존재하는 엘리먼트까지 매 엘리먼트마다 역 인덱스(Inverted Index)를 생성하게 된다. 또한 이진 테이블 방식을 사용한 자체 조인(Self-Join)으로 엘리먼트 사이의 종속 관계를 처리하기 때문에 복잡한 경로의 XQuery에서 상당한 부담이 따른다. 관련 연구[12]에서는 XML Query의 전문 검색 중에서 포함 질의(Containment query)에 대해서 관계형 테이블을 이용한 해결 방식을 제안하고 있다. 이 방식은 XML 문서의 텍스트와 엘리먼트 각각에 대해서 키워드가 나타나는 위치를 XML 문서의 시작부터 몇 번째 단어인지를 계산하여 (begin, end)의 숫자를 부여하고 이 숫자를 포함한 역 인덱스를 생성한다. 이 숫자는 자식-부모 관계를 파악하기 위한 모든 포함 질의에 그 값을 비교하는데 사용된다. 하지만 이 값은 문서의 변경이 발생할 경우 변경 부분 이후에 있는 모든 원소에 대해 (begin, end)를 다시 계산하여야 하며 역인덱스의 많은 부분에 대한 변경이 필요하다. 가장 최근에 발표된 관련 연구[13]에서는 루트 엘리먼트에서 최종 단말 엘리먼트까지의 경로가 특정 역 인덱스에 저장되기 때문에 다른 기법들에서 나타난 문제점은 상당히 극복되었다. 그러나 이 기법에서도 역시 XML 문서의 변경 처리에 있어서 변경된 엘리먼트 이후의 엘리먼트에 대한 역 인덱스를 재생성해야 하는 또 다른 문제점이 존재한다. 이에 본 논문에서는 이러한 문제점들을 해결하여 보다 효율적으로 XML 질의와 변경을 처리할 수 있는 역 인덱스 기법을 제안한다.

3. XML 변경과 전문 검색을 위한 역 인덱스 설계 및 구현

관계형 데이터베이스에 XML 문서를 저장하는 여러 기

법들 중에서 XML 문서를 테이블의 특정 컬럼에 저장하고, 그 컬럼에 대한 역 인덱스를 생성하는 기법은 엘리먼트, 애트리뷰트, 그리고 중첩된 엘리먼트 사이에서의 계층 구조를 효과적으로 관리하여 정확한 질의 검색을 할 수 있다는 장점을 가지고 있다. 또한, 전체 XML 문서가 저장된 특정 컬럼을 통해서 XML 문서 전체를 결과로 쉽게 반환할 수 있다. 이러한 이유로 본 논문에서는 이 기법을 사용해서 XML 문서를 저장하며 (그림 1)은 본 논문에서 제안하는 기법을 설명하기 위한 예제 XML 문서이다. 본 논문에서는 2장에서 제시된 관련 기법[11-13]들이 전혀 고려하지 못했던 XML 문서의 변경과 다양한 종류의 전문 검색 기능을 동시에 효과적으로 처리할 수 있는 역인덱스 기법인 XFTS (XML Full-Text Search)를 제안한다.

```
<books>
  <book>
    <title> Data on the Web </title>
    <author>
      <family>Kim</family><given>Young Chul</given>
      <family>Lee</family><given>Eun Suk</given>
      <family>Hong</family><given>Gil Dong</given>
    </author>
    <summary>
      This book mainly mentions
      <keyword>semistructured data</keyword>
      <keyword>database</keyword>
      <keyword>XML</keyword>
    </summary>
  </book>
</books>
```

(그림 1) 예제 books XML 문서

3.1 XFTS 역 인덱스 기법의 구현

본 논문에서 제안하는 역 인덱스 기법 XFTS는 다음의 <표 1>과 같이 4개의 테이블을 사용한다.

〈표 1〉 XFTS 역 인덱스 기법을 위한 테이블

XML_Documents (id, docname, isidx, contents)
Location (docid, pathid, path, depth, path_cnt)
Element (docid, eid, name, sibord, pathid, key_count, value)
Word (word, position, depth, docid, eid, pathid)

XFTS 역 인덱스 기법에서는 애트리뷰트에 대한 정보를 저장하기 위한 테이블은 엘리먼트에 대한 정보를 저장하는 테이블 Element와 유사하기 때문에 편의상 생략하기로 한다. 각각의 테이블에 대한 설명은 다음과 같다. 먼저 XML_Documents 테이블은 실제 XML 문서를 저장하기 위한 것으로 사용자가 XML 문서 전체를 결과로 반환하기를 원하는 경우에 최대 4GB까지 저장할 수 있는 CLOB 타입의 contents 컬럼을 통해서 처리할 수 있다. 이 테이블의 id 컬럼은 저장된 XML 문서의 고유한 식별자를 나타내며 docname 컬럼은 저장된 XML 문서의 이름을 나타낸다. 그리고 isidx 컬럼은 contents 컬럼에 저장되어 있는 XML 문서에 대한 역 인덱스의 생성 여부를 나타낸다. isidx 컬럼의 값이 'true'인 XML 문서는 이미 역 인덱스가 생성되었다는 의미이므로 실제 역 인덱스를 생성해야 하는 문서는 isidx 컬럼의 값이 'false'로 되어 있다.

Location 테이블은 XML 문서 내에 존재하는 경로에 대한 정보를 저장한다. 이 테이블의 docid 컬럼은 각 경로가 존재하는 XML 문서를 구별하기 위한 식별자이고 pathid 컬럼은 각 경로들을 구별하기 위한 식별자를 나타며, path 컬럼은 루트 엘리먼트에서부터 시작하여 최종 단말 엘리먼트까지의 경로를 나타낸다. 또한 depth 컬럼은 루트 엘리먼트에서부터 단말 엘리먼트까지의 깊이를 나타내며 path_cnt 컬럼은 XML 문서 내에서 나타난 루트 엘리먼트에서부터 단말 엘리먼트까지의 동일한 경로의 횟수를 표현한다. 본 논문에서는 path 컬럼의 값을 입력할 때 XPath에서 각 경로를 구별하기 위해서 사용된 '/' 문자 대신에 접두사로 '~' 문자를 덧붙인 '~/'을 사용한다[14].

Element 테이블은 XML 문서 내에 존재하는 엘리먼트들에 대한 세부 정보를 저장하며 sibord 컬럼은 XQuery의 기능 중에서 '//author/family[2]'처럼 동일한 부모 엘리먼트의 자식들인 형제 엘리먼트 사이에서의 순서와 관련된 질의를 처리한다. 형제를 찾는 XQuery를 처리하기 위해서는 '~%/author~/family'인 경로를 Location 테이블의 path 컬럼에서 찾고 Element 테이블에서 sibord가 2인 엘리먼트를 찾으면 된다. value 컬럼은 각 엘리먼트의 전체 컨텐츠를 저장하며 key_count 컬럼은 불용어와 함께 동일한 키워드를 제거한 각 엘리먼트의 컨텐츠를 단어수로 나타낸다. 이러한 컬럼들을 통해서 특정 경로 내에 특정 키워드나 구를 포함하는 엘리먼트를 검색하는 XQuery를 융통성 있게 처리할 수 있다.

Word 테이블은 XQuery 중에서 경로 정보가 주어지지 않는 단순한 키워드 검색을 위한 것이다. 이 테이블의 word 컬럼은 엘리먼트의 컨텐츠 중에서 불용어를 제외한 각 키워드를 나타내며 position 컬럼은 특정 엘리먼트 내에 존재하는 각 키워드에 대한 상대적 위치 정보를 나타낸다. XQuery에서 언급하는 키워드간의 거리 연산(distance)[12]을 위하여 본 논문에서는 특정 엘리먼트 내에 존재하는 키워드들 간의 거리 연산만 고려한다. 본 논문에서는 position 컬럼을 통해서 특정 엘리먼트에 있는 키워드들 간의 거리 연산을 처리할 수 있다. depth 컬럼은 각 경로의 구별자인 pathid가 동일한 Location 테이블의 depth 컬럼 값에 1을

증가시킨 것으로 각 키워드의 깊이를 나타낸다.

(그림 2), (그림 3), (그림 4)는 각각 예제 books XML 문서를 파싱하여 얻은 Location, Word, Element 역 인덱스의 내용을 나타내고 있다.

Location (docID, pathID, path, depth, path_cnt)
(1, 1, ~/books, 1, 1)
(1, 2, ~/books~/book, 2, 1)
(1, 3, ~/books~/book~/title, 3, 1)
(1, 4, ~/books~/book~/author, 3, 1)
(1, 5, ~/books~/book~/author~/family, 4, 3)
(1, 6, ~/books~/book~/author~/given, 4, 3)
(1, 7, ~/books~/book~/summary, 3, 1)
(1, 8, ~/books~/book~/summary~/keyword, 4, 3)

(그림 2) books XML 문서의 Location 역 인덱스 내용

Word (word, position, depth, docID, eID, pathID)

(Data, 1, 4, 1, 3, 3)	(GIL, 1, 5, 1, 10, 6)
(Web, 2, 4, 1, 3, 3)	(Dong, 2, 5, 1, 10, 6)
(Kim, 1, 5, 1, 5, 5)	(book, 1, 4, 1, 11, 7)
(Young, 1, 5, 1, 6, 6)	(mentions, 2, 4, 1, 11, 7)
(Chul, 2, 5, 1, 6, 6)	(semistructured, 1, 5, 1, 12, 8)
(Lee, 1, 5, 1, 7, 5)	(data, 2, 5, 1, 12, 8)
(Eun, 1, 5, 1, 8, 6)	(database, 1, 5, 1, 13, 8)
(Suk, 2, 5, 1, 8, 6)	(XML, 1, 5, 1, 14, 8)
(Hong, 1, 5, 1, 9, 5)	

(그림 3) books XML 문서의 Word 역 인덱스 내용

Element(docID, eID, name, sibord, pathID, key_count, value)
(1, 1, books, 1, 1, 0, null)
(1, 2, book, 1, 2, 0, null)
(1, 3, title, 1, 3, 2, Data on the Web)
(1, 4, author, 2, 4, 0, null)
(1, 5, family, 1, 5, 1, Kim)
(1, 6, given, 2, 6, 2, Young Chul)
(1, 7, family, 3, 5, 1, Lee)
(1, 8, given, 4, 6, 2, Eun Suk)
(1, 9, family, 5, 5, 1, Hong)
(1, 10, given, 6, 6, 2, Gil Dong)
(1, 11, summary, 3, 7, 2, This book mainly mentions)
(1, 12, keyword, 1, 8, 2, semistructured data)
(1, 13, keyword, 2, 8, 1, database)
(1, 14, keyword, 3, 8, 1, XML)

(그림 4) books XML 문서의 Element 역 인덱스 내용

다음의 예는 본 논문에서 제안하는 XFTS 역 인덱스 기법을 통한 XML 전문 검색의 대표적인 것들이다.

① Word 테이블의 word 컬럼을 통해서 단순 키워드 검색을 처리할 수 있다.

```
contains(text(), 'XML')
SELECT E.DOCID, E.ELID
FROM ELEMENT E, WORD W
WHERE W.WORD LIKE '%XML%'
AND W.DOCID = E.DOCID AND W.EID = E.EID ;
```

(그림 5) 단순 키워드를 검색하는 질의의 SQL으로의 변환

② Element 테이블의 value와 key_count 컬럼을 통해서 tight 종속 질의를 처리할 수 있다[12].

```
//keyword ='semistructured data'
SELECT E.DOCID, E.ELID
FROM LOCATION L, ELEMENT E
WHERE L.PATH LIKE '~%/keyword'
AND E.VALUE LIKE '%semistructured data%' AND
E.KEY_COUNT = 2
AND L.DOCID = E.DOCID AND L.ID = E.PATHID ;
```

(그림 6) tight 종속 질의의 SQL으로의 변환

③ Word 테이블의 position 컬럼을 통해서 엘리먼트 범위 내에서 키워드간의 거리 연산을 수행한다.

```
//book/title/Distance('Data', 'Web') <= 1
SELECT E.DOCID, E.EID, E.VALUE
FROM LOCATION L, ELEMENT E, WORD Data, WORD Web
WHERE L.PATH LIKE '~%/book/title/'
AND Data.WORD = 'Data' AND Web.WORD = 'Web'
AND L.DOCID = E.DOCID AND L.ID = E.PATHID
AND Data.DOCID = Web.DOCID AND Data.EID = Web.EID
AND Data.PATHID = L.ID AND Web.PATHID = L.ID
AND Web.POSITION - Data.POSITION > 0
AND Web.POSITION - Data.POSITION <= 1
AND E.DOCID = Data.DOCID AND E.EID = Data.EID
AND E.DOCID = Web.DOCID AND E.EID = Web.EID ;
```

(그림 7) 엘리먼트 내의 키워드간 거리 연산 질의의 SQL으로의 변환

3.2 XML 문서의 변경 처리

본 논문에서는 XML 문서의 변경을 효과적으로 처리할 수 있도록 역 인덱스를 구성하였다. 현재까지 연구된 많은 기법들은 XML 문서 내의 엘리먼트와 키워드의 변경에 대해서 이미 생성되어진 역 인덱스를 재생성 하거나 [11-13] 각 키워드의 위치 정보에 일정한 여백을 비워둠으로써 역 인덱스를 재생성하는 문제를 해결하려고 했다[15]. 그러나 XML 문서에서 변경이 자주 발생하면 비워두었던 번호 여백이 전부 소모되기 때문에 결국은 역 인덱스를 재 생성해야 하는 문제가 발생한다. 따라서 이러한 기존의 기법들은 XML 문서의 변경을 완전히 처리한다고 볼 수 없다.

3.2.1 기존 엘리먼트 내의 키워드 변경 처리

키워드의 추가와 삭제는 유사하기 때문에 추가에 대한 절차만을 살펴본다. 키워드의 추가 시에는 추가되어야 할 엘리먼트에 대한 정보를 바탕으로 Element 테이블과 Word 테이블에서 관련된 튜플을 변경한다. (그림 8)에서 예제 books XML 문서의 keyword 엘리먼트 중에서 컨텐츠가 'database'인 것을 'database system'으로 변경하기 위한 절차를 보이고 있다.

```
/ 1. Location 테이블에서 키워드를 저장할 docid, pathid를 얻는다. /
SELECT DOCID, PATHID INTO L_DOCID, L_PATHID
FROM LOCATION
WHERE PATH = '~/books~/book~/summary~/keyword' ;
/ 2. docid, pathid를 바탕으로 elid를 얻는다 /
SELECT ELID INTO E_ELID
FROM ELEMENT
WHERE NAME = 'KEYWORD' AND VALUE = 'database' AND
DOCID = L_DOCID
AND PATHID = L_PATHID ;
/ 3. Element의 value에 키워드를 추가해서 변경을 한다. /
UPDATE ELEMENT(KEY_COUNT, VALUE)
SET (KEY_COUNT + 1, 'database system'
WHERE DOCID = L_DOCID AND ELID = E_ELID AND
PATHID = L_PATHID ;
/ 4. Word 테이블에 키워드를 추가하기 위한 정보를 얻는다. /
SELECT POSITION, DEPTH INTO W_POSITION, W_DEPTH
FROM WORD
WHERE DOCID = L_DOCID AND EID = E_ELID AND PATHID
= L_PATHID ;
/ 5. Word 테이블에 키워드를 추가한다. /
INSERT INTO WORD('system', W_POSITION, W_DEPTH,
L_DOCID, E_ELID, L_PATHID) ;
```

(그림 8) keyword 엘리먼트 내의 키워드 'system'을 추가하기 위한 절차

3.2.2 엘리먼트와 키워드의 입력 처리

XML 컨텐츠를 다른 XML 문서에 추가하는 경우를 살펴보자. 이러한 변경을 처리하기 위해서 Location 테이블의 path 컬럼에서 추가될 엘리먼트까지의 경로가 한 번도 나타나지 않은 경우에는 추가될 엘리먼트의 경로 정보를 삽입하고 이 정보를 바탕으로 Element 테이블과 Word 테이블의 관련 튜플들을 변경한다. 그러나 이미 추가할 엘리먼트까지의 경로가 존재할 경우에는 Location 테이블에서 삽입될 경로의 path_cnt 컬럼을 1만큼 증가시키고 Element 테이블에서 삽입될 엘리먼트 이후의 엘리먼트에 대한 sibord 컬럼의 값을 1씩 증가시킨다. 그 다음으로 Element 테이블과 Word 테이블에서 관련 튜플들을 변경한다.

(그림 9)는 추가될 엘리먼트까지의 경로가 이미 존재하는 경우의 예로 books XML 문서에 책의 저자로 'Cheon'이라는 컨텐츠를 가진 family 엘리먼트를 author 엘리먼트의 5번째 자식 엘리먼트로 추가하기 위한 절차를 보이고 있다.

```
/ 1. 추가될 엘리먼트까지의 경로가 이미 존재하는지 검사한다. /
SELECT PATH_CNT INTO L_PATH_CNT
FROM LOCATION
WHERE PATH = '~/books~/book~/author~/family' ;
/ 2. 이미 존재(l_path_cnt >= 1)하기 때문에 path_cnt를 1만큼 증가
   시킨다. /
UPDATE LOCATION(PATH_CNT) SET PATH_CNT =
PATH_CNT + 1
WHERE PATH = '~/books~/book~/author~/family' ;
/ 3. 같은 부모 엘리먼트를 가진 엘리먼트 중에서 삽입될 엘리먼트
   이후의 엘리먼트에 대한 sibord 컬럼의 값을 1만큼 증가시킨다. /
UPDATE ELEMENT(SIBORD) SET SIBORD = SIBORD + 1
WHERE PATH LIKE '~%/books~/book~/author' AND SIBORD >
4 ;
/ 이후의 절차는 엘리먼트 내에 키워드를 추가하는 절차와 동일하다. /
```

(그림 9) family 엘리먼트를 추가하기 위한 절차

3.2.3 엘리먼트의 삭제 처리

본 논문에서는 엘리먼트의 삭제에 의해서 XML 문서가 변경될 경우에 삭제될 엘리먼트가 텍스트를 제외한 자식 엘리먼트를 가지고 있다면 반드시 그 자식 엘리먼트도 함께 삭제되어야 한다고 가정한다. 삭제될 엘리먼트가 하위 자식 엘리먼트를 가지고 있지 않을 경우에는 삭제될 엘리먼트에 대한 역 인덱스 정보만을 삭제하면 된다. 그렇지 않고 삭제될 엘리먼트가 자식 엘리먼트를 가지고 있을 경우에는 부모 엘리먼트를 삭제함으로써 발생되는 트리거를 통해서 자식 엘리먼트에 대한 튜플을 삭제한다. 삭제될 엘리먼트까지의 경로가 처음 나타난 것인지 이미 존재하는 것인지를 알아보기 위해서는 Location 테이블의 path_cnt 컬럼을 사용한다. 이 컬럼의 값이 2이상이면 삭제될 엘리먼트까지의 경로가 이미 두 번 존재하는 것이기 때문에 path_cnt를 1만큼 감소시키고 삭제될 엘리먼트 이후에 나타나는 엘리먼트의 sibord 컬럼의 값을 1씩 증가시킨다. 그 다음 절차로 튜플 기반 트리거를 통해서 Element 테이블, Word 테이블의 관련 튜플들을 변경한다. 그러나 path_cnt가 1이면 삭제될 엘리먼트까지의 경로가 한 번 나타난 것이기 때문에 Location테이블에서 관련 경로 정보를 가진 튜플을 삭제하고 path_cnt가 2이상일 때의 절차와 동일하게 처리한다.

이처럼 본 논문에서 제안하는 기법은 XML 문서의 변경을 처리하기 위해서 변경된 엘리먼트와 키워드의 역 인덱스에 대해서 부분적인 변경이 필요하기 때문에 상당히 효율적이다.

4. 성능 평가 모델

본 장에서는 본 논문에서 제안하고 있는 XFTS 기법의 실험 환경을 기술한다. XML 전문 검색을 위한 역 인덱스를 생성하기 위해서 Java 1.4.2와 JDOM API를 사용하였다. 실험에 사용한 상용 데이터베이스 시스템은 오라클 9i이고 이것은 리눅스 2.4.20, 펜티엄 Ⅲ 1.2GHz 듀얼 CPU, 메인 메모리 1.5GB에 설치되어 있다. 각 테이블의 인덱스는 주키와 함께 조인이 발생되는 컬럼에 생성되었다. 클라이언트는 윈도우 2000 Professional, 펜티엄 Ⅳ 1.4GHz에 설치되어 있고 메인 메모리는 768MB이다. 본 장의 나머지 부분에서는 실험에 사용된 XML 문서와 XML 질의, 그리고 성능 평가 척도에 대해서 설명한다.

4.1 실험 XML 문서

본 논문의 실험에서는 세 개의 XML 문서를 사용하였다. 첫 번째 문서는 셰익스피어 희곡의 내용을 포함하고 있는 Shakespeare's Work이고 두 번째 문서는 DBLP의 서지 목록 내용을 담고 있는 DBLP이고 세 번째 문서는 실제 데이터가 아닌 가상의 경매 데이터를 담고 있는 Auction이다 [16-18]. <표 2>는 실험에 사용된 문서의 타입과 개수, 전체 크기, 그리고 각 문서의 평균 엘리먼트의 깊이를 나타내고 있다.

〈표 2〉 실험에 사용된 XML 문서에 대한 정보

종 류	타 입	문서의 개수	전체 크기(MB)	평균 깊이
Shakespeare's work	Real	37	7.53	5.95
DBLP	Real	1	50.3	4.00
Auction	Synthetic	1	115.5	7.02

4.2 실험 XML 질의

본 논문에서는 XML 전문 검색을 위한 언어로서 XQuery의 문법과 유사한 언어를 사용하였다. 전문 검색 기능 중에서 이미 정보 검색 분야에서 오랜 기간 동안 연구해 온 질의 검색은 제외하고 XQuery의 전문 검색 기능 중에서 XML 문서의 계층 구조 특성에 의해서 추가된 기능에 대해서만 실험하였다. <표 3>은 실험에 사용된 XML 질의를 나타내고 있다.

<표 3>에서 나타난 질의 종류에서 중간 문자는 질의에 사용된 XML 문서를 나타내고 마지막 숫자는 질의 번호를 나타낸다. 그리고 각 질의에 대한 결과로서는 관련 연구 [11-13]에서 사용되었던 키워드가 포함된 XML 문서의 고유 식별자 대신에 각 엘리먼트에 대한 정보를 나타내었다. 이것은 XML 문서에 대한 식별자를 바탕으로 각 엘리먼트에 대한 정보를 얻기 위한 처리 과정을 거쳐야 하는 관련 연구에서의 비효율적인 면을 개선한 것이다.

〈표 3〉 실험에 사용된 XML 질의

질의 종류	XML 질의
QS1	contains(text(), 'XML')
QS2	/PLAY/TITLE/'The Comedy of Errors'
QS3	/PLAY/ACT/SCENE/SPEECH/SPEAKER/'DUKE SOLINUS'
QS4	//SPEAKER='AEGEON'
QS5	//TITLE/'ACT'
QS6	//ACT//SPEECH/SPEAKER='KING JOHN'
QS7	//SCENE/*/LINE/'love'
QD1	contains(text(), 'SQL')
QD2	/DBLP/ARTICLE/TITLE/'SQL Reunion'
QD3	//AUTHOR='Frank Manola'
QD4	//INPROCEEDINGS/TITLE/'Dynomite'
QA1	contains(text(), 'pencil')
QA2	/SITE/REGIONS/ASIA/ITEM/LOCATION/'Korea'
QA3	//name='preventions'
QA4	//item/payment/'Personal Check'

4.3 성능 평가 척도

본 논문에서 제안하는 역 인덱스 기법의 성능을 평가하기 위해서 다음과 같은 3가지의 성능 평가 척도를 사용하였다.

① 결과의 정확성 : 질의 결과의 내용이 같은가를 비교하였다. 중복 결과를 제거한 후 그 결과가 같은지를 비교하였다.

② 역 인덱스 생성 성능 척도 : XML 질의를 처리하기 위해서 사용되어지는 역 인덱스의 공간 크기와 역 인덱스를 만들기 위해서 사용되어진 생성 시간을 측정하였다.

③ 질의 처리 시간 복잡도 척도 : 역 인덱스가 저장되어진 테이블에 접근하고 결과를 얻는데까지 걸린 시간을 측정하였다. 측정 시간의 오차를 줄이기 위해서 반복적으로 5회 실험하였고 결과 시간은 5회의 평균으로 나타내었다.

5. 실험 결과

본 장에서는 4장에서 열거한 성능 평가 척도를 바탕으로 본 논문에서 제안하는 XFTS 기법의 성능을 평가한다. 또한, XML 문서의 크기 변화와 경로 길이의 변화에 따른 질의 처리 시간 복잡도의 성능도 함께 평가한다.

앞으로 편의상 D. Florescu[11]가 제안한 인덱스 기법을 I_INDEX 기법, C. Zhang[12]이 제안한 역 인덱스 기법을 C_INDEX 기법이라고 부르고 본 논문에서 제안하는 기법을 XFTS_INDEX 기법이라고 부르기로 한다.

5.1 역 인덱스 생성 성능

<표 4>는 역 인덱스를 생성하는데 걸린 시간을 나타내고 있다. 여기서 C_INDEX 기법, XFTS_INDEX 기법, I_INDEX 기법 순으로 생성 시간이 오래 걸린다는 것을 알 수 있다. I_INDEX 기법은 XML 문서의 깊이가 깊어짐에 따라서 그 깊이만큼 중복되는 정보를 저장하기 때문에 역 인덱스를 생성하는데 시간이 오래 걸린다. C_INDEX 기법과 XFTS_INDEX 기법은 XML 문서의 깊이와 상관없이 일정하게 증가하는 것을 볼 수 있다. C_INDEX 기법이 XFTS_INDEX 기법보다 시간이 적게 걸리는 이유는 질의를 처리하기 위해서 보다 적은 정보를 생성하기 때문이다.

〈표 4〉 제안 기법별 역 인덱스의 생성 시간(sec)

	I_INDEX 기법	C_INDEX 기법	XFTS_INDEX 기법
Shakespeare's work	890.258	209.619	488.202
DBLP	4880.978	1322.540	2678.732
Auction	43387.022	1913.811	3837.301

〈표 5〉 제안 기법별 역 인덱스의 크기(MB)

	I_INDEX 기법	C_INDEX 기법	XFTS_INDEX 기법
Shakespeare's work	101	13	21
DBLP	429	78	110
Auction	3000	243	373

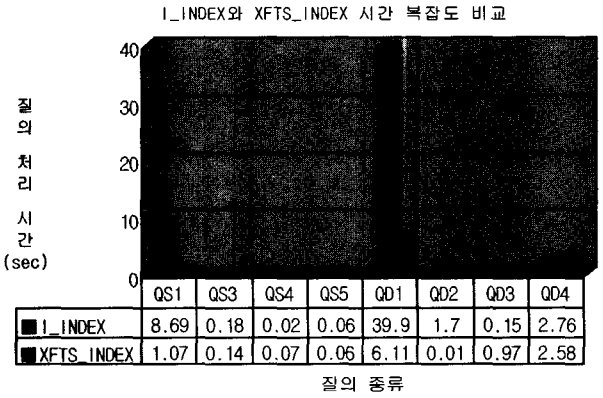
<표 5>에서 각각의 XML 문서에 대해서 생성되어진 역 인덱스의 크기를 나타내고 있다. 이 표를 통해서 I_INDEX, XFTS_INDEX, C_INDEX 기법의 순서로 좋은 성능을 보인다는 것을 알 수 있다. 이런 성능 차이의 원인은 I_INDEX 기법은 같은 엘리먼트에 대해서 중복된 Word_Type 역 인덱스를 생성하기 때문이다. 최악의 경우에 루트 엘리먼트에서 단말 엘리먼트까지의 깊이가 깊어질 경우에는 단말 엘리먼트에 포함된 키워드에 대한 역 인덱스를 생성하기 위해서 단말 엘리먼트까지의 깊이만큼 중복된 Word_Type 역 인덱스를 생성하게 된다. 그리고 C_INDEX 기법이 XFTS_INDEX 기법보다 약 25%의 성능 향상을 보이는데 그 이유는 C_INDEX에서는 포함 연산을 위한 계층 정보로(begin, end) 숫자만을 사용하며 계층 정보를 위한 역인덱스를 따로 만들지 않기 때문이다. 반면에 XFTS_INDEX는 계층 정보로 루트에서 단말까지의 패스 스트링을 따로 저장한다. 하지만 C_INDEX 기법은 XML 문서가 변경 되었을 때 변경된 부분 다음에 있는 모든 원소의 (begin, end)를 다시 부여해야 하는 어려움이 있지만 XFTS_INDEX 기법은 C_INDEX 기법에 비해 추가적으로 생성된 Location 테이블에 새로운 패스 스트링을 추가하든지 아니면 기존에 존재하는 패스 스트링의 path_cnt 컬럼 값을 증가시킴으로써 효과적으로 처리할 수 있다. 이처럼 C_INDEX 기법은 공간 복잡도에 있어서는 XFTS_INDEX 기법보다 좋은 성능을 보이지만 XML 문서의 변경을 처리하기 위하여 각 원소의 (begin, end) 값을 새로 부여해야 하므로 매우 비효율적이다.

5.2 질의 처리 시간 복잡도

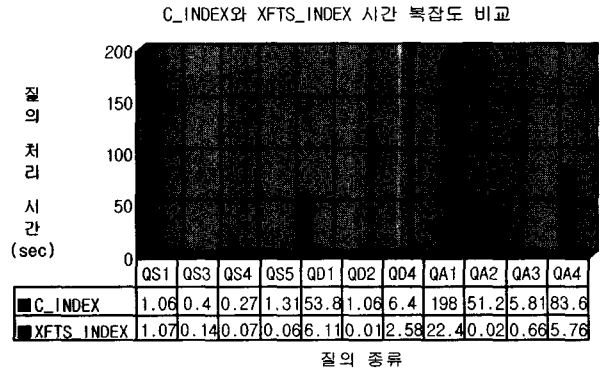
질의 처리 시간 성능 비교는 XFTS_INDEX 기법과 기존 기법인 I_INDEX 기법, C_INDEX 기법을 각각 따로 실험하였다. 이렇게 비교한 이유는 I_INDEX 기법을 통한 역 인덱스 생성 시에 문서의 크기가 115MB인 Auction 문서의 경우에 다른 기법들에 비해서 역 인덱스의 크기가 훨씬 커져서 Auction 문서에 대한 질의 처리 시간을 측정할 수 없었기 때문이다. (그림 10)은 I_INDEX 기법과 XFTS_INDEX 기법의 질의 처리 시간 성능을 비교한 것이다. (그림 10)에서 알 수 있듯이 QS4와 QD3 질의를 제외하고 본 논문에서 제안하는 XFTS_INDEX 기법이 더 좋은 성능을 보인다. QS1과 QD1은 같은 종류의 질의로써 I_INDEX 기법과 XFTS_INDEX 기법에서 상당한 성능 차이를 보이는데 그 이유는 I_INDEX 기법에서는 특정 단어가 존재하는 엘리먼트를 검색함에 있어서 XML 문서에 대한 번호와 함께 각 엘리먼트의 고유한 번호를 검색하기 위해서 'IN' 연산을 수행해야 하기 때문이다. 그에 비해 XFTS_INDEX 기법은 '=' 연산을 통해 수행하기 때문에 실행 시간을 단축할 수 있다. QS1과 QD1을 비교해 보면 실험 XML 문서의 크기가 커짐에 따라 그 만큼 실행 시간이 길어진 것을 알 수 있다. 나머지 QS3, QS5, QD2, QD4도 이와 같은 이유로 성능 차이를 보인다. 그러나 QS4와 QD3는 tight 종속 질의로 I_INDEX 기법은 XML 문서에 나타나는 각 엘리먼트에 대한 정보를 EleName 역 인덱스로 생성한다. 그래서 한 번의 '=' 연산을 통해서 빠르게 검색할 수 있다. 그에 비해 XFTS_INDEX 기법은 경로에 대한 정보를 검색하는데 'LIKE' 연산을 한 번 수행해야 하기 때문에 실행 시간이 더 걸린 것이다.

(그림 11)은 C_INDEX 기법과 XFTS_INDEX 기법의 질의 처리 시간 성능을 비교한 것이다. (그림 11)에서 볼 수 있듯이 XFTS_INDEX 기법이 C_INDEX 기법보다 QD1, QA1, QA2, QA4에서 훨씬 좋은 성능을 보인다. 또한 나머지 질의에서도 같은 결과를 보인다. 이러한 결과의 주된 원

인은 다음과 같다. QD1과 QA1 질의에서는 조인을 할 때 XFTS_INDEX 기법은 '=' 연산을 통해서 이루어지지만 C_INDEX 기법은 수 비교 연산을 통해 이루어진다. 같은 종류의 질의인 QD1과 QA1에서 실험 XML 문서의 크기 변화가 실행 시간에 상당한 영향을 미쳤다. QD2, QD4, QA2, QA4 질의에서도 역시 실행 시간에 문서의 크기가 영향을 미쳤음을 알 수 있다. 특히, QA2와 QA4 질의에 대한 C_INDEX 기법은 문서의 크기 변화와 함께 복잡한 경로에 대한 질의 처리를 위해서 여러 번의 조인이 발생한다. 그러나 XFTS_INDEX 기법은 Location. 테이블에서 '=' 연산을 통해 복잡한 경로라 할지라도 쉽게 찾을 수 있기 때문에 좋은 성능을 보인다.



(그림 10) I_INDEX기법과 XFTS_INDEX 기법의 시간 복잡도 성능 비교

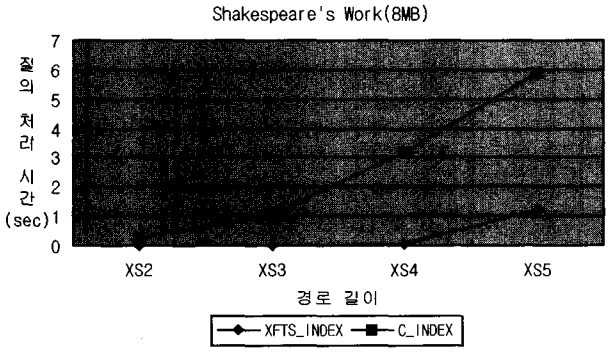


(그림 11) C_INDEX기법과 XFTS_INDEX 기법과의 시간 복잡도 성능 비교

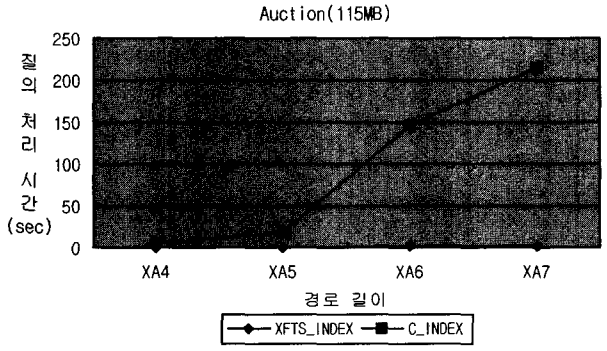
5.3 XML 문서의 크기 변화와 경로 길이에 따른 시간 복잡도 성능

(그림 12)와 (그림 13)은 XML 문서의 크기 변화와 사용자 쿼리의 경로 길이에 따른 시간 복잡도를 분석한 것이다. (그림 12)와 (그림 13)의 X축에 사용된 질의 종류에서 가운데 문자는 XML 문서을 나타내고 마지막 숫자는 경로 길이를 나타낸다. 성능 비교는 C_INDEX 기법과 XFTS_INDEX 기법을 사용한다. 실험에 사용된 XQuery는 반환 될 결과의 수를 고려해서 사용되었다.

(그림 12)와 (그림 13)에서 볼 수 있듯이 C_INDEX 기법은 경로 길이가 길어질수록 처리 시간도 길어졌다. 그러나 본 논문에서 제안하는 XFTS_INDEX 기법은 경로 길이가 길어지는 것과는 상관없이 일정한 처리 시간이 걸렸다. 예외적으로 (그림 12)의 XFTS_INDEX 기법에서 경로 길이가 5일 경우에 처리 시간이 갑자기 증가한 이유는 결과 튜플의 수가 다른 질의에 비해서 훨씬 많아서 결과를 가져오는데 오랜 시간이 걸렸기 때문이다. (그림 13)에서의 C_INDEX 기법은 처리 시간에 있어서 (그림 12)에서의 처리 시간보다 더 큰 차이를 보였다. 그 이유는 실험에 사용된 XML 문서의 크기가 상당히 커졌기 때문이다. (그림 13)에서의 C_INDEX 기법은 XQuery의 경로 길이가 길어짐에 따라서 처리 시간도 상당히 증가하였다. 그렇지만 XFTS_INDEX 기법은 경로 길이와는 상관없이 일정한 처리 시간을 보였다. 종합 해보면 C_INDEX 기법은 XML 문서의 크기가 커지고 경로 길이가 길어질수록 XQuery를 처리하는데 오랜 시간이 걸린다. 그러나 본 논문에서 제안하는 XFTS_INDEX 기법은 문서의 크기와 경로 길이와는 상관없이 항상 일정한 처리 시간을 유지한다.



(그림 12) Shakespeare문서의 경로 길이 변화에 따른 시간 복잡도 성능 비교



(그림 13) Auction문서의 경로 길이 변화에 따른 시간 복잡도 성능 비교

5.4 XML 변경 성능

I_INDEX와 C_INDEX는 XML 변경을 전혀 고려하지 않고 만들어진 인덱스이므로 직접적인 성능을 비교하는 것은 매우 어렵다. 하지만 XFTS 방법은 기존의 C_INDEX에서 사용하는 번호 부여 방식(Numbering scheme)과는 달리 XML 변경 연산이 발생할 때 삽입 또는 삭제가 이루어지는 부분 주변에는 전혀 영향을 주지 않는다. 따라서 XML 문서의 변경이 발생할 경우 변경되어야 할 역인덱스가 최소화 되어 XML 문서 변경이 쉽게 이루어질 수 있다.

6. 결 론

본 논문에서는 XML 데이터를 저장하기 위한 기법으로 지금까지 꾸준하게 발전해 온 관계형 데이터베이스를 사용하여 XML 문서의 변경시 역 인덱스의 재생성 없이 부분적인 변경만으로 XML 문서의 변경 내용을 반영하면서 다양한 기능의 XML 전문 검색 질의를 효율적으로 처리 할 수 있는 역 인덱스 기법을 제안하였다. 본 논문에서 제안한 XFTS_INDEX 기법이 기존에 연구되었던 I_INDEX 기법과 C_INDEX 기법에 비해 향상된 점은 다음과 같다. 첫째, 검색 시간이 단축되었다. 여러 응용 분야에서 XML 데이터를 활용하려는 노력이 활발히 이루어짐으로써 XML 데이터의 크기도 상당히 커지고 있는 추세이다. 이러한 상황에서 XML 데이터에 대한 검색 성능 평가를 좌우하는 중요한 요소는 검색에 걸린 시간이다. 둘째, XML 데이터의 변경을 효과적으로 처리할 수 있다. 현재 XML 데이터는 주로 디지털 정보를 표현하고 교환하기 위한 수단으로 사용되고 있다. XML 데이터 사용의 보편화에는 XML 형식의 데이터는 언제든지 확장 또는 변경 될 수 있다는 장점이 큰 역할을 차지하였다. 이러한 점에서 XML 데이터는 변경이 자주 일어날 수 있다. 이러한 변경을 본 논문에서 제안한 기법이 쉽게 처리한다는 사실은 상당한 성과라 할 수 있다.

향후 연구 과제로는 성능의 향상을 위해서 원래의 XML 데이터보다 상당히 큰 역 인덱스 저장 공간이 필요하기 때문에 현재의 XML 검색 기능을 그대로 유지 또는 발전시키면서 상대적으로 역 인덱스의 크기를 줄일 수 있는 압축 기법의 연구가 필요하다.

참 고 문 헌

[1] T. Bray, J. Paoli, C. Sperberg-McQueen, "Extensible Markup Lanauage(XML) 1.0," http://www.w3c.org/TR/1998/REC-xml-19980219, 1998.

[2] S. Abiteboul, D. Quass, J. McHugh, J. Widom, J. Wiener, "The Lorel query language for semistructured data," International Journal on Digital Libraries, 1997.

[3] A. Deutsch, M. Fernandez, D. Florescu, A. Levy, D. suciu, "A Query Language for XML," Proceedings of 8th International World Wide Web Conference, 1999.

[4] J. Clark, S. DeRose, "XML path language(XPath) Version 1.0," http://www.w3c.org/TR/1999/REC-xpath-19991116, 1999.

[5] J. Robie, D. Chambelin, D. Florescu, "Quilt : an XML Query Languange," 2000.

[6] D. Chamberlin, D. Florescu, J. Robie, J. Simeon, M. Stefanescu, "XQuery : a query language for XML," Technical report, 2001.

[7] A. Brown, M. Fuchs, J. Robie, P. Wadler, "XML Schema : Formal Description," http://www.w3c.org/TR/2001/WD-xmlschema-formal-20010320/, 2003.

[8] D. Lipkin, J. Marsh, H. Tompson, N. WalshXSL, "Transformations(XSLT) Version 1.0," http://www.w3c.org/TR/1999/REC-xslt-19991116, 1999.

[9] S. Buxson, "XQuery and XPath Full-Text Requirements," http://www.w3.org/TR/2003/WD-xquery-full-text-requirements-20030502/, 2003.

[10] S. Yahia, P. Case, "XQuery and XPath Full-Text Use Cases," http://www.w3.org/TR/2003/WD-xmlquery-full-text-use-cases-20030214/, 2003.

[11] D. Florescu, D. Kossman, I. Manolescu, "Integrating keyword search into XML query processing," Proceedings of the 9th International World Wide Web Conference, 2000.

[12] C. Zhang, J. Naughton, D. DeWitt, Q. Luo, G. Lohman, "On supporting containment queries in relational database management systems," Proceedings of the ACM SIGMOD International Conference on the Management of Data, 2001.

[13] C. Seo, S. Lee, H. Kim, "An efficient inverted index techniques for XML Documents using RDBMS," Information and Software Technology 45, pp.11-22, 2001.

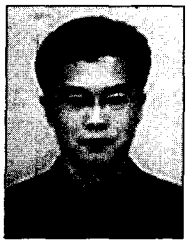
[14] M. Yoshikawa, T. Amagasa, "XRel : a path-based approach to storage and retrieval of XML Documents using relational database," ACM Transactions on Internet Technology 1, pp.110-141, 2001.

[15] Q. Li, B. Moon, "Indexing and Querying XML Data for Regular Path Expressions," Proceedings of the 27th VLDB Conference, 2001.

[16] OASIS Home Page, Shakespeares's XML Document Reference, http://www.oasis-open.org/cover/bosakSahkespeare200.html, cited, Sep., 2003.

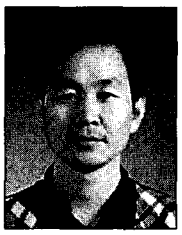
[17] CS Department University of Trier, DBLP XML Document Reference, http://www.informatik.uni-trier.de/ley/db, cited, Sep., 2003.

[18] MonetDB, Auction XML Document Reference, http://monetdb.cwi.nl/xml/Benchmark/benchmark.html, cited, Sep., 2003.



천 윤 우

e-mail : ywcheon@idatabank.com
2002년 계명대학교 컴퓨터공학과(학사)
2004년 계명대학교 컴퓨터공학과 (공학 석사)
2004년~현재 (주)데이타뱅크 시스템즈
관심분야 : RDBMS, XQuery



홍 동 권

e-mail : dkhong@kmucc.keimyung.ac.kr
1985년 경북대학교 전자공학과(학사)
1992년 University of Florida 전자계산학과 (석사)
1995년 University of Florida 전자계산학과 (박사)
1985년~1990년 한국전자통신연구원
1996년~1997년 한국전자통신연구원
1997년~현재 계명대학교 공학부 컴퓨터공학 전공 조교수
관심분야 : 능동 실시간 데이터베이스, 병렬 처리, 성능 평가, 시뮬레이션, 멀티미디어 처리