論文
2-6-9

XPath 질의를 이용한 DB2XML 알고리즘 설계 및 구현 (Design and implementation of algorithms for DB2XML using XPath query)

김 노 환* 정 충 교**
(No-Whan Kim) (Choong-Kyo Jeong)

요 약

XML은 이미 웹 상에서 상업적인 데이터 교환을 위한 표준으로 정착되었다. 대부분의 기관들은 XML 문서가 EDI의 형태로서 각종 문서교환용으로 이용할 수 있기를 원하지만, 불행하게도 아직까지는 기존의 상업적인 데이터는 대부분 관계형 데이터베이스에 저장되어 있다. 따라서, 이들 기존 관계형 데이터베이스에 저장되어 있는 데이터를 XML 문서 형태로 변환해서 문서교환에 이용할 필요가 있다.

관계형 데이터베이스의 데이터를 XML로 변환하기 위해서는 관계형 데이터베이스 테이블의 각 필드를 XML로 매핑하여 XML 문서로 출판할 수도 있겠지만, 두 개 이상의 데이터베이스 테이블을 결합해서 하나의 XML 문서를 만드는 경우, DTD와 연관된 단순 매핑 만으로는 문제를 해결하지 못하므로 조인(join)을 실행해야만 한다. 본 논문에서는 조인을 통해서 생성된 엘리먼트들이 보이도록 XML을 위한 뷰를 만들고, 이 뷰를 이용하여 관계형 데이터베이스의 내용을 XML로 변환하기 위한 알고리즘을 제안하고 이를 구현하고자 한다.

ABSTRACT

XML has already been settled down as a standard for the exchange of commercial data on the web. Though most institutions have a wish to use XML document as an EDI form for all sorts of document exchange, it is unfortune that previous commercial data is still saved in the relational form of data base. Therefore, it is necessary that data saved in the relational form of data base be transformed into XML document form for the better use of document exchange.

In order to transform relational form of data base to XML, one solution is to publish XML document by way of mapping each field of the relational form of data base table onto XML. However, in the case of building one XML document out of more than two data base tables, a join should be performed since a mere mapping associated with DTD can not solve the problem. In this paper, we build the view for the XML in which alignments generated from the join are presented, and then through this view the contents with the relational form of data base should be transformed into XML. Briefly speaking, This paper aims to propose algorithm concerned with this transformation and to realize it.

* 정회원 : 동우대학 인터넷통신과 조교수
** 정회원 : 강원대학교 전기전자·정보통신공학부 부교수

논문접수 : 2001. 6. 12.
심사완료 : 2001. 6. 21.

1. 서론

XML(eXtensible Markup Language)은 이미 웹 상에서 상업적인 문서 교환을 위한 표준으로서 시스템 통합을 위한 기술로 정착되었다. XML의 네스트(nested), 자기서술적(Self-describing) 구조는 어플리케이션을 위한 문서교환을 위해서 이미 유용하게 사용되고 있으며, 다수의 기관들이 XML을 위한 스키마로서 DTD(Document Type Definition) 표준안을 제안한 바 있고, 전자상거래와 전자도서관 등 다양한 분야에서 표준 DTD가 개발되어 상용화되는 등 XML 기술은 급속도로 발전하고 있다.

그러나 XML의 놀라운 발전과 개발 환경에도 불구하고, 새로운 웹 기반 어플리케이션 환경에서도 대부분의 운용 데이터들은 기존의 RDB 시스템에 저장되고 있다는 점은 매우 의미가 있다[1].

관계형 데이터 베이스에 저장된 데이터를 XML 문서형태로 출판하기 위한 메커니즘의 필요에 따라 많은 연구가 있었지만 규범적인 DTD에 1:1로 매핑하는 연구가 대부분으로[2], 이러한 접근 방법은 공개된 DTD가 사용자가 원하는 관계형 스키마에 정확하게 일치할 수 없기 때문에 매우 제한적일 수밖에 없었다. 또한 사용자들은 한 관계 데이터를 여러 용도의 XML 문서로 매핑하기를 원한다는 사실에 주목할 필요가 있다.

본 논문에서는, 관계형 데이터를 XML 뷰로 변환하게 되면 아무리 복잡한 매핑도 처리가 가능하므로, [그림 4]에 제안된 XML 뷰에 [그림 5]에 제안된 XPath로 XML 뷰에 질의하고, 결과를 XML로 볼 수 있는 일반적이고 동적이며 효과적인 툴을 만들기 위한 알고리즘을 제안하고 이를 구현하고자 한다. 향후 이러한 시스템은 매우 폭 넓게 사용될 것으로 기대된다.

본 논문의 2장에서는 관련분야에 대한 특성에 대해 기술하고, 3장은 제안된 변환구조와 뷰에 대해 설명한다. 4장은 제안된 알고리즘의 구현과정을 도출하였고, 5장은 결론 및 향후 계획을 제시한다.

2. 관련 연구

웹 상의 XML 문서에서는 데이터를 효율적으로 추출하고 질의하기 위해서 다음과 같은 다양한 질의어들이 사용되고 있다.

ODBC2XML 툴은 사용자가 내장된 SQL을 이용해서 XML 문서를 정의하고 XML 뷰를 구성하도록 하므로 기본적으로 범용성이 결여되어 있다.

오라클의 XSQL 툴은 고정된 정의로서 관계형 테이블과 속성 이름에 XML 태그를 직접 매핑하는 규범적 변환처리를 사용하므로, 필드명을 다른 용도로 사용하기 위한 1:1 매핑 등 임의처리가 불가하므로, 충분한 일반화를 제공하지 못한다.

IBM의 DB2XML Extender는 XML에 관계형 데이터를 결합하고 관계형 테이블에 XML 데이터를 분리하는 것을 지원하는 DAD(Data Access Definition) 언어를 제공한다. DAD의 결합특성은 관계형 데이터로부터 임의의 XML에 대한 생성을 지원하지만 질의어를 지원하지는 못한다.

XML-QL(Query Language)은 SQL의 확장판으로 선언 형태의 언어로서, W3C 기술노트로 제안되었다.[3] 기존의 SQL처럼 조건을 명시하는 "where"절과 질의의 결과를 XML로 재구성하기 위한 "construct"절로 구성이 되어 있으며, 질의, 구조, 변환, 그리고 XML 데이터의 통합 및 새로운 XML data의 생성을 지원한다. 그러나 XQL이나 XPath에 비해 질의가 복잡하고 표현이 긴 단점이 있다.

XQL(XML Query Language)은 XSL의 패턴정의 부분을 확장한 언어로서, XML 트리의 각 엘리먼트 노드를 따라 이동하게 되고 결과로 패턴과 일치하는 노드들의 집합을 반환하도록 하여, XSL에 비해서 훨씬 간단하게 문장을 구성할 수 있도록 제안되었다. 그러나 XML-QL의 "construct"절이 없으므로 XML 문서 구성시 어려움이 있다.[4]

XML-GL(Graphic Language)은 XML 문서에 대한 질의를 그래프 형태로 나타내는 질의어로, 사용자가 사용하기 쉽도록 "where"절과 "construct"절은 비쥬얼 인터페이스를 이용하여 정의한다. 그러나, 질의가 복잡해지면 그래프도 복잡해지므로 읽기가 어려워진다.

XML을 위한 다른 언어로는 XSL[5], XMAS[6], Lorel[7], 그리고 YATL 등이 있다.

지금까지의 데이터베이스를 XML로 변환하기 위한 방법은 1대1 매핑 방법으로 자연스럽지 못한 처리였다. 이러한 문제점을 해결하는 변환 알고리즘은 향후 인터넷 문서가 HTML이 아닌 XML이 될 경우 XML에 질의하여 원하는 결과를 보다 분명하게 질의

하고 결과를 얻도록 하는데 많은 공헌을 할 것으로 기대한다.

3. 제안된 DB2XML 변환 구조

XPath를 이용하여 RDB에 질의하려면, RDB의 내용을 볼 수 있는 XML 뷰(View)가 필요하다.

이러한 DB2XML 변환과정을 설명하기 위해서 DTD, 뷰 및 질의어를 다음과 같이 정의하였다.

[그림 1]은 실험에 이용한 DTD로서, name, e-mail, url, link의 직원관련 정보를 나타낸다. 여기서, name은 직원 성명, e-mail은 직원의 e-mail 주소, url은 직원의 url, link는 직원과 연관된 직원으로서 고용자의 상급자 또는 부하 직원을 나타낸다.[8]

```
<?xml version="1.0" encoding="US-ASCII"?>
<!ELEMENT personnel (person+)>
<!ELEMENT person (name,email*,url*,link?)>
<!ATTLIST person id ID #REQUIRED>
<!ELEMENT name (family?|given?)+>
<!ELEMENT family (#PCDATA)>
<!ELEMENT given (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT url (#PCDATA)>
<!ELEMENT link (manager?,subordinates?) >
<!ELEMENT manager (#PCDATA) >
<!ELEMENT subordinates (#PCDATA) >
```

[그림 1] 고용자에 대한 정보(personnel.dtd)

[Fig. 1] Information for personnel

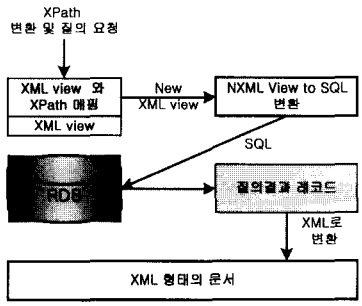
[그림 2]는 [그림 1]과 연관된 RDB 스키마이다.

```
personnel(*pid, nid, lid, email, url)
name(nid, family, given)
link(lid, manager, subordinates)
```

[그림 2] 고용자를 위한 RDB 스키마

[Fig. 2] RDB schema for personnel

[그림 3]은 본 논문에서 제안한 DB2XML 변환 구조로서, 응용 프로그램을 이용하여 RDB에 접근한 후, 제안된 알고리즘에 의해서 데이터를 XML로 변환할 수 있다.



[그림 3] DB2XML 변환 구조

[Fig. 3] Transform Structure for DB2XML

이 구조는 DB 관리자가 DB에 XML 뷰를 정의하고, 사용자가 XPath로 질의하면, XML 뷰와 XPath 질의를 매핑하여 New XML 뷰를 생성하고, 생성된 뷰 질의어를 SQL 질의어로 변환한다. 변환된 SQL 질의어가 RDB에 질의하면 질의 결과로 원하는 레코드를 추출하여, 적절한 태그 처리 알고리즘을 통해서 XML 문서를 생성하게 된다.

```
 1: personnel { |
 2:   person ($PS:=PERSON) { $PS/pid |
 3:     name ($NM:=NAME) { |
 4:       family {
 5:         $NM[$PS/nid = $NM/nid]/family }
 6:       given {
 7:         $NM[$PS/nid = $NM/nid]/given }
 8:     }
 9:     email { $PS/email }
10:     url { $PS/email }
11:     link ($LN:=LINK) { |
12:       manager {
13:         $LN[$PS/lid=$LN/lid]/manager }
14:       subordinates {
15:         $LN[$PS/lid=$LN/lid]/subordinates }
16:     }
17:   }
18: }
```

[그림 4] XPath 질의를 위한 XML 뷰

[Fig. 4] XML view for XPath query

[그림 4]는 [그림 2] personnel 관계형 데이터베이스 스키마를 위한 XML 뷰이다. [그림 4]에서 줄 1은 루트 엘리먼트인 personnel을 나타낸 것으로서, "{"는 personnel 엘리먼트에 관련한 룰(SQL과 대응되는 부분)을 포함하기 위한 표시이고, "|"는 해당 엘리먼트가 자식 엘리먼트를 가지고 있음을 나타낸다. 줄 2는 person 엘리먼트로서, "("와 ")" 사이에 관계형 데이터베이스에 존재하는 PERSON 테이블 명이 나

타나면 이를 관련 변수 "$PS"에 할당한다. 줄 3은 name 엘리먼트를 나타내고, name 엘리먼트에 테이블 이름 NAME을 변수 $NM에 할당하고 있다.

줄 5는 family 엘리먼트를 위해서 데이터베이스 스키마와 XPath 질의를 매핑하는 과정으로, NAME은 테이블명이고, NM은 간략히 사용하기 위한 재명명(Rename) 테이블명이며, $NM은 관련 변수명이고, NM.nid는 NM 테이블의 nid 필드를 의미한다.

상기 문장을 SQL로 변환시키면 다음과 같다.

select 다음에 *family*가 온 것은 XPath 질의에 의한 것이며, XPath에서 "["와 "]" 사이의 문장은 XPath에서 조건절을 나타내므로, SQL의 Where절과 일치함을 알 수 있다.

줄 9는 email 엘리먼트로서, PERSON 테이블에서 email 필드를 추출하라는 질의표현이다.

어플리케이션은 직접적으로 관계형 데이터베이스에 접근하지 못하지만 XML 뷰를 통해서는 가능하므로, XPath로 사용자 질의를 작성하면 된다.

XML을 위해 설계된 XPath 질의어는 절대 및 상대패스를 이용한 질의가 가능하며, 필터링 처리를 위한 기호로 "["나 "]"를 사용한다. 결국, 위에서 설명한 [그림 4]의 XML 뷰를 사용자에게 보여주고, [그림 5]와 같은 XPath 질의를 하게 하면 된다.

```
.//person/link/manager[.//name/family="Pfeiffer"]
```

[그림 5] 사용자의 XPath 질의

[Fig. 5] User's XPath query

[그림 5]의 XPath 질의어에서,[9][10][11]

▶ ".//person/link/manager"는 루트로부터 임의의 경로("//")에 존재하는 person을 찾고, person의 자식인 link, link의 자식인 manager 엘리먼트를 찾아서, 정보를 추출하라는 것이다.

▶ [.//name/family="Pfeiffer"]은 SQL에서 where절과 유사한 것으로서, 바로 위의 질의에 대한 조건 절로, 임의의 경로에 존재하는 name 엘리먼트 자식인 family 엘리먼트의 내용이 "Pfeiffer"인 조건을 만족하는 manager를 찾으라는 것이다.

[그림 6]은 [그림 4]와 [그림 5]를 결합하여 새롭게 생성된 New XML view이다.

```
 1: personnel { |
 2:   person ($PS:=PERSON) { |
 3:     name ($NM:=NAME) { |
 4:       family {
 5:         $NM[$PS/nid = $NS/nid and
                .//name/family="Pfeiffer"]/family }
 6:         link ($LN:=LINK) { |
 7:         manager {
 8:           $LN[$PS/lid=$LN/lid]/manager
 9:         }
10:     }
11:   }
12:}
```

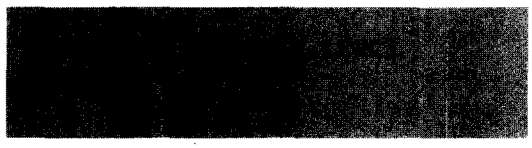
[그림 6] New XML 뷰[그림 4, 5를 매핑]

[Fig. 6] New XML view by mapping Fig 4 & 5

[그림 6]은 [그림 4]의 XML 뷰에 [그림 5]의 XPath 질의를 하는 경우, 일치하는 엘리먼트에 해당하는 정보를 나타내는 것으로, [그림 6]을 SQL로 변환시키면 원하는 레코드를 추출할 수 있다.

[그림 6]의 줄 5는 family가 "Pfeiffer"인 것을 가져오라는 조건 질의가 부가된 명령문으로, SQL로 표현하면 PS와 NM 테이블을 조인하고 NM 테이블에서 family 필드의 내용이 "Pfeiffer" 것에 해당하는 family를 보이라는 의미이다.

[그림 6]을 SQL로 변환하면 다음과 같다.



이 SQL 명령문은 where절 "X.nid = Y.nid and X.lid = L.lid"을 통해서 테이블 X,Y,L을 조인하고, 조인된 테이블에서 X.family가 Pfeiffer에 해당하는 사람의 manager를 찾으라는 질의이다.

4. 결합(XPath 와 XML 뷰) 및 SQL 변환 알고리즘 구현

[그림 7]은 XML 뷰를 위한 기본 문법으로, 본 논문에서 직접 독창적으로 디자인한 것이다.

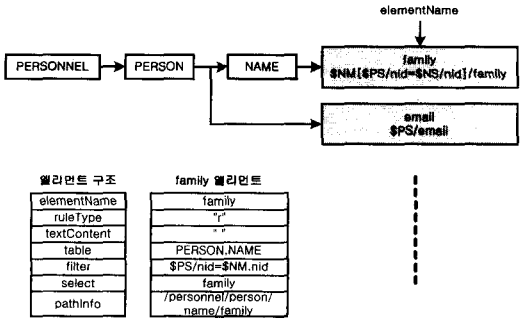
```
Node :- Tag, Rule, [Node]
Rule :- Variable, [Condition]
Condition :- TableName | "[" Filter "]"
Variable :- "(" Var ":=" Value ")" | Variable
Filter :- AND(Filter,Filter) | OR(Filter,Filter) |
          Not(Filter) | Term | Term RelOp Term
Relop :- "=" | "<" | "<=" | ">" | ">="
```

[그림 7] XML 뷰를 위한 구문

[Fig. 7] Grammar for XML view

[그림 8]의 엘리먼트 저장구조는 elementName, ruleType, textContent, table, filter, select, path Info로 되어 있다. family 엘리먼트의 저장 형태를 살펴보면, 우선 elementName은 "family"로서, 이와 같이 엘리먼트 이름 정보를 따로 정한 이유는 임의의 엘리먼트 이름이 들어오더라도 해당 엘리먼트에 룰이 있으면 임의의 엘리먼트를 이름으로 하는 구조에 룰 정보를 편리하게 저장하고 관리하기 위한 것이다.

[그림 8]과 같은 구조화를 통한 정보 저장이 이후에 SQL로 변환할 때 템플릿 기능을 하게 된다. 즉, 이 elementName을 이용하여 태그를 정의할 수 있다.



[그림 8] XML 뷰 저장구조

[Fig. 8] Storage structure for XML view

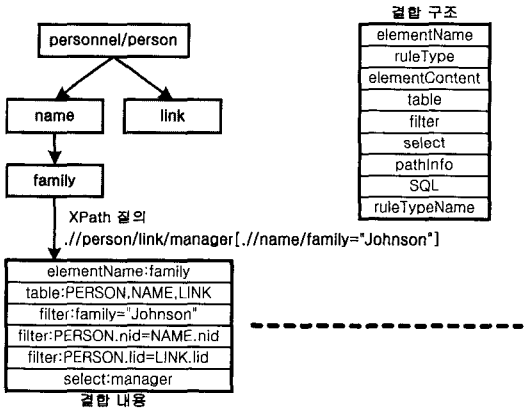
ruleType은 "r/t/x"로 표시할 수 있다. "r"은 해당 엘리먼트의 내부에 질의정보를 갖고 있는 것이고, "t"는 해당 엘리먼트가 질의가 아닌 텍스트 형태의 값을 갖고 있는 것이며, "x"는 해당 엘리먼트에 아무런 정보가 없는 상태를 말한다.

table은 관계형 데이터베이스의 테이블로 매핑 하기 위해서 각각의 정보를 분리하여 저장한 것이다. 파싱하는 동안 메모리에 이와 같은 정보를 보관하고 있다가, XML 뷰와 XPath가 매핑이 이루어져서 새로운 New XML 뷰가 만들어지면, SQL로 변환할 때 이를 참조한다.

filter는 [$PS/nid = $NS/nid]이라고 되어 있다. 즉, PERSON 테이블의 nid와 NAME 테이블의 nid가 같은 조건을 나타낸다.

select는 SQL의 Select절에 해당하며, pathinfo는 사용자 XPath와의 매핑을 위해서 필요하다.

[그림 9]는 XML 뷰에 XPath를 매핑처리 하는 과정을 나타내고 있다. 이와 같은 결합 구조는 elementName, ruleType, elementContent, table, filter, select, pathInfo, SQL, Rule Type Name으로 구성된다. XML 뷰와 일치하는 부분은 이 결합 구조에서도 같은 역할을 한다. [그림 9]에서 family 엘리먼트 아래에 있는 XPath 질의를 가지고 XML 뷰에 매핑시킨다. 일치하는 필터, 테이블을 결합하여 family 엘리먼트 내부를 테이블 구조로서 표현한 것이다.



[그림 9] XML 뷰와 XPath의 결합구조

[Fig. 9] Composition Structure of XML view & XPath

여기서, SQL은 해당 엘리먼트 내부에 룰(table, filter, select, rule type)을 다음과 같이 SQL 질의로 변환한 후에 SQL 변수에 저장한다.

[그림 10]은 XML 뷰에 대한 내부 표현 저장 알고리즘으로, XML 뷰를 읽어서, elementName, Var, Value, textContent, tableName, filterName, Select Name, pathInfo로 분리한다.

[그림 9]의 결합 구조에 있는 변수에 대한 정보를 VarInfo 변수에 저장한 다음 [그림 9]의 결합구조의 XML→SQL 변환 알고리즘에서 이용한다.

[4]를 참조한 결과, XPath가 XQL과 유사하므로, [그림 9]의 RuleType Name은 다음과 같은 형태로 질의를 분류할 수 있었다.

가. Select Children and Descendant[SC]

특정 엘리먼트의 상대 패스나 절대패스 상에 존재하는 자식 엘리먼트, 자손 엘리먼트 또는 임의의 깊이를 갖는 엘리먼트를 선택하기 위한 질의로 주로 이용된다.

나. Filters[FI]

link[manager]와 같은 형태의 질의로서, link 엘리먼트의 자식인 manager 엘리먼트의 인스턴스가 존재하는가를 질의한다.

다. Equivalence[EQ]

name[family = "Jhonson"]과 같은 형태의 질의로서, name 엘리먼트의 자식인 family 엘리먼트가 Jhonson인 것이 있으면 name을 보이라는 것이다. 여기에는 =, !=와 같은 조건은 물론 본 논문에서는 대소비교와 연산까지를 포함시켰다.

```
XML뷰를 메모리에 읽어 구조화()
{
    r2xClause = ReadViewFile()

    for (i=0; i<ubound(rxinfoValue) {
     (elementName, Var, Value, textContent,
     tableName, filterName, SelectName,
        pathInfo ) = decompose(r2xClause)

     If (isEmpty(Var) <> true) {
        VarInfo(i,0) = Var;
        VarInfo(i,1) = Value;
     }
        r2xinfoValue(i).elementName,        textContent,
tableName, selectName, pathInfo =
                elementName, textContent, tableName,
selectName, pathInfo
        }
  }
```

[그림 10] XML 뷰에 대한 내부표현 저장 알고리즘

[Fig. 10] Storage Algorithm for internal expression of XML view

기타 분리된 정보들은 내부 표현에 저장된 후, XML 뷰와 XPath를 매핑할 때 저장된 알고리즘 정보를 이용하는데 사용한다.

[그림 11]은 XML 뷰에 XPath를 매핑하는 알고리즘이다. 먼저 XPath 질의를 읽어들인 다음 이를 elementName과 filterInfo로 분리한다. 만일 질의가 [그림 5]와 같다면, elementName 변수에는 manager가 filterInfo 변수에는 [.//name/family="Pfeiffer"]가 할당된다.

[그림 9]에 저장하기 위해서, table, filter, select ... 등으로 각각 분리하여 저장한 질의 내용을 통하여 질의 형태를 3가지 유형별로 분류하였고, 유형은 XPath 질의 특성으로 질의를 분석하였다. 이러한 특성을 바탕으로 XML 뷰와 XPath 질의를 매핑하여 새로운 XML 뷰를 내부 표현으로 저장하였다가 SQL로 변환할 때 이를 이용하는 알고리즘을 사용하였다.

```
XML뷰 결합 처리()

{xpathClause=ReadxpathFile(),
 (elementName,filterInfo)=decompose(xpathClause)
rule = RuleType(elementName, FilterInfo)
 if (rule = 'select children' or 'select descendants') {
   /* .//name */
   FindElement(elementName)
   /* rxinfoValue을 참조하여 해당하는 엘리먼트
      정보를 가져온다. */
   r2xinfo(i).elementName = elementName,
   elementContent, pathInfo
   r2xinfo(i).tableName(0) = tableName,
   r2xinfo(i).filterInfo(0) = filterInfo,
   r2xinfo(i).selectInfo(0) = selectInfo,
   i++
   /* rxinfoValue에서 엘리먼트정보를 찾은 다음
      해당 정보를 r2xinfo에 넣는다 */
  (r2xinfo(i).elementName,selectInfo,=rewrite(elementName)
```

```
}else if (rule = 'filters') {
   /* .//name[family] */
   FindElement(elementName, filterInfo)
   r2xinfo(i)=elementName,elementContent,pathInfo
   r2xinfo(i).tableName(j) = tableName,
   r2xinfo(i).filterInfo(j) = filterInfo,
   r2xinfo(i).selectInfo(j) = selectInfo,
   j++, i++

}else if(rule = 'Equivalence') {
   /* .//name[family = 'Johnson'] */
   FindElement(elementName, filterInfo)
   r2xinfo(i)=elementName,elementContent,pathInfo,operator
   r2xinfo(i).tableName(j) = tableName,
   r2xinfo(i).filterInfo(j) = filterInfo,
   r2xinfo(i).selectInfo(j) = selectInfo,
   j++, i++
}
```

[그림 11] XML 뷰를 XPath에 매핑하기 위한 알고리즘

[Fig. 11] Algorithm for mapping XML view on XPath

[그림 12]는 XML 뷰를 SQL로 변환하는 알고리즘으로서 [그림 11]의 알고리즘과 유사하다.

[그림 12]는 XML 뷰와 XPath를 매핑하여 이미 특정 엘리먼트가 어떤 유형인지와 SQL로 변환하기 위한 정보를 내부적으로 포함하고 있다. 이러한 정보는 바로 [그림 11]의 알고리즘으로 변환시킬 수 있다.

즉, 엘리먼트의 유형을 알고 있기 때문에 유형이 "select children" 혹은 "select descendant"이면 SQL로 "select ~ from ~ where ~" 형태로 변환된다.

왜냐하면, 이미 내부 정보로 select, table, filter 변수에는 XPath와 매핑이 이루어진 다음에 이와 같은 정보를 내부구조 정보로 보관하고 있기 때문이다.

두 번째 유형인 "filter"의 처리는 특정 엘리먼트 하부에 존재하는 엘리먼트의 인스턴스가 적어도 하나 존재하느냐는 질의이므로, 이는 SQL로 "select ~ from ~ where ~ and exists(select ~ from ~ where ~)"의 형태로 변환시킬 수 있다.

세 번째 유형인 "Equivalence"의 처리는 모든 필터들을 "and"로 연결하는 형태로서, SQL로 "select ~ from ~ where ~ and ~ and ~ …"로 변환시킬 수 있다.

[그림 10], [그림 11], [그림 12]의 알고리즘에 의해서 관계 데이터베이스의 내용을 XML로 변환시킬 수 있다. 이 방법은 질의를 통해서 데이터베이스의 일부 내용을 XML로 변환시키는 특성을 갖고 있다.

```
generlateSQL()
{
      for (i=0; i<ubound(r2xinfo()) {
         if (rule = 'select children' or 'select
         descendants')
      {
            /* author/first-name */
         r2xInfo(i).sqlUserQuery=select ~ from ~
         where ~
         }else if (rule = 'filters') {
            /* book[excerpt] , book[excerpt][title] */
         r2xInfo(i).sqlUserQuery=select ~ from ~
         where ~ and exists(select ~ from ~ where ~)
         }else if(rule = 'Equivalence') {
         /* author[last-name = 'bob'] , degree[from
         != 'Harvard'] */
         r2xInfo(i).sqlUserQuery=select ~ from ~
         where ~ and ~ and ~
      }
}
```

[그림 12] New XML 뷰→SQL로 변환하기 위한 알고리즘

[Fig. 12] Algorithm for transforming New XML view to SQL

5. 결론 및 향후 계획

XML은 이미 웹 상에서 상업적인 데이터 교환을 위한 표준으로 정착되었다. 대부분의 기관들은 XML 문서가 EDI의 형태로서 각종 문서교환용으로 이용할 수 있기를 원하지만, 불행하게도 아직까지는 기존의 상업적인 데이터는 대부분 관계형 데이터베이스에 저장되어 있다.

따라서, 이들 기존 관계형 데이터베이스에 저장되어 있는 데이터를 XML 문서 형태로 변환해서 문서 교환에 이용할 필요가 있다.

관계형 데이터베이스의 데이터를 XML로 변환하기 위해서는 관계형 데이터베이스 테이블의 각 필드를 XML로 매핑하여 XML 문서로 출판할 수도 있겠지만, 두 개 이상의 데이터베이스 테이블을 결합해서 하나의 XML 문서를 만드는 경우, DTD와 연관된 단순 매핑 만으로는 문제를 해결하지 못하므로 조인(Join)을 실행해야만 한다. 본 논문에서는 조인을 통해서 생성된 엘리먼트들이 보이도록 XML을 위한 뷰를 만들고, 이 뷰를 이용하여 관계형 데이터베이스의 내용을 XML로 변환하기 위한 알고리즘을 제안한

후, 이를 구현하였다.

향후 본 논문에서 제안된 방법을 통해서, XPath 질의로서 간단히 데이터베이스 문서를 XML로 변환한다면, 향후 XML 형태의 빈번한 문서 교환을 위해서나 웹 상에서 유용한 문서의 검색처리를 위한 방법으로서 매우 의미가 있을 것으로 생각한다.

※참고문헌

[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] 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-q-19980819/

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

[5] J. Clark. XSL Transformation(XSLT) specification, 1999, http://www.w3c.org/TR/WD-xslt.

[6] B. Ludaescher, Y. Papakonstantinou, P. Velikhov, and V. Vianu. View definition and DTD inference for xml., In Workshop on semistructured Data and Nonstandard Data Formats, January 1999.

[7] J.McHugh and J. Widom. Query optimization for XML. In Proceedings of VLDB, Edinburgh, UK, September 1999.

[8]. http://www.w3.org/XML

[9] James Clark, Steve DeRose, "XML Path Language(XPath) Version1.0", http://www.w3.org/TR/xpath

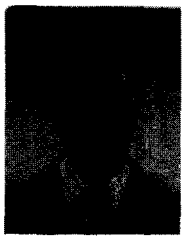
[10] 이형문,http://www.xmlab.com/xml_course/xsl

[11] http://myhome.naver.com/yaguza/xml_file/xhapter9.htm

김 노 환

1978년 숭실대학교 전자공학과 (공학사)
1985년 연세대학교 대학원 (공학석사)
2000년 강원대학교 대학원 박사과정 수료
1980년 금성전기(주) 기술연구소
1983년 현대전자산업(주) 시스템 연구소
1993년~현재: 동우대학 인터넷통신과 조교수
관심분야 : XML, 인터넷응용

정 충 교



서울대학교 전기공학과 졸업 (공학사)
한국과학기술원(KAIST) 졸업 (공학석사)
한국과학기술원(KAIST) 졸업 (공학박사)
1989년 LG정보통신(주) 책임연구원
1995년~현재 : 강원대학교 부교수
관심분야 : network(TCP/IP, ATM)