데이터베이스의 개념구조에 기반한 XML 문서의 색인 및 질의 스키마의 설계 및 구현

추 교 남[†] · 우 요 섭[††]

요 약

본 논문에서는 다양한 질의 처리를 위하여 데이터베이스 색인 스키마의 특징인 반구조적 정보를 구조 정보화할 수 있는 방법과 보다 빠르고 최적화된 질의처리 방법을 제안하고자 한다. XML 트리에 추가된 번호 정보를 비트열로 변형하여 트리의 구조 변경없이 트리 노드간의 구조 정보를 나타낸다. 그리고 이 과정에서 생성되는 기타 구조 정보들을 검색하여 색인 스키마에 추가한다. 또한, 질의 스키마에서는 색인 스키마를 이용하여 절대 경로 질의 표현식 뿐만 아니라 상대 경로 질의 표현식의 경우에도 주어진 노드 정보를 통하여 상위의 노드를 복원한다. 이러한 점은 하나의 질의를 통하여 파생 질의 표현식을 작성할 수 있다는 장점이 있다. 그리고 질의 처리 과정에서는 색인 스키마와 질의 스키마를 이용하여 비트열 사이의 비트 연산을 함으로써 응답시간을 최소화하고 색인 파일의 노드별 레코드의 정보만으로 정확한 결과를 검색할 수 있다.

키워드 : XML, 문서색인, 데이터베이스

Design and Implementation of XML Indexing and Query Scheme Based on Database Concept Structure

Choo Kyo Nam[†] · Woo Yo Seob[††]

ABSTRACT

In this paper, we propose a new indexing technique to solve various queries which have a strong good point not only database indexing schema take advantage of converting from semi-structured data to structured data but also performance is more faster than before. We represent structure information of XML document between nodes of tree that additional numbering information which can be bit-stream without modified structure of XML tree. And, We add in indexing schema searching incidental structure information in the process. In Querying schema, we recover ancestor nodes through give information of node using indexing schema in complete path query expression as well as relative path query expression. Therefore, it takes advantage of making derivative query expression with given query. In this process, we recognize that indexing and querying schema can get searched result set faster and more accurate. Because response time is become shorter by bit operating, when query occur and it just needs information of record set earch node in database.

Key Words : XML, Document Indexing, Database

1. 서 론

XML 문서의 사용 요구가 증가되고 문서의 양도 방대해짐에 따라, 그 문서들을 XML 표준에 의해 정의하기 위한 정보량 또한 급격히 증가하고 있다. 이에 따라 XML 문서 정보를 보다 효율적으로 검색하고 관리할 수 있는 방법[1-5]과 함께, 데이터베이스 등의 저장 매체에 XML 문서 정보를 효율적으로 저장하려는 연구도 활발하게 진행되고 있다[6-11]. 이러한 연구들은 동일한 구조를 가진 여러 XML 문서들에 대한 효율적인 경로 검색 지원을 목표로 하고 있다. 그러나 구조가 다른 여러 XML 문서들로부터 원하는 경로를 찾기 위해서는 각 XML 문서들에 대해 각각의 색인어를 구성하고, 검색해야만 한다.

검색 대상이 노드간 조상-자손(Ancestor-Descendant) 관계를 포함하는 경우, 관계있는 모든 노드를 탐색해야 하기 때문에, 검색 성능이 급격히 떨어지게 된다. 이를 보완하기 위한 색인 방법이 제안되기도 하였으나, 루트 노드가 아닌 중간 경로에서 조상-자손 관계가 나타나는 경우, 성능이 여전히 떨어지는 단점을 가진다[5]. 또한, 구조가 다른 XML 문서들을 함께 검색할 수 있는 통합 색인 방법[11-16]도 제안되었다. 이들은 정보 검색 분야에서 사용하는 역 색인에 XML의 구조적 성질을 반영시켜 확장하는 방식을 취하였다. 색인어를 포함하는 모든 노드 정보들을 데이터베이스내의 테이블에 저장하여 안정적으로 관리할 수 있게 구성하였으

※ 본 연구는 산업자원부, 한국산업기술평가원 지정 인천대학교 멀티미디어 연구센터의 지원에 의한 것입니다.
† 준 회 원 : 인천대학교 대학원 정보통신공학과 박사과정
†† 정 회 원 : 인천대학교 정보통신공학과 교수
논문접수 : 2006년 1월 20일, 심사완료 : 2006년 5월 2일

며, 효율적인 질의를 처리하는 기능을 경로 탐색에 이용함으로써 노드간 경로 검색을 효율적으로 지원하였다. 그러나, [6-9]에서 제안된 색인 기법들은 XML 문서의 수가 증가함에 따라 비교와 검색해야 하는 색인 데이터가 증가하게 되어 검색 성능이 저하되며, [15-16]은 구조가 다른 문서가 증가할 수록 성능이 떨어지는 단점을 보였다. 이러한 특징은 대량의 XML 문서들에 대한 노드간 경로 탐색을 어렵게 만드는 요인으로 작용한다.

이에, 본 논문에서는 복잡한 구조 검색을 피하고 비트열 기반의 XML 문서의 색인 알고리즘을 제안한다. 또한, 제안하는 색인 구조를 기반으로 하여 XML 문서에 대한 효율적인 질의처리가 가능하도록 한다.

2. 관련 연구

XML 문서는 복잡하고 다양한 구조로 나타날 수 있기 때문에 XML 문서에 맞는 구조 기반 검색 기법이 필요하게 되었고, 이런 구조 기반 질의를 효율적으로 처리하기 위해 XML 문서의 색인 기법이 필요하게 되었다. XML 문서에 대하여 질의가 발생하였다면 질의를 구성하는 원소나 속성, 텍스트등간의 구조 관계를 빠르게 결정해 내야하며, 이런 구조 관계로 발생할 수 있는 모든 경우를 문서 구조내에서 효율적으로 검색할 수 있어야 한다. 이를 위한 XML 문서 정보 검색에 관한 여러 선행 연구가 진행되어 왔다.

먼저 살펴볼 것은, XML 문서의 색인 방법에 관한 연구이다. 만약, XML 문서상에 "Section"이라는 원소의 하위에 "Figure"라는 원소에 대한 질의가 발생 하였다면, "Section" 원소 이하의 모든 서브 트리(Sub-Tree)를 검색하여 "Figure"를 찾는 것이다. 그러나 이러한 방법은 큰 부하를 초래하여 탐색 시간을 증가시키는 단점이 있다[6, 8, 12, 14]. 에서는 이러한 기본 검색 구조에 부가적인 정보를 이용하여 보다 효율적으로 검색을 수행하려는 시도를 찾아볼 수 있다. XML 문서 트리 상에 번호 부여 기법(Numbering Scheme)을 추가하여 보다 빠르게 구조 정보를 결정할 수 있도록 하였다. 이 경우, 질의에 대해서는 성능이 우수하나 문서의 구조 변화가 빈번한 경우 문서 트리에 대한 번호 부여가 다시 이루어져야 한다는 문제점과 이에 따른 색인 테이블의 갱신이 자주 발생한다는 단점이 있다.

다른 선행 연구로는 XML 문서의 구조 병합(Join) 알고리즘에 관하여 살펴볼 필요가 있다. 구조 병합 알고리즘은 번호 부여 기법을 향상시키기 위해 제안되었고, 이는 XML 질의를 최적화하는데 그 핵심을 두고 있다[6-9, 14-16]. 예를 들어 원소와 원소 간의 병합이나, 원소와 속성의 병합, 또 문서상에 존재하는 문자열 값과 원소, 속성들 간의 병합을 통해 보다 효율적 질의 수행을 할 수 있도록 제안된 것이다. 그 방식에는 정보검색분야의 전통적 색인 방법인 역 색인(Inverted Index)과 관계형 데이터베이스를 대상으로 주로 수행되었다. 그러나 이러한 방법은 효과적인 질의를 위해 다양한 형태의 병합연산을 수행해야 하며, 부모-자식(Parent-Child)관계와 같은 구조 관계를 결정해 내기 위한 테이블 간의 병합 연산 을 수행하는 과정에서 불필요한 병합 연산도 발생하게 된다. 또한, 질의의 결과를 반환 할 때에는 XML로 복원하여야 하기 때문에, 복원에 필요한 구조 정보를 함께 저장해야하는 문제점을 가지게 되었다.

이에 본 논문에서는 XML 문서 자체에 모든 노드들을 구별할 수 있는 비트 스트림 구조를 XML 문서 트리에 추가하여 검색하고자하는 임의의 노드를 많은 수의 테이블 간의 조인 연산 없이 빠르게 검색할 수 있는 방법을 제안하고자 한다.

3. 데이터베이스 구조 기반의 색인과 질의 스키마

3.1 색인 스키마

본 논문에서 제안하는 색인 과정은 우선 XML 문서를 트리 구조로 변환하고 XML의 각 노드들에 순차적으로 각 트리의 레벨을 기준으로 연속적인 번호를 부여하여 확장된 트리 구조로 재구성한다. 다음으로 트리 구조 내의 노드 구조 정보를 효율적으로 처리하기 위하여 64비트의 정적 공간에 각 레벨에 존재하는 노드 수만큼 가변적인 비트를 할당하고 고유한 비트 스트림을 생성한다.

■ 색인 스키마를 구성하기 위한 수행하는 과정은 다음과 같다.

① XML 문서를 DOM(Document Object Model) 트리를 사용하여 트리로 구성한다.
② 작성된 트리에 각 레벨의 노드에 대하여 순차적인 번호를 부여한다.
③ 번호를 부여 받은 확장 트리가 재구성 된다.
④ 확장 트리에서 각 노드에 부여된 번호를 이용하여 비트 스트림을 구성한다.

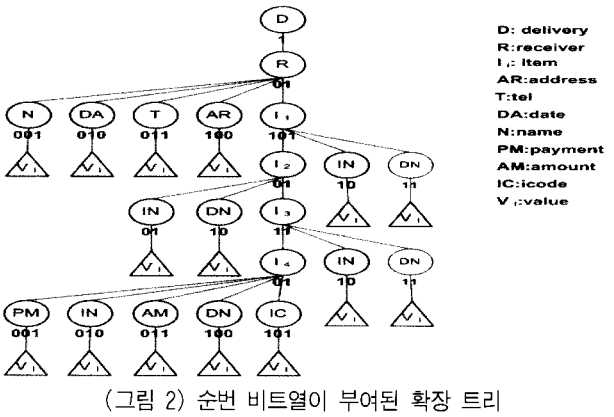
3.1.1 XML 트리의 번호 부여 기법

XML문서의 예를 (그림 1)에서 보였다. (그림 2)는 각 레벨의 노드마다 순차적으로 번호를 부여하고 각 레벨의 노드 수를 표현 할 수 있는 비트를 할당한다. 루트 노드로부터 시작하여 최하위의 노드들까지 순차적 번호 부여를 하게 된다. XML 문서 트리의 레벨을 기준으로 번호를 부여하게 되면 특정 노드를 기준으로 동일한 자식노드가 하위의 같은 레벨에서 연속하여 나타나거나 수직적으로 반복되어 나타나는 모든 노드 구조에 대하여 유연하게 색인을 할 수 있게 된다.

이때 트리의 노드마다 부여하는 번호 중에 비트 0으로 시작하는 것은 제외하고 비트 1부터 부여하도록 한다. 이는 질의 표현식을 비트열로 변환하는 과정에서 비트열 값이 1인 값은 부여된 값임을 쉽게 알 수 있으나, 0비트의 값은

```
<delivery>
  <receiver address="incheon" tel="123-4567" date="2004-04-01" >
    <name> kimiljin </name>
     <item iname="book" dn="0">
       <item iname="computer" dn="1">
          <item iname="xml" dn="2">
            <item iname="XML 입문" dn="3"
          payment="true"  icode="20040312BE" fee="2000"
                    amount="10" >
              <raddress>University of Incheon, do-hwa dong
                        Nam-gu
                    Incheon, korea</raddress>
                <price> 2000 </price>
          </item>
       </item>
      </item>
     </item>
  </receiver>
          ....
          <중략>
          ....
</delivery>
```

(그림 1) XML 문서의 예



(그림 2) 순번 비트열이 부여된 확장 트리

부여된 비트 값인지 아닌지를 확인하여야 하는 번거로움이 발생하기 때문이다. 예를 들어 001의 경우 최하위 비트 1의 값은 해당 노드에 대하여 순차적으로 부여받은 것임을 알 수 있으나 최하위 비트 1을 제외한 00은 트리 내에서 번호로 부여 받은 것인지를 확인하여야만 한다.

이 연산을 수행하고 난 후에는 루트 노드를 기점으로 각 노드들의 경로를 따라 부여된 번호를 연결하게 되면 각 경로마다 유일한 비트열이 생성된다. 이때 이 값 자체는 루트 노드로부터 경로가 끝나는 노드까지의 상하의 관계 즉 부모-자식, 조상-후손, 형제등과 같은 트리의 모든 노드들의 구조 정보를 나타내게 된다.

3.1.2 XML 구조 정보의 비트열 생성

(그림 2)에서 루트 노드로부터 연속적으로 노드를 방문하면서 각 노드에 부여된 비트열을 얻고 이 비트열을 할당된 고정 비트 공간의 최하위 비트에서 시작하여 각 레벨 별로 비트열 값을 저장한다. <표 1>의 L0부터 L7까지에 표현된 비트 값들은 (그림 2)의 각 노드별 계층적 관계에 대한 비

〈표 1〉 각 노드에 할당된 비트열

0_i : '0'비트의 i개 반복

레벨	L_7	L_6	L_5	L_4	L_3	L_2	L_1	L_0
전체 비트수	←			64 비트				→
노드별 비트 스트림	0_{49}	0_3	0_2	0_2	0_2	0_3	0_2	1
							01	1
						001	01	1
						100	01	1
						101	01	1
					01	101	10	1
					10	101	01	1
				01	01	101	01	1
				11	01	101	01	1
			01	11	01	101	01	1
			10	11	01	101	01	1
		001	01	11	01	101	01	1
		010	01	11	01	101	01	1

트열을 나타낸 것이다. 이 비트열은 모든 노드들과 일대일로 대응이 되며 고유하고 유일한 값이다. 또한, 각 노드의 원소별 비트열은 유일한 값이기 때문에 XML 문서로 다시 복원할 수 있다. 비트열의 전체 크기는 64 비트로 고정된 공간을 할당한다. 하지만 XML 문서의 레벨이 늘어나거나 레벨별 노드의 수가 증가하면 고정된 공간인 64비트를 초과할 수 있다. 이때는 64비트 할당 공간보다 더 큰 공간을 할당하기만 하면 된다.

그러나 할당 공간이 증가함에 따른 비트열에 대한 전체 알고리즘은 영향을 받지 않는다. 이는 수많은 문서를 처리해야 하는 경우에도 색인 스키마는 변화 없이 비트열 구조만 확장되는 장점을 가진다. 이렇게 얻어진 노드별 비트열은 다시 16진수 형태로 변환되고, 데이터베이스 색인 스키마 테이블의 각 필드 별로 저장된다.

3.1.3 색인 테이블의 구성

색인 테이블에는 사용자의 질의가 발생됐을 때 질의 분석에 필요한 각 노드의 이름, 고유한 비트열, 부모 노드의 비트열과 레벨값 등이 저장된다. (그림 3)는 색인 파일의 스키마 구조를 나타낸 것이다. 각 필드의 역할은 다음과 같이 정의된다.

① N_name : XML 트리 노드의 이름
② B_value : 각 노드의 비트열 값
③ Tb_len : 해당 노드 비트열 전체의 길이 값
④ B_len : 현 노드에 가변적으로 할당되는 비트열의 길이 값
⑤ Level : 현 노드가 XML 트리 상에 존재하는 레벨
⑥ P_value : 현 노드의 부모 노드의 비트열
⑦ Ctype : 현 노드가 원소인지 속성인지를 나타냄
⑧ Position : 현 노드가 부모로부터의 몇 번째 자식 노드인지를 나타냄
⑨ D_num : XML 문서의 문서 번호
⑩ Data : 각 노드의 데이터

N_name	B_value	Tb_len	B_len	Level	P_value	Ctype	Position	D_num	Data

(그림 3) 색인 스키마 테이블 필드

N_name	B_value	Tb_len	B_len	Level	P_value	Ctype	Position	D_num	Data
tel	0000001B	6	3	2	00000003	1	3	1	5558417

(그림 4) 색인 스키마 테이블의 예

(그림 4)은 tel 노드의 구조 정보를 나타낸 레코드 구조이다. N_name 필드에는 노드 이름인 tel 저장되고, XML 트리에서 얻을 수 있는 64비트의 비트열은 B_value 필드에 16진수 형태로 저장한다. 루트노드로부터 시작된 tel의 비트 스트림 총 길이는 Tb_len 필드에 저장되고 B_len은 tel이 속한 레벨에서 표현되는 비트열의 길이를 저장한다. 결국 총 비트열 길이인 Tb_len 필드값과 현 노드의 비트열 길이인 B_len 값과의 차로 현 노드의 상위 비트의 총 길이를 얻을 수 있다. 다음으로 Level 필드는 루트 노드를 '0' 레벨로 시작하여 속성 tel이 속한 레벨 값이 저장되고, P_value 필드는 tel의 한 레벨 상위인 부모 노드의 비트열이 저장된다. 질의 발생시 질의의 절대 경로 복원, 질의의 결과 반환 시 부모의 형제 노드의 추적과 같은 부모에 관련된 연산을 수행할 때 참조할 필드이다.

Ctype 필드는 tel이 원소인지 속성인지를 구분하는 필드이다. 이 필드도 질의 표현식 내용을 정확하게 판단하여 적절한 결과를 반환하기 위하여 사용된다. Position 필드는 상위 노드의 자식으로써 tel의 순차적인 위치를 저장하는 필드이다. D_name 필드는 tel이 속한 문서 번호로써, 질의 표현식에서 얻고자하는 결과들을 문서 번호로써 구분하고 특정 문서에 대한 정확한 결과를 얻고자 할 때 사용된다. Data 필드는 tel가 가지고 있는 데이터를 저장하게 된다. 원소의 경우 시작 태그와 종료 태그 사이의 텍스트 값을 가지는 경우와 아무 값도 가지지 않는 경우가 있다. 속성 값의 경우도 문자열 값이나 원소 생성에 관련된 실수값을 가지는 경우도 있다.

(그림 5)는 (그림 2)의 XML 트리로부터 얻어진 색인 정보를 저장한 테이블이다. 이때, 저장된 노드별 레코드 셋은 유일한 값이다. 그러므로 레코드 세트를 이용한 질의 처리 방식은 기존에 연구에서 결과 세트를 찾기 위해 단편적인 데이터들이 저장된 테이블간의 연관성에 기초한 병합 방식과 부수적 연산을 수행하는 경우 보다는 훨씬 효율적이다. 질의가 특정 문서를 대상으로 하였다면, 문서 정보 D_num 필드를 참조하여 보다 쉽게 문서에 대한 질의를 수행할 수 있으며, 질의에 문서 정보가 없을 시에도 D_num 필드를 참조하여 반환 된 결과 중에서 서로 다른 문서의 노드 정보간

N_name	B_value	Tb_len	B_len	Level	Data	P_value	Ctype	Position	D_num
delivery	00000001	1	1	0	없음	00000000	0	1	1
receiver	00000003	3	2	1	없음	00000001	0	1	1
name	0000000B	6	3	2	홍길동	00000003	0	1	1
date	00000013	6	3	2	040301	00000003	1	2	1
tel	0000001B	6	3	2	5558417	00000003	1	3	1
address	00000023	6	3	2	인천	00000003	0	4	1
item	0000002B	6	3	2	없음	00000003	0	5	1
item	0000006B	8	2	3	없음	0000002B	0	6	1
iname	000000AB	8	2	3	book	0000002B	1	7	1
dn	000000EB	8	2	3	0	0000002B	1	8	1
iname	0000016B	10	2	4	computer	0000006B	1	1	1
dn	0000026B	10	2	4	1	0000006B	1	2	1
item	0000036B	10	2	4	없음	0000006B	0	3	1
① item	0000076B	12	2	5	없음	0000036B	0	1	1
iname	00000B6B	12	2	5	xml	0000036B	1	2	1
dn	00000F6B	12	2	5	2	0000036B	1	1	1
payment	0000176B	15	3	6	true	0000076B	1	2	1
② iname	0000276B	15	3	6	xml 입문	0000076B	1	3	1
amount	0000376B	15	3	6	2	0000076B	0	4	1
dn	0000476B	15	3	6	3	0000076B	1	5	1
icode	0000576B	15	3	6	4031BE	0000076B	0	6	1

(그림 5) 색인 테이블의 예

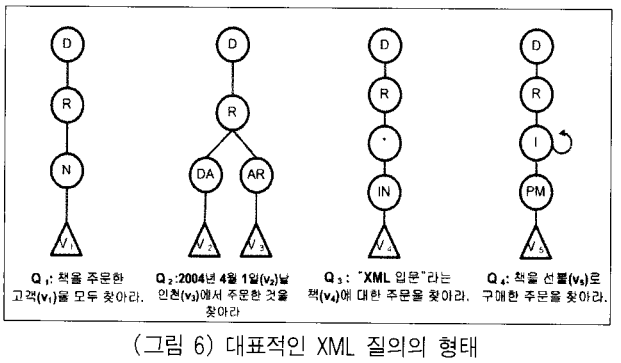
의 결합으로 표현될 수 있는 질의 형태를 방지할 수 있다.

예를 들어, (그림 5)의 ② iname 노드에 대한 검색 과정을 살펴보도록 하자. ② iname 노드의 비트열 필드(B_value 필드)값 000276B는 XML 문서 트리 상의 iname 노드의 위치를 알려 준다. 그리고 부모 노드의 비트열 정보(P_value)를 통하여 상위 노드를 찾을 수도 있다. (그림 5)의 ① item 노드의 비트열이 ② iname의 상위 노드 비트열 0000076B과 동일하기 때문에 ① item이 ② iname의 부모 노드임을 알 수 있다. 이러한 연산을 반복한다면, ② iname로부터 루트 노드사이에 속한 어떠한 노드도 검색할 수 있다. 만약 문서의 양이 증가하여 부모의 비트열이 일치하는 경우도 발생할 수 있으나, 이때는 레벨 정보, 원소나 속성의 이름, 데이터를 비교함으로써 구별해 낼 수 있다. 또 ② iname 노드의 형제 노드를 검색하는 질의가 주어졌다면, 별다른 병합 연산 없이 ② iname 노드의 D_num 필드와 Level 필드와 같은 값을 갖는 모든 노드 정보는 형제 노드가 된다. 만약 문서에 제안된 색인 파일 구조에 저장할 수 없는 상이한 구조 정보가 추가되었다면, 기존의 색인 테이블의 변형없이 새로운 구조 정보를 나타내는 필드를 추가함으로써 색인 알고리즘의 확장성을 보장할 수 있다.

3.2 질의 스키마

XML 문서에 대하여 질의가 발생되었다면, 먼저 데이터베이스와 정합을 위하여 주어진 XPath 표현식에 대하여 전체 질의 경로를 색인 테이블에서 참조하여 비트열로 변환한다. 이때 질의 표현식이 절대 경로가 아닌 상대 경로로 주어졌다면 이를 전체 질의 경로(Full Query Path)로 복원해야 한다. 질의 표현식에 나타난 원소나 속성의 이름을 색인 테이블의 상위 노드 정보 필드를 참조하여 루트 노드에 이를 때까지 복원할 수 있다. 완전한 질의 표현식으로 복원되어졌다면, 이를 색인 스키마에서 얻어진 데이터베이스 색인 정보에서 적절한 질의 결과를 검색하게 된다.

질의 표현식을 비트열로 변환하는 가장 큰 이유는 변환된 비트열을 색인 테이블과 다시 정합을 시에 비트열 자체가 중복없는 고유한 값이기 때문에 가장 빠르게 원하는 결과값을 검색할 수 있다. 게다가 정합되는 비트열은 문자열 정합보다 매우 빠르게 수행되므로 연산 시간도 짧아질 수 있다. (그림 6)는 일반적 XML 질의의 4가지 형태를 나타낸 것이다.



(그림 6) 대표적인 XML 질의의 형태

〈표 2〉 XML 질의 Q1의 비트열 연산

0$_i$: '0'비트의 i개 반복

노드 \ 비트 연산		64 비트		
노드 Delivery	And 연산	0_{61}	00	1
노드 Receiver		0_{61}	01	1
(a)		0_{61}	00	1
노드 Receiver	And 연산	0_{58}	000	011
노드 Name		0_{58}	001	011
(b)		0_{58}	000	011

(그림 6)의 질의 Q1(/Delivery/Receiver/Name)에 대하여 (그림 5)을 참조하여 XPath 표현식을 구성하는 각 원소나 속성의 비트열로 변환할 수 있다. 우선 XPath 형태의 질의가 주어지면 표현식 상의 최상위 노드의 비트열을 색인 파일로부터 얻어온다. <표 2>에서 보는 바와 같이 질의 Q1의 노드 Delivery와 노드 Receiver의 AND 연산이다. 여기서 우리는 AND 연산의 특성을 알아볼 필요가 있다. 비트 간의 AND 연산을 수행할 때 비교하는 비트열 간의 비트 값이 같다면 연산의 결과도 비교 대상인 비트열 값이 반환되는 점을 이용하여 비트열 간의 비트 연산을 수행하였다. 노드 Delivery와 노드 Receiver의 AND 연산을 하게 되면 비트 값이 1인 경우만 남고 나머지는 모두 0비트가 된다. 연산의 결과 (a)는 모든 비트가 0 비트이고 최하위 비트 값만 1이다.

만약 연산의 결과 (a)와 노드 Receiver의 비트 스트림 값이 최하위 비트로부터 시작하여 일부 비트가 일치하였다면 우리는 연산 결과 (a)가 노드 Receiver의 비트열에 포함되어 있다는 것을 알 수 있다. 이때 일치하는 모든 비트를 분석한다면 더욱 명확하게 비교 노드간의 구조 관계인 상,하위 관계 혹은 형제 관계를 판별해 낼 수 있다. 하지만 <표 2>에서 보듯이 노드 Delivery와 노드 Receiver의 AND 연산의 결과 (a)는 노드 Delivery의 비트 스트림과 일치한다. 이러한 결과는 노드 Receiver의 비트열에는 노드 Delivery의 비트열이 포함되어 있다는 것이고, 또 노드 Delivery의 비트열에 대한 총길이가 짧으므로 노드 Delivery가 노드 Receiver의 상위 노드임을 말해준다.

두 번째 질의 형태 Q2(/Delivery/Receiver/[@Date=V_2]/[@Address=V_3])에 대한 처리 과정도 첫 번째 질의 처리 과정과 마찬가지로 노드 Receiver와 노드 Name의 AND 연산이다. 이번에도 연산의 결과 (b)의 값이 노드 Receiver의 비트 스트림과 일치한다. 이 또한 노드 Receiver가 노드 Name의 상위 노드임을 말해준다. 그래서 연산 결과 (a)와 (b)를 통하여 노드 Delivery는 노드 Receiver와 노드 Name의 상위이고 노드 Receiver는 노드 Name의 상위 노드가 됨을 알 수 있다. 그러므로 질의 Q1의 비트 스트림 값은 노드 Name의 비트 스트림 값이 대표할 수 있다. 이와 같은 과정으로 상위 관계는 확인 할 수 있으나 부모 관계는 아직 알 수 없음으로 이를 위하여 (그림 5)의 색인 테이블을 참조하여야만 한다. 이때 Level 필드 값은 하위 노드보다 레벨 값이 1 작은 값을 가

〈표 3〉 XML 질의 Q2의 비트열 연산

0_i: '0'비트의 i개 반복

노드	비트 연산	64 비트		
노드 Delivery	And 연산	0_{61}	00	1
노드 Receiver		0_{61}	01	1
(c)		0_{61}	00	1
노드 Receiver	And 연산	0_{58}	000	011
노드 Date		0_{58}	010	011
(d)		0_{58}	000	011

노드	비트 연산	64 비트		
노드 Delivery	And 연산	0_{61}	00	1
노드 Receiver		0_{61}	01	1
(e)		0_{61}	00	1
노드 Receiver	And 연산	0_{58}	000	011
노드 Address		0_{58}	100	011
(f)		0_{58}	000	011

지면 부모이고, P_value 필드 값과 이전에 확인한 상위 노드의 비트 스트림 값과 비교함으로써 부모 관계를 확인 할 수 있다. 이상으로 질의 Q1이 성공적으로 비트 스트림으로 변환되어졌다.

질의 Q2는 질의 Q1이 확장된 형태이므로 여러 개의 하위 질의로 분해하여 질의 Q1에 적용했던 방식을 이용하여 정확한 질의 표현식으로 표현되었는지를 확인하면 된다. 질의 Q2의 각 노드별 비트열은 색인 파일로부터 얻어 올 수 있다. 이때도 역시 색인 파일의 Level 필드와 P_value 필드를 확인함으로써 부모 관계를 알 수 있고, 질의 Q1과 같이 수행하여 주어진 질의 Q2를 <표 3>의 연산 결과에 의하여 질의 표현식의 신뢰성을 확인할 수 있다. 그러므로 질의 Q2의 비트 스트림 값은 노드 Date의 비트 스트림 값과 노드 Address의 비트열 값의 의하여 분리된 질의로 표현될 수 있다.

질의 Q3(/Delivery/Receiver/*/[IN=V_4])은 상위의 일부 노드와 최종단의 노드가 주어진 형태의 질의이므로 주어진 상위 일부 노드에 대해서는 질의 Q1과 같은 방법으로 질의 내용을 확인한다. 질의 Q3의 경우 중간 경로를 알 수 없기 때문에 생략된 경로를 복원하는 방법으로 우선 최종단의 노드 값(V4)을 색인 파일로부터 얻어온다. 그 값들과 상위 노드의 비트 열과 비트 연산 XOR를 하여 주어진 질의에 나타나있는 상위 노드 비트열(/Delivery/Receiver/)과 종단 노드(v4)의 비트 스트림이 제거된 중간 노드의 비트열과 같은 비트를 가지는 모든 경로들을 추출해 낸다. 결과 검색을 위하여 <표 4>의 연산 결과 (g)와 (f)를 통하여 얻어진 질의 표현식들을 가지고 색인 테이블을 참조하여 검색한다.

〈표 4〉 XML 질의 Q3의 비트열 연산

0_i: '0'비트의 i개 반복

노드	비트 연산	64 비트			
노드 Delivery	And 연산	0_{61}		00	1
노드 Receiver		0_{61}		01	1
(g)		0_{61}		00	1
노드 Receiver	XOR 연산	0_{49}	000	000000000	011
노드 IN		0_{49}	010	011101101	011
(h)		0_{49}	010	011101101	000

질의 Q4(/Delivery/Receiver//Item[Payment=V_5])는 반복 구조가 나타나는 노드 이전까지 질의 Q1과 같은 연산을 수행하여 <표 5>의 연산 결과 (i)를 얻을 수 있다. 반복 구조를 갖는 노드가 시작되면 현 노드의 자식 노드 중에 자신과 같은 구조를 같은 노드가 있는지 확인을 하고 만약 존재한다면 반복적으로 자식 노드에 대해서도 같은 방식을 적용하여 하위 노드를 검색한다. 이런 연산을 반복 구조를 갖는 노드가 끝날 때까지 계속 진행하여 <표 5>의 (j)부터 (m)까지 얻어 낼 수 있고, 이때 이전의 질의 Q1에서 수행한 것과 같이 각각 노드들의 Level 필드 값과 P_value 필드 값을 비교하여 부모 관계를 확인하여야 한다. 이 작업을 마치면, 이후 노드에 대하여 주어진 질의 표현식의 최하위 노드(V5)와 같은 값을 갖는 자식 노드를 포함하고 있는지 검색한다. 연산의 결과로 주어진 질의의 최종단 노드(V5)의 비트 스트림 길이와 일치하는 부분이 포함된 비트열을 얻을 수 있다. (<표 5>의 연산 결과(n)).

〈표 5〉 XML 질의 Q4의 비트열 연산

0_i: '0'비트의 i개 반복

노드	비트 연산	64 비트		
노드 Delivery	And 연산	0_{61}	00	1
노드 Receiver		0_{61}	01	1
(i)		0_{61}	00	1
노드 Receiver	XOR 연산	0_{58}	000	011
노드 Item		0_{58}	101	011
(j)		0_{58}	101	000
노드 Item	XOR 연산	0_{56}	00	101011
노드 Item		0_{56}	01	101011
(k)		0_{56}	01	000000
노드 Item	XOR 연산	0_{54}	00	01101011
노드 Item		0_{54}	11	01101011
(l)		0_{54}	11	00000000
노드 Item	XOR 연산	0_{52}	00	1101101011
노드 Item		0_{52}	01	1101101011
(m)		0_{52}	01	0000000000
노드 Item	XOR 연산	0_{49}	000	011101101011
노드 payment		0_{49}	001	011101101011
(n)		0_{49}	001	000000000000

4. 실험 결과

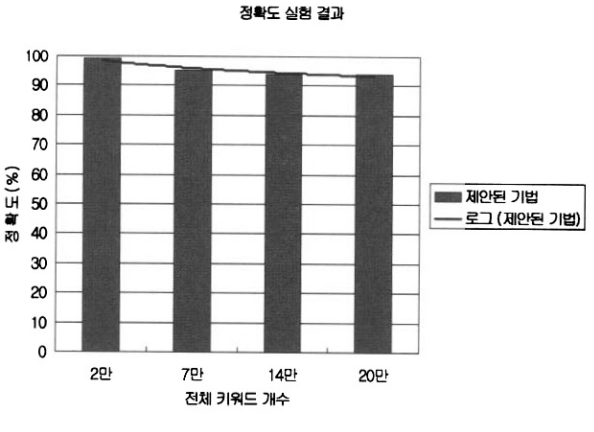
4.1 질의 수행의 정확도

본 논문에서 제안한 XML 색인 시스템의 질의 수행 정확도 실험을 위해 <표 6>과 같은 실험데이터를 구축하였다.

<표 6>의 실험 데이터 중 XPath 깊이가 10인 질의 종류를 1000회 수행되었고, 결과는 (그림 7)과 같다. (그림 7)에서 보는 바와 같이 질의 표현식에 나타나는 원소 및 속성, 텍스트의 개수가 증가 하더라도 정확도는 99% 이상이었음을 알 수 있다. 에러률이 1%이내였는데, 그 원인으로는 문서의 복잡도가 증가함에 따라 XML 문서 트리의 레벨별 노드 수가 증가하게 된다. 이로 인해 각 노드의 구조 정보를 표현할 수 있는 비트 스트림의 길이가 64비트를 초과하게 되어 질의 처리 연산이 실패하거나 질의 처리가 성공하더라도 적절하지 못한 결과를 반환하였다.

〈표 6〉 정확도 평가를 위한 실험 데이터

의 미	값
전체 문서 개수	1000
평균 문서 크기	10KB
문서당 평균 원소 개수	128
문서당 평균 속성의 개수	98
원소의 평균 깊이	5.24
문서당 평균 키워드 개수	226
원소의 평균 자식 개수	4.29
K-ary 기반 색인에서 평균 K값	10
전체 원소 개수	128,942



(그림 7) XML 질의에 대한 검색 결과의 정확도

4.2 질의 처리의 응답 시간 실험

질의 처리의 응답 시간 실험을 위하여 실험 데이터와 질의는 [10]에서 INRIA와 XRel의 비교 실험에 사용된 셰익스피어의 희곡을 사용하였다. 실험을 위한 데이터는 <표 7>에서 보는 바와 같고, 전체 문서의 원소 수와 출현 빈도수는 <표 8>과 같다.

〈표 7〉 응답시간 평가를 위한 실험 데이터

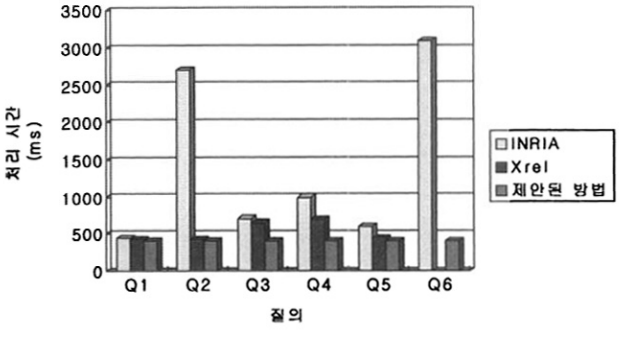
항 목	값
문서 개수	35
문서의 총 크기(MB)	6.91
평균 크기(KB)	197.4
원소 노드의 수	170,728
텍스트 노드의 수	140,018
경로의 개수	57
문서의 깊이	6
문서당 평균 원소 노드의 수	4,878
문서당 평균 속성 노드의 수	4,000

〈표 8〉 XML 문서의 중요 노드의 개수

원 소	개 수	비 율(%)
ACT	178	0.10
TITLE	982	0.57
SCENE	715	0.41
STAGEDIR	5,974	3.49
SPEECH	29,544	17.30
SPEAKER	29,544	17.54
LINE	102,294	59.91

비교 대상으로는 INRIA의 Single Inlining Edge 테이블과 XRel 간에 이루어졌으며, 실험 대상의 테이블 및 색인의 구성은 각각 [10]과 [16]의 내용을 참고로 하였다. 하지만, XRel의 경우 '*' 연산자를 지원하지 않아 사용자의 부가적인 정보 입력이 필요하기 때문에 실험에 사용된 질의에서는 이를 생략하였다. 제안된 질의 처리 방법에 대한 성능 평가는 변환된 질의 SQL문이 데이터베이스에 전달된 시점부터 질의 결과가 출력되는 시점까지의 시간을 측정하였다. 각 질의에 대하여 100회씩 반복하여 수행된 결과의 평균값을 사용하였다.

(그림 8)에서 보는 바와 같이 질의수행에 있어서 기존의 INRIA와 XRel이 제안한 방법에는 주어진 질의에 대한 결과를 검색하기 위하여 병합 연산을 수행하지만 본 논문에서



(그림 8) 질의에 대한 검색 결과의 응답 시간

제안한 질의 방식의 경우 이미 색인 파일 접근 이전에 최적화된 질의 형태로 변환하였기 때문에 데이터베이스 접근 이후에 발생하는 테이블 간의 병합 연산이 없어 INRIA과 XRel의 연산 결과와 큰 차이를 보였다. 즉 위의 실험 결과에서 주어진 질의에 대하여 부모-자식, 형제 관계 등의 구조 관계를 찾아내기 위한 병합 연산을 본 논문에서는 이미 XML 문서 트리의 비트 스트림과 다른 구조 정보를 이용하여 분석하였기 때문에 질의 응답 시간을 줄일 수 있었다.

5. 결 론

본 논문에서는 데이터베이스의 개념 구조에 기반한 XML 문서의 효율적인 색인 및 질의 스키마를 제안하였다. 문서의 깊이가 깊고 복잡한 문서 구조를 대상으로 질의 정확도와 응답 시간 측정을 수행하여 제안한 색인과 질의 방법에 대한 효율성을 보였다. 본 논문에서 제안한 색인 및 질의 스키마는 XML 문서상의 구조 정보와 데이터들을 검색하여 많은 정보를 관리할 수 있는 데이터베이스 등의 저장매체에 저장할 수 있다. 하지만 XML 문서가 변경되면 XML 문서 트리에 부여하는 번호들이 변경될 수 있으며, 그로 인한 부가적인 처리가 필요할 수 있다. 향후 연구에서는 XML 문서 트리가 변경되어도 비트열의 변경이 발생하지 않는 비트열의 동적 모델에 관한 연구가 필요하다. 또한, XML 문서 색인에 필요한 구조 정보가 증가함에 따른 데이터베이스의 테이블의 크기가 커지는 것에 대응하기 위한 색인 파일 구조의 확장에 대한 연구가 추가적으로 이루어져야 할 것으로 생각된다.

참 고 문 헌

[1] Tuong Dao, Ron Sacks-Davis, and James A. Thom, "An Indexing Schema Structured Documents and its Implementation", Proceedings of the 5th International Conference on Database Systems for Advanced Application, pp.125-134, 1997.

[2] Roy Goldman and Jennifer Widom, "DataGuides : Enabling Query Formulation and Optimization in Semistructured Databases", Proceedings of VLDB, pp.436-445, August 1997.

[3] Tova Milo and Dan Suciu, "Index structures for Path Expressions", Proceedings of ICDT, pp.277-295, January, 1999.

[4] Brian F. Cooper, Neal Samfple, and Michael J. Franklin, etc, "A Fast Index for Semistructured Data", Proceedings of VLDB, pp.341-350, January, 2001.

[5] Chin-Wan Chung, Jun-Ki Min, and Kyu-seok Shim, "APEX : An Adaptive Path Index for XML Data", Proceedings of SIGMOD, pp.121-132, June, 2002.

[6] C. Zhang, J. Naughton and D. DeWitt, Q. Luo, etc, "On Supporting Containment Queries in Relational Database Management Systems", Proceedings of SIGMOD, 2001.

[7] Daniela Florescu, Donald Kossman, and Ioana Manolescu, "Intergrating Keyword Search into XML Query Processing", WWW/Computer Networks, pp.33(1-6) : 119-135, 2000.

[8] Shurug Al-Khalifa, H. V. Jagadish, Nick Koudas and Jignesh M. Patel, etc, " Structural Joins : A Primitive for Efficient XML Query Pattern matching", Proceedings of ICDE, pp.141-153, February, 2002.

[9] Shu-Yao Chien, Zografoula Vagena and Donghui Zhang, etc, "Efficient Structural Joins on Indexed XML Documents", Proceedings of VLDB pp.263-274, August, 2002.

[10] Masatoshi Yoshikawa and Toshiyuki Amagasa, "XRel : A Path-Based Approach to Storage and Retrieval of XML Documents Using Relational Databases", Proceedings of TOIT, Vol.1, No.1, pp.110-141, August, 2001.

[11] Chiyoung Seo, Sang-Won Lee, and Hyoung-Joo Kim, "An Efficient Inverted Index Technique for XML Documents Using RDBMS", Information and Software Technology (Elsevier Science), Vol. 45, Issue 1, pp.11-22, January, 2003.

[12] P.F. Dietz and D. Sleator, "Two Algorithms for Maintaining Order in a List", Proceedings of STOC, 1987.

[13] T. Grust, "Accelerating XPath Location Steps", Proceedings of SIGMOD, 2002.

[14] Q. Li and B. Moon, "Indexing and Querying XML Data for Regular Path Expression", Proceedings of VLDB, 2001.

[15] J. McHugh and J. Widom, "Optimization for XML", Proceedings of VLDB, 1999.

[16] D. Floresc and D. Kossman, "A Performance Evaluation of Alternative Mapping Schemes for Storing XML Data in a Relational Database", Technical Report, INRIA, 1999.

추 교 남

e-mail : kyonam@incheon.ac.kr

1997년 인천대학교 정보통신공학과(학사)

1999년 인천대학교 정보통신공학과 (공학석사)

2003년~현재 인천대학교 정보통신공학과 박사과정

관심분야 : 한국어정보처리, XML, 인공지능

우 요 섭

e-mail : yswooo@incheon.ac.kr

1986년 한양대학교 전자통신공학과(학사)

1988년 한양대학교 전자통신공학과 (공학석사)

1992년 한양대학교 전자통신공학과 (공학박사)

1992년~현재 인천대학교 정보통신공학과 교수

관심분야 : 한국어정보처리, XML, 네트워크프로그래밍