XML 실체뷰를 이용한 XQL 질의 처리

김 수 희[†]·문 찬 호[†]·강 현 철[††]·서 상 구[†††]

요 약

XML 문서의 구조적인 특징을 고려한 XML 질의 처리에 관한 연구가 활발히 수행되고 있다. 이들 연구들은 하나의 XML 문서 또는 XML 저장소를 대상으로 효과적인 검색을 수행하기 위한 XML 질의어의 개발이나 확장, 그리고 질의 최적화를 중심으로 수행되고 있다. XML 문서 검색의 성능 향상을 위해서 XML 저장소에는 XML 문서들 외에 그들로부터 도출된 XML 뷰를 실체뷰로 저장해 둘 수 있는데, 이들 실체뷰는 XML 질의 처리에 이용될 수 있다. 본 논문에서는 XML 저장소에 저장된 하부 XML 문서들로부터 XQL을 통해 정의된 XML 실체뷰가 지원될 경우 이를 이용한 XQL 질의 처리에 대해 연구한다. 이를 위해 실체뷰를 지원하는 XML 저장소의 구조를 기술하고, XQL 질의가 제기되었을 때 관련 실체뷰를 이용한 처리가 가능한지를 판별한 후 그에 따라 원래의 XQL 질의를 해당 실체뷰에 대한 XQL 질의로 변환하는 알고리즘을 제시한다.

XQL Query Processing Using XML Materialized Views

SooHee Kim[†]·Chanho Moon[†]·Hyunchul Kang[††]·Sang-Koo Seo[†††]

ABSTRACT

Recently, much research on XML query processing considering the structural characteristics of XML documents is conducted, focusing primarily on the development and/or extension of XML query languages and on query optimization for effective retrieval of XML documents. In the XML repository, the XML views derived from the underlying XML documents could be stored materialized for effective retrieval of XML documents, and could be capitalized on for XML query processing. In this paper, assuming that the XML views defined in XQL are materialized in the XML repository, we investigate XQL query processing that capitalizes on them. We describe the storage structure of the XML repository which supports the materialized views, and propose an algorithm that determines whether the given XQL query can be processed with the relevant materialized view and accordingly transforms the original query into one against it.

키워드 : XML, XQL, 질의 변환(query rewriting), 실체뷰(materialized view)

1. 서 론

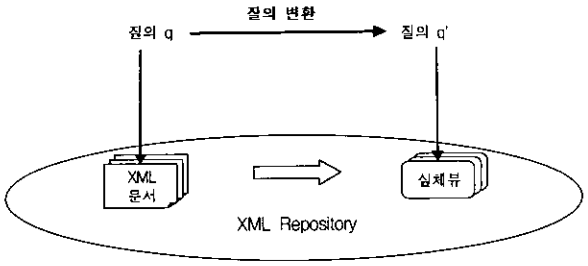
XML(eXtensible Markup Language)[1]이 차세대 웹 문서 표준으로 제안되면서, E-Commerce 등과 같은 응용에서 XML 문서를 활용하려는 노력이 활발히 이루어지고 있다[2]. 이를 위해 XML 문서의 구조정보 표현 방법 및 저장 시스템에 대한 연구가 수행되고 있으며[3-6], XQL(XML Query Language)[7, 8], XML-QL[9], XPath[10], Quilt[11], XQuery[12] 등과 같은 XML 표준 질의어의 제안도 이루어지고 있다.

뷰는 이질적인 데이타의 통합과 데이타 여과(filtering) 기능을 통해서 사용자가 요구하는 데이타를 제공한다. 인터넷의 확산으로 현재 웹에서는 다양한 데이타 저장소(repository)에 많은 양의 XML 문서들이 산재되어 있는데, 이들 XML 데이타에 대해서도 뷰는 유용한 개념이다[13]. 뷰는 질의 처리의 성능 향상을 위해 실체뷰(materialized view)[14]로 유지할 수 있는데, 뷰가 실체뷰로 제공되면 관련된 질의의 효율적인 처리에 이용될 수도 있다. 즉, 하부 데이타베이스에 대한 원래의 질의를 실체뷰에 대한 질의로 변환하여 수행함으로써 질의 응답 시간을 줄일 수 있다. XML 뷰 및 XML 실체뷰에 대해서는 최근에 연구가 시작되고 있다[13, 15, 16].

XQL[7, 8]은 1998년 W3C에 의해 제안된 범용적인 XML 질의어로서 현재 eXcelon[17] 등과 같은 대표적인 XML 저장 시스템들에 의해 널리 지원되고 있다. 본 논문에서는 XML 저장소 내에서 XQL로 정의된 XML 실체뷰가 지원될 경우, 이를 이용한 XQL 질의 처리에 관하여 연구하였다. (그림 1)에서 보는 것처럼 XML 저장소에 어떤 DTD를 참

※ 본 연구는 한국과학재단 목적기초연구(2001-1-303-001-3)지원으로 수행되었음.
† 준 회 원 : 중앙대학교 대학원 컴퓨터공학과
†† 정 회 원 : 중앙대학교 컴퓨터공학과 교수
††† 정 회 원 : 광운대학교 경영정보학과 교수
논문접수 : 2000년 12월 4일, 심사완료 : 2001년 9월 28일

조하는 여러 XML 문서들과 이들로부터 도출된 XML 실체뷰들이 있다고 하자. XML 저장소에 대한 XQL 질의 q가 주어졌을 때 그 결과를 관련된 실체뷰로부터도 얻을 수 있다면 질의 변환을 통해 q를 실체뷰에 대한 질의 q'으로 변환하여 수행하면 방대한 양의 XML 저장소 내 문서들에 대한 검색을 피할 수 있어 질의 응답 시간을 줄일 수 있다.



(그림 1) XML 실체뷰를 이용한 XQL 질의 처리

뷰 또는 실체뷰를 이용한 질의 처리는 주로 관계 데이타베이스 시스템에서 많은 연구가 수행되었다[18-25]. XML은 트리 구조로 문서의 구조 정보를 나타내며 XQL 질의는 그 구조에 따라 경로 표현식(path expression)으로 질의의 대상이 되는 엘리먼트를 지정한다. 따라서 XML 실체뷰를 이용한 XQL 질의 처리에는 기존의 레코드 기반 관계 데이타 모델 하에서 실체뷰를 이용한 질의 처리에서와는 다른 기법이 요구된다. 본 논문에서는 XQL 질의가 제기되었을 때 관련 XQL 질의로 정의된 실체뷰를 이용한 처리가 가능한지를 판별하고, 그에 따라 원래의 XQL 질의를 해당 실체뷰에 대한 XQL 질의로 변환하는 알고리즘을 제시한다.

본 논문의 구성은 다음과 같다. 2절에서는 관련 연구를 기술하며, 3절에서는 실체뷰를 이용한 질의 처리 모델을 기술한다. 4절에서는 XML 실체뷰의 정의와 실체뷰를 지원하는 XML 저장소의 구조에 대해 기술한다. 5절에서는 주어진 XQL 질의에 대하여 관련 실체뷰를 이용한 처리 가능 여부 판별 문제를 다루고, 6절에서는 XQL 질의 변환 알고리즘을 제시한다. 마지막으로 7절에서 결론을 맺는다.

2. 관련 연구

XML 실체뷰를 이용한 질의 처리에 관한 연구는 아직 없으며, 가장 관련된 연구로는 반구조적 데이타에 대한 뷰를 이용한 질의 변환(query rewriting)[26], 질의 응답[27], 그리고 질의 포함(query containment)[28] 문제에 관한 연구가 있다.

[26]에서는 반구조적 데이타를 포함하는 이질적인 데이타 소스의 통합을 위한 TSIMMIS 시스템[29]에서 OEM[30]으로 모델링한 반구조적 데이타에 대한 질의 및 뷰 표현을 위해 사용하는 질의어 TSL(Tree Specification Language)에 대한 질의 변환 문제를 연구하였다. 기존 관계 뷰를 이용한 질의 변환을 위해 사용되는 기법들인 containment mapping, chase, unification 등을 확장하여, 반구조적 데이타베이스 D에 대한 질의 q와 D에 대한 뷰의 집합 $V = \{V_1, \cdots, V_n\}$이 주어졌을 때, V내의 뷰를 최소한 하나 접근하여 q의 결과와 동일한 결과를 얻을 수 있는 새로운 질의 q'을 작성하는 알고리즘을 제시하였다.

[27]에서는 반구조적 데이타나 웹 데이타와 같이 그래프로 표현되는 데이타베이스에 대한 질의 체계의 하나인 정규 경로 질의(regular path query)로 질의와 뷰가 정의될 경우 뷰를 이용하여 질의에 응답하는 문제를 연구하였다. 정규 경로 질의는, 객체를 나타내는 노드들과 객체 간을 잇는 레이블이 있는 에지(labeled edge)들로 구성된 그래프로 모델링된 데이타베이스에 대해 제기되는 것으로, 에지의 레이블 값들로 구성된 알파벳을 갖는 정규 표현식(regular expression)으로 표현되며 해당 데이타베이스 내의 경로(즉, 에지 씨퀀스) 양 끝에 있는 두 객체 쌍의 집합을 구하는 것이다.

[28]에서는 반구조적 데이타를 레이블이 있는 방향성 그래프(labeled directed graph)로 모델링한 데이타베이스에 대한 질의를 표현할 수 있는 질의어인 StruQL0에 대하여 서로 다른 두 질의의 결과 간의 포함 관계를 규명하는 질의 포함(query containment) 문제를 연구하였다. 질의 포함 문제는 뷰를 이용한 질의 변환에 응용될 수 있다. StruQL0은 STRUDEL 웹 사이트 관리 시스템[31]의 질의어인 StruQL[32]의 부분 집합으로 그래프로 표현된 데이타에 대한 정규 표현식을 포함한 질의를 표현할 수 있다. 이 연구에서는 정규 표현식으로 표현된 반구조적 질의 간의 포함 관계를 다루고 있다.

이들 연구들은 모두 그래프 기반의 반구조적 데이타베이스로부터 생성된 뷰를 이용한 반구조적 질의 처리 문제를 다룬 것으로, 뷰의 소스로 XML 문서를 그리고 뷰의 정의 및 질의 표현을 위한 질의어로 XQL과 같은 XML 질의어를 대상으로 하지 않았다.

3. 실체뷰를 이용한 질의 처리 모델

본 절에서는 실체뷰에 대해 설명하고, 질의 결과와 실체뷰 간의 포함 관계에 따라 실체뷰를 이용한 질의 처리 유형에 대해 기술한다.

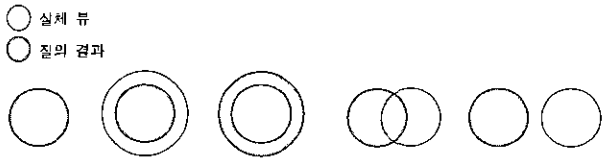
3.1 실체뷰

실체뷰는 단순히 뷰 정의만을 시스템 목록에 저장하는 것이 아니라 뷰 내용을 직접 저장하는 기법에 의해 제공되는 뷰를 말한다[14]. 따라서 뷰를 도출한 하부 데이타의 변경 시 그에 해당하는 뷰 내용도 변경하여 일관성을 유지해야 하는 뷰 갱신(view refresh)을 필요로 한다. 실체뷰 갱신에는 하부 데이타의 변경시 뷰의 내용을 모두 버리고 새로

뷰를 재생성(recomputation)하는 방법과 하부 데이타의 변경 시 실체뷰의 내용중 해당 변경으로 인해 영향을 받는 부분만 반영하는 점진적 갱신(incremental refresh) 방법이 있다. 이들 기법에 대한 자세한 사항은 본 논문의 범위 밖이므로 이하 더 다루지 않으며 XML 실체뷰의 갱신 기법[16]과 반구조적 데이타에 대한 실체뷰의 갱신 기법[33-35]을 참조한다.

3.2 실체뷰를 이용한 질의 처리 유형

질의 처리에 실체뷰를 이용하기 위해서는 주어진 질의 결과의 전부 또는 일부가 실체뷰로부터 도출 가능해야 한다. 질의 결과와 실체뷰 내용 간의 포함 관계는 (그림 2)와 같이 다섯 가지로 분류할 수 있다[36].

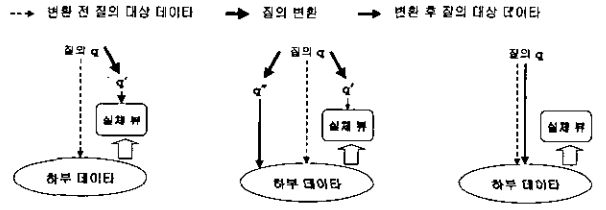


(a) 일치 (b) 포함 (c) 완전중복 (d) 부분중복 (e) 분리

(그림 2) 실체뷰와 질의 결과 간의 포함관계[36]

(그림 2)에서 (a)는 실체뷰와 질의 결과가 일치하는 경우를 나타낸 것이며, (b)는 실체뷰가 질의 결과를 포함하는 경우이다. (c)는 (b)와 반대로 실체뷰가 질의 결과에 완전히 포함됨을 나타내며, (d)는 실체뷰와 질의 결과가 일부만 겹침을 나타낸다. (e)는 실체뷰와 질의 결과 사이에 아무런 포함 관계가 성립하지 않는 경우를 나타낸다.

(그림 2)에 나타낸 실체뷰와 질의 결과 간의 포함 관계에 따라 질의 처리에 실체뷰를 이용하는 방법은 (그림 3)과 같이 크게 세 가지 유형으로 나누어 볼 수 있다. 이때 실체뷰로부터 질의의 결과를 얻기 위해서는 주어진 질의를 뷰에 대한 질의로 변환하는 질의 변환이 필요하다.



(a) 유형 MV (b) 유형 UD+MV (c) 유형 UD

(그림 3) 실체뷰를 이용한 질의 처리의 유형

(그림 3)에서 (a)의 유형 MV(Materialized View Only)는 (그림 2)의 (a)일치나 (b)포함에 해당하는 경우의 질의 처리 방법으로 주어진 질의 q를 실체뷰에 맞는 질의 q'으로 변환하여 실체뷰로부터 원하는 결과를 모두 얻을 수 있음을 나타낸다. (b)의 유형 UD+MV(Both Underlying Documents and Materialized View)는 (그림 2)의 (c)완전 중복이나 (d)부분 중복에 해당하는 경우의 질의 처리 방법으로 질의 결과의 일부는 실체뷰에 존재하지만 일부는 실체뷰에 존재하지 않는 경우이다. 따라서 유형 UD+MV는 주어진 질의 q를 실체뷰에 대한 질의 q'과 하부 데이타에 대한 질의 q'' 으로 변환하여 각각으로부터 결과를 얻는다. (c)의 유형 UD(Underlying Documents Only)는 주어진 질의의 결과를 모두 하부 데이타로부터 얻는 경우이다. (그림 2)의 (e)분리의 경우 이러한 질의 처리 유형을 사용하여야 하며, (b)포함의 경우에도 질의 처리 비용을 고려하여 유형 UD와 같은 질의 처리가 가능하다.

4. XML 실체뷰를 지원하는 저장소 구조

본 절에서는 XQL 질의를 통한 XML 실체뷰의 정의, 실체뷰를 지원하는 XML 저장소의 구조, 그리고 그 구조를 이용한 XQL 질의 처리 과정에 대해 기술한다.

4.1 XML 실체뷰

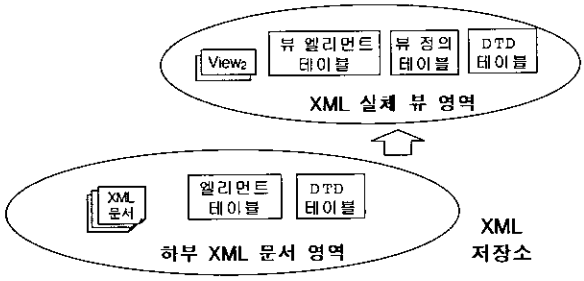
XQL은 화일의 디렉토리 경로를 나타내는 것과 같은 경로 표현식을 이용하여 질의의 대상이 되는 XML 문서의 **목적 엘리먼트(target element)**를 표시하고 그 외 조건을 표시하는 필터 연산자([ ]), 비교 연산자(=, !=, lt, gt, …) 등을 이용하여 질의를 표현한다[7]. 예를 들어, XQL 질의 bookstore/magazine은 bookstore 엘리먼트의 자식 엘리먼트 중 magazine 엘리먼트를 루트로 하는 서브 트리들을 모두 검색한다. XQL 질의는 하나의 XML 문서에서부터 XML 저장소 내의 모든 문서에 대해 수행될 수 있다. [7]의 XQL 명세에서는 XQL 질의의 결과 형태를 특정 형식으로 한정하고 있지 않으며, 노드, 노드 리스트, XML 문서, 배열, 또는 기타 구조를 그 대상으로 열거하고 있다. 본 논문에서는, XQL 질의의 결과로 하나의 루트(/) 엘리먼트를 가진 잘 형성된(well-formed) XML 문서를 반환한다고 가정하였다. bookstore/magazine의 경우, 복수개의 magazine 서브 트리가 검색되면 이들 magazine 엘리먼트들을 모두 자식 엘리먼트로 하는 루트 엘리먼트를 갖는 XML 문서가 반환된다.

본 논문에서 XML 뷰는 XQL 질의 표현식으로 정의되어 이름이 주어진다고 가정하였다. 즉, XML 실체뷰의 내용은 XQL 질의의 결과로 검색된 서브 트리들을 루트 엘리먼트로 묶은 것이며, 뷰의 소스인 하부 XML 문서와 마찬가지로 XML 문서로서 저장된다. 사용자 또는 응용 프로그램은 반복적으로 자주 제기되는 XQL 질의 결과를 뷰로 정의하여 실체뷰로 유지할 수 있다. 현재 [7]의 XQL 명세는 뷰 정의를 위한 구문을 정의하지 않고 있는데 뷰의 이름을 부여하기 위한 구문 확장이 필요하다. 단, 뷰에 부여된 이름의 역할이 적절히 규정되어 정의된 뷰가 다른 XQL 질의로부터 참조 가능해야 하는데, 위에서 언급한대로 XQL 질의 결과가 하나의 XML 문서로 반환될 경우에는 뷰의 이름이

문서의 이름이 된다.

4.2 XML 저장소 구조

XML 저장소 구조는 (그림 4)와 같이 **하부 XML 문서 영역**(이하, 문서 영역)과 **XML 실체뷰 영역**(이하, 뷰 영역)으로 구성된다. XML 문서의 저장 방법은 크게 분할 방법과 비분할 방법으로 나눌 수 있다[6]. 분할 방법은 XML 문서를 엘리먼트 단위로 분할해서 분할하는 기법이고, 비분할 방식은 문서 전체를 BLOB(Binary Large OBject) 혹은 CLOB (Character Large OBject)의 형태로 저장하는 기법이다. 본 논문에서는 XML 문서를 분할 방법에 의해 현재 가장 널리 사용되고 있는 관계 DBMS에 저장하는 경우를 연구 대상으로 하였다.



(그림 4) XML 저장소의 구성

문서 영역은 XML 문서들을 엘리먼트 단위로 분할하여 저장한 엘리먼트 테이블과 DTD 정보를 저장한 DTD 테이블로 구성되며, 뷰 영역은 실체뷰의 내용을 엘리먼트 단위로 분할하여 저장한 뷰 엘리먼트 테이블, 뷰 정의를 저장하는 뷰 정의 테이블, DTD 테이블로 구성된다. 다음은 이들에 대한 설명이다.

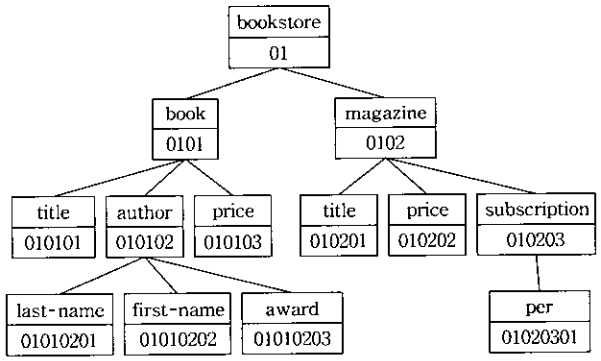
```
<!ELEMENT bookstore ( book*, magazine * ) >

<!ELEMENT book ( title, author*, price ) >
<!ELEMENT author ( first-name, last-name, award ) >
<!ELEMENT first-name ( #PCDATA ) >
<!ELEMENT last-name ( #PCDATA ) >
<!ELEMENT award ( #PCDATA ) >
<!ELEMENT magazine ( title, price, subscription ) >
<!ELEMENT title ( #PCDATA ) >
<!ELEMENT price ( #PCDATA ) >
<!ELEMENT subscription ( per ) >
<!ELEMENT per ( #PCDATA ) >
```

```
<bookstore>
    <book>
        <title> Seven </title>
        <author>
            <first-name> Joe </first-name>
            <last-name> Bob </last-name>
            <award> Review </award>
        </author>
        <price> 12 </price>
    </book>
    <magazine>
        <title> Tracking Trenton </title>
        <price> 2.50 </price>
        <subscription>
            <per> year </per>
        </subscription>
    </magazine>
    ……
</bookstore>
```

(그림 5) DTD와 XML 문서의 예

- **DTD 테이블** : DTD를 엘리먼트 단위로 분할하여 저장한 테이블이다. DTD 테이블의 레코드 구조는, DTD 식별자(DTDID), DTD의 트리 구조에서 해당 엘리먼트의 위치를 표시하는 DTD의 엘리먼트 식별자(EID), 그리고 엘리먼트의 이름(Ename) 컬럼으로 구성된다. 엘리먼트 식별자는 [4]에서 기술한 경로 엘리먼트 식별자를 사용하였다. 경로 엘리먼트 식별자는 엘리먼트의 식별자를 할당함에 있어 자신이 부모 엘리먼트의 몇 번째 자식 엘리먼트인가를 두자리 수로 나타내고 이를 부모 엘리먼트의 식별자 끝에 추가한 것이다. 예를 들어, 조상 엘리먼트의 식별자가 '0101'인 경우, 첫 번째와 두 번째 자식 엘리먼트의 식별자는 각각 '010101'과 '010102'가 된다. (그림 5)는 DTD와 이를 따르는 XML 문서의 예를 나타낸 것이며, (그림 6)은 (그림 5)의 DTD의 트리 구조와 그에 따른 DTD 테이블의 내용을 나타낸 것이다.



DTDID	EID	Ename
1	01	bookstror
1	0101	book
1	010101	title
1	010102	author
1	010203	price
1	01010201	first-name
1	01010202	last-name
1	01010203	award
1	0102	magazine
1	010201	title
1	010202	price
1	010203	subscription
1	01020301	per

(그림 6) DTD의 트리 구조 및 DTD 테이블

- **엘리먼트 테이블** : XML 저장소의 문서 영역에 존재하는 모든 XML 문서는 엘리먼트 단위로 분할하여 엘리먼트 테이블에 저장된다. 엘리먼트 테이블의 레코드 구조는, 해당 엘리먼트의 XML 문서 식별자(DID), 해당 엘리먼트의 DTD 내의 위치를 나타내는 DTD 엘리먼트 식별자(DTDEID), 해당 XML 문서의 트리 구조에서의 위치를 표시하는 엘리먼트 식별자(EID), 엘리먼트 이름(Ename), 그리고 해당 엘리먼트가 단말 노드일 경우의 내용(contents) 컬럼으로 구성된다. (그림 7)은 (그림 5)의 XML 문서가 엘리먼트 테이블에 저장된 예를 나타낸 것이다.

DID	DTDEID	EID	Ename	Contents
1	01	01	bookstror	-
1	0101	0101	book	-
1	010101	010101	title	Seven
1	010102	010102	author	-
1	010103	010203	price	12
1	01010201	01010201	first-name	Joe
1	01010202	01010202	last-name	Bob
1	01010203	01010203	award	Review
1	0102	0102	magazine	-
1	010201	010201	title	Tracking Trenton
1	010202	010202	price	2.5
1	010203	010203	subscription	-
1	01020301	01020301	per	year
...	...	...	...	...

(그림 7) 엘리먼트 테이블

- **뷰 정의 테이블** : XML 실체뷰의 정의를 저장하는 테이블이다. 뷰 정의 테이블의 레코드 구조는, 뷰의 식별자(ViewDID), 뷰 정의를 나타내는 XQL 질의 표현식(ViewDef), 그리고 뷰를 도출한 하부 XML 문서들이 만족하는 DTD 식별자(DTDID) 컬럼으로 구성된다.

- **뷰 엘리먼트 테이블** : XML 실체뷰의 내용은 XML 문서 형태를 취하고 있다. 이를 엘리먼트 단위로 분할하여 저장한 테이블이다. 뷰 엘리먼트 테이블의 레코드 구조는, 해당 엘리먼트가 속한 하부 XML 문서와 뷰의 식별자(BaseDID와 ViewDID), 해당 엘리먼트의 DTD 내의 위치를 나타내는 DTD 엘리먼트 식별자(DTDEID), 실체뷰의 XML 문서 트리 구조에서 해당 엘리먼트의 위치를 표시하기 위한 엘리먼트 식별자(EID), 엘리먼트 이름(Ename), 그리고 해당 엘리먼트가 단말 노드일 경우의 내용(Contents) 컬럼으로 구성된다. (그림 8)은 뷰 정의 테이블과 뷰 엘리먼트 테이블의 예를 보여준다.

ViewDID	ViewDef	DTDID
1	bookstore/book/author[first name = "Joe"]	1
2	bookstore/magazine/subscription	1
...	...	...

(a) 뷰 정의 테이블

BaseDID	ViewDID	DTDEID	EID	Ename	Contents
1	1	010102	010102	author	-
1	1	01010201	01010201	first-name	Joe
1	1	01010202	01010202	last-name	Bob
1	1	01010203	01010203	award	Review
1	2	010203	010203	subscription	-
1	2	01020301	01020301	per	year
...	...	...	...	...	...

(b) 뷰 엘리먼트 테이블

(그림 8) 뷰 정의 테이블과 뷰 엘리먼트 테이블

4.3 XQL 질의 처리

4.2절에서 제시한 실체뷰를 지원하는 XML 저장소의 저장 구조하에서 주어진 XQL 질의의 처리 과정은 다음과 같다. 예를 들어, bookstore/book/author 라는 질의가 주어지면 DTD 테이블에서 목적 엘리먼트인 author의 EID값 '010102'를 검색한 후, 엘리먼트 테이블에서 DID값 별로 DTDEID의 값이 '010102'인 것과 '010102'의 자손 엘리먼트의 식별자 값에 해당되는 레코드를 모두 검색하여 해당 문서에 대한 질의 처리를 수행한다. 이와 같이 문서 별로 검색된 서브 트리들은 모두 하나의 루트(/) 엘리먼트의 서브 트리가 되어 잘 형성된 XML 문서를 질의 결과로 구성한다.

한편, 질의 처리에 실체뷰가 이용될 경우에는 질의 변환에 의해 생성된 질의가 해당 실체뷰를 대상으로 수행된다. 이때는 엘리먼트 테이블 대신 뷰 엘리먼트 테이블을 검색한다. 이 두 테이블간의 차이는 뷰 엘리먼트 테이블에 BaseDID 컬럼이 추가되었고, 엘리먼트 테이블의 DID 컬럼을 뷰 엘리먼트 테이블에서는 ViewDID 컬럼이 대체하고 있다는 점뿐이다. 따라서, 질의가 요구하는 문서별 서브 트리는 해당 실체뷰를 식별하는 ViewDID 값을 갖는 레코드들에 대해서 검색되며, 그 방법은 위에서 기술한 엘리먼트 테이블에서 DID 값 별로 서브트리를 구성하는 레코드들을 검색하는 과정과 동일하다.

5. XML 실체뷰를 이용한 XQL 질의 처리 유형

본 절에서는 주어진 XQL 질의에 대하여 관련 실체뷰를 이용한 처리 가능 여부 판별 문제를 다룬다. 먼저 몇가지 예를 통하여 XQL 질의가 주어졌을 때 이를 그와 연관된 실체뷰를 이용하여 처리 가능한 경우와 불가능한 경우를 알아본 후, 이를 판별하기 위한 방법을 제시한다. 또한 판별 결과에 따라 (그림 3)의 세가지 질의 처리 유형 MV, UD+MV, UD 중 어느 유형에 해당하는지를 결정하는 방법을 제시한다.

5.1 예

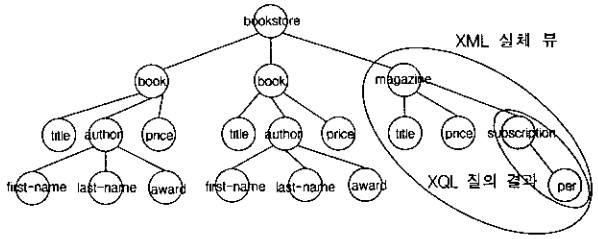
본 논문에서는 XQL로 실체뷰를 정의하고 이를 XQL 질의 처리에 이용함에 있어 다음을 가정하였다.

- XML 실체뷰는 XML 저장소 내 단일 DTD를 참조하는 XML 문서들로부터 도출된 것이다.
- XQL 질의는 XML 저장소 내 문서들 중 단일 DTD를 참조하는 문서들에 대해 제기된다.
- XQL 질의에서 필터 연산자로 조건을 명시할 경우 conjunctive normal form을 취한다.
- XQL 질의 중 엘리먼트의 속성을 묻는 질의 등 기타 기능은 다루지 않는다.

이와 같은 가정에 따라 XQL 질의는 엘리먼트에 대한 아무런 조건 명시 없이 경로만으로 구성된 것과 필터 연산자를 통해 조건을 명시한 것으로 대별할 수 있다. (그림 9)는 질의와 실체뷰 둘 다에 조건이 명시되지 않은 경우의 예를 나타낸 것이다. (그림 9) (a)는 bookstore/magazine으로 정의된 실체뷰 v가 있다고 할 때, bookstore/magazine/subscription이라는 질의 q가 제기된 경우이다. v는 bookstore 엘리먼트의 자식 엘리먼트 중 magazine 엘리먼트를 모두 검색하는 것으로 그 결과에 해당되는 서브 트리를 그림에 표시하였다. 한편, q는 bookstore 엘리먼트의 자식 엘리먼트인 magazine 엘리먼트의 자식 엘리먼트 중 subscription 엘리먼트를 모두 검색하는 것으로 그 결과에 해당되는 서브 트리를 그림에 음영으로 표시하였다. q의 결과가 v의 내용에 포함되므로 이는 (그림 2)의 (b) 포함의 경우에 해당되며 따라서 (그림 3)의 질의 처리 유형 MV로 처리 가능하다. 한편, (그림 9) (b)는 bookstore/book/author로 정의된 실체뷰 v가 있다고 할 때, bookstore/book이라는 질의 q가 제기된 경우이다. q의 결과가 v의 내용을 완전히 포함하고 있으므로 (그림 2)의 (c) 완전 중복의 경우에 해당되며 따라서 (그림 3)의 질의 처리 유형 UD+MV로 처리 가능하다.

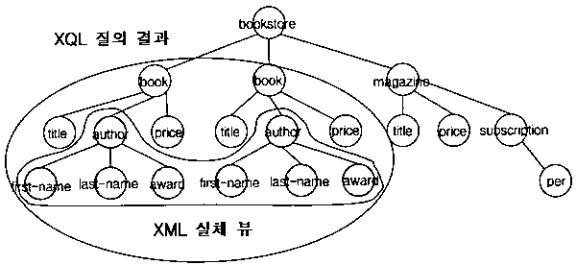
(그림 10)은 질의와 실체뷰 둘 다에 조건이 명시된 경우의 예를 나타낸 것이다. (그림 10) (a)는 bookstore/book[author/first-name = "Jane"]으로 정의된 실체뷰 v가 있다고 할 때, bookstore/book/author[first-name = "Joe"]라는 질의 q가 제기된 경우이다. v는 bookstore 엘리먼트의 자식 엘리먼트 중 book 엘리먼트를 모두 검색하되 자식인 author 엘리먼트의 first-name 엘리먼트 값이 "Jane"인 것만 검색하는 것이다. 한편, q는 bookstore 엘리먼트의 자식인 book 엘리먼트의 자식 중 author 엘리먼트를 모두 검색하되 first-name 엘리먼트 값이 "Joe"인 것만 검색하는 것이다. q의 결과는 v의 내용과 별개의 것이므로 (그림 2)의 (e) 분리의 경우에 해당되며 따라서 (그림 3)의 질의 처리 유형 UD로 처리해야 한다.

한편, (그림 10) (b)는 bookstore/book[author/first-name = "Jane"]로 정의된 실체뷰 v가 있다고 할 때, bookstore/book/ author[first-name = "Jane" and last-name = "Poster"]라는 질의 q가 제기된 경우이다. 필터 내의 and는 Boolean AND이다. 따라서 q의 결과가 v의 내용에 포함되므로 이는 (그림 2)의 (b) 포함의 경우에 해당되며 따라서 (그림 3)의 질의 처리 유형 MV로 처리 가능하다.



- XML 실체 뷰 정의 : bookstore/magazine
- XQL 질의 : bookstore/magazine/subscription

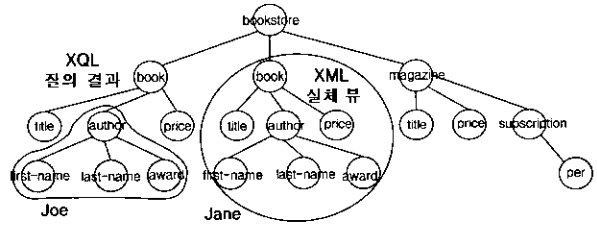
(a) 포함



- XML 실체 뷰 정의 : bookstore/book/author
- XQL 질의 : bookstore/book

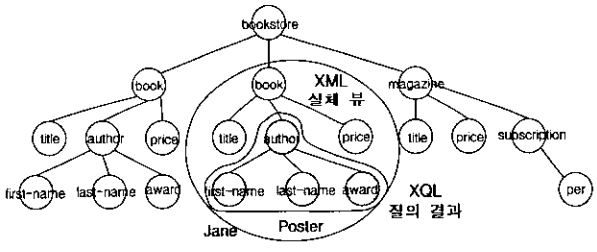
(b) 완전 중복

(그림 9) 조건이 명시되지 않은 실체뷰와 질의 결과 간의 포함 관계 예



- XML 실체 뷰 정의 : bookstore/book[author/first-name = "Jane"]
- XQL 질의 : bookstore/book/author[first-name = "Joe"]

(a) 분리

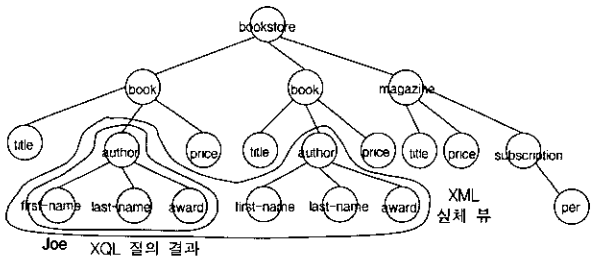


- XML 실체 뷰 정의 : bookstore/book[author/first-name = "Jane"]
- XQL 질의 : bookstore/book/author[first-name = "Jane" and last-name = "Poster"]

(b) 포함

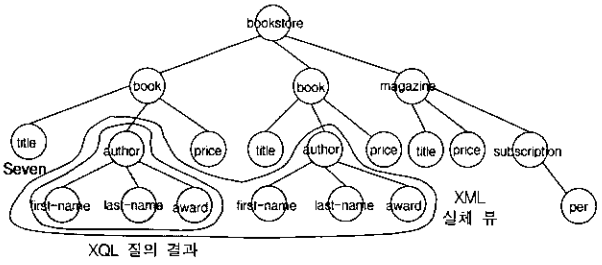
(그림 10) 조건이 명시된 실체뷰와 질의 결과 간의 포함 관계 예

(그림 11)은 실체뷰는 조건이 명시되지 않은 것인데 질의는 조건이 명시된 경우의 예를 나타낸 것이다. (그림 11) (a)는 bookstore/book/author로 정의된 실체뷰 v가 있다고 할 때, bookstore/book/author[first-name = "Joe"]라는 질의 q가 제기된 경우이다. q의 결과가 v의 내용에 포함되므로 이는 (그림 2)의 (b) 포함의 경우에 해당되며 따라서 (그림 3)의 질의 처리 유형 MV로 처리 가능하다.



- XML 실체 뷰 정의 : bookstore/book/author
- XQL 질의 : bookstore/book/author[first-name = "Joe"]

(a) 포함



- XML 실체 뷰 정의 : bookstore/book/author
- XQL 질의 : bookstore/book[title = "Seven"]/author

(b) 포함

(그림 11) 조건이 명시되지 않은 실체뷰와 조건이 명시된 질의 결과 간의 포함 관계 예

한편, (그림 11) (b)는 bookstore/book/author로 정의된 실체뷰 v가 있다고 할 때, bookstore/book[title = "Seven"]/author라는 질의 q가 제기된 경우이다. q는 bookstore 엘리먼트의 자식인 book 엘리먼트의 자식 중 author 엘리먼트를 모두 검색하되 book 엘리먼트의 자식 엘리먼트 인 title 값이 "Seven"인 것만 검색하는 것이다. 그림에서 볼 수 있듯이 q의 결과는 v의 내용에 포함되어 (그림 2)의 (b) 포함의 경우에 해당된다. 그러나 q를 처리할 때 접근해야 하는 title 엘리먼트가 v의 내용에 포함되어 있지 않으므로 v를 이용한 q의 처리는 불가능하다. 따라서 (그림 3)의 질의 처리 유형 UD로 처리해야 한다.

5.2 질의 처리 유형의 결정

질의 q와 실체뷰 v가 주어졌을 때, v를 이용한 q의 처리 가능 여부를 판별하고 그에 따라 해당 질의 처리 유형을 결정하는 것은, q와 v의 경로 간의 비교 그리고 q와 v의 조건 간의 비교를 통해 가능하다. 예를들어 (그림 9) (a)의 경우, q의 경로는 bookstore/magazine/subscription이고 v의 경로는 bookstore/magazine인데, 한 문서 내에서 전자는 후자가 지정하는 서브 트리의 서브 트리를 지정하는 것이다. 따라서 (그림 2)의 (b)포함 관계에 해당됨을 알 수 있는 것이다.

q(v)가 명시한 질의의 목적 엘리먼트까지의 경로를 **q.path (v.path)**로 나타내고 이를 q(v)의 **목적 엘리먼트 경로**라고 부르자. 이는, 만약 q 또는 v가 목적 엘리먼트 또는 그 자손 엘리먼트에 대한 조건을 명시하고 있을 경우에는 그 조건을 제외한 경로를 의미한다. 예를 들어 (그림 9)~(그림 11)의 경우, q.path와 v.path는 각각 <표 1>과 같다.

두 경로 p1, p2 간의 연산 $<_p$를 다음과 같이 정의하자 : p1이 p2의 prefix와 일치할 경우 p1 $<_p$ p2이 성립한다. 예를 들어 (그림 10) (a)의 경우, q.path = bookstore/book/author이고 v.path = bookstore/book이므로, v.path $<_p$ q.path이다. q와 v는 모두 조건을 명시하고 있지 않으므로 v.path $<_p$ q.path가 성립되면, MV 유형에 의해 v를 이용한 q의 처리가 가능함을 알 수 있다. (그림 9) (b)의 경우에는, q.path = bookstore/book이고 v.path = bookstore/book/author 이므로, q.path $<_p$ v.path이다. q와 v는 모두 조건을 명시하고 있지 않으므로 q.path $<_p$ v.path가 성립되면, UD + MV 유형에 의해 v를 이용한 q의 처리가 가능함을 알 수 있다.

한편, q와 v의 목적 엘리먼트 또는 그 자손 엘리먼트에 대한 조건 명시가 있을 경우에는 경로 비교 외에 이들 조건도 서로 비교해야 한다. q(v)에서 목적 엘리먼트 또는 그

〈표 1〉 목적 엘리먼트 경로의 예

그 림	q	q.path	v	v.path
9a	bookstore/magazine/subscription	bookstore/magazine/subscription	bookstore/magazine	bookstore/magazine
9b	bookstore/book	bookstore/book	bookstore/book/author	bookstore/book/author
10a	bookstore/book/author[first-name = "Joe"]	bookstore/book/author	bookstore/book[author/first-name = "Jane"]	bookstore/book
10b	bookstore/book/author[first-name = "Jane" and last-name="Poster"]	bookstore/book/author	bookstore/book[author/first-name = "Jane"]	bookstore/book
11a	bookstore/book/author[first-name = "Joe"]	bookstore/book/author	bookstore/book/author	bookstore/book/author
11b	bookstore/book[title = "Seven"]/author	bookstore/book[title = "Seven"]/author	bookstore/book/author	bookstore/book/author

〈표 2〉 목적 엘리먼트 조건의 예

그 림	q	q.con	v	v.con
9a	bookstore/magazine/subscription	{ }	bookstore/magazine	{ }
9b	bookstore/book	{ }	bookstore/book/author	{ }
10a	bookstore/book/author[first-name = "Joe"]	{bookstore/book/author/first-name = "Joe" }	bookstore/book[author/first-name = "Jane"]	{bookstore/book/author/first-name = "Jane" }
10b	bookstore/book/author[first-name = "Jane" and last-name = "Poster"]	{bookstore/book/author/first-name = "Jane", bookstore/book/author/last-name = "Poster" }	bookstore/book[author/first-name = "Jane"]	{bookstore/book/author/first-name = "Jane" }
11a	bookstore/book/author[first-name = "Joe"]	{bookstore/book/author/first-name = "Joe" }	bookstore/book/author	{ }
11b	bookstore/book[title = "Seven"]/author	{ }	bookstore/book/author	{ }

자손 엘리먼트에 대한 조건이 필터 연산자를 통해 conjunctive normal form으로 [p_1 and p_2 and … p_n]과 같이 주어졌을 때 이를 집합 {P_1, P_2, …, P_n}으로 표현한 것을 **q.con(v.con)**으로 나타내고 이를 **목적 엘리먼트 조건**이라 부르자. 이때, P_i는 p_i에서 조건이 주어진 대상 엘리먼트를 해당 문서 내의 완전 경로(full path)로 바꾸어 표현한 것이다(i = 1, …, n). 예를 들어 (그림 10) (b)의 경우, q의 조건은 [first-name = "Jane" and last-name = "Poster"]이고 v의 조건은 [first-name = "Jane"]이다. 따라서, q.con = {bookstore/book/author/first-name = "Jane", bookstore/book/author/ last-name = "Poster"}이고, v.con = {bookstore/book/author/first-name = "Jane"}이다. 만약 q 또는 v에 조건이 명시되지 않았다면 q.con와 v.con는 공집합이 된다. 예를 들어 (그림 9)~(그림 11)의 경우, q.con과 v.con은 각각 <표 2>와 같다. q.con와 v.con는 집합이므로 둘 간의 비교는 집합 간의 비교로 표현할 수 있다. 예를 들어 (그림 10) (b)의 경우, v.con ⊂ q.con가 된다. (그림 9)의 경우처럼 q와 v에 모두 조건이 명시되지 않으면 q.con와 v.con이 모두 공집합이므로 v.con = q.con이다.

(그림 9)의 경우처럼 v.con = q.con이면 q.path와 v.path의 비교를 통해 질의 처리 유형을 결정할 수 있다. 반면, (그림 10) (b)의 경우처럼 조건 명시가 있는 경우에는 q.path와 v.path의 비교와 더불어 q.con와 v.con의 비교를 통해 질의 처리 유형이 결정된다. (그림 10) (b)의 경우에는, q.path $<_p$ q.path이고 v.con ⊂ q.con이므로 MV 유형에 의해 v를 이용한 q의 처리가 가능함을 알 수 있다. 이상과 같이 q.path와 v.path의 비교 그리고 q.con과 v.con의 비교를 통해 질의 처리 유형을 결정하는 방법을 정리하면 <표 3>과 같다.

〈표 3〉 XML 실체뷰를 이용한 XQL 질의 처리의 유형

조건 \ 경로		v.path = q.path	v.path $<_p$ q.path	q.path $<_p$ v.path	otherwise
조건이 명시되지 않은 경우	-	MV	MV	UD + MV	UD
조건이 명시된 경우	v.con = q.con	MV	MV	UD + MV	UD
	v.con ⊂ q.con	MV	MV	UD + MV	UD
	v.con ⊃ q.con	UD + MV	UD + MV	UD	UD
	otherwise	UD	UD	UD	UD

6. XQL 질의 변환 알고리즘

5절에서 기술한 바와 같이 XQL 질의 q와 그에 연관된 XML 실체뷰 v가 주어지면 v를 이용한 q의 처리 유형을 결정할 수 있다. 본 절에서는 MV 질의 처리 유형을 위한 질의 변환을 수행하는 알고리즘 rewrite_XQL_query()를 제안한다.

rewrite_XQL_query()는, 각각 질의와 실체뷰를 정의하는 XQL 표현식 q와 v를 입력받아 MV 유형의 질의 처리가 가능할 경우 v에 대한 XQL 질의 표현식 q′을 반환하고, 불가능할 경우에는 NULL을 반환한다. rewrite_XQL_query()는 다음과 같이 작동한다. 먼저, DTD 테이블을 참조하여 q와 v로부터 q.path, q.con, v.path, v.con을 구한 후, 이들 간의 비교를 통해 <표 3>에서 정리한대로 MV 유형으로 v를 이용한 q의 처리가 가능한지를 판단한다. 불가능할 경우에는 NULL을 반환하고 가능할 경우에는 q와 v로부터 q′을 생성한다. 생성된 q′에 필터 연산자를 통한 조건이 명시되어 있을 경우, 조건의 대상 엘리먼트를 나타내는 완전 경로(이들은 q.con과 v.con으로부터 도출된 것임)를 q′의 목적 엘리먼트 경로 이후의 자손 경로로 변환한다(refine_query()). 알고리즘 rewrite_XQL_query()를 기술하면 (그림 12)와 같고 알고리즘에서 사용된 표기(notation)는 <표 4>와 같다. <표 4>의 표기 중 q.path와 q.con은 5절에서 정의하였고, q.elem과 q.level은 다음과 같이 정의된다.

- **q.elem(v.elem)** : 경로 q의 목적 엘리먼트
- **q.level(v.level)** : 경로 q의 레벨 값으로 루트 엘리먼트로부터 목적 엘리먼트까지의 해당 문서 트리의 높이

예를 들어, q가 bookstore/book[title = "Seven"]/author[first-name = "Joe"]일 경우, q.elem은 author이고 q.level은 3이다. 또한, a_path(p,l)과 d_path(p,l)은 경로 p를 레벨 값

〈표 4〉 표기(notation)

표 기	설 명
sub(*con1, con2*)	목적 엘리먼트 조건 *con1*에 *con2*의 내용이 포함될 때는 *con1* - *con2*를 반환하고, 아닐 경우 NULL을 반환한다
and(*con*)	목적 엘리먼트 조건 *con*이 공집합이면 NULL을 반환하고, {P}이면 P를 반환하고, $\{P_1, \cdots, P_n\}$일 경우 P_1 and ⋯ and P_n을 반환한다
filter(*cnf*)	conjunctive normal form으로 나타낸 조건 *cnf*가 NULL이면 NULL을 반환하고, 아닐 경우 [*cnf*]를 반환한다
q.path	XQL 질의 경로 표현식 q의 목적 엘리먼트 경로
q.con	XQL 질의 경로 표현식 q의 목적 엘리먼트 조건
q.level	XQL 질의 경로 표현식 q의 레벨
q.elem	XQL 질의 경로 표현식 q의 목적 엘리먼트
a_path(*p, l*)	경로 p를 레벨 값 *l*을 기준으로 분할 시 조상 경로
d_path(*p, l*)	경로 *p*를 레벨 값 *l*을 기준으로 분할 시 자손 경로

*l*을 기준으로 분할하였을 때 생기는 조상 경로와 자손 경로를 각각 나타낸다. 예를 들어, 경로 bookstore/book[title = "Seven"]/author[first-name = "Joe"]를 레벨 2에서 분할하면 조상 경로는 bookstore/book[title = "Seven"]이 되고, 자손 경로는 /author[first-name = "Joe"]가 된다. 레벨 3에서 분할하면 조상 경로는 원래의 경로 그대로 bookstore/book[title = "Seven"]/author[first-name = "Joe"]이 되고, 자손 경로는 NULL이 된다.

알고리즘 rewrite_XQL_query()를 적용하면, (그림 9) (a)의 경우에는 q′ = /magazine/subscription가, (그림 10) (b)의 경우에는 q′ = /book/author[last-name = "Poster"]가, 그리고 (그림 11) (a)의 경우에는 q′ = /author[first-name = "Joe"]가 반환된다. <표 5>는 이들 경우에서 q′의 경로 표현식 스트링을 구성하는 요소(서브스트링)들을 모두 나타낸 것으로, 알고리즘 rewrite_XQL_query()가 세 경우 모두 q′을 정확히 생성함을 보여준다.

알고리즘 rewrite_XQL_query()는 주어진 질의 q와 실체뷰 v에 대한 XQL 질의 경로 표현식을 구성하는 스트링의 탐색(traversal), 비교 및 조작(manipulation)을 통해 질의 변환을 수행한다. 이들 작업은 모두 이들 경로 표현식 스트링을 구성하는 엘리먼트 이름을 구성하는 서브스트링을 단위로 수행된다. 예를 들어, v = bookstore/book/author일때, 경로 표현식 스트링은 bookstore, book, 그리고 author의 세 엘리먼트로 구성되어 있고 이들 간의 구분자(delimiter)는 '/'이다. v로부터 v.path 또는 v.elem의 추출은 경로 표현식 스트링에서 이들 세 엘리먼트를 차례로 탐색하여 수행된다. 또한 v.path = q.path = bookstore/book/author일때 이들이 동일한지의 여부는, 이들 세 엘리먼트를 차례대로 탐색하며 상

〈표 5〉 MV 유형 질의 변환의 예

요 소 \ 그 림	9a	10b	11a
q	bookstore/magazine/subscription	bookstore/book/author[first-name = "Jane" and last-name = "Poster"]	bookstore/book/author[first-name = "Joe"]
v	bookstore/magazine	bookstore/book[author/first-name = "Jane"]	bookstore/book/author
v.elem	magazine	book	author
v.level	2	2	3
d_path(q, v.level)	/subscription	/author[first-name = "Jane" and last-name = "Poster"]	NULL
d_path(q, v.level).path	/subscription	/author	NULL
q.con	{ }	{bookstore/book/author/first-name = "Jane", bookstore/book/author/last-name = "Poster"}	{bookstore/book/author/first-name = "Joe"}
v.con	{ }	{bookstore/book/author/first-name = "Jane"}	{ }
q.con-v.con	{ }	{bookstore/book/author/last-name = "Poster"}	{bookstore/book/author/first-name = "Joe"}
and(q.con-v.con)	NULL	bookstore/book/author/last-name = "Poster"	bookstore/book/author/first-name = "Joe"
filter(and(q.con-v.con))	NULL	[bookstore/book/author/last-name = "Poster"]	[bookstore/book/author/first-name = "Joe"]
initial q′	/magazine/subscription	/book/author[bookstore/book/author/last-name = "Poster"],	/author[bookstore/book/author/first-name = "Joe"]
refined q′	/magazine/subscription	/book/author[last-name = "Poster"],	/author[first-name = "Joe"]

응하는 위치의 엘리먼트 간에 비교하여 판단하게 된다.

(그림 12)에 기술한 것과 같이, 알고리즘 rewrite_XQL_query()는 q와 v로부터 q(v).path 및 q(v).con의 추출, 이들 간의 비교, v.elem의 추출, v.level의 계산, q의 경로 분할, <표 4>에서 정의한 sub, and, 그리고 filter 연산을 통한 스트링 조작, 그리고 이들 과정에서 구해진 서브스트링들의 접합(concatenation)을 수행한다. 이들 작업은 모두 해당 경로 표현식 서브스트링의 레벨 수 만큼의 엘리먼트 탐색을 거쳐 수행된다. 이 레벨 값의 상한값(upper bound)은 주어진 질의 q가 접근하는 XML 문서와 관련 실체뷰 v를 도출한 XML 문서들이 준수하는 DTD 트리의 높이로 결정된다. 예를 들어, (그림 5)의 DTD 트리의 높이는 4이다. 일반적으로 DTD 트리의 높이를 h라고 하면, 알고리즘 rewrite_XQL_query()가 q′을 생성하는 과정의 복잡도는 O(h)이다.

```
rewrite_XQL_query(v, q, q′) {

/*  v : XML 실체뷰 정의
    q : XQL 질의
    q′ : v에 대한 XQL 질의
*/

/*  MV 유형 질의 처리 가능 여부 판단  */
    if( not ( (v.path = q.path or v.path <p q.path)
              and (v.con = q.con or v.con ⊂ q.con)
      )
    ) return(NULL)

/*  MV 유형 질의 변환  */
    q′ = "/" + v.elem + d_path(q, v.level).path + filter (and (sub
         (q.con, v.con)))
    /* +는 string concatenation  */
    q′ = refine_query(q′)
    return(q′)
}
```

(그림 12) XQL 질의 변환 알고리즘

7. 결 론

본 논문에서는 XML 저장소에서 XQL로 정의된 실체뷰를 지원할 경우, 저장소의 하부 문서들에 대해 제기된 XQL 질의 처리를 위해 관련된 실체뷰를 이용하는 기법에 대해 연구하였다. 이를 위해 먼저 실체뷰를 이용한 질의 처리 모델을 기술하고, 실체뷰를 지원하는 XML 저장소의 구조를 제시하였다. 그리고 주요 연구 이슈인 주어진 XQL 질의에 대하여 관련 실체뷰를 이용한 처리 가능 여부 판별과 그에 따른 질의 처리 유형 결정 방법을 제시하였다. 실체뷰를 이용한 질의 처리 유형은 세가지로 나뉜다. 첫째, 질의의 결과를 모두 실체뷰로부터 얻는 유형(MV), 둘째, 질의 결과 일부를 실체뷰로부터 얻고 나머지 결과를 하부 XML 문서들로부터 얻는 유형(UD+MV), 그리고 셋째, 질의 결과를 모두 하부 XML 문서들로부터 얻는 유형(UD)이다. 이들 중 MV 유형에 대하여 주어진 질의 q와 관련 실체뷰 v에 대하여 q와 동일한 결과를 얻는 v에 대한 질의 q′을 생성하는 질의 변환 알고리즘 rewrite_XQL_query()를 제시하였다.

향후 연구 과제로는 첫째, XML 실체뷰를 이용한 XQL 질의 처리의 성능 평가가 필요하다. 즉, 제기된 XQL 질의를 XML 저장소 내의 하부 XML 문서들에 대해 수행하는 경우에 비해 실체뷰로부터 결과를 얻는 것이 얼마나 질의 응답 시간의 단축을 가져오는지를 규명하는 것이 필요하다. 특히, 해당 실체뷰를 도출한 하부 XML 문서에 발생한 변경을 실체뷰에 반영하는 데 드는 비용을 함께 고려하는 성능 평가가 필요하다. 둘째, UD+MV 질의 처리 유형에 대한 연구가 필요하다. 이 유형에서는 질의 q와 관련 실체뷰 v가 주어졌을 때, q의 결과를 일부는 v로부터, 일부는 하부 문서로부터 검색한다. 따라서 v에 대한 질의 q′과 하부 문서에 대한 질의 q″을 생성하는 알고리즘이 필요하다. 또한, q′의 결과와 q″의 결과를 적절히 통합하여 원래의 질의 q의 결과를 구성하는 절차가 필요하다. 즉, 질의 변환 문제와 결과 통합 문제를 같이 고려해야 한다.

참 고 문 헌

[1] T. Bray et al., "Extensible Markup Language (XML) 1.0," http : //www.w3.org/TR/1998/REC-xml-19980210, 1998.

[2] S. Abitcboul et al., "Activc Views for Electronic Commerce," Proc. Int'l Conf. on VLDB, pp.138-149, 1999.

[3] J. Shanmugasundaram et al., "Relational Databases for Querying XML Documents : Limitations and Opportunities," Proc. Int'l Conf. on VLDB,pp. 302-314, 1999.

[4] 연제원 외, "XML 문서 구조검색을 위한 저장 시스템 설계", 한국정보과학회, '99년 봄학술발표논문집, 제26권 제1호, pp. 3-5, 1999.

[5] 이용석, 손기락, "XML 문서 저장 시스템 설계 및 구현", 한국정보과학회, '98년 가을학술발표논문집, 제25권 제2호, pp. 347-349, 1998.

[6] 한상웅, 홍의경, "ORDBMS를 이용한 XML 저장 시스템 설계", 한국정보과학회, 2000년 가을학술발표논문집, 제27권 제2호, pp.3-5, 2000.

[7] J. Robie et al., "XML Query Language(XQL)," http : //www.w3.org/TandS/QL/QL98/pp/xql.html, 1998.

[8] J. Robie et al., "The Design of XQL," http : //www.texcel.no/whitepapers/xql-design.html, 1998.

[9] A. Deutsch et al., "XML-QL : A Query Language for XML,"

http : //www.w3.org/TR/1998/NOTE-xml-ql-19980819/, 1998.

[10] J. Clark and S. DeRose., "XML Path Language (XPath) Version 1.0," http : //www.w3.org/TR/xpath, 1999.[11] D. Chamberlin et al., "Quilt : An XML Query Language for Heterogeneous Data Sources," http : //www.almaden.ibm.com/cs/people/chamberlin/quilt_lncs.pdf, 2000.

[12] D. Chamberlin et al., "XQuery 1.0 : An XML Query Language," http : //www.w3.org/TR/xquery, 2001.

[13] S. Abiteboul, "On Views and XML," Proc. ACM Symp. on Principles of Database System, pp.1-9, 1999.

[14] A. Gupta and I. Mumick, "Maintenance of Materialized Views : Problems, Techniques, and Applications," Bulletin of TCDE, Vol.18, No.2, pp.3-18, Jun. 1995.

[15] Y. Papakonstantinou and V. Vianu, "DTD Inference for Views of XML Data," Proc. of 19th ACM SIGACT-SIGMOD-SIGART Symp. on PODS, pp.35-46, 2000.

[16] 임재국 외, "XML 실체뷰 관리 프레임워크", 한국정보과학회, 2000년 가을학술발표논문집, 제27권 제2호, pp.243-245, 2000.

[17] eXcelon co., http : //www.exceloncorp.com.

[18] R. Pottinger and A. Levy, "A Scalable Algorithm for Answering Queries Using Views," Proc. of the 26th Intl Conf. on VLDB, pp.484-495, 2000.

[19] A. Levy, "Answering Queries Using Views : A Survey," http : //www.cs.washington.edu/homes/alon/site/YearPage_C1477634.html, 2000.

[20] A. Levy et al., "Answering Queries Using Views," Proc. of the 14th ACM SIGACT-SIGMOD-SIGART Symp. on PODS, pp.95-104, 1995.

[21] H. Yang and P. Larson, "Query Transformation for PSJ-Queries," Proc. Int'l Conf. on VLDB, pp.245-254, 1987.

[22] S. Chaudhuri et al., "Optimizing Queries with Materialized Views," Proc. Int'l Conf. on Data Eng., pp.190-200, 1995.

[23] O. Duschka and M. Genesereth, "Answering Queries Using Recursive Views," Proc. Symp. on PODS, pp.109-116, 1997.

[24] S. Abiteboul and O. Duschka, "Complexity of Answering Queries Using Materialized Views," Proc. Symp. on PODS, pp.254-265, 1998.

[25] F. Afrati et al., "Answering Queries Using Materialized Views with Disjunction," Proc. ICDT, pp.435-452, 1999.

[26] Y. Papakonstantinou and V. Vassalos, "Query Rewriting for Semistructured Data," SIGMOD Proc. Int'l Conf. on Management of Data, pp.455-466, 1999.

[27] D. Calvanese et al., "Answering Regular Path Queries Using Views," Proc. Int'l Conf. on Data Eng., pp.389-398, 2000.

[28] D. Florescu et al., "Query Containment for Conjunctive Queries with Regular Expressions," Proc. PODS, pp.139-148, Jun. 1998.

[29] H. Garcia-Molina et al., "The TSIMMIS Approach to Mediation : Data Models and Languages," J. of Intelligent Information Systems, Vol.8, No.2, pp.117-132, 1997.

[30] Y. Papakonstantinou et al., "Object Exchange Across Heterogeneous Information Sources," Proc. Int'l Conf. on Data Engineering, pp.251-260, 1995.

[31] M. Fernandez et al., "STRUDEL : A Web-Site Management System," Proc. SIGMOD Int'l Conf. on Management of Data, pp.549-552, 1997.

[32] M. Fernandez et al., "A Query Language for a Web-Site Management System," SIGMOD Record, Vol.26. No.3, pp. 4-11, Sept. 1997.

[33] D. Suciu, "Query Decomposition and View Maintenance for Query Languages for Unstructured Data," Proc. Int'l Conf. on VLDB, pp.227-238, 1996.

[34] Y. Zhuge and H. Garcia-Molina, "Graph Structured Views and Their Incremental Maintenance," Proc. Int'l Conf. on Data Engineering, pp.116-125, 1998.

[35] S. Abiteboul et al., "Incremental Maintenance for Materialized Views over Semistructured Data," Proc. Int'l Conf. on VLDB, pp.38-49, 1998.

[36] D. Lee and W. Chu, "Conjunctive Point Predicate-based Semantic Caching for Web Databases," Extended Abstract, Tech. Rep. TR-980030, UCLA, Sep. 1998.

김 수 희

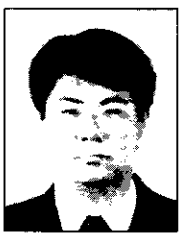
e-mail : soohee@lge.com

1999년 중앙대학교 컴퓨터공학과 졸업 (공학사)

2001년 중앙대학교 컴퓨터공학과 대학원 졸업 (공학석사)

현재 LG전자㈜ 재직 중

관심분야 : 이동 데이터베이스, XML 등



문 찬 호

e-mail : moonch@rose.cse.cau.ac.kr

1997년 중앙대학교 컴퓨터공학과 졸업 (공학사)

1999년 중앙대학교 컴퓨터공학과 대학원 졸업(공학석사)

현재 중앙대학교 컴퓨터공학과 박사과정 재학 중

관심분야 : Web 데이타베이스, XML 등

강 현 철

e-mail : hckang@cau.ac.kr

1983년 서울대학교 컴퓨터공학과 졸업 (공학사)

1985년 U. of Maryland at College Park, Computer Science(M.S.)

1987년 U. of Maryland at College Park, Computer Science(Ph.D.)

1988년~현재 중앙대학교 컴퓨터공학과 교수

관심분야 : 이동 데이타베이스, 웹 데이타베이스, DBMS 저장 시스템 등

서 상 구

email : skseo@daisy.gwu.ac.kr

1984년 서울대학교 컴퓨터공학과 졸업 (공학사)

1986년 한국과학기술원 전산학과(M.S.)

1995년 한국과학기술원 전산학과(Ph.D.)

1999년~현재 광운대학교 경영정보학과 조교수

관심분야 : 데이타웨어하우징, 웹 데이타베이스, 데이타베이스 최적화 등