XML가상뷰에서 XQL 질의처리 시스템 구현

김천식 손기락
한국외국어대학교 컴퓨터공학과
{dol, ksohn}@san.hufs.ac.kr

XQL query processing system implementation on the XML virtual view

Chunshig Kim° Kirack Sohn
Dept. of Computer Engineering, Hankuk University Of Foreign Studies

요약

XML은 웹상에서 상업 데이터 교환을 위한 표준이 되었다. 하지만, 대부분의 상업적인 데이터가 관계형 데이터베이스 시스템에 저장되어 있다. 만일 XML의 특성을 이용한다고 할 때 XML문서로서 관계형 데이터를 출판할 필요가 있게 된다. 즉, 관계형 데이터를 XML로의 출판에 관한 많은 연구 및 상업적 이용이 있어왔다. 본 논문에서는 XML 뷰에 해당하는 R2X에 XQL로 질의하여 새로운 R2X뷰 언어를 만든 다음 이를 RDB엔진을 이용하여 결과를 얻어 XML로의 출판에 관한 연구를 하였다. 또한 효과적인 출판을 위한 방법을 고안하고 질의 처리를 제안하였다. 향후, 이를 바탕으로 상업적인 분야에 도움을 줄 것으로 기대한다.

1. 소개

XML은 WWW에서 상업적 데이터 교환을 위한 표준으로서 등장했다. XML의 포함형태(nested), 자기서술(Self-describing)구조는 애플리케이션을 위한 데이터교환에 사용되고는 있지만, 아직까지 불편한 점이 있다. 사실, 이미 여러 기관에서 XML을 위한 스키마로서의 DTD (Document Type Description) 표준안을 제안했다. DTD는 전자상거래와 부동산등 다양한 분야를 위해서 개발되었다. XML의 환경에도 불구하고, 대부분의 운용데이터, 새로운 웹기반 애플리케이션을 위해서 조차 관계형 데이터베이스 시스템에 저장되고 있다는 점은 매우 중요하다[1]. 따라서 XML문서의 형태로 관계형 데이터를 출판하는데 필요한 메커니즘이 필요로 하게 된다. 이러한 방법에 많은 연구가 있었지만 대부분이 규범적인 DTD로 변환시키는 연구가 대부분이었다[2]. 이러한 접근은 매우 제한적이다. 왜냐하면 공개된 DTD가 적당한 관계형 스키마에 정확하게 일치할 수 없기 때문이다. XML문서들은 각각의 DTD에 순응한다. 이와 같은 관점에서 뷰를 이용하여 질의를 처리하는 SilkRoute[3]가 있었다. 이 시스템은 XML 뷰에 XML-QL[4]질의 결과를 받아 출력하는 형태였다.

XQL은[5] XML질의언어로 향후 더욱 범용성이 높을 것으로 기대되는 언어이다. 따라서 XML 가상뷰(virtual view)인 R2X를 만들고 XQL로 질의하여 결과를 얻는 시스템은 매우 폭 넓게 사용될 것으로 본다. R2X는 본 논문에서 제한한 뷰이다. 본 논문에서는, 변형된 XQL로[6] 관계형 데이터베이스에 질의하고, 결과를 XML로 볼 수 있는 일반적이며, 동적이며, 효과적인 툴인 R2X 툴을 제안한다. XQL에는 또한 템플릿이란 특성이 없다. 따라서 XML출판에 효율적인 XQL구조로의 변경과 내부구조의 처리 방법을 제안하였다.

2. *R2X(Relational to XML)* 툴의 구조

XQL을 이용하여 RDB에 질의하기 위해서는 사용자가 RDB를 XML로 볼 수 있는 뷰가 필요하다. 이와 같은 뷰를 정의하기 위해서 본 논문에서 정의한 것이 R2X뷰이다. 즉, RDB의 스키마의 일부를 적절하게 R2X뷰로 변환시킨다. 이후 사용자는 R2X뷰 만을 보고 XML문서에 XQL로 질의하여 질의 결과를 얻도록 하는 것이다. 이와 같은 이유로 실험을 위해 다음과 같은 DTD를 설계하였다 (그림 1). (그림 2)는 (그림 1)의 XML DTD를 RDB 스키마로 표현한 것이다. R2X툴의 구조는 (그림 3)과 같은 모습이다. 이것은 애플리케이션 프로그램을 통해서 데이터를 접근하여 애플리케이션과 관계형 데이터 사이의 미들웨어로 작동한다.

(그림 4)는 bib(본 논문에서 이용하는 서적 데이터)(그림 2)에 대한 뷰 이다. 뷰(그림 4)를 보고 (그림 5)를 이용하여 질의를 던진다.

R2X툴의 질의 결합은 사용자 질의와 R2X뷰 질의를 취하고 결합된 질의는 (그림 6)에 있다.

(그림 7)은 질의 결합에 대한 구조를 묘사한 것이다. 질의 구성자(composer)는 XQL질의 (U)를 패턴과 필터로 분리하고 R2X뷰 내부 표현으로 변환시킨다. 패턴 매처(pattern matcher)는 뷰 트리에서의 XQL패턴을 평가하고 일치하는 패턴 정보를 찾아낸다. Rewriter는 U의 패스정보와 V의 R2X툴 정보를 취하여 변환정보를 얻은 다음 XQL내부표현구조에 보관한다.

```
<?xml encoding="US-ASCII"?>
<!ELEMENT bib(article)*>
<!ELEMENT article(author+, title, journal?) >
<!ATTLIST article  year CDATA #REQUIRED
                   month CDATA #IMPLIED
                   volume CDATA #IMPLIED
                   number CDATA #IMPLIED
                   pages CDATA #IMPLIED>
<!ELEMENT publisher (#PCDATA)>
<!ELEMENT address (#PCDATA)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (first_name?, last_name)>
<!ELEMENT first_name  (#PCDATA)>
<!ELEMENT last_name (#PCDATA)>
<!ELEMENT journal (#PCDATA)>
```

그림 1 : bib.dtd

```
ARTICLE(Title, Journal, Year, Month, Volume, Number,
Pages, Author_id)
AUTHOR (Author_id, First_name, Last_name)
```

그림 2 : bib에 대한 RDB 스키마

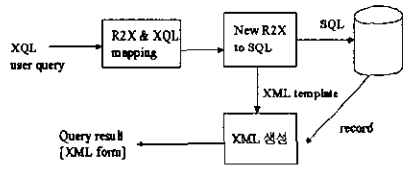

그림 3. R2X툴의 구조

```
bib { |
  article { |
        title ($AR := ARTICLE) { $AR[Year > 1980]/title }
        author ($AU := AUTHOR) {
            $AU[$AU/author_id= $AR/author_id]/first_name
        }
        Journal { $AR[Year > 1980]/journal }
        year {  $AR[Year > 1980]/Year  }
        month {  $AR[Year > 1980]/month  }
        volume { $AR[Year > 1980]/volume }
        number { $AR[Year > 1980]/number }
        pages { $AR[Year > 1980]/pages }
  }   }
```

그림 4. R2X뷰 QUERY(V)

```
results { |
   result { |
       article { |
           title { .//article/title [.//author="Norman"] }
           author{ .//article/author[.//author="Norman"]  }
           Journal { //journal[.//author="Norman"]  }
           year { //article/year[.//author="Norman"]  }
       }  }  }
```

그림 5 : XQL user query(U)

3. 결합(XQL과 R2X뷰) 및 SQL변환 알고리즘

(그림 8)는 뷰의 저장구조이다. (그림 9)은 XQL 구조를 나타내며 R2X와 XQL변환처리를 위한 내부 표현을 나타낸 것이다.

```
results { |
   result { |
       article { |
           title { ($AR := ARTICLE)
                   ($AU:= AUTHOR) {
              $AR[Year> 1980 and author="Norman" and
                    $AR/author_id=$AU/author_id]/title
           }
           author {
              $AU[year > 1980 and autho="Norman" and
              $AR/author_id=$AU/author_id]/first_name
           }
           Journal {
              $AR[year > 1980 and author="Norman" and
               $AR/author_id=$AU/author_id]/journal
           }
           year {
              $AR[year > 1980 and author="Norman" and
              $AR/author_id=$AU/author_id]/year
           }      }      }      }
```

그림 6. 합성 R2X Query (C)

XQL구조(그림 9)중 "TypeName"는 [5]를 참조하여 XQL질의 형태를 유형별로 분리할 수 있었다.

유형으로는

- "Selecting Children and Descendant" 인 경우

이 질의는 특정 엘리먼트의 자식엘리먼트나 손자엘리먼트 또는 임의의 깊이를 갖는 엘리먼트를 선택하기 위한 질의이다.

- "Filters" 인 경우

이 질의는 book[excerpt]와 같은 형태의 질의이다.

- "Boolean Expression" 인 경우

이 질의는 필터('[', ']')내부에 and, or, $not $이 오는 형태이다.

- "Equivalence" 인 경우

author[last-name = 'Bob'] 과 같은 질의이다. 본 논문에서는 대소비교와 연산까지를 포함하는 형태로 본다.

- "Union and Intersection" 인 경우

연산자를 $Union$, $Intersection$을 사용하는 형태의 질의이다.

첫 번째 유형은 정확하게 table, filter, select가 하나만 존재하는 형태이고 두 번째, 세 번째, 다섯 번째 유형은 여러 개의 table, filter, select가 존재하는 형태로서 이들 정보를 단순히 저장한 다음 SQL로 변형할 때 이용한다. 네 번째 유형은 table, filter, select를 하나로 결합 처리한 후 SQL로의 변형할 때 이들 정보를 이용한다. 이와 같이 유형별로 질의를 처리하기 위하여 "Type Name"정보를 이용한다. (그림 9)은 Title엘리먼트 밑에 있는 질의를 가지고 R2X에 변환시킨다. 일치하는 Filter와 Table을 결합하여 Title엘리먼트 내부 구조로서 표현한 것이다. 특정 엘리먼트가 어떤 유형인지를 파악한 후 다음의 알고리즘을 이용하여 R2X를 SQL로 바꿀 수 있다.

R2X를 SQL로 바꾸는 알고리즘은 다음과 같다. 유형에 따라서 아래와 같이 변환한다 :

- "Selecting Children and Descendant"인 경우

SQL로 "select ~ from ~ where ~ " 형태

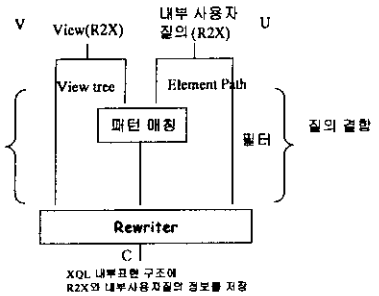

그림 7 : 질의 결합(Composition) 다이어 그램

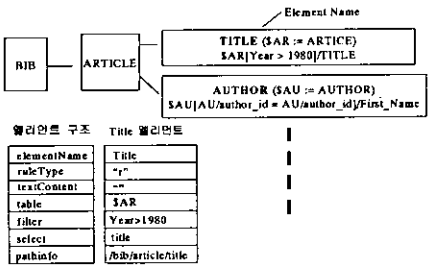

그림 8. R2X 뷰 저장구조

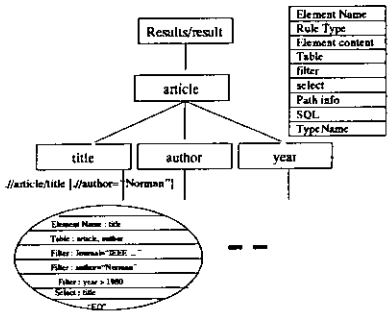

그림 9. R2X 와 User XQL의 결합

• "filter" 인 경우
SQL로 "select ~ from ~ where ~ and exists(select ~ from ~ where ~)" 의 형태
• "Equivalence"인 경우
SQL로 "select ~ from ~ where ~ and ~ and ~ .. " 의 형태
• "Union and Intersection"인 경우
두 개이상의 SQL을 Union으로 결합하거나 Intersection으로 결합하는 형태
• "Boolean Expression"인 경우
SQL로 "select ~ from ~ where ~ exists(select ~ from ~ where ~) and exists(select ~ from ~ where ~) . . . " 형태

4. 결론

XML은 웹상에서 상업 데이터 교환을 위한 표준이 되었다. 그러나 현재 대부분의 상업적인 데이터가 관계형 데이터베이스 시스템에 저장된다. 만일 XML을 이용해서 데이터를 교환할 때 XML문서로 관계형 데이터를 출판할 필요가 있게 된다. 즉, 관계형 데이터를 XML로 출판에 관한 많은 연구 및 상업적 이용이 있어왔다. 하지만 사용상의 많은 불편함이 존재한다. 따라서, 본 논문에서는 R2X툴의 구조를 제한하여 이러한 문제점의 해결을 제안하였다.

R2X툴은 XML에서 관계 데이터의 뷰를 만들고 질의하기 위한 일반적이며, 동적이며, 효과적인 프레임워크이다. 이 작업은 가상뷰와 사용자 질의를 결합하기 위한 알고리즘 및 R2X뷰를 SQL로 변환하기 위한 알고리즘이다. R2X툴은 많은 장점이 있다. 사용자에 의해서 요청 받은 관계형 데이터만이 구체화(materialized)되고, 데이터는 요구 즉시 항상 만들어진다. 그리고 관계형 엔진은 일반적인 계산의 대부분을 효율적으로 수행한다. 이러한 형태의 질의 처리시스템은 [3]과 유사하지만 본 논문에서 대상언어가 XQL[5]이므로 실질적인 처리에 있어서는 많은 차이가 있음을 보였다. 예를 들면 XML-QL은[4] 템플릿이 있었지만 XQL은[5] 템플릿이 없는 언어이며 XQL의 질의처리는 패스형태의 질의 처리라는 면에서 SQL이나 XML-QL과은 차이가 있다. 이러한 문제는 (그림 8)의 구조를 이용하여 해결하였다. 이러한 문제해결을 바탕으로 XQL의 범용성은 향후 상업적 이용에 기여할 것으로 기대된다.

5 참고 문헌

[1] J. Shanmugasundaram, E. Shekita, R. Barr, M. Carey, B. Lindsay, H. Pirahesh, B. Reinwald, "Efficiently Publishing Relational Data as XML Documents", VLDB Conference, September 2000. Click here for the slides.

[2] Ronald Bourret, XML and Databases, "http://www.rpbourret.com/xml/XMLAndDatabases.html

[3] Mary F. Fernandez, Wang Chiew Tan, Dan Suciu: SilkRoute: trading between relations and XML. WWW9 / Computer Networks 33(1-6): 723-745(2000)
[DBLP:journals/cn/FernandezTS00]

[4] Alin Deutsch , Mary Fernandez , Daniela Florescu, Alon Levy , Dan Suciu , "XML-QL: A Query Language for XML" ,http://www.w3.org/TR/1998/NOTE-xml-ql-19980819/

[5] Jonathan Robie, Joe Lapp, David Schach, XML Query Language (XQL), http://www.w3.org/TandS/QL/QL98/pp/xql.html

[6]"Jonathan Robie (Software AG) <jonathan.robie@sagus.com>" XQL (XML Query Language), http://www.ibiblio.org/xql/xql-proposal-1999Aug09.html