XQuery에서의 XML 데이터 특성을 고려한 group by 지원을 위한 질의 표현 기법에 대한 연구

이 민 수[†] · 조 혜 영[††] · 오 정 선[††] · 김 윤 미[†††] · 송 수 경[†††]

요 약

현재 널리 채택되고 있는 XML은 플랫폼에 의존하지 않는 데이터 표현 형식으로 B2B 응용 프로그램이나 워크플로우 상황에서처럼 느슨하게 연결된(loosely coupled) 이기종 시스템 간에 정보를 교환하는 데 매우 유용하게 사용되고 있다. XML의 이러한 장점 때문에 점차 증가하는 XML에 대한 관리 및 검색에 대한 요구 사항에 대처할 수 있도록 강력한 질의 언어인 XQuery가 만들어졌다. 문서의 검색을 위한 질의 언어인 XQuery는 다양한 데이터 소스로부터 가져온 XML 데이터를 고유한 구조를 가진 질의 결과로 구성할 수 있도록 설계되었으며 현재 XML 질의 언어의 표준이다.

XQuery는 반복문 등을 포함하는 강력한 검색 기능을 지원하나 데이터를 그룹화 하는 경우에는 질의 표현이 상대적으로 어렵고, 복잡한 형태를 취한다. 따라서 본 논문에서는 XQuery에 그룹화 처리를 위한 명시적인 groupby절을 도입한 질의 표현 기법을 모색함으로써 XML 데이터의 재구성과 집계 함수 처리를 위한 그룹화를 보다 효율적으로 처리할 수 있도록 하였다. 이를 위해서 XQuery에 groupby절을 도입하기 위한 EBNF(Extended Backus-Naur Form)를 제안하고, 네이티브 XML 데이터베이스인 eXist 기반의 XQuery 그룹화 질의 처리 시스템을 구현하였다.

키워드 : 그룹화, XQuery, XML 데이터베이스

Research on supporting the group by clause reflecting XML data characteristics in XQuery

Lee, Minsoo[†] · Cho, Hyeyoung[††] · Oh, Jung-Sun[††] · Kim, Yun-mi[†††] · Song, Soo-kyung[†††]

ABSTRACT

XML is the most popular platform-independent data expression which is used to communicate between loosely coupled heterogeneous systems such as B2B Applications or Workflow systems. The powerful query language XQuery has been developed to support diverse needs for querying XML documents. XQuery is designed to configure results from diverse data sources into a uniquely structured query result. Therefore, it became the standard for the XML query language.

Although the latest XQuery supports heavy search functions including iterations, the grouping mechanism for data is too primitive and makes the query expression difficult and complex. Therefore, this work is focused on supporting the groupby clause in the query expression to process XQuery grouping. We suggest it to be a more efficient way to process grouping for restructuring and aggregation functions on XML data. We propose an XQuery EBNF that includes the groupby clause and implemented an XQuery processing system with grouping functions based on the eXist Native XML Database.

Key Words : Grouping, XQuery, XML Database

1. 서 론

XML(eXtensible Markup Language)[1]은 구조적 또는 반구조적인 데이터를 저장 및 전송하기 위한 마크업 언어이다. 많은 종류의 XML 데이터 소스에서 폭넓게 적용할 수 있도록 만들어진 XQuery[2]라고 하는 질의 언어는 W3C XML Query Working Group XML Query 1.0 요구 사항에서 밝힌 내용 및 XML Query Use Cases의 사용 예에 맞게 만들어졌다. 이것은 질의를 간결하게 만들고 쉽게 이해할 수 있는 언어가 되도록 한 것이다.

복잡한 구조를 가지는 XML에서의 그룹화는 데이터를 재구성하거나 집계 함수를 처리하는 경우에 주로 이용된다. 하지만, 그룹화의 중요성이 더 큼에도 불구하고, 그룹화를 지정하고 구현하기가 쉽지 않다. XML문서의 일부 속성이나 하위 요소가 존재하지 않거나 하위 요소가 반복되는 등의

※ 이 논문은 2004년도 한국학술진흥재단의 지원에 의하여 연구되었음.(KRF-2004-041-D00572)
† 종신회원 : 이화여자대학교 컴퓨터학과 교수
†† 정 회 원 : 이화여자대학교 대학원 컴퓨터학과
††† 준 회 원 : 이화여자대학교 컴퓨터학과 석사과정
논문접수 : 2006년 3월 31일, 심사완료 : 2006년 6월 9일

구조적 유연성에 의한 영향을 많이 받기 때문이다. 따라서 이러한 XML 그룹화를 지원하기 위한 질의 언어의 개선과 질의 처리 기법의 연구가 필요하다.

XML 데이터 소스에서 폭넓게 적용할 수 있도록 만들어진 XQuery 질의 언어를 이용하여 XML 데이터의 재구성과 집계 함수 처리를 위해 기존의 FLWR(For-Let-Where-Return) 표현식에서 groupby절을 확장함으로써 그룹화를 쉽게 명시하도록 하였으며 eXist(An Open Source Native XML Database)[3] 기반으로 질의 처리기를 구현하였다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구로 groupby 질의를 지원하기 위한 XML 질의 처리 기술들에 대해 설명한다. 3장에서는 기존 XQuery 문법에 groupby 질의를 도입하기 위한 EBNF(Extended Backus-Naur Form)를 제안하고, 실제 XML 데이터에 groupby절을 적용시킨 질의를 소개한다. 4장에서는 구현된 시스템에 대한 이해를 돕기 위해 groupby 질의를 처리하기 위한 과정에 대해 설명한다. 5장에서는, eXist 기반의 XQuery 그룹화 질의 처리 시스템 구현에 대한 개발 환경과 groupby 질의를 처리하는 과정을 보여준다. 여기서 eXist XML 데이터베이스 시스템을 이용하였다. 6장에서는 groupby를 추가한 XQuery의 그룹화 모델링 능력을 비교하고 마지막으로 7장에서는 결론 및 향후 연구 방향을 제시한다.

2. 관련 연구

2.1. groupby 질의를 지원하기 위한 XML 질의 처리 기술

XML 데이터를 검색하기 위한 질의 언어로는 XPath[4], XQuery 등이 제안되었으며, 이러한 질의 언어들은 반구조적(semi-structured) 특징을 지니는 XML 데이터 구조를 탐색하기 위하여 경로 수식을 기반으로 하고 있다. XML과 같은 반구조적 데이터 모델은 기존의 관계형 데이터 모델이나 객체지향 데이터 모델과는 많은 차이점들이 있어서 기존의 데이터베이스 시스템에 저장하려면 XML 데이터를 다른 데이터 모델의 데이터로 변환하는 과정이 필요하게 된다. 물론 데이터를 검색할 때에도 질의를 변환하고 수행된 질의 결과를 다시 XML 형태로 변환하는 과정을 수행해야 한다. 이와 같은 작업에 따른 부담이 너무 크기 때문에, XML 전용 데이터베이스 시스템을 이용하여 XML 데이터를 저장, 관리하는 것이 가장 효율적이다. XML과 같은 반구조적 데이터 모델 기반의 데이터베이스 시스템과 관련된 연구로는 Lore 시스템, Timber 시스템, GApply를 들 수 있다.

Lore 시스템[5]은 반구조적 데이터베이스 시스템으로 원래 반구조적 데이터를 질의하기 위해 설계되어 점차 XML 데이터로 확장되었다. XML을 위한 선언적 질의 언어 개발, XML 데이터에 대해 새로운 검색 기술 개발, 효율적인 XML 질의 처리기 개발에 목적을 두고 있다. 타입 강제(type coercion)와 관련한 메커니즘을 포함하며 사전에 문서의 구조를 모르는 경우에 특히 유용한 강력한 표현식을 허용한다. Lore에서 쓰이는 질의언어는 Lorel로서 OQL과 매우 유사하며 명시적으로 groupby를 지원하지는 않으나 질의계획을 생성하는 과정에서 경로별 그룹화가 필요하여 내부적인 그룹화를 수행한다. 질의계획에서 사용되는 질의 연산자로는 관계형 연산자와 동일한 Scan, Join, Project, Select 이외에 SetOp, ArithOp, CreateSet, Groupby 등이 있다. 먼저 SetOp은 set 연산(union, intersect and except)을 처리하며, ArithOp은 수치 연산(addition, multiplication)을 처리한다. CreateSet는 임의의 서브 질의를 패키징하여 앞으로 사용할 수 있도록 하며, 마지막으로 Groupby는 경로별 그룹화를 처리한다.

Timber 시스템[6]에서는 XML 데이터는 원래의 트리 구조 그대로 저장하므로 관계형 데이터베이스 시스템[7, 8]에 저장할 때 발생하는 이질적인 데이터 모델간의 부조화 문제가 발생하지 않는다. XML 데이터를 기존의 객체지향 모델과 유사한 형태로 저장할 경우에, XPath 질의에 대하여 각각의 노드들을 하나씩 탐색하여 결과를 찾아내는 튜플 단위(tuple-at-a-time) 방식을 사용하기 때문에 질의 처리 속도가 크게 저하되었다. Timber 시스템은 질의 처리를 위하여 같은 이름을 갖는 엘리먼트들을 모아서 인덱스로 구성하고, 질의를 탐색할 때는 각각의 엘리먼트들의 집합을 조인하는 집합 단위(set-at-a-time) 방식을 취하여 성능을 향상시켰다. 또한 이러한 질의 처리방식에 효율성을 더하기 위하여 효율적인 선택율 추정법을 이용한 질의 계획 생성기를 제공하고 있다. Timber 시스템에서는 TAX[9, 10]라는 트리 대수(tree algebra)를 내부적으로 정의하여 잠재적으로 포함되는 그룹화 구성에 관한 내용을 지정하고, 이를 이용해 XQuery의 내포된 질의를 TAX의 그룹화 질의로 재구성하는 일련의 방식을 취하였다.

XML에서의 그룹화 처리를 위한 연산자를 제안하고, 기존 관계형 엔진에 고르게 통합될 수 있는 GApply 연산자를 통해 그룹화를 지원하는 연구가 있었다[11]. 관계 데이터베이스에 저장된 데이터를 XML 형태로 제공할 때 관계형 모델과 XML 모델의 차이를 보완하기 위해서는 튜플의 그룹을 가리키는 변수가 필요한데 이를 GApply 연산자를 통해서 제공한다. GApply(GCols,PGQ)라는 구문을 사용하여 GCols는 그룹핑하는 열을 표시하며 입력 튜플 흐름은 이에 따라 나누어지고 PGQ(per-group query)는 각 그룹에 적용되는 질의이다.

3. XQuery에서의 groupby절 확장

본 절에서는 현재 XQuery에서 지원되고 있지 않은 group by절을 도입한 그룹화 구성의 필요성에 대해서 알아본다. 또한 현재 XQuery에서 지원되고 있지 않은 groupby절을 도입한 구문과 group by절을 사용한 질의의 유형을 살펴본다.

3.1 Groupby절 도입의 의미와 필요성

기존의 내포된 형태의 XQuery를 이용하면 XML 데이터

```
FLWGRExpr  ::= (ForClause | LetClause)+ WhereClause?
                    groupbyClause? "return" ExprSingle
ForClause  ::= <"for" "$"> VarName TypeDeclaration?
                    PositionalVar? "in" ExprSingle ("," "$"
                    VarName TypeDeclaration? PositionalVar?
                    "in" ExprSingle)*
LetClause  ::= <"let" "$"> VarName TypeDeclaration? ":="
                    ExprSingle ("," "$" VarName
                    TypeDeclaration?":=" ExprSingle)*
WhereClause  ::= "where" ExprSingle
groupbyClause  ::= ("groupby") groupbySpecList
groupbySpecList  ::=  ( NgroupbySpec | SgroupbySpec )
SgroupbySpec ::= "[" groupbySpec ("," groupbySpec)* "]"
NgroupbySpec ::= groupbySpec ("," groupbySpec)*
groupbySpec  ::= ExprSingle
```

(그림 1) XQuery에 groupby절을 도입하기 위한 EBNF

에 대한 그룹화 질의를 작성할 수 있으나 매우 복잡한 형태의 질의가 된다. 본 연구에서는 XQuery에 groupby절을 추가함으로써 일부 XML 그룹화를 하는 복잡한 내포된 형태의 XQuery들에 대해서 동일한 의미를 갖는 질의를 쉽게 작성할 수 있도록 하고자 한다.

이미 사용되고 있는 SQL에서의 groupby절과 본 연구에서 도입하고자 하는 XQuery에서의 groupby절에 대하여 다음과 같이 비교할 수 있다. XQuery는 트리 기반의 질의 언어이고 SQL은 테이블 기반의 질의 언어로서 이들간에 groupby를 지원하는 의미는 약간의 차이가 있다. RDB에서의 groupby는 열의 값들에 대하여 그룹을 구성하여 테이블의 레코드들을 모으게 되는 반면에 XML에서의 그룹화는 엘리먼트들에 대한 그룹화를 하여 트리 구조에서 관련이 있는 서브트리들을 서로 같은 그룹으로 묶게 하는 효과가 있어서 구조를 함께 유지하면서 그룹화를 하게 된다. 그리고 그룹화를 하는 기준이 테이블의 열들이 아니라 특정 계층 구조를 따라서 여러 개의 엘리먼트들에 따라 그룹화를 하게 된다. 그리하여 테이블 기반의 SQL에서는 그룹화를 할 때 결과의 표현이 하나의 레벨로 표현되어서 비교적 간단하게 질의 결과를 보여줄 수 있으나 트리 기반의 기존의 XQuery에서는 그룹화를 트리의 계층구조에 따라 여러 레벨로 할 수 있어서 부득이하게 내포된 형태로 질의를 구성하게 되는 경우가 많다. 따라서, 여러 레벨에 걸쳐서 그룹화를 표현할 수 있도록 하게 하되 내포된 형식을 피하도록 XQuery에 groupby 절을 제공하게 되면 사용자는 훨씬 쉽게 질의를 작성할 수 있게 된다.

또한 XQuery는 for와 같이 순차적인 구문을 포함하는 질의어인데 반해 SQL의 비순차적인 모델을 따르는 groupby를 사용하는 경우 의미상의 문제가 없는지에 대하여 살펴볼 필요가 있다. XQuery에서 실제로 그룹화를 할 수 있는 방법은 매우 많다. 특히 for를 포함하는 절차적인 질의어 속성이 있어서 SQL에서는 지원하지 못하는 그룹화도 가능하다. 그러나 SQL에서는 쉽게 표현이 되는 그룹화를 기존의 XQuery에서 표현하려면 매우 복잡하게 된다. 그러나 이러한 그룹화들은 매우 잘 정의된 것으로 특정 엘리먼트를 기준으로 그룹화하는 등의 것으로 SQL에서의 groupby 의미를 가져와서 트리 구조에 적용하게 될 경우 매우 편하게 사용할 수 있고 의미적으로 간단하게 매핑이 된다. 다만 groupby를 XQuery에 추가한다고 해서 기존의 XQuery에서 표현할 수 있는 모든 형태의 그룹화를 지원하지는 못한다. 즉 일부 의미가 분명하고 잘 정의된 그룹화만이 groupby절에서 적용이 가능하게 되고 이들을 본 연구에서는 4가지 유형으로 나누어 제안하였다.

본 연구에서 기존 XQuery에 그룹화 키워드인 group by절을 도입하는 방안을 제안함으로써 다음과 같은 효과를 얻을 수 있을 것으로 기대된다.

첫째, SQL에서처럼 group by절을 사용하여 그룹화를 표현할 수 있게 된다. 현재 XQuery에서는 내포된 FLWR식(nested FLWR expression)과 조인을 통해 그룹화 처리를 하고 있어 질의 문장의 표현이 복잡하다.

둘째, 동일 레벨상의 컬럼에 대해서만 그룹화가 가능한 SQL과는 달리 임의의 레벨상의 바인딩 변수에 의한 그룹화가 가능하다. XML의 다양한 계층 구조를 고려할 때 레벨에 구애받지 않는 그룹화가 반드시 필요하다.

셋째, 내포된 FLWR식 안에 다시 내포된 FLWR식을 가지는 복잡한 계층 구조의 XML문서를 쉽게 생성할 수 있게 된다.

3.2 Groupby절을 도입한 XQuery 구문

기존 XQuery 문법에서 groupby절을 도입하여 (그림 1)과 같이 EBNF(Extended Backus-Naur Form)를 정의하여 추가하였다.

3.3 Groupby절을 이용한 4가지 질의 유형

(그림 2)는 예제 XML 데이터인 'bib.xml'을 보여주며 이 데이터를 이용해서 groupby절을 도입한 4가지 질의 유형은 다음과 같다.

```
<?xml version="1.0" encoding="utf-8"?>
<bib>
    <book year="1994">
        <title>TCP/IP Illustrated</title>
        <author>Stevens W.</author>
        <publisher>Addison-Wesley</publisher>
        <price>65.95</price>
    </book>
    <book year="1992">
        <title>Advanced Programming in the Unix environment</title>
        <author>Stevens W.</author>
        <author>Abiteboul Serge</author>
        <publisher>Addison-Wesley</publisher>
        <price>65.95</price>
    </book>
    <book year="2000">
        <title>Data on the Web</title>
        <author>Abiteboul Serge</author>
        <author>Buneman Peter</author>
        <author>Suciu Dan</author>
        <publisher>Morgan Kaufmann Publishers</publisher>
        <price>39.95</price>
    </book>
    <book year="1999">
        <title>The Economics of Technology and Content for Digital TV</title>
        <editor>CITI</editor>
        <publisher>Kluwer Academic Publishers</publisher>
        <price>129.95</price>
    </book>
</bib>
```

(그림 2) 예제 XML 데이터 (bib.xml)

XQuery 유형 1:
하나의 바인딩 변수에 의한 groupby

```
<results>
 {
for $b in doc("bib.xml")/bib/book,
    $a in $b/author
group by $a
return
   <result>
      { $a }
      <titles>
         { $b/title }
      </titles>
   </result>
 }
</results>
```

XQuery 유형 2:
집계함수와 함께 사용된 groupby

```
<results>
 {
 for $b in doc("bib.xml")/bib/book,
     $a in $b/author
 group by $a
 return
   <result>
      { $a }
      <title count>
         { count($b/title) }
      </title count>
      <titles>
         { $b/title }
      </titles>
   </result>
 }
</results>
```

(그림 3) 유형 1과 유형 2의 groupby 질의

(그림 3)의 XQuery유형 1은 groupby절을 도입한 XQuery로 작성한 것으로 하나의 바인딩 변수에 의한 groupby 질의 문장이다. 실험 XML 데이터('bib.xml')에 대해서 각 저자별로 출간한 도서의 제목을 검색하기 위해 구성된 질의이다.

(그림 3)의 XQuery 유형 2는 집계함수와 함께 사용된 groupby 질의 문장이다. 실험 XML 데이터('bib.xml')에 대해서 각 저자별로 출간한 도서의 제목들과 도서의 개수를 검색하여 구성된 질의이다.

(그림 4)의 XQuery 유형 3은 groupby절을 도입한 XQuery로 작성한 것으로 두 개 이상의 바인딩 변수에 의한 groupby 질의 문장이다. 실험 XML 데이터('bib.xml')에 대해서 각 저자별 및 출판사별로 검색하여 출간한 도서의 제목들을 검색하기 위해 구성된 질의이다.

XQuery 유형 3:
두 개 이상의 바인딩 변수에 의한 groupby

```
<results>
 {
for $b in doc("bib.xml")/bib/book,
      $a in $b/author,
      $y in $b/@year
group by $a, $y
return
  <result>
        { $a }
        <year  titles>
           <year>{ $y }</year>
           <titles>
              { $b/title }
           </titles>
        </year  titles>
    </result>
 }
</results>
```

XQuery 유형 4:
집합으로 구성된 바인딩 변수에 의한 groupby

```
<results>
{
for $b in doc("bib.xml")/bib/book
let $a := $b/author
group by [ $a ]
return
        <result>
           <author  set> { $a }
  </author  set>
     <titles>
        { $b/title }
     </titles>
     </result>
}
</results>
```

(그림 4) 유형 3과 유형 4의 groupby 질의

(그림 4)의 XQuery 유형 4는 groupby절을 도입한 XQuery로 작성한 것으로 집합으로 구성된 바인딩 변수에 의한 groupby 질의 문장이다. 실험 XML 데이터('bib.xml')에 대해서 각 저자들의 집합별로 출간한 도서의 제목을 검색하여 구성된 질의이다. 만약 groupby 절이 없는 현재의 XQuery 구문을 사용한다면 6절에 보인 것과 같이 복잡한 내포된 구문을 사용하여야만 동일한 결과를 얻을 수 있다.

4. eXist를 이용한 XQuery 그룹화 질의 처리 기법

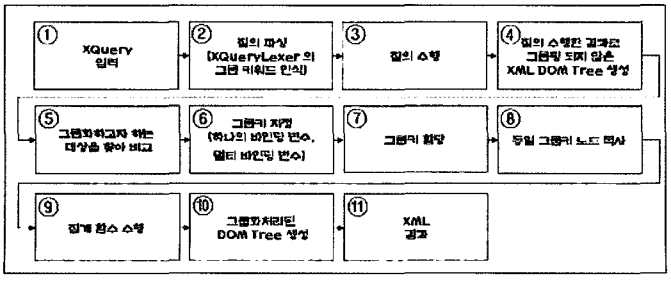
4.1 그룹화 질의 처리 과정 개요

eXist 데이터베이스에서 그룹화 질의 처리 과정을 (그림 5)에서 보여준다. XQuery 질의가 들어오면, XQueryLexer가 그룹 키워드를 인식하여 파싱을 한다. 질의 파싱 단계가 끝난 후에 질의를 수행하고, 질의 수행 결과는 아직 그룹핑 되지 않은 XML DOM 트리를 생성하게 된다. 그룹핑 되지 않은 질의 결과를 가지고 그룹화하고자 하는 대상을 찾아 비교를 한다. 그룹화를 위해 그룹키(group key)를 지정해 주어 바인딩 변수를 처리하도록 하였다. 그룹키는 그룹화하고자 하는 대상에게 값을 할당해주는 것이다. 이때, 그룹화하고자 하는 대상의 하위 노드들 및 형제 노드까지 모두 찾아 그룹키 값을 부여하고, 값이 동일한 경우에는 첫 번째 자식 노드와 동일한 레벨에서 그룹화하고자 하는 대상의 하위 노드들을 복사한다. 집계 함수를 함께 사용하여 그룹화하면, 이 단계에서 복사된 노드들의 수 및 값들을 처리하게 된다. 마지막으로 그룹화가 처리된 DOM 트리를 생성하고, 결과 XML이 출력된다.

4.2 groupby를 처리하기 위한 주요 3 단계

(그림 2)의 예제 XML 데이터에 보인 것처럼 서지목록 데이터인 bib.xml 문서를 (그림 3)의 유형 1의 예제 질의를 사용하여 groupby를 처리하기 위한 주요 과정에 대한 알고리즘을 (그림 6)에 보이며 다음과 같이 3단계로 설명할 수 있다.

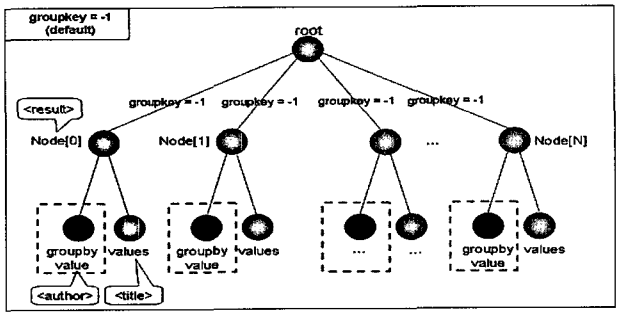
(그림 7)의 DOM(Document Object Model) 트리에서 루트(root)는 <results>이고, Node[0]에서 Node[n]까지는 <result> 노드를 의미한다. groupby value 노드는 그룹화 하고자 하는 대상(element)이며, values 노드는 groupby value의 하위 노드를 말한다. 따라서 그룹화 처리 1단계로 groupby value를 비교하여 그룹키를 지정하는데 Node[0]에서 Node[n]까지 중간 레벨(Intermediate Level)의 노드들을 비교한다. 먼저 Node[0]의 groupby value와 Node[1]의 groupby value를 비교하여 동일한 경우 동일한 그룹키를 준다.



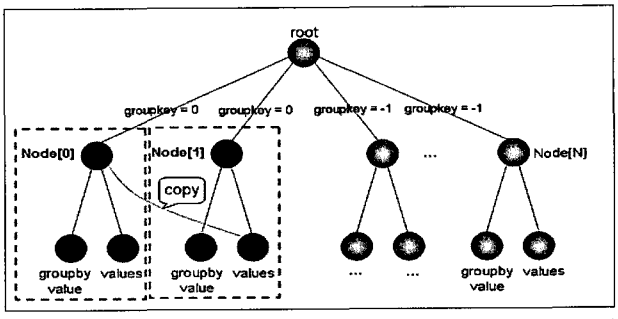
(그림 5) eXist에서의 그룹화 질의 처리 과정

```
Procedure processGroups (NodeImpl node[])
{
     setGroupKeys(node[]);       // 1 단계 : 그룹키 지정
     linkNodes(node[]);          // 2 단계 : 그룹에 따라 노드를 링크함
     removeNodeLinks(node[]);    // 3 단계 : 중복되는 링크 제거
}

Procedure setGroupKeys (NodeImpl node[])          // 1 단계
{
    int groupkey = -1;
    for (i = 0; i < node.length - 1; i++) {
     for (j = i + 1; j < node.length; j++) {
            //두 개의 groupkey를 비교하여 동일하면 0을 return 하는 것
            if (compareTo (node[i], node[j])) {
                // node[i]의 groupkey가 -1 이 아니면 node[i]의 groupkey를
                //return 하고 아니면 마지막으로 사용한 groupkey +1 을 return
                if (node[i].getGroupKey() == -1)  {  //새로운 groupkey 가 필요
                        groupkey++ ;
                        node[i].setGroupKey(groupkey);
                        setNodeChildGroupKey(node[i].getFirstChild(), groupkey);
                 } else {
                        groupkey = node[i].getGroupKey();
                 }
                 node[j].setGroupKey(groupkey);
                 //groupkey를 해당 node와 child node들에 setting
                 setNodeChildGroupKey(node[j].getFirstChild(), groupkey);
            }
        }
    }
}

Procedure linkNodes (NodeImpl node[])             // 2단계
{
    for (i = 0; i < node.length - 1; i++) {
      for (j = i + 1; j < node.length; j++) {
        if(nodeIsNotLinked(node[i]) {
            if (compareTo (node[i], node[j])) {
                linkNodes(node[i], node[j]);
          }
        }
      }
   }
}

Procedure removeNodeLinks (NodeImpl node[])    // 3단계
{
    for (i = 0; i < node.length - 1; i++) {
            if(nodeIsLinked(node[i]) {
                deleteFromRoot(node[i]);
            }
   }
  }
```

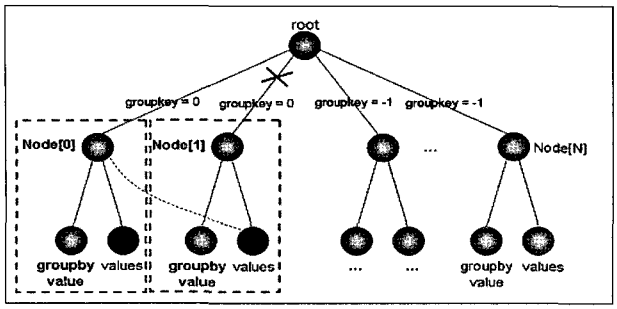
(그림 6) 그룹화를 처리하기 위한 주요 알고리즘



(그림 7) 그룹화 처리 1단계로 groupby value 노드들 간의 비교



(그림 8) 그룹화 처리 2단계로 groupkey가 같은 노드들 간의 연결



(그림 9) 그룹화 처리 3단계로 중복방지를 위한 노드 링크 삭제

2단계는 (그림 8)에서 보인 바와 같이 groupkey 값이 동일한 노드들을 찾아 각 그룹의 첫 노드에 values 노드들을 연결해 준다. 즉, Node[0].groupkey 값과 Node[1].groupkey 값이 동일하면 Node[0]에 Node[1]의 values 노드를 링크한다. 이런 작업을 Node[n]까지 계속 반복한다. Node[n]까지 모든 작업이 끝나게 되면 동일한 그룹키를 가진 노드들끼리 링크가 생기게 된다.

3단계는 (그림 9)에서 보인 바와 같이 그룹키에 따라 새로이 연결된 노드들이 중복되지 않도록 하기 위해서 루트 노드로부터 기존의 연결을 제거한다. 즉, Node[0]에 Node[1]의 values 에 대한 링크를 포함시켰으므로 Node[1]에 대한 링크를 루트(root)로부터 삭제시킨다. 이런 작업을 Node[n]까지 반복한다. 처리가 완료되면 그룹화 처리가 완료된 DOM 트리 구조가 완성된다.

5. eXist 기반의 XQuery 그룹화 질의 처리시스템 구현

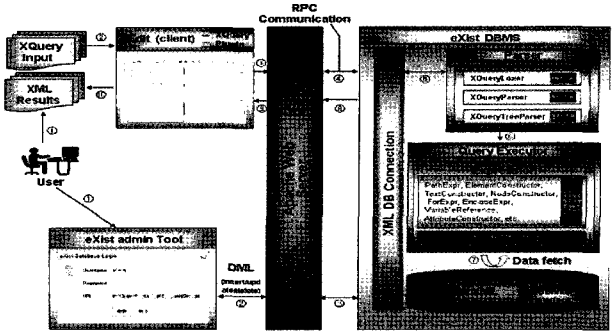
5.1 개발 환경

eXist 시스템 기반의 XQuery 그룹화 질의 처리 시스템을 구현하기 위한 개발 환경은 서버로는 eXist(버전 1.0b2)와 클라이언트로는 jEdit(버전 4.2)를 사용하였다. <표 1>에 보인 것처럼 eXist를 실행하기 위해 apache-ant(버전 1.6.3)가 우선 설치되어야 하며, jEdit에서 XQuery를 사용하기 위해서 필요한 것이 XQuery + XSLT 플러그인(plugin)이다.

(그림 10)은 eXist Server와 jEdit Client를 이용하여 XQuery의 groupby절을 지원하는 질의 처리 순서를 나타낸 시스템 구조도를 보여주고 있다.

〈표 1〉 XQuery에 groupby절을 도입하기 위한 개발환경

■ 개발환경		
서 버 (Server)	eXist 1.0	1) URL: http://exist.sourceforge.net/ 2) License: Open Source 3) Latest Stable Release : eXist-1.0b2-build-1107.jar
클라이언트 (Client)	jEdit 4.2	• groupby를 인식할 수 있도록 확장 - 설치 필수 요건 : 어텐터 3가지 1) jEdit 4.2 (http://www.jedit.org) 2) the XQuery plugin 3) eXist (at least snapshot 040927)
통합개발환경 (Integrated Development Environment : IDE)	Eclipse SDK 3.0.2	• eXist 소스 코드 디버깅 1) URL: http://www.eclipse.org/ 2) License: Open Source 3) Entry last updated: June, 2003
개발 언어 (Developer Language)	• JDK 1.4.2 • XQuery 1.0	• J2SE URL: http://java.sun.com/j2se/1.4.2/download.html • XQuery URL: http://www.w3.org/TR/2005/CR-XQuery-20051103



(그림 10) eXist에 groupby절을 지원하기 위한 확장된 시스템 구조

질의 처리 순서를 다음과 같이 확장 설계하였다. 우선 jEdit와 XQuery 플러그인을 통해 XQuery 입력을 받는다. 입력된 XQuery 요청에 대해서 아파치 웹 서버(Apache Web Server)를 통해서 eXist 데이터베이스 관리 시스템으로 요청한다. eXist 시스템은 요청된 질의에 대해서 Parser 모듈을 통해 파싱(parsing) 작업을 한다.

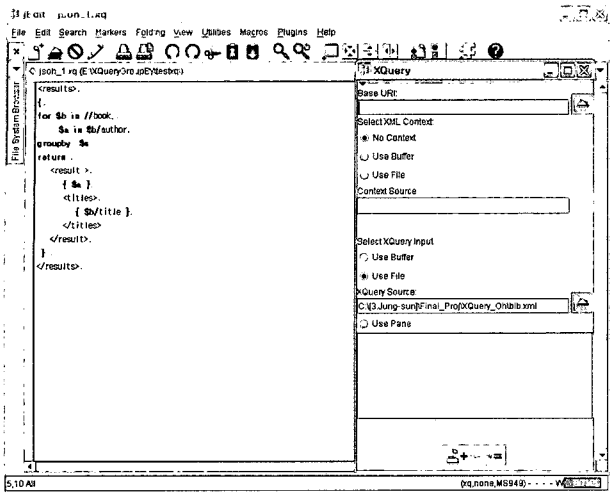
Parser모듈에서 확장된 부분은 다음과 같다. (그림 10)의 ⓐ에 표시된 것처럼 XQueryLexer는 XQuery의 어휘 분석을 처리하며, 그룹화를 나타내기 위한 groupby 키워드를 추가하였다. (그림 10)의 ⓑ에 표시된 것처럼 XQueryParser에서는 XQuery AST(Abstract Syntax Tree) 생성 시에 group by 구문 처리 함수를 추가하였다. (그림 10)의 ⓒ에 표시된 것처럼 XQueryTreeParser에서는 AST를 분석해서 내부적으로 사용하는 질의 표현식으로 변환하는 함수를 수정하여, AST에서 groupby 가 널(null)이 아닌 경우 GroupSpec 클래스 배열을 생성한 후 결과 질의 표현식에 저장하도록 하였다.

질의를 수행하는 Query Executor모듈에서 확장된 부분은 다음과 같다. PathExpr은 XQueryTreeParser에서 groupby 값이 있는 경우 GroupSpec에 해당 값을 지정한 후, ForExpr에서 각 노드 단위로 result 레벨들을 GroupedValueSequence 클래스 타입의 배열에 저장한다. (그림 10)의 ⓓ에 표시된 것처럼Query Executor의 Expression에서 groupby 문장에 대한 Expression 관리 클래스를 선언하고, XQuery와 XPath 실행 클래스들을 구현하였다.

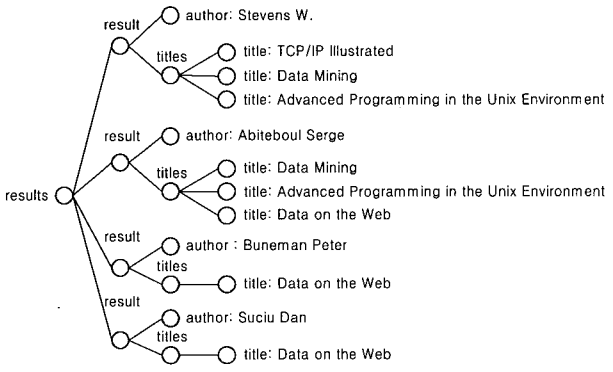
(그림 10)의 ⓒ에 표시된 것처럼 그룹화 항목을 비교하기 위해 GroupedValueSequence를 정의하여 GroupSpec에서 정의한 groupby 항목 값을 저장한 후 이 값들을 비교해 동일한 값인 경우 해당 노드에 대해서 동일한 그룹키를 부여한다. 이 때 groupby로 묶여야 하는 모든 노드(자식 노드 및 형제 노드)들에 대해서 동일한 그룹키 값을 부여한다. 수행된 결과에 대해 groupby를 처리한 후, 결과를 반환한다.

5.2 eXist 기반의 XQuery 그룹화 질의 처리 시스템의 실행화면

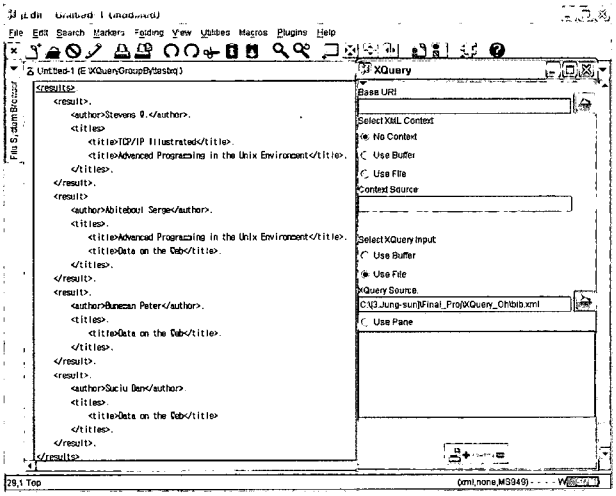
본 절에서는 구현된 시스템에서 XQuery groupby 질의를 수행한 결과들을 보여준다. eXist 소스 코드 디버깅을 하기 위해서 이용되는 자바의 대표적인 통합개발환경으로 Eclipse와 jEdit이 있으며 jEdit에서 XQuery 질의를 작성하고 실행



(그림 11) jEdit에서의 유형 1의 질의 요청 화면



(그림 12) 유형 1의 질의 예제에 대한 예상되는 질의 결과



(그림 13) jEdit에서의 유형 1의 질의 결과 화면

할 수 있다.

(그림 11)은 XQuery 유형 1의 예제를 입력한 것으로서 서지목록 데이터인 bib.xml 실험 데이터에서 각 저자별로 출간한 도서의 제목들을 검색하여 출력하기 위해 하나의 바인딩 변수에 의한 groupby 질의를 jEdit 클라이언트에서 입력한 화면이다. 이 XQuery질의에 대한 예상되는 결과는 (그림 12)에서 트리 구조로 보여주는 것처럼 저자별로 도서가 그룹화된 것으로서 (그림 13)은 jEdit 클라이언트에서 XQuery 질의가 처리되어 나타난 결과 XML 화면이다.

6. 그룹화를 지원하기 위한 XQuery 모델링 능력의 비교

<표 2>는 groupby를 지원하는 XQuery와 다른 XML 질의 언어들의 기능을 비교한 것이다.

XQuery에 groupby를 추가하는 경우에 다음과 같이 각 질의 유형별로 기존의 복잡한 XQuery 그룹화 질의들이 사용자가 작성하기에 편하도록 내포된 구조가 제거되어 매우 단순화된 형태로 표현되며 길이도 대체로 짧아진다.

〈표 2〉 groupby를 지원하는 XQuery와 다른 XML 질의어들과의 비교

	group by를 지원하는 XQuery	LOREL	XML-QL	XML-GL	XSL	XQL
그룹화 질의	Yes	Yes	No	Yes	No	No
집계 연산	Yes	Yes	No	Yes	Partially	Partially
내포된 질의	Yes	Yes	Yes	No	Yes	No

```
<results>
{
    let $a := doc("bib.xml")//author
ⓐ  for $author in distinct-values($a/text())
    return
      <result>
        <author>{ $author }</author>
        <titles>
        {
ⓑ     for $b in doc("bib.xml")/bib/book
        where some $ba in $b/author
                satisfies ($ba/text() = $author)
        return
          $b/title
        }
        </titles>
      </result>
}
</results>
```

(a) groupby 를 사용하지 않은 경우

```
<results>
{
 for $b in doc("bib.xml")/bib/book,
       $a in $b/author
 group by  $a
 return
    <result>
      { $a }
      <titles>
        { $b/title }
      </titles>
    </result>
}
</results>
```

(b) groupby 를 사용한 경우

(그림 14) 유형 1의 XML 그룹화 질의에 대한 모델링 능력 비교

```
<results>
  {
    let $a := doc("bib.xml")//author
ⓐ for $author in distinct-values($a/text())
    let $t := ⓑ for $b in doc("bib.xml")/bib/book
                  where some $ba in $b/author
                    satisfies ($ba/text()= $author)
                  return
                    $b/title
  return
       <result>
          <author>{ $author }</author>
          <title-count>{ count($t) }</title-count>
          <titles>{ $t }</titles>
       </result>
  }
</results>
```

(a) groupby를 사용하지 않은 경우

```
<results>
 {
  for $b in doc("bib.xml")/bib/book,
        $a in $b/author
  group by  $a
  return
     <result>
        { $a }
        <title-count>
           { count($b/title) }
        </title-count>
        <titles>
           { $b/title }
        </titles>
     </result>
 }
</results>
```

(b) groupby를 사용한 경우

(그림 15) 유형 2의 XML 그룹화 질의에 대한 모델링 능력 비교

```
<results>
 {
   let $a := doc("bib.xml")//author
ⓐ for $author in distinct-values($a/text())
  return
       <result>
          <author>{ $author }</author>
          {
            let $year :=
doc("bib.xml")/bib/book[author=$author]/year
      ⓑ for $y in distinct-values($year/text())
          return
            <year-titles>
              <year>{ $y }</year>
              <titles>
              {
            ⓒ for $b in
doc("bib.xml")/bib/book[author=$author and year=$y]
                  return
                $b/title
              }
             </titles>
          </year-titles>
          }
      </result>
 }
</results>
```

(a) groupby를 사용하지 않은 경우

```
<results>
 {
  for $b in doc("bib.xml")/bib/book,
        $a in $b/author,
        $y in $b/@year
  group by $a, $y
  return
     <result>
        { $a }
        <year-titles>
           <year>{ $y }</year>
           <titles>
              { $b/title }
           </titles>
        </year-titles>
     </result>
  }
</results>
```

(b) groupby를 사용한 경우

(그림 16) 유형 3의 XML 그룹화 질의에 대한 모델링 능력 비교

```
<results>
 {
 ① let $au1 := ⓐ for $b1 in doc("bib.xml")/bib/book
                where exists($b1/author)
                return
                  <author-set>{
                        for $a1 in $b1/author
                        order by $a1
                        return $a1
                              }
                  </author-set>
 ② let $au2 := ⓑ  for $b2 in doc("bib.xml")/bib/book
                where exists($b2/author)
                return
                  <author-set>{
                        for $a2 in $b2/author
                        order by $a2
                        return $a2
                              }
                  </author-set>
 ③ let $au3 := union($au1, $au2)
 ⓒ for $au4 in $au3/author-set
   return
     <result>
        { $au4 }
        <titles>
          {
       ⓓ for $b in doc("bib.xml")/bib/book
        where exists($b/author)
        let $au := ⓔ let $au5 := $b/author
                   return
                     <author-set>{
                       for $a3 in $b/author
                       order by $a3
                       return $a3
                             }
                     </author-set>
        where deep-equal($au,$au4)
        return
            $b/title
          }
        </titles>
     </result>
 }
</results>
```

(a) groupby 를 사용하지 않은 경우

```
<results>
{
for $b in doc("bib.xml")/bib/book
let $a := $b/author
return group by [ $a ]
        <result>
          <author-set>{ $a }</author-set>
             <titles>
               { $b/title }
             </titles>
        </result>
}
</results>
```

(b) groupby 를 사용한 경우

(그림 17) 유형 4의 XML 그룹화 질의에 대한 모델링 능력 비교

이들 예제에서 보는 것처럼 groupby가 없는 기존의 XQuery를 이용하여 그룹화 질의를 작성할 경우 내포된 질의를 작성하여야 하며 이는 질의를 매우 작성하기도 어렵고 이해하기도 어렵게 하여 모델링 능력을 저하시킨다. 기존의 XQuery에 groupby를 확장함으로써 명시적으로 groupby를 지원하여 groupby 변수들을 지원할 수 있게 되며 이에 따라 집계값을 구하거나 집합으로 된 변수에 대한 그룹화 질의도 매우 쉽게 작성할 수 있게 되어 XQuery의 모델링 능력을 향상시킬 수 있다.

7. 결론 및 향후 연구

XML(eXtensible Markup Language)은 데이터를 저장 및

전송하기 위한 마크업 언어로서 구조적 또는 반구조적인 정보에 가장 널리 사용되는 형식이다. 이러한 XML 데이터는 계속 증가하고 있는데 이러한 XML 데이터의 편리한 검색에 대한 요구사항에 대처할 수 있도록 강력한 질의 언어인 XQuery가 만들어졌다. XQuery는 다양한 데이터 소스로부터 가져온 XML 데이터를 고유한 구조를 가진 질의 결과로 구성할 수 있도록 설계되어 XML 질의 언어의 표준이 되었다. 현재 XQuery는 반복문 등을 포함하는 검색 기능을 지원하며 질의 표현이 상대적으로 어렵고, 복잡한 형태를 취한다. 또한 groupby 기능이 제공되지 않아 중첩된 내포구조와 조인연산을 이용하여 그룹화 처리를 하고 있다.

본 논문에서는 XQuery에 groupby 절을 도입한 질의 처리기의 구현으로 데이터의 재구성과 집계처리를 위한 그룹화 구성이 쉽도록 하였다. 기존의 SQL에처럼 groupby를 사용하여 그룹화 및 다양한 레벨에서의 그룹화 그리고 여러 변수 및 집합 기반의 그룹화가 가능하도록 구현하였다. 본 논문에서 제안한 기법을 적용하기 위해서 XQuery에 group-by 절을 도입하여 EBNF를 제안하고, eXist 네이티브 XML 데이터베이스 기반의 XQuery 그룹화 질의 처리 시스템에서 groupby를 지원하는 기능을 추가하여 확장함으로써 group-by 질의 처리가 가능하도록 하였다. 향후 연구로는 XQuery에서 groupby 질의를 고려하여 효율적인 질의 계획을 수립할 수 있도록 확장 및 구현할 계획이다.

참 고 문 헌

[1] XML(eXtensible Markup Language), http://www.w3.org/XML/

[2] XQuery (XML Query Language), http://www.w3.org/XML/Query/

[3] eXist(An Open Source Native XML Database), http://exist.sourceforge.net/

[4] XML Path Language (XPath) 2.0, http://www.w3.org/TR/2005/WD-xpath20-20050404/

[5] J. McHugh, S. Abiteboul, R. Goldman, D. Quass and J. Widom, "Lore: A Database Management System for Semistructured Data", SIGMOD Record, 26(3), pp.54-66, September, 1997.

[6] H. V. Jagadish, Shurug Al-Khalifa, Adriane Chapman, Laks V.S. Lakshmanan, Andrew Nierman, Stelios Paparizos, Jignesh M. Patel, Divesh Srivastava, Nuwee Wiwatwattana, Yuqing Wu and Cong Yu. "TIMBER: A Native XML Database", VLDB Journal, Vol.11, Issue 4, pp.274-291, 2002.

[7] D. Chatziantoniou and K. A. Ross, "Querying multiple features of groups in relational databases", VLDB, pp.295-306, 1996.

[8] D. Chatziantoniou and K. A. Ross, "Groupwise processing of relational queries", VLDB, pp.476-485, 1997.

[9] H. V. Jagadish, Laks V.S.Lakshmanan, Divesh Srivastava and Keith Thompson. "TAX: A Tree Algebra for XML", In Proc. DBPL Conf., pp.149-164, Frascati, Italy, Sep. 2001.

[10] Stelios Paparizos, Shurug Al-Khalifa, H. V. Jagadish, Laks Lakshmanan, Andrew Nierman, Divesh Srivastava and Yuqing Wu, "Grouping in XML", In: EDBT 2002 Workshop on XML-Based Data Management (XMLDM'02), pp.128-147, 2002.

[11] S. Chaudhuri, R. Kaushik and J.F. Naughton, "On Relational Support for XML Publishing: Beyond Sorting and Tagging", SIGMOD, pp.611-622, 2003.

이 민 수

e-mail : mlee@ewha.ac.kr

1992년 서울대학교 컴퓨터공학과(학사)

1995년 서울대 대학원 컴퓨터공학과 (공학석사)

1995년~1996년 LG전자 미디어통신연구소 연구원

2000년 University of Florida 컴퓨터공학과 공학박사

2000년~2002년 미국 Oracle Corporation, Senior Member of Technical Staff

2002년~현재 이화여자대학교 컴퓨터학과 조교수

관심분야 : 데이터웨어하우스, 지식기반 시스템, 웹 데이터베이스

조 혜 영

e-mail : theodora.cho@samsung.com

2000년 이화여자대학교 화학과 (학사)

2004년 이화여자대학교 과학기술대학원 컴퓨터학과(공학석사)

2004년~현재 삼성전자 연구원

관심분야 : 데이터웨어하우스, 데이터베이스, XML, WAP

오 정 선

e-mail : agamaster@msn.com

2003년 숙명여자대학교 컴퓨터과학과 (학사)

2006년 이화여자대학교 과학기술대학원 컴퓨터학과(공학석사)

2006년~현재 SK 커뮤니케이션즈

관심분야 : 데이터웨어하우스, XML 데이터베이스, 임베디드 DBMS

김 윤 미

e-mail : cherish11@ewhain.net
2005년 이화여자대학교 컴퓨터학과(학사)
2005년~현재 이화여자대학교 컴퓨터학과 석사과정
관심분야 : 데이터웨어하우스, XML 데이터베이스, 데이터마이닝

송 수 경

e-mail : happymint@ewhain.net
2006년 이화여자대학교 컴퓨터학과(학사)
2006년~현재 이화여자대학교 컴퓨터학과 석사과정
관심분야 : XML 데이터베이스, 데이터마이닝, 임베디드 DBMS