# 관계형 데이터베이스 환경에서의 XQuery Processor 설계 및 구현

정민경, 홍동권
계명대학교 정보통신대학 컴퓨터 공학과
e-mail: skallet@kmu.ac.kr

# Design and Implementation of XQuery processor using Relational Technologies

Min-Kyoung Jung, Dong-Kweon Hong
Kei-Myoung University

요 약

XML이 발표되면서 대용량의 XML을 효과적으로 관리하는 여러 가지 방법들이 연구되고 있다. 특히 지금까지 상업적, 기술적으로 성공적이고 안정된 데이터 모델인 관계형 데이터베이스를 활용하는 여러 가지 방법들이 연구되고 있다. 본 논문은 관계형 DBMS를 사용하여 XML 질의어인 XQuery를 SQL로 변환하여 처리하는 효율적인 방법을 제안한다. 우선 본 논문에서 제안하는 방식은 XML문서를 분할하여 관계형 테이블에 저장하는 분할방식을 사용하며, 분할된 관계형 테이블을 이용하여 XPath를 포함한 XQuery의 기능을 실행하는 SQL을 생성하여 관계형 DBMS에서 SQL을 실행하는 방식을 사용한다. 제안한 XQuery 처리방식은 먼저 XQuery의 구문 분석을 통하여 AST(Abstract Syntax Tree)를 생성하고, AST를 순회하면서 SQL문장을 생성한다. 생성된 SQL문장은 XML 문서의 경로를 사용함으로써 XQuery 연산의 조인 횟수를 감소시키며, 각 노드마다 부여된 순서 정보를 효과적으로 사용하여 문서의 원래 순서에 맞는 XML 부분을 생성하는 방법을 제시한다. 그리고 실제 제안된 시스템을 개발하여 그 성능을 평가한다.

## 1. 서론

W3C(World Wide Web Consortium)에 의해 제안된 XML(Extensible Mark up Language) 이 문서교환의 표준으로 지정된 후 인터넷, 음악, 과학, 디지털 도서관 등과 같은 매우 다양한 분야에서 활용되고 있다. 이렇듯 XML의 폭넓은 활용으로 인해 다양한 형태의 XML문서를 효율적으로 저장, 검색, 색인하기 위한 XQuery Processing System의 연구가 필요하게 되었다. 이러한 연구를 바탕으로 eXist, X-Hive, Galax등과 같은 XML전용 데이터베이스가 많이 생겨나게 되었으며 기존의 상용 RDBMS인 경우 오라클에서는 XML을 저장할 수 있는 XML Type과 오라클 10g에와서는 XML전문 검색어인 XPath를 포함한 XQuery구문, 다양한 XPath 함수까지 처리할 수 있는 기능을 지원하고 있다.

하지만 위의 시스템 모두 특정 연구 분야 대해 설계 된 것이 아니라 일반적인 범위 내에서 XML을 저장하고 질의를 처리할 수 있도록 구현된 것이므로 연구 특성에 맞게 이용자가 특수한 기능을 추가한다거나 필요에 맞게 수정할 수 없으며 현재 시스템에서 제공하는 기능에만 의존해야 한다.

따라서 본 논문에서는 관계형 데이터베이스를 이용하여 XML Query Processor를 설계 및 구현하고 이에 특수한 여러 기능들을 추가 및 수정할 수 있게 함으로써 FullText 검색뿐만 아니라 영상 처리 혹은 웹 서비시스와 같은 특수한 분야에서도 융통성 있게 응용될 수 있도록 그 활용성을 높이기 위함이다. 그리고 질의를 처리하는 면에 있어서는 지금까지 논란이 되었던 새로운 노드의 삽입으로 인한 불필요한 갱신현상과 중간 결과값의 복잡한 표현과정을 줄임으로써 본 연구와 유사한 다른 연구 방법들보다 훨씬 효율적인 XQuery 처리 알고리즘을 제안하고 이를 직접 구현하여 그 기능을 평가한다.

본 논문의 구성은 다음과 같다. 2장에서는 XML

*본연구는 한국과학재단 목적기초연구 (R01-2003-000-10001-0)지원으로 수행되었음

문서를 관계형 테이블에 사상하기 위한 XML 색인 테이블을 설계하고 3장에서는 새로운 Dewey_order 인코딩 기법을 제시한다. 4장에서는 2장에서 제시한 XML 색인 테이블을 바탕으로 XQuery식을 SQL로 변환하는 알고리즘을 다룬다. 5장에서는 본 연구에서 구현한 시스템의 성능을 평가하고 마지막으로 결론 및 향후 과제를 제시한다.

## 2. 인덱스 테이블의 설계

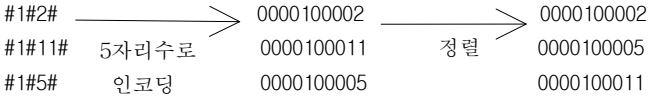
본 논문에서는 비슷한 내용의 XML 문서들을 모아 하나의 논리적인 컬렉션을 생성하여 관리한다. 이는 여러 개의 XML문서들을 쓰임새에 따라 일정하게 분류하여 효율적으로 관리하기 위함이다. 본 연구에서 제안하는 XML 색인 테이블은 다음과 같이 XML 요소들의 각 타입과 대응하는 5개의 table을 사용하며 이들 모두 컬렉션 단위로 존재한다.

Collection_Document(id, name)
Collection_Llocation(id, pathid, path, depth, path_cnt)
Collection_Word(id, position, depth, word, eid, pathid)
Collection_Attribute(id, eid, name, value)
Collection_Element(id,eid, name, pathid, info, value key_count ,sibord, pid, numbering)

위의 색인테이블들은 Dewey Order 기법을 이용한 XQuery 질의 처리기[6]와 동일한 구조를 가지며 그 내용 또한 동일하다.

## 3. Dewey_order 인코딩 기법

각 노드들을 XML 문서에 내재된 순서대로 정렬해야 할 경우 Dewey_order값을 기준으로 데이터를 비교하게 된다. 하지만 이를 사전식 순서대로 비교하여 데이터를 정렬하므로 부정확한 값이 출력된다. 따라서 본 연구에서는 각 노드들마다 부여된 Dewey order값을 0을 포함한 특정 자리수로 변환하고 이를 기준으로 올바른 순서대로 정렬한다.




본 연구에서 제시한 Dewey_order 인코딩 기법은 XQuery 식을 처리하거나 결과값을 XML형태로 재생성할 때 매우 유용하게 사용된다.

## 4. XQuery를 SQL로 처리하는 알고리즘

계층적인 질의 구조를 지닌 XQuery를 구조적인 관계형 언어로 처리하는 데는 여러 가지 문제점이 발생한다. 첫 번째로 XQuery를 실행할 때 발생되는 중간 결과값들을 관계형 환경에서 처리할 수 있어야 한다. 몇몇 연구에서는 중간 결과값이 발생할 때 마다 반복적으로 뷰를 생성함으로써 이를 해결하고 있다. 하지만 이렇게 다수의 뷰를 생성할 경우 최종 결과값을 질의할 때 현재까지 생성된 모든 SQL문들이 한꺼번에 중첩되어 실행되므로 질의 성능이 떨어지게 된다. 두 번째로 그동안 많은 논란이 되었던 엘리먼트의 constructor 처리이다. 만약 최종 결과 트리에 새로운 노드를 삽입하여 다른 형태의 XML fragment를 반환해야 할 경우 삽입이 발생된 지점부터 각 노드마다 부여된 순서정보들이 모두 변경되어야 하는 문제점이 발생한다. 따라서 본 연구에서는 이러한 문제점들을 다음과 같은 방법으로 해결한다. 첫 번째로 XQuery의 for, let, where, 혹은 XPath 함수들의 중간 결과값을 표현하기 위한 복잡한 과정을 모두 생략함으로써 쿼리의 성능을 높인다. 두 번째로 새로운 엘리먼트를 construction 할 경우 삽입할 노드명과 이와 결합될 기존의 노드명을 단순히 String 형태로 연결함으로써 삽입으로 인한 불필요한 갱신현상을 줄인다. 단 새로운 노드가 삽입될 정확한 위치를 찾기 위해 소수의 뷰를 형성한다. 만약 엘리먼트 constructor가 없는 XQuery일 경우 아무리 여러 개의 조건들과 함수들이 있더라도 본 기법에서는 중간 결과값들의 복잡한 표현 과정을 모두 생략하므로 최종 결과값들을 한번에 찾을 수 있다.

이러한 기법을 바탕으로 본 연구에서 개발한 XQuery Processor 시스템은 아래의 XQuery grammar를 모두 지원한다.

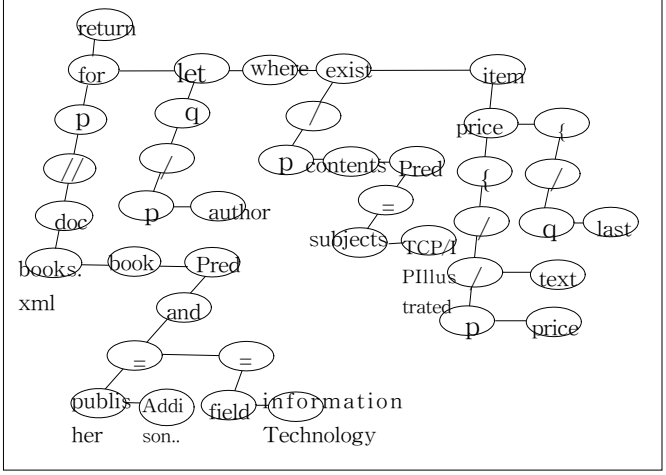
```
XQuery_Flower ::= For x in XPath_F
                | Let x := XPath_F
                | Where XPath_F
                | Return XML_Fragment_Construct
Axis ::= '/'
       | '//'
 x ::= variables

XPath_F ::= XPath                    Function ::= count()
          | x 'Axis' XPath_F                  | text()
          | XPath 'Function'                  | contains()
                                              | exist()
                                              | empty()
XML_Fragment_Construct ::= Complex_Construct
                         | '{' XPath_F '}'
Complex_Construct ::= <New_item> '{' XPath_F '}'
       | <New_item New_attr = "{XPath_F}"> '{' XPath_F '}'
```

[figure 2] XQuery grammar

*본연구는 한국과학재단 목적기초연구
(R01-2003-000-10001-0)지원으로 수행되었음

본 연구에서는 1중 For문의 XQuery는 6단계로 2중 for문인 경우 5단계로 또한 construction(결과 트리에 새로운 노드를 삽입)이 없는 구문인 경우 1개의 SQL문으로 모두 처리한다. 우선 주어진 XPath를 다음과 같은 AST (Abstract Syntax Tree)트리로 변환하고 이를 순회하면서 정확한 시점에 SQL변환 알고리즘을 적용한다. 아래의 예시는 1중 루프와 item노드의 construction을 포함하는 XQuery 구문이며 이를 AST 트리로 표현한 것이다.

```
 *출판사가 Addison-Wesley인 IT분야의  책을 검색하되 TCP/IP
Protocol과 관련된 책의 가격과 작가의 last 이름을 출력하라.
*for $p in doc('books.xml')//book[publisher='Addison-Wesley'
  and field =   'Information Technology']
 let $q := $p/author
 where exists($p/contents[subjects="TCP/IP Protocol"])
 return <item price = "{$p/price/text()}"> {$q/last} </item>
```




[figure 3] AST tree

본 시스템에서는 위와 같이 1개의 for문을 포함한 XQuery식을 아래의 알고리즘으로 처리한다.

XQuery To SQL(AST node)

1) AST 트리를 순회하되 constructor 노드(itme)의 attribute값으로 들어갈 엘리먼트 price를 만나면  현재까지 AST를 순회하면서 생성한 SQL문을 적절히 결합하여  attr_table 이라는 뷰를 생성한다. 단 관계형 테이터베이스(오라클의 경우)에서 기본적으로 제공되는 rownum값도 함께 추출한다.

attr_table(docid, eid, name, dewey_n, value, info, path, pathid, sibord, row_id)

2) constructor 노드(item)의 text값으로 들어갈 엘리먼트 last노드를 만날 경우 현재까지 AST를 순회하면서 생성한 SQL문을 적절히 결합하여 text_table이라는 뷰를 생성한다.

text_table(docid, eid, name, dewey_n, value, info, path, pathid, sibord)

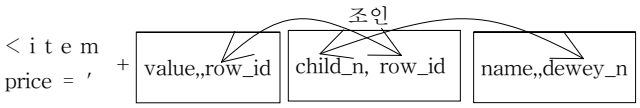
3) 2)과정을 수행했을 때는 AST 트리의 순회를 완료한 것이며 3)단계부터는 AST의 트리와는 상관없이 1중 루프를 포함한 XQuery FLWOR 식이라면 모두 일률적으로 거치게 되는 단계이다. 2)단계에서 생성한  text_table 뷰를 for문에 출현하는 노드(book엘리먼트)로 그룹핑하여 그중 최고 root노드만을 골라 group_view를 생성한다. 이때 book의 자식노드인 Publisher와 field 노드는 각각 'Addison-wesley'와 'Information Technology'라는 text를 소유하고 있어야 되며 각 그룹의 root노드를 추출할 때 fixed_digit라는 저장 프로시져를 통해 인코딩된 Dewey_n값을 검색하여 child_n라는 컬럼명으로 뷰를 생성한다.

group_view(child_n)

4) 3)과정에서 생성한 group_view에 rownum값과 인코딩된 Dewey_n값을 추출하여 group_seq라는 뷰를 생성한다. 이때 각각의 컬럼명을 row_id, child_n 이라한다.

group_seq(child_n, row_id)

5) 4)과정에서 생성한 group_seq라는 뷰의 child_numbering 컬럼값과 동일한 값을 가진 엘리먼트들을 text_table(item노드가 삽입될 위치)에서 그리고 row_id컬럼과 동일한 값을 가진 엘리먼트들을 attr_table(item노드의 attribute로 들어갈 값)에서 검색한 뒤 이들을 1:1 결합하여 concatenation_table이라는 뷰를 생성한다. 이때 새롭게 삽입될 item엘리먼트와 price 어트리뷰트의 문자열과 또한 name 컬럼의 문자열을 결합함으로써 새로운 노드의 삽입으로 인한 불필요한 갱신현상을 줄인다.




conctenation_table(t.name,t.dewey_n,t.value,,,t.sibord, a.row_id)

6) concatenation_table과 text_table을 결합하고 중복된 값을 제거하여 최종 결과뷰를 생성한다. 여기서 중복된 값이란 concatenation_table과 동일한 eid를 가진 text_table의 행들이다.

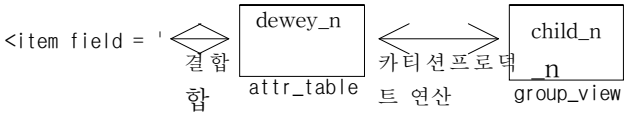
last_table(docid, eid, name, dewey_n, value, info)

지금까지 살펴본 내용과는 달리 2중 for문인 XQuery 경우 5단계로 나눠 처리한다. group_view를 생성하는 단계까지는 1중 for문의 처리방식과 동일하나 group_seq뷰를 생성하는 과정은 거치지 않고 concatenation_table뷰와 최종 last뷰를 생성한다. 이때 카티션 프로덕트 연산을 하여 처리한다는 점이 1중 for문의 XQuery 처리방식과 다르다.

nested XQuery To SQL(AST node)

1)~3) XQuery To SQL 알고리즘과 동일

4) group_view의 child_n값과 동일한 값을 가진 행을 text_table(item노드가 삽입될 위치)에서 검색한 뒤 attr_table의 행(item노드의 attribute로 들어갈 값)의 Dewey_n값을 인코딩한 값을 기준으로 카티션 프로덕트 연산을 하여 2중 for문을 수행한 것과 동일한 개수의 item노드를 생성한다.




conctenation_table(t.name, t.dewey_n, t.value,,,t.sibord, a.row_id)

5) item노드의 자식으로 출현할 서브트리들을 text_table로부터 검색하되 attr_table과 카티션 프로덕트 연산을 한다. 그리하여 4)과정에서 생성한 item노드와 동일한 개수의 서브트리를 생성하고 concatenation 테이블과 UNION All 연산을 한 뒤 최종 결과뷰를 생성한다. 단 아래의 최종 결과 뷰는 서로 동일한 Dewey_n값을 가지게 되는데 이때 row_id 컬럼을 기준으로 정렬함으로써 결과

*본연구는 한국과학재단 목적기초연구
(R01-2003-000-10001-0)지원으로 수행되었음

뷰를 XML 형태로 재생성할 때 XML문서에 내재된 순서대로 데이터를 로드할 수 있다.

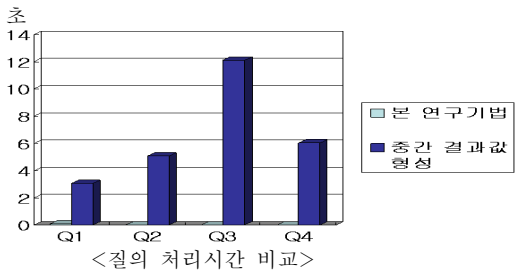
last_table(docid, eid, name, dewey_n, value, info, row_id)

위의 XQuery변환 알고리즘을 통해 알 수 있듯이 본 기법에서는 for, let, where절과 같은 불필요한 중간 결과값의 표현과정을 생략하고 return절에 해당하는 최종 결과값만을 질의함으로써 질의 성능을 높일 수 있다. 이로써 construction이 없는 XQuery 구문인 경우 1개의 SQL문으로 모두 처리 한다. 또한 새로운 엘리먼트의 constructor가 발생할 경우 해당 엘리먼트의 이름을 문자열로 결합함으로써 삽입으로 인한 복잡한 갱신 현상을 줄인다. 그 외에 XQuery의 for문에 해당하는 중간 결과값들을 한 번에 검색하여 질의 성능을 높이되 dewey order값으로 이들을 적절히 그룹핑함으로써 마치 for문을 실제로 수행한 것과 동일한 효과를 발휘한다.

## 5. 성능평가

본 XQuery Processor 시스템의 구현환경은 팬티엄 4 3.00GHz, 메인 메모리 512MB, window2000운영체제이며 JDK1.4.05와 JDBC, PL/SQL를 통해 구현하였다. 이 실험의 측정은 본 연구기법과 XQuery의 각 중간 결과값들을 뷰로써 표현하여 처리하는 기존의 연구기법[3]과의 질의 처리 시간을 비교한 것이다. 이에 사용된 XML문서는 9.58KB크기로 도서에 관한 정보를 담고 있다.

```
(Q1):for $p in doc('books.xml')//book
return <item title = "{$p/title/text()}"> {$p} </item>
(Q2):for $p in doc('books.xml')//book
where $p/@year = '2000'
return <item person "{$p//last/text()}"> {$p//last} </item>
(Q3):for $p in doc('books.xml')//book
let $q := $p/title[contains(text(),'Illustrated')]
where $p/@year = '1994'
return <item person = "{$p//last/text()}"> {$q} </item>
(Q4):for $p in doc('books.xml')//book[field = 'Information Technology']
let $q := $p/contents[subjects = 'XML Query Language']
where $q/notion[contains(text(), 'XQuery')]
return $q/homepage
```




<질의 처리시간 비교>

위의 그래프에서 보는 봐와 같이 본 연구에서 구현한 XQuery Processor의 시스템이 중간 결과값을 모두 형성하여 쿼리를 처리하는 기존의 방식[3]보다 훨씬 우수한 성능을 발휘한다.

## 6. 결론 및 향후과제

지금까지 본 연구에서는 XQuery FLWOR식을 관계형 언어로 효율적으로 처리할 수 있는 XQuery 변환 알고리즘을 제시하였다. 또한 이러한 알고리즘을 바탕으로 본 연구에서는 XQuery Processor를 실제로 구현하고 테스트함으로써 중간 결과값을 모두 형성하여 XQuery를 처리하는 기존의 방식들보다 훨씬 우수함을 증명하였다.

이렇듯 효율적인 XQuery 기술들을 더욱 더 연구하고 향후 특정 연구 분야에서 필요로 하는 특수한 기능들을 본 시스템에 추가 및 수정함으로써 융통성 있게 활용할 수 있는 방안이 계속 연구되어야 할 것이다.

## 참고문헌

[1]Yoshikawa, M., Amagasa, T, "XRel: path-based approach to storage and retrival of XML Documents using relation database" ACM Transactions on Internet Technology(TIOT)

[2]Igor Tatarinoy, Stratis D. Yiglas, Kevin B eyer, Javavel Shanmugasundaram, Eugene Shekita, Chun Zhang "Storing and Querying Orderd XML Using a Relational Database System" SIGMODE Conference 2002

[3]David Dehaan, David Toman, Mariano P. Consens, M.Tamer Ozsu "A Comprehensive XQuery to SQL Translation using Dynamic Interval Encoding" SIGMODE Conference 2003

[4]Torsten Grust, Sherif Sakr, Jens Teubner "XQuery on SQL HOSTS" VLDB Conference 2004

[5]Yang Chu, Liang-Tien Chia, and Sourva S. Bhowmick "SM3+: An XML Database Solution for the Management of MPEG-7 Descriptions" DEXA 2005

[6]정민경, 홍동권 "Dewey order 기법을 이용한 RDBMS 환경에서의 XQuery 질의 처리기 설계 및 구현" 정보처리학회 춘계학술발표대회 2005

*본연구는 한국과학재단 목적기초연구 (R01-2003-000-10001-0)지원으로 수행되었음