XQL를 지원하는 XML 문서 저장 시스템

허명식*, 손기락

한국외국어대학교 컴퓨터공학과

Design and Implementation of an XML Document Storage System supporting XQL

Myeong-Sik Hur, Kirack Sohn

Dept. of Computer Science & Engineering, Hankuk University of Foreign Studies

요약

XML 문서와 같은 구조적 데이터는 관계형 데이터베이스에 저장하고 인터넷 응용 프로그램을 개발하는데 많은 이점을 가지고 있다. 또한 이러한 구조적 문서에 대한 질의 언어를 지원하는 것은 유용하다. 이에 본 논문에서는 XML 문서에 대한 질의 언어로 제안되어진 XQL을 효율적으로 지원할 수 있도록 XML 문서의 각 엘리먼트를 관계형 데이터베이스의 테이블로 매핑시키는 방법과 XQL을 SQL로 변환하는 방법을 제시하고 또한 XML 문서의 DTD를 통해 해당하는 문서의 테이블을 자동으로 생성하는 방법을 제시한다. 이를 통해 XQL을 효율적으로 지원하는 XML 문서 저장 시스템을 설계하고 구현한다.

1. 서론

최근 인터넷 사용과 정보의 양이 급증하면서 인터넷상의 정보를 보다 효과적으로 사용하고자 하는 연구가 활발히 진행되었다. 지금까지 인터넷상의 대부분의 정보는 HTML 문서로 구성되어 있었으나 HTML은 단지 문서의 재현을 위한 정보를 나타내는 하나의 정의된 DTD(Document Type Definition)를 사용하기 때문에 각 문서의 엘리먼트를 의미 있는 정보로 표현하는 기능이 부족하다. 이에 W3C(World Wide Web Consortium)에서는 차세대 웹 문서의 표준으로 XML (Extensible Markup Language)을 1998년에 지정하였다[1]. XML 문서는 구조적 데이터를 표현할 수 있으며 사용자가 정의한 DTD를 만족하는 트리 구조를 가지고 있다. 이러한 XML 문서는 구조적 문서를 표현하는데 유용하여 여러 시스템에 저장되어진 많은 다양한 정보를 표현하는데 사용되어진다. 이런 XML 문서를 데이터베이스를 이용하여 효과적으로 저장하고 관리할 수 있는 시스템이 요구되어지고 있으며 또한 많은 연구가 이루어지고 있다. XML 문서가 데이터베이스에 저장되면 문서는 다른 데이터와 같이 질의되어 질 수 있고 동시성, 복구 제어등의 많은 이점을 가지고 있다. 하지만 관계형 데이터베이스의 표준 질의 언어인 SQL은 정확한 데이터베이스 구조를 알아야 데이터를 질의 할 수 있다. 이에 W3C 에서는 XML 문서에 대한 범용적인 질의 언어로 XQL(XML Query Language)을 1998년에 제안하였다[2]. 이러한 XQL의 가장 큰 특징은 문서의 엘리먼트를 항해(navigate)하기 위해 경로를 사용하고 있다. 이를 통해 데이터베이스 구조에 대한 정확한 지식없이 데이터베이스에 저장된 XML 문서에 대해 질의가 가능하다.

이에 본 논문은 관계형 데이터베이스를 이용하여 XQL를 효과적으로 지원하기위한 XML 문서 저장시스템을 설계하고 구현한다.

2. 관련연구

XML 문서는 기본이 되는 엘리먼트를 기준으로 트리 구조로 구성되어 있다. 기존에 이러한 구조를 가지는 XML 문서를 관계형 데이터베이스에 저장하는 여러 방식이 연구 되어져왔다. 아래에서는 이와 같은 구조를 갖는 문서의 저장시스템에 대한 관련연구를 기술한다.

2.1 리스트형태를 이용한 방법

각 엘리먼트 노드들은 자신의 고유 번호를 갖고 있고 트리의 상위 노드는 하위 노드들의 고유번호를 리스트 형태로 갖고 있다. 이렇게 자신의 하위 노드에 대한 번호를 이용하여 트리 구조에서의 검색이나 서브 트리에 대한 구조적 검색을 할 수 있다[3]. 트리 구조에서 특정 노드를 찾기 위해서는 트리 항해를 하여 탐색해야 하므로, 많은 탐색 시간이 소요되며, 특정 노드의 하위 서브 트리에 대한 질의나 트리 구조에서의 특정 레벨에 관련된 질의를 하기 어렵거나 효율이 낮아 문서의 운용 데이터베이스로 사용하기에는 부적합하다.

2.2 경로 엘리먼트 ID(Path Element ID)를 이용한 방법

경로 ID는 자신이 자식 엘리먼트 중 몇 번째 엘리먼트인가를 표시하는 엘리먼트 ID를 자신의 조상에 대한 엘리먼트 ID로부터 계속 이어받아 구성된다[4]. 즉 자기 자신에 대한 접근 경로가 된다. 이와 같은 경로 엘리먼트 ID를 사용할 경우, XML 문서의 트리 구조상 어느 위치의 엘리먼트라도 쉽게 검색해 낼 수 있다. 그러나 반복되는 엘리먼트가 존재하면 경로 엘리먼트 ID가 무한이 증가되는 경우가 발생하므로 관계형 데이터베이스에 저장하는 것이 어렵다.

2.3 DFS Numbering을 이용한 방법

트리의 루트 노드로부터 DFS 방식으로 노드를 방문하여 처음 방문할 때의 순서와 자신의 자식 노드의 방문이 모두 끝난 뒤의 노드의 방문 순서를 1쌍으로 구성한다[3]. 이렇게 구성된 DFS Numbering 순서쌍은 문서의 트리 구조상에서 특정 노드에 속해있는 하위 트리에 대한 질의를 하나의 SQL 문장으로 처리할 수 있다. 하지만 몇 단계 위 또는 몇 단계 아래의 특정위치 엘리먼트에 대한 질의 처리는 DFS Numbering 하나만으로는 어렵고 부가적인 속성 필드를 필요로 하거나, 추가적인 검색 및 연산이 필요하다. 또한 문서를 데이터베이스에 추가나 삭제시 문서를 트리로 구성하여 DFS Numbering 한 뒤에 저장해야 하는 작업이 필요하다.

본 논문에서는 XQL의 효율적인 지원을 위해 DFS Numbering 등의 관련연구를 활용할 것이다.

3. XML 문서의 저장 시스템

3.1 엘리먼트의 위치 값 저장

구조적 문서인 XML 문서를 저장하기 위해 전체문서를 저장하고 엘리먼트는 전체문서에서의 시작위치와 종료위치를 저장하도록 한다. XML 문서의 특성상 자손 엘리먼트의 시작위치와 종료위치는 부모 엘리먼트의 시작위치보다 크고 종료위치보다 작은 위치정보를 가지게 되므로 DFS Numbering 과 동일한 효과를 가진다. 그리고 트리의 깊이에 대한 레벨 정보를 추가함으로써 구조적 문서에 대한 효율적인 질의가 가능하도록 하였다

3.2 스키마 생성

XML 은 문서의 구조를 정의하는 DTD 를 가지는데 이를 데이터베이스 스키마를 생성하는데 이용한다. <그림 1>은 DTD 의 예를 보여주는 것이다. 엘리먼트는 테이블로 속성들은 테이블의 필드로 구성하는 것을 기본 규칙으로 모든 엘리먼트를 테이블로 생성하지 않고 <ATTLIST>를 가지거나 PCDATA, CDATA 등의 타입을 가지는 엘리먼트만을 T_ELEMENT 테이블을 상속 받아 특성과 내용을 테이블의 필드로 추가하여 생성하게 된다. 이때 <ATTLIST>안에 속성의 이름은 테이블의 필드명으로 들어가게 된다. 하나의 데이터베이스에는 여러 DTD 의 문서를 저장할 수 있지만, 같은 이름의 엘리먼트를 가지는 것 중 <ATTLIST>의 내용이나 엘리먼트의 타입이 다른 엘리먼트를 가지는 다른 DTD 의 문서는 같은 데이터베이스에 저장되지 않는다.

```
<!DOCTYPE simple [
<!ELEMENT document (title,author+ summary*)>
<!ATTLIST document
        trackNum CDATA #REQUIRED>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT summary (#PCDATA)>
]>
```

<그림 1> DTD 의 예

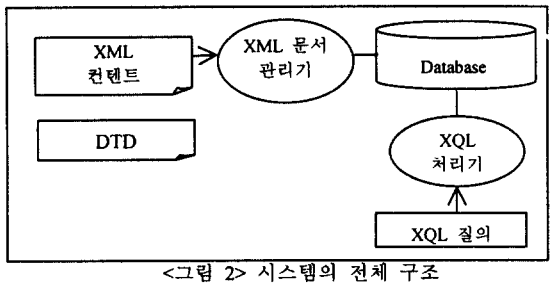
3.3 XML 데이터 타입 지원

XML 의 엘리먼트와 속성은 XML TYPE 으로 정의된다. 이를 관계형 데이터베이스에서 지원하기 위해서는 각각의 XML TYPE 을 built-in 데이터 타입으로 생성하여 제공한다. ID, IDREF, CDATA, PCDATA, ENTITY, NMTOKEN, NOTATION 등은 스트링 타입으로, IDREFS, ENTITES, NMTOKENS 등은 스트링 리스트 타입으로 생성한다. 단 ID, IDREF 를 관계형 데이터베이스에서 효과적으로 지원하기 위해서는 ID 를 프라이머리 키로 선언하고 IDREF 는 포린키로 선언해야 하지만 본 저장 시스템에서는 이를 효과적으로 지원하지 못한다.

4. 전체 시스템의 설계

4.1 저장 시스템의 전체 구조

본 저장 시스템은 <그림 2>와 같이 XML 문서를 입력 받아 문서의 DTD 에 따라 저장 스키마를 생성하고 XML 문서를 데이터베이스에 입력하는 XML 문서관리기와 XQL 의 질의를 입력 받아 XQL 를 파싱하고 해당하는 결과를 출력하는 XQL 처리기로 구성된다.



<그림 2> 시스템의 전체 구조

4.1.1 XML 문서관리기

XML 문서의 DTD 를 URL 또는 인스턴스로 입력받아 새로운 DTD 일 경우 3.2 절에서와 같이 스키마를 생성하고 3.1 절에서와 같이 XML 문서를 파싱하여 엘리먼트의 위치 값과 함께 데이터베이스에 저장한다.

4.1.2 XQL 처리기

사용자의 XQL 질의를 파싱하여 해당하는 SQL 과 함수를 호출하여 질의에 대한 결과값을 표시한다.

4.2 저장 스키마

다음 <그림 3>는 데이터베이스의 저장 테이블 구조를 나타내고 있다. T_ELEMENT 테이블은 엘리먼트를 저장하는 베이스 테이블로 엘리먼트의 태그 값과 레벨 정보 그리고 XML 문서의 위치정보를 저장하는 START_OFFSET, END_OFFSET 를 가지고 있으며 이 테이블을 통해 문서의 구조적 질의를 수행한다. 3.2 절에서의 정의에 따라 DERIVED ELEMENT 테이블은 ELEMENT 테이블을 상속 받아 동적으로 생성되며 엘리먼트의 태그 이름이 테이블의 이름이 되며 엘리먼트의 특성이 이 테이블의 특성 이름의 필드가 된다.

T_ELEMENT 테이블		
ID	INTEGER(PK)	엘리먼트의 ID
DOC ID	INTEGER(FK)	T_DOCUMENT 참조
TAG NAME	CHAR(50)	element tag 이름
LEVEL	INTEGER	TREE 의 위치
START_OFFSET	INTEGER	시작 위치
END_OFFSET	INTEGER	종료 위치

DERIVED ELEMENT 테이블		
ID	INTEGER(PK)	엘리먼트의 ID
CONTENT	XMI_TYPE	내용
ATTRI NAME	XMI_TYPE	Attribute

T_DOCUMENT 테이블		
ID	INTEGER(PK)	문서 ID
DATA	CLOB	문서 전체 저장

<그림 3> 데이터베이스의 저장 테이블 구조

XQL 를 SQL 로의 변환

5.1 XQL 의 구조

XQL 은 URI 디렉토리 항해 문법과 흡사하지만 파일 구조를 통해 항해를 정의하는 것 대신에 XML 트리의 엘리먼트를 통한 항해한다. 엘리먼트는 각각 '/' 와 '//'로 구분되어진다. 본 논문에서는 엘리먼트의 구분자와 엘리먼트를 하나의 쌍으로해서 처리하며 연속해서 나오는 쌍에 대해서는 앞에서 나온 결과 테이블을 이용하여 질의하게 된다.

5.2 XQL 에서 SQL 로 변환하는 기본적인 패턴

다음은 이러한 쌍을 SQL 로 변환할 때 사용되는 기본적인 패턴으로 질의에 따라 추가되거나 변경되어진다.

① SELECT * FROM t_element
WHERE tag_name = :tag

② SELECT * FROM [element_name]
WHERE [attr_name] = :value

③ SELECT * FROM t_element e
WHERE tag_name = :tag AND
EXISTS (SELECT * from element
WHERE start_offset > e.start_offset AND
end_offset < e.end_offset AND
doc_id = e.doc_id)

④ SELECT * FROM t_element, tmp
WHERE t_element.tag_name = :tag AND
t_element.start_offset > tmp.start_offset AND
t_element.end_offset < tmp_end_offset AND
t_element.doc_id = tmp.doc_id

⑤ SELECT * FROM [element_name], tmp
element_name.start_offset > tmp.start_offset AND
element_name.end_offset < tmp_end_offset AND
element_name.doc_id = tmp.doc_id AND
[attr_name] = :value

⑥ SELECT * FROM t_element e, tmp
WHERE t_element.tag_name = :attribute_name AND
t_element.start_offset > tmp.start_offset AND
t_element.end_offset < tmp.end_offset AND
t_element.doc_id = tmp.doc_id
EXISTS (SELECT * from element
WHERE tag_name = :tag_value
start_offset > e.start_offset AND
end_offset < e.end_offset AND
doc_id = e.doc_id) AND

⑦ 범위에 관련된 질의는 위의 패턴을 실행한 결과에서 범위에 해당하는 레코드를 새로운 테이블에 추가하여 결과를 반환한다.

5.3 XQL 를 SQL 로 변환하는 알고리즘

```
separator = gettoken()
element = gettoken()
filter = gettoken()

num_of_wildchar = 1

if (filter = null)
  if (separator = '/')
    if (element = '*')
      num_of_wildchar++
    else
      num_of_wildchar = 1
      패턴 ① 사용
      // where 절에  level  = 0 추가
  else
      패턴 ① 사용
else if (filter = 포함관련)
  패턴 ③ 사용
else if (filter = 범위관련)
  패턴 ⑦ 사용
else
  패턴 ② 사용

while (separator != EOQ)
  separator = gettoken()
  element = gettoken()
  filter = gettoken()

  if (filter = null) {
    if (element = '*')
      num_of_wildchar++
    else if (separator = '/') {
      패턴 ④ 사용
// where 절에  element.level =tmp.level + num_of_wildchar
      num_of_wildchar = 1
    }
    else {
      패턴 ④ 사용
// where 절에  element.level > tmp.level + num_of_wildchar++
      num_of_wildchar = 1
    }
  }
    else if (filter = 포함관련)
      패턴 ⑥ 사용
    else if (filter = 범위관련)
      패턴 ⑦ 사용
    else
      패턴 ⑤ 사용
  }
}
```

6. 결론

XQL 를 효율적으로 지원할 수 있도록 XML 문서의 내용 및 구조 정보를 관계형 데이타베이스의 테이블로 매핑시키는 방법과 XQL 를 SQL 로 변환하는 방법을 제시하였다. 또한 XML 문서의 DTD 를 통해 해당하는 문서의 테이블을 자동으로 생성하는 방법을 제시하였고 이를 통해 XQL 를 효율적으로 지원하는 XML 문서 저장 시스템을 설계하고 구현하였다.

7. 참고문헌

[1] Extensible Markup Language(XML) 1.0 ,"http://www.w3.org/TR/1998/REC-xml".
[2] XML Query Language(XQL) ,"http://www.w3.org-/TandS/QL/QL98/pp/xql.html".
[3] 이용석, 손기락 "XML 문서 저장 시스템 설계 및 구현", 정보과학회 할술발표 논문집(I),1998.
[4] 연제원, 조정수, 이강찬, 이규철, "XML 문서 구조검색을 위한 저장 시스템 설계",정보과학회 할술발표 논문집(B), 26 권 1 호, 1998.