제20회 한국정보처리학회 추계학술발표대회 논문집 제10권 제2호 (2003. 11)

관계형 데이터베이스를 이용한 XQuery 전문검색

천윤우, 홍동권
계명대학교 정보통신대학
email:{ywcheon,dkhong}@kmu.ac.kr

XQuery Full-Text Search in RDBMS

Yun-Woo Cheon, Dong-Kweon Hong
College of Information and Communication, Keimyung University

요 약

XML이 인터넷상에서 디지털 정보를 표현하고 교환하기 위한 표준이 되어감에 따라 최근까지 XML을 저장하고 검색하기 위한 역인덱스 기법에 대한 연구가 활발히 진행되고 있다. 본 논문에서는 XML 전문 검색을 위한 새로운 역인덱스 구조를 제안한다. 기존에 연구된 역인덱스 기법을 통한 키워드 검색 기능을 더욱 보완하고 최근에 W3C에서 새로운 기능으로 추가된 전문 검색 기능을 구현한다.

1. 서론

최근 XML(eXtensible Markup Language)은 인터넷상에서 디지털 정보를 표현하고 교환하기 위한 표준으로 확고한 자리를 굳혔다. 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 방대한 양의 데이터를 다양한 형태의 프로그램으로 자동 처리할 수 있는 새로운 컴퓨팅 패러다임을 제공하는 획기적인 것으로 평가되고 있다.

본 논문에서는 최근 W3C에서 채택하고 있는 XML 문서의 전문 검색 기능을 기존의 관계형 기술을 활용하는 방식을 채택하여 구현하는 기법을 연구하고, 연구 결과를 활용하여 관계형 모델을 활용한 전문 검색 기능의 구현 방법들에 대한 장단점을 알아보고 XML 전문 검색 기능을 위한 관계형 모델의 가능성 및 한계점을 알아본다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구에 대해서 기술하고, 3장에서는 기존 연구에서의 문제점에 대해서 기술한다. 4장에서는 본 논문에서 제안하는 인덱스 구조 및 질의 처리에 대해서 기술하고 5장에서는 구현된 시스템에 대한 성능을 평가하고 마지막으로 6장에서 결론 및 향후 과제를 제시한다.

2. 관련연구

XML 모델과 관계형 모델은 여러 가지 면에서 많은 차이점이 있으며 그 차이점에 따라 데이터의 표현, 질의어의 형식 및 기능이 매우 다르다. 비록 차이점이 존재하지만 이미 성숙 단계에 이른 관계형 기술을 최대한 활용하여 관계형 모델을 이용한 XML 문서의 저장 기법, XML Query를 SQL로 변환하는 방법에 대한 연구가 진행되어 왔다. 하지만 XML Query에서의 전문 검색 기능은 2003년 2월에 처음으로 "Working Draft"가 발표된 만큼 그 기능과 구현 방법에 대한 연구는 별로 진행된 것이 없다.

2.1XMLQuery

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 많은 문제점이 있다. 이로 인해 새로운 질의어가 W3C에서 XQuery 표준화로 진행되고 있다. XQuery[1] 구문 형태는 XML Schema, XSLT, XPath[2]의 내용들에 의해서 특히 IBM의 Quilt[4] 연구 내용을 많이 수용하고 있다. XQuery는 초기에는 데이터의 검색 부분만 강조했으나 점차 데이터의 변경(Update) 기능도 고려하고 있으며, 2003년에 와서는 XML 전문 검색(full-text search) 부분도 표준의 한 부분으로 포함되고 있다.

2.2 키워드 검색 및 XQuery 전문 검색 기능

최근까지 XML 문서에서의 키워드 검색에 대한 연구가 진행되어 왔다[3,7]. 특히 [3]에서는 키워드 검색을 지원하기 위해서 각 엘리먼트의 단어별로 word_element 역인덱스를 생성하고 엘리먼트와 엘리먼트 사이의 종속 관계를 나타내기 위해서 binary 역인덱스를 생성한다.

2003년 2월 W3C에서는 "XQuery and XPath

* 본 연구는 한국과학재단 목적기초연구 (R01-2003-000-10001-0)지원으로 수행되었음.

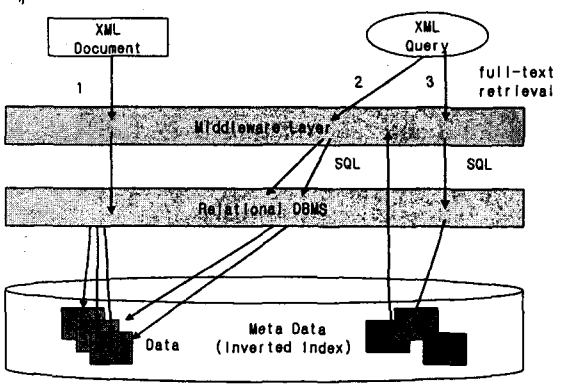
Full-Text Requirement"와 "XQuery and XPath Full-Text Use Cases" 2건의 Working Draft가 발표되었다. 이 2가지의 문서는 향후 XML Query에 추가될 full-text 기능에 대한 부분이 포함되어져 있다.

3. 기존 연구에서의 문제점

XML 문서에서의 키워드 검색에 대한 연구는 활발히 진행되어져 왔지만 전문 검색에 대한 연구는 별로 진행된 것이 없다. 키워드 검색에 대한 연구[3]에서도 같은 단어에 대해서 루트 엘리먼트 에서부터 실제 그 단어가 존재하는 엘리먼트까지 매 엘리먼트마다 word 인덱스를 생성하기 때문에 상당한 공간의 낭비를 초래한다. 더불어 binary 인덱스로 엘리먼트 사이의 종속 관계를 설정하기 때문에 복잡한 경로의 XQuery에서는 조인에 상당한 부담이 따른다. 가장 최근에 발표된 연구에서는 이 같은 문제점은 상당히 극복되었다[5]. 그러나 여기서 제안하는 방법 역시 XML 문서에서 엘리먼트 내의 약간의 텍스트 변경에도 불구하고 그 엘리먼트 이후의 역인덱스를 다시 생성해야 하는 문제점을 가진다. 이에 본 연구에서는 이러한 문제점들을 해결함과 더불어 XML 문서에서의 전문 검색 기능을 처리할 수 있는 역인덱스 구조를 제안한다.

4. 제안하는 인덱스 구조 및 질의 처리

먼저 본 논문에서의 성능 평가는 [3]에서 제안한 인덱스 기법과의 비교를 통해 이루어진다. 앞으로 본 논문이 제안하는 인덱스 기법을 XFTS(XML Full-Text Search)라고 하며 [3]에서 제안한 인덱스 구조는 I 인덱스라고 부른다. 그리고 XML 전문 검색을 위한 언어로써 XQuery의 문법과 유사한 언어를 사용한다. XQuery의 전문 검색 기능을 구현하기 위해서 관계형 데이터베이스를 활용하는 방법을 사용한다. 전문 검색 기능은 XQuery의 전문 검색 기능 중에서 XML 문서의 계층 구조 특성에 의해서 추가된 기능만을 위한 구현 방법을 제안한다. 본 논문에서는 관계형 데이터베이스에 XML문서를 저장하는 방법으로 XML문서를 테이블의 특정 컬럼에 저장하고, 그 컬럼에 대한 역인덱스를 생성하는 방법을 사용한다. (그림 1)에서는 XML 문서 자체와 그것의 역인덱스를 관계형 데이터베



(그림 1) 관계형 데이터베이스를 이용한 XML 데이터 관리

이스에 저장하고 XQuery 입력시 처리과정을 보이고 있다. 본 논문에서 제안하는 XFTS 역인덱스는 <표 1>과 같이 네 개의 테이블에 저장된다. 각 테이블의 주키는 밑줄로 표현되었으며 그것에 대한 설명은 다음과 같다. XML_DOCUMENTS 테이블은 실제 XML 문서를 저장하기 위한 것이다. LOCATION 테이블은 루트 엘리먼트에서 시작하여 최종 엘리먼트까지의 패스정보를 나타내기 위한 컬럼인 PATH와 각 패스를 구별하기 위한 ID, 그리고 각 단말 엘리먼트까지의 깊이를 나타내기 위한 컬럼인 DEPTH로 구성된다. ELEMENT 테이블은 각 엘리먼트의 XML 문서에서의 세부 정보를 나타내기 위한 것이다. 이 테이블의 DOCID, EID, NAME, PATHID는 의미그대로이며 SIBORD 컬럼은 형제 엘리먼트 사이에서의 순서를 나타내고, 엘리먼트 컨텐츠의 불용어를 제외하지 않고 나타낸 컬럼인 VALUE와 VALUE 컬럼중에서 공백을 기준으로 불용어를 제외하고 같은 키워드를 제외한 단어수를 나타낸 KEY_COUNT 컬럼등으로 구성된다. WORD 테이블은 XQuery에서 구조적인 정보가 주어지지 않았을 때 단순한 텍스트 키워드 검색을 위해서 필요한 것으로써 엘리먼트 컨텐츠에서 불용어를 제외한 각 키워드를 나타내는 컬럼인 WORD와 그 키워드의 엘리먼트 내에서의 상대적 위치 정보를 나타내는 POSITION, 키워드의 깊이를 나타내는 DEPTH 컬럼등으로 구성된다. 엘리먼트의 정확한 패스사용으로 정확한 결과 값을 얻기 위해서 각 path에 '~'을 덧붙여서 사용한다[6]. 그래서 각 패스는 '~/'로 시작하게 된다.

XFTS 역인덱스 구조는 XML 스키마가 급격하게 변화하지 않는다고 가정한다면 특정 엘리먼트의 컨텐츠에 약간의 변경의 경우 가장 최근의 연구[5]에서 제안한 역인덱스 구조보다 탁월한 성능을 보인다. 그 이유는 연구[5]에서 제안한 역인덱스 구조는 엘리먼트와 컨텐츠에 대해서 XML 문서 전체를 통한 상대적인 위치에 번호를 부여하기 때문에 엘

<표 1> XFTS 테이블 스키마

XML_DOCUMENTS(ID, DOCNAME, ISIDX, CONTENTS)
LOCATION (PATH, ID, DEPTH)
ELEMENT (DOCID, EID, NAME, SIBORD, PATHID, KEY_COUNT, VALUE)
WORD (WORD, POSITION, DEPTH, DOCID, EID, PATHID)

리먼트 컨텐츠될 경우, 그 엘리먼트 이후에 나타나는 엘리먼트와 컨텐츠에 대해서 전부 수정을 해야 하지만 본 논문에서 제안하는 역인덱스 구조는 변경이 일어난 그 엘리먼트와 컨텐츠에 대해서만 약간의 수정이 요구되므로 상당한 융통성이 있기 때문이다. XML 문서에서 한 엘리먼트의 범위를 넘어선 컨텐츠 사이의 거리 연산은 별다른 의미가 없다. 그래서 본 논문에서는 [7]에서 정의된 proximity 속성과는 달리 특정 엘리먼트내의 컨텐츠에 대한 거리 연산만이 유효하다고 가정한다. 본 논문에서 제안하는 XFTS 인덱스 구조를 통해 처리할 수 있는 현재까지의 연구에서는 수행할 수 없었던 XML 쿼리의 예이다.

- Word 테이블을 통해서 다음과 같이 XML 문서에 대해서 구조적인 정보가 주어지지 않았을 때 처리할 수 있다.

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

- Element 테이블의 key_count 컬럼을 통해서 전문 검색 기능을 수행한다.

```
        //TITLE= 'Shakespeare Love'
SELECT E.DOCID, E.ELID, E.NAME
FROM LOCATION L, ELEMENT E
WHERE L.PATH = '~%/TITLE' AND
E.VALUE = 'Shakespeare Love' AND E.VAL_COUNT = 2
AND L.ID = E.PATHID;
```

- Word 테이블의 position컬럼을 통해서 엘리먼트 컨텐츠 proximity containment query[7]를 수행한다.

```
//ACT/SCENE/STAGEDIR/
            Distance('Lords', 'Soldiers') <= 3
SELECT E.DOCID, E.EID, E.NAME, E.VALUE
FROM LOCATION L, ELEMENT E,
WORD Lords, WORD Soldiers
WHERE L.PATH LIKE '~%/ACT~/SCENE~/STAGEDIR'
AND Lords.WORD = 'Lords'

AND Soldiers.WORD = 'Soldiers'AND Lords.DOCID =
Soldiers.DOCID
AND Lords.EID = Soldiers.EID
AND L.ID = Lords.PATHID
AND L.ID = Soldiers.PATHID
AND Soldiers.POSITION - Lords.POSITION > 0
AND Soldiers.POSITION - Lords.POSITION <= 3
AND E.DOCID = Lords.DOCID AND E.EID = Lords.EID
AND E.DOCID = Soldiers.DOCID
AND E.EID = Soldiers.EID;
```

5. 성능 평가

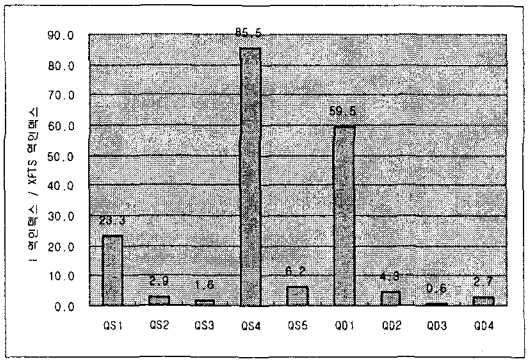
5.1 실험 환경

본 장에서는 본 논문에서 제안하고 있는 XML 전문 검색 기능의 실험 환경을 기술한다. 역인덱스를 생성하기 위해서 JDOM과 Java 1.4.1을 사용했다. 또한 이 실험이 행해진 상용 DBMS는 리눅스 2.4.20, 펜티엄 3 1.2GHz 듀얼 CPU에서 설치되어 있고 메인 메모리는 1.5GB이다. 관계형 DBMS의 인덱스로 주키만을 사용했다. 클라이언트는 윈도우 2000 Professional에서 펜티엄 4 1.4GHz이고 메인 메모리는 768MB이다. 실험에서 사용된 XML 문서와 생성된 역인덱스의 크기, 역인덱스를 저장한 테이블의 크기는 <표 2>와 같다.

5.2 실험 결과

시간 복잡도 측정을 위해서 Shakespeare's Work* XML 문서와 DBLP** XML 문서를 사용했다. 그리고 정확한 역인덱스 성능을 평가하기 위해서 비교적 간단한 쿼리로 실험했다. (그림 2)의 X축 레벨에서 중간의 영문자는 사용된 XML 문서의 종류를 나타내고 마지막 번호는 쿼리번호를 나타낸다.

(그림 2)와 (그림 3)은 [3]에서 연구된 역인덱스 기법과 본 논문에서 제안한 역인덱스 기법과의 시간 복잡도와 공간 복잡도를 비교한 것이다. (그림 2)에서 알 수 있듯이 QD3를 제외한 모든 경우에 XFTS인덱스가 I역인덱스보다 좋은 성능을 보인다. 특히 XML 문서에 대한 구조적인 정보없이 키워드만을 사용하는 QS1,QD1과 강한 종속 쿼리인 QS4에서 훨씬 좋은 성능을 보인다. 이런 성능 차이의 주된 원인은 I 역인덱스에서는 구체적인 구조 정보가 주어지지 않으면 XFTS 인덱스 구조에 비해서 훨씬 많은 중복된 레코드를 검색해야 하기 때문이다. 또한 구체적인 구조 정보를 주지 않는 [7]에서 제안된 간접적인 종속 속성이 두 번 이상 나타나면 I역인덱스에서는 구조를 찾을 수 없기 때문에 XQuery를 처리 할 수가 없다. 그렇



(그림 2) 시간 복잡도 성능 비교

<표 2> 실험 XML 문서의 크기 및 생성 인덱스 크기

	Shakespeare's Work	DBLP
문서의 크기	8	50
XFTS 역인덱스 크기	20	110
XFTS 테이블 크기	33	175
I 역인덱스 크기	104	440
I 테이블 크기	152	642

지만 본 논문에서는 각 엘리먼트의 경로를 저장하고 있기 때문에 그 처리가 가능하다. 그리고

XML 문서에서 나타나는 특정 엘리먼트의 키워드거리 연산의 경우에도 I역인덱스에서는 각 키워드에 대한 위치 정보가 없기 때문에 XQuery가 불가능하지만 XFTS에서는 엘리먼트내의 각 컨텐츠 키워드별로 번호를 부여했기 때문에 처리가 가능하다.

DBLP XML 문서의 크기를 조정하면서 실험한 공간

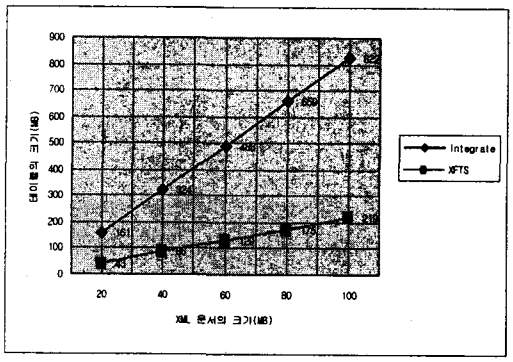
* http://www.oasis.org

** http://dblp.uni-trier.de/xml/

복잡도에 있어서도 (그림 3)에서 보듯이 XFTS가 거의 4배의 성능 향상을 보인다. 이런 성능 차이의 주된 이유는 I역인덱스 기법에서는 하나의 키워드에 대해서 중복 역인덱스를 구성하지만 XFTS에서는 한번만 구성하기 때문이다. 또한, XFTS에서는 여러 번 나타나는 같은 경로에 대해서는 한 번만 역인덱스를 생성하기 때문에 공간의 효율을 높일 수 있는 것이다.

6. 결론 및 향후 과제

최근까지 키워드 검색에 대한 연구는 많이 진행되어 왔지만 전문 검색에 대한 연구는 거의 없었다. 이에 지금까지 연구된 키워드 검색에 비해 훨씬 좋은 성능을 보이면서 전문 검색을 가능케 하는 인덱스 기법을 본 논문에서 제안했다. 가장 최근에 발표된 연구[5]의 문제점으로 제기된 특정 엘리먼트의 컨텐츠의 변경에 있어서도 본 논문의 XFTS 역인덱스 구조는 융통성을 보인다. 그러나 아직도 원래의 XML 문서의 크기보다 상당히 큰 역인덱스를 저장하기 위한 공간이 필요하기 때문



(그림 3) 공간 복잡도 성능 비교

에 이러한 공간을 줄이기 위한 압축 기법의 연구가 앞으로의 과제로 남아 있다.

참고 문헌

[1] D. Chamberlin, D. Florescu, J. Robie, J.Simeon, M. Stefanescu, XQuery: a query language for XML, Technical report, W3C Working Draft, February 2001.

[2] J. Clark, S. DeRose, XML path language (XPath) Version 1.0., Technical report, W3C Recommendation, November 1999.

[3] D. Florescu, D. Kossman, I. Manolescu, Integrating keyword search into XML query processing, Processing of the Ninth International World Wide Web Conference (2000).

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

[5] C. Seo, S.W Lee, H.J Kim, An efficient inveted index techniques for XML documents using RDBMS, Information and Software Technology 45(2003) 11-22.

[6] 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 (1) (2001) 110-141.

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