# 관계형 DB에서 Local Order 방식을 이용한 XQuery 변환기 설계 및 구현

이명숙*, 홍동권*, 손유익*
*계명대학교 컴퓨터공학과
e-mail:mslee@kmu.ac.kr

# XQuery transformer Using Local Order in Relational DB

Myung-Suk Lee*, Dong-Kweon Hong*, You-Ek Son*
*Dept of Computer Engineering, Kei-myung University

요 약

관계형 데이터베이스(RDBMS)의 내용을 XML로 표현하게 되면 지금까지 축적되어온 방대한 비즈니스 데이터들을 지속적으로 이용할 수 있게 된다. 다양한 형태로 저장된 XML 문서에서 원하는 데이터를 추출하고 변환하는 작업을 위해 W3C에서는 XPath를 핵심으로 하는 XQuery를 새로운 질의어 표준으로 발표하였다. RDBMS에 XQuery 질의어로는 구조가 다르기 때문에 질의어를 사용할 수가 없다. 따라서 본 논문에서는 XQuery 질의어로 RDBMS에서 검색이 가능하도록 XQuery 질의어를 SQL문장으로 변환시켜 검색할 수 있는 변환기를 설계하였다. 이 변환기는 복잡한 Numbering 기법을 사용하지 않고 child-parent relationship만 사용하므로 효율적인 데이터의 업데이트를 처리할 수 있다.

## 1. 서론

최근 관계형 데이터베이스를 이용한 XML 문서의 저장 및 질의 처리와 관련한 연구가 활발히 진행되어 왔다. XML은 확장성과 문서 관계성이 좋기 때문에 새로운 정보 공유 환경의 표준 매체로 자리 잡아가고 있다. 다양한 형태로 저장된 XML 문서에서 원하는 데이터를 추출하고 변환하는 작업을 위해 일반적으로 질의어를 사용하게 되는데 W3C에서는 XML 문서에 대한 질의어로서 구조적 질의어인 XPath를 핵심으로 하는 XQuery를 새로운 질의어의 표준으로 발표하였다.[1,2]

그러나 반구조적 문서로 표시하는 XML 문서와 구조적으로 표시하는 관계형 데이터베이스는 구조적으로 일치하지 않기 때문에 XML 문서를 관계형 데이터베이스에 저장 및 질의 처리를 하기에는 많은 어려움이 존재한다. 따라서 XML 문서를 관계형 데이터베이스에 저장하고 질의를 처리하기 위해서는 추가적인 처리과정이 필요하다.

관계형 데이터베이스의 내용을 XML로 표현하게 되면 지금까지 축적되어온 방대한 비즈니스 데이터들을 지속적으로 이용할 수 있게 된다. 따라서 관계형 데이터베이스 시스템에 XQuery로 질의를 하고자 할 때 XML 문서에서 필요한 정보를 검색할 수 있도록 하기 위해서는 XQuery를 SQL로 변환하는 연구가 필요하다.

본 논문에서는 관계형 데이터베이스에 저장된 문서에 질의를 하기 위해 XQuery를 사용하게 되면, 관계형 데이터베이스의 질의어인 SQL로 바꾸어 관계형 데이터베이스에 질의를 한 후 결과를 화면에 보여주는 기능을 가진 변환기 프로그램을 설계 및 구현하였다. XQuery를 SQL로 바꾸는 프로그램을 이하 '변환기'로 하며, 변환기는 자바 언어로 구현이 되어있다. XQuery 구문과 어휘를 분석하는 Lexer와 Parser는 eXist 프로그램의 한 부분을 임의로 수정하여 사용하였다.

본 논문의 구성은 다음과 같다. 2장에서는 XPath, XQuery 및 Order Encodeing 방법들에 대해 알아보고, 3장에서는 변환기 설계, 4장에서는 XQuery 질의어를 SQL로 변환, 마지막으로 5장에서는 결론 및 향후 연구 방향을 기술한다.

※ 본 연구는 한국과학재단 목적기초연구 (R01-2003-000-10001-0)지원으로 수행되었음.

## 2. 관련 연구

### 2.1 XPath

XPath는 W3C에서 버전1.0으로 권고하고 있으며[1] 문서에서 노드의 하위 집합을 선택하기 위해 단순한 구문을 제공하는 XML에 대해 정의된 질의어이다. XPath는 URL경로 표기법을 사용하여 XML 문서의 계층적인 구조를 논리적으로 탐색하는데 이 구조는 요소노드(element node), 속성 노드(attribute node), 값 노드(text node)를 사용하여 트리구조로 형성된다. XPath는 노드의 경로를 나타내기 위해 위치 경로(location path)를 사용하는데, 경로 위치 표현 방식에 있어서는 루트 엘리먼트로부터 시작되는 절대경로 방식과 사용자에 의해서 정해진 엘리먼트 또는 컨텍스트 엘리먼트에 의해서 결정되는 상대경로 방식[3]이 있다.

위치 경로는 축약된 문법과 축약되지 않은 문법을 이용해 정의할 수 있고, 본 논문에서는 축약된 문법을 대부분 사용하고 있다. 축약된 문법은 일반적으로 많이 쓰이고 있는 매우 편리한 형식이다. 그러나 좀 더 세밀한 검색 조건이 필요한 특별한 상황에서는 반드시 축약되지 않은 형식을 사용해야 한다. 표 2-1은 XPath의 축약된 표기식을 나타내고 있다.

표 2-1. XPath의 축약된 표기식

| 축약되지 않은 문법 | 축약된 문법 |
|---|---|
| child:: | / |
| attribute:: | @ |
| /descendant-or-self::node() | // |
| parent::node() | .. |

### 2.2 XQuery

XQuery는 XML 질의를 위한 새로운 W3C 표준으로서 W3C XML Query Working Group를 통해 정의되었다.[2] W3C XML Query Working Group이 가장 최근 발표한 XQuery 표준은 아직 완성된 표준이 아니기 때문에 XQuery 표준은 현재에도 계속 발전하고 있다. XQuery는 XPath에 근거한 경로 표현과 For, Let, When, Return(FLWR)로 표현된다.

### 2.3 Order Encoding 방법들

XML 문서를 저장하고 질의하기 위해 XML 문서를 관계형 데이터베이스 시스템으로, 이러한 기반위에서 XML 질의를 SQL 질의로 바꾸는 연구가 많이 이루어져 왔다.[3] 많은 논문에서 관계 데이터베이스 시스템을 기반으로 XML 문서를 저장하고 검색할 수 있게 하기 위해서 넘버링 기법에 의한 XML 문서 저장 기법을 사용하였다. Order Encoding 방법을 채택한 넘버링 기법으로서는 각각의 노드에 번호를 할당하는 Global Order Encoding, 로컬 순서를 가지고 각각의 노드는 형제들 사이에 상대적 위치를 가지는 Local Order Encoding, 이전의 두 가지 방법을 복합적으로 사용하는 Dewey Order Encoding[4] 방법이 있다. 그림2-1은 이러한 Order Encoding 방법들을 나타내고 있다.

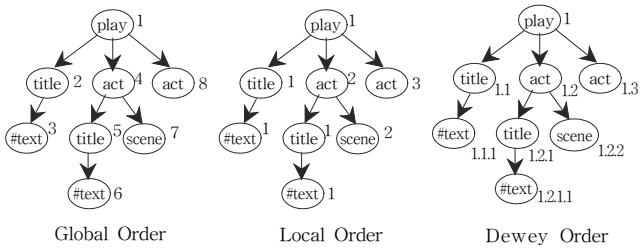



그림 2-1. Order Encoding 방법들

## 3. 변환기의 설계

### 3.1 변환기 구성도

본 절에서는 XQuery 질의어를 SQL로 변환하는 변환기 시스템을 그림3-2와 같이 설계하였다.

XML 문서는 Analyzer에 의해 미리 분석되어 관계형 데이터베이스의 테이블에 삽입이 되어 있다. XQuery 변환기는 XQuery를 입력 받으면 가장 먼저 Lexer로 XQuery 구문이 보내어 진다. Lexer에서 토큰을 생성하여 Parser로 토큰을 보내면 Parser에서 그 토큰을 이용하여 AST(Abstract Syntax Tree)를 만들어낸다. AST는 자식과 형제 노드를 가지는 트리로 괄호 형태로 표시된다. Parser에 의해 생성된 AST를 SQL Converter에서 순회하면서 XQuery를 SQL로 변환 하게 된다. 변환된 SQL을 가지고 RDBMS에 질의를 하고 결과는 CLOB Type의 XML 형태로 보여준다.

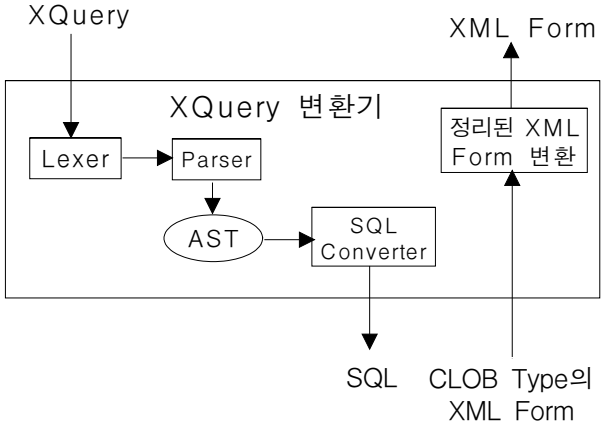



그림 3-1. XQuery to SQL 변환기 구성도

3.2 XQuery 구문 분석 및 파스 트리(AST)

XPath식(doc("books.xml")//book[title="Data on the Web"]//author[first="Serge"])을 통해서 AST의 형태를 살펴보면 표3-1과 같다. Parser를 통해 구문 분석 후 결과를 출력한 것이다. "|110|"과 같이 숫자로 되어 있는 부분은 토큰 번호를 나타낸다. 각 번호에 따른 토큰 타입이 미리 정의 되어 있다.

표 3-1. XPath식에 대한 AST

| doc("books.xml")//book[title="Data on the Web"]//author[first="Serge"] |
|---|
| ( //\|110\| ( //\|110\| ( doc\|12\| books.xml\|43\| ) book\|4\| ( Pred\|5\| ( =\|42\| title\|4\| Data on the Web\|43\| ) ) ) author\|4\| ( Pred\|5\| ( =\|42\| first\|4\| Serge\|43\| ) ) ) |

표3-1의 AST를 괄호로 표현된 것을 보기 쉽게 트리 형태로 그려 보면 그림3-2과 같다. 그림3-2에서 보듯이 AST는 자식, 형제 노드를 가지는 트리로 AST를 깊이 우선 탐색으로 처리한다. 가장 왼쪽의 단말노드를 처리한 후 형제 노드를 검색하고 처리하는 것을 반복해서 한다. 각각의 "Pred" 마다 서브 쿼리를 생성하고 두개의 쿼리가 생성되면 두 조건을 조인 한다. 그리고 그 다음 "Pred" 가 나올 때마다 이전에 생성된 SQL과 계속해서 같은 방식으로 조인을 해 나가면 된다. 이런 규칙을 통해 XPath를 SQL로 변환을 할 수가 있다.

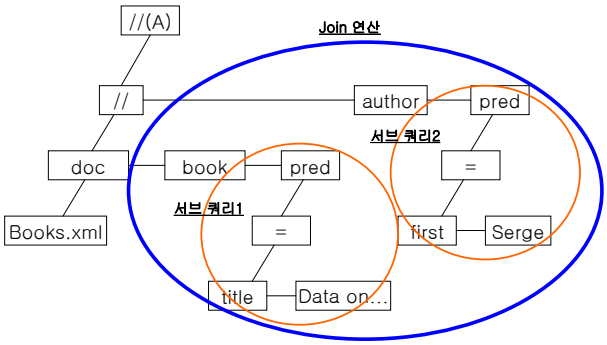



그림 3-2. SQL 변환 규칙

## 4. XQuery 질의어를 SQL로 변환

이 장에서는 그림3-2의 AST를 표4-1의 SQL 변환 알고리즘을 통해 SQL로 변환되는 과정을 설명하고 있다. 다음 표4-1은 XQuery를 SQL로 변환하는 알고리즘을 pudeo code로 간단히 나타내고 있다.

pathExpr 함수가 파싱된 AST를 받아서 160가지의 토큰타입에 따라 노드를 처리하고 나면 SQL 큐에 SELECT문이 저장되어 있다. makeSQL함수는 실제 SQL문을 출력하는 문인데 sequence를 두어 각각의 서브쿼리를 처리하고 조인하는 과정을 거치게 된다.

표 4-1. SQL 변환 알고리즘

```
pathExpr(AST ast)
{
    switch(ast.getType)          //노드의 type을 체크
       case PREDICATE
           predicate(ast.getfirstchild())
                        //predicate함수를 호출하여 pred처리
              SQL문 생성, 큐에 저장
       case FUNCTION
           functionCall(ast)    // functionCall 함수를 호출
              찾아올 문서의 SQL문 생성, 큐에 저장
       case LITERAL_or
              or처리
              SQL 생성, 큐에 저장
       case LITERAL_and      // or과 같음
       case SLASH
              pathExpr(ast.getFirstChild())
                 //'/'의 자식을 pathExpr로 넘겨서 다시 반복
                  if(ast.getNextSibling() != null)
                     pathExpr(ast.getNextSibling());
       case DSLASH                          // SLASH와 같음
       case   QNAME
       case XML_PI;
         ⋮
         ⋮     //    160개의    XQueryParserTokenTypes이
                 case문으로 되어 있다.
}

makeSQL( )
{
sequence=1
size=SQLQueue.size( )
if size==1
  // 하나의 elect문 생성, sub_query가 없는 상태
else
    while(!SQLQueue.isEmpty( ))
    {
       if(sequence==1)
          // pred1(가장 내부에 있는)을 위한 SELECT문 생성
       else if(sequence==size)
            // 조인을 위한 SELECT문 생성
       else
             // pred2를 위한 SELECT문 생성
      sequence++
    }
}
```

표4-2은 표3-1의 예제 식에 대한 SQL문 결과를 보여주고 있다. 서브 쿼리 1은 조건을 만족하는 경로 "//book/title"을 찾아서 그 경로의 조상인 "//book" 노드(AST를 보면 조건식을 만족하는 결과 노드)를 찾아서 반환하는 SQL문을 작성한다. 서브 쿼리1에서 조건의 조상 노드인 "//book" 노드를 반환하는 이유는 그 다음 조건과의 조인을 위해서이다.

서브 쿼리2도 1과 마찬가지로 조건을 만족하는 경로 "//book/author/first"를 찾아서 그 경로의 조

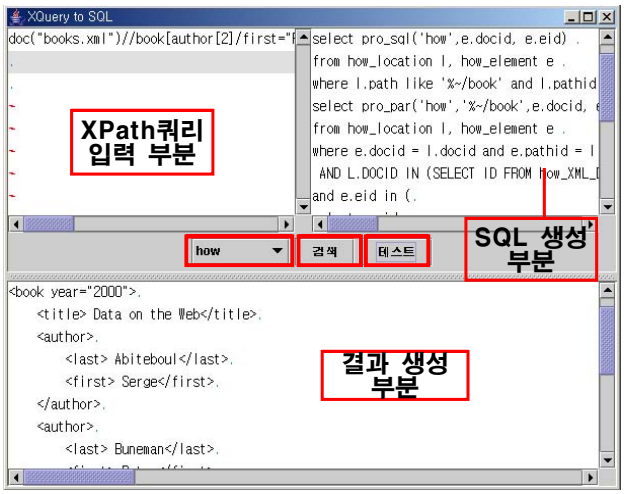
상인 "//book/author" 노드를 찾아서 반환한다. 이 두개의 서브 쿼리가 생성되면 이 두 조건을 하나로 만들어주는 조인연산을 수행 한다.

표 4-2. 생성된 SQL

```
SELECT Pro_sql('how', e.docid, e.eid)
FROM how_location l, how_element e
// 서브 쿼리 1과 2를 조인하기 위한 쿼리
WHERE l.path LIKE '%~/book~/author'
        AND l.pathid = e.pathid and l.docid = e.docid
        AND e.docid IN ( SELECT id
                         FROM   how_xml_documents
                         WHERE docname = 'books.xml')
        AND e.eid IN (
// 서브 쿼리 2
            SELECT Pro_par('how','%~/book~/author',e1.docid, e1.eid)
             FROM how_location l1, how_element e1
             WHERE l1.path LIKE '%~/book~/author~/first'
                    AND l1.pathid = e1.pathid and l1.docid = e1.docid
                    AND e1.value LIKE 'Serge'
                    AND e1.docid IN (SELECT id
                                     FROM   how_xml_documents
                                    WHERE docname = 'books.xml')
                   )
        AND  Pro_par('how','%~/book', e.docid, e.eid)
        IN (
// 서브 쿼리 1
            SELECT Pro_par('how','%~/book', e2.docid, e2.eid)
            FROM   how_location l2, how_element e2
            WHERE l2.path LIKE '%~/book~/title'
                    AND l2.pathid = e2.pathid and l2.docid = e2.docid
                    AND e2.value LIKE 'Data on the Web'
                    AND e2.docid IN ( SELECT id
                                      FROM how_xml_documents
                                     WHERE docname = 'books.xml')
          )
```

그림4-1은 프로그램 실행 화면을 보여주고 있다. XPath 쿼리 입력 부분에 XPath 식을 입력 한 후 컬랙션을 선택하고 검색을 누른다. SQL 생성 부분에 XPath 식에 대한 SQL문이 생성되어 나타난다. 텍스트를 클릭하면 SQL 문이 실행되어 아래쪽 결과 생성 부분에 그 결과가 나타난다.

그림 4-1. 실행화면




## 5. 결 론

계층적인 구조를 가지는 XML은 평면 구조를 가지는 관계데이터베이스와는 구조가 다르기 때문에 관계데이터베이스에 XML 문서를 저장 및 질의 처리하기 위해서는 별도의 처리과정이 필요하다. 이에 따라 넘버링과 같은 XQuery를 SQL 질의로 변환할 수 있는 방법들이 연구되었다. 그러나 이 방법은 삽입, 삭제 시 재 넘버링 해야 한다는 단점과 오버헤드가 증가하고 별도의 경로를 저장해야하는 공간이 필요한 단점을 가지고 있었다.

따라서 본 논문에서는 삽입, 삭제 시 노드를 재구성 하지 않고 부모노드와 자식노드를 찾을 수 있는 구조로 XQuery를 SQL로 변환하여 질의할 수 있도록 설계 하였다. 테이블의 데이터 저장방법은 XFTS[5] 기법을 적용하였다. XFTS 기법의 장점인 삽입과 삭제의 용이함을 잘 이용하기 위해 관계형 데이터베이스에 정의되어 있는 테이블 구조를 변환하지 않고 SQL 문을 완성 하였다. 그러나 결과 SQL 문이 복잡해지게 되는 단점을 가지고 있다.

복잡한 SQL 구문과 프로시저의 사용으로 인해 성능은 Numbering 기법에 비해 다소 떨어질지 모르지만 내용의 변화가 많은 XML 문서에서는 이 기법이 Numbering 기법에 비해서는 더 효율성이 높은 장점이 있다. 향후 두 가지 방법의 장점을 모아 문서의 삽입, 삭제와 질의 처리시의 성능도 고려한 최적화에 대한 연구가 필요할 것이다.

## 참고문헌

[1] J. Clark and S. DeRose, "XPath 1.0: XML Path Language," W3C Recommendation, http://www.w3.org/tr/XPath, 1999.

[2] D. Chamberlin, et al., "XQuery 1.0: An XML Query Language," W3C Working Draft, http://www.w3.org/TR/2001/WD-xquery-20010607, 2001.

[3 P. Bohannon, J. Freire, P. Roy, J. Simeon, "From XML Schema to Relations: A Cost-based Approach to XML Storage", ICDE 2002.

[4] Igor Tatarinov, Stratis D. Viglas, et, "Storing and Querying Ordered XML Using a Relational Database System", ACM *SIGMOD*, 2002.

[5] 천윤우, 홍동권, " 관계형 모델에서 XML 변경과 전문 검색을 지원하기 위한 역 인덱스 구축 기법," 한국정보처리학회 논문지, Vol. 11, No. 03, pp. 0509~0518, 2004.