효율적인 XML 질의 처리를 위한 XQuery 질의의 정규화

(Normalization of XQuery Queries for Efficient XML Query Processing)

김 서 영 † 이 기 훈 †† 황 규 영 †††
(Seo-Young Kim) (Ki-Hoon Lee) (Kyu-Young Whang)

요 약 XML이 웹 상에서의 정보 표현, 통합, 교환을 위한 표준이 됨에 따라 다양한 XML 질의 언어들이 제안되었으며, World Wide Web Consortium(W3C)은 XQuery를 XML 질의 언어의 표준으로 권고하였다. XQuery는 SQL과 유사하게 중첩 질의를 허용하므로, 중첩된 XQuery 질의를 동일한 의미를 가지면서 보다 효율적으로 실행될 수 있는 질의로 변환하는 정규화 규칙들이 제안되었다. 하지만 제안된 정규화 규칙들은 제한적인 형태의 중첩 질의에만 적용되는 문제점을 가지고 있다. 특히, FLWR 표현식의 where 절에 있는 중첩을 처리할 수 없다.

본 논문에서는 SQL 질의의 정규화 규칙들을 확장하여 XQuery 질의의 정규화 규칙들을 제안한다. 제안한 정규화 규칙들은 FLWR 표현식의 모든 절에 나타나는 중첩을 처리할 수 있다. 본 논문의 주요 공헌은 다음과 같다. 첫째, 상관과 집계의 유무에 따라 XQuery 질의의 중첩 유형을 분류하고, 각 유형 별로 정규화 규칙들을 제안한다. 둘째, 중첩된 XQuery 질의에 정규화 규칙들을 적용하는 세부 알고리즘들을 제안한다.

키워드 : XML 질의 처리, XQuery, 정규화

Abstract As XML becomes a standard for data representation, integration, and exchange on the Web, several XML query languages have been proposed. World Wide Web Consortium(W3C) has proposed XQuery as a standard for the XML query language. Like SQL, XQuery allows nested queries. Thus, normalization rules have been proposed to transform nested XQuery queries to semantically equivalent ones that could be executed more efficiently. However, previous normalization rules are applicable only to restricted forms of nested XQuery queries. Specifically, they can not handle FLWR expressions having nested expressions in the where clause.

In this paper, we propose normalization rules for XQuery queries by extending those for SQL queries. Our proposed rules can handle FLWR expressions having nested expressions in every clause. The major contributions of this paper are as follows. First, we classify nesting types of XQuery queries according to the existence of correlation and aggregation. We then propose normalization rules for each nesting type. Second, we propose detailed algorithms that apply the normalization rules to nested XQuery queries.

Key words : XML Query Processing, XQuery, Normalization

· 본 연구는 첨단정보기술연구센터를 통하여 한국과학재단으로부터 지원을 받았음

† 학생회원 : 삼성전자 기술총괄 소프트웨어센터 연구원
sykim@mozart.kaist.ac.kr
†† 비 회 원 : 한국과학기술원 전산학과
drlee@mozart.kaist.ac.kr
††† 종신회원 : 한국과학기술원 전산학과 교수
: kywhang@mozart.kaist.ac.kr

논문접수 2004년 4월 6일
심사완료 2004년 7월 2일

1. 서 론

XML(eXtensible Markup Language) 문서는 구조 정보를 추가할 수 있는 문서로서, 웹의 새로운 표준 문서로 많이 사용되고 있다[1]. 이에 따라, 대량의 XML 문서들에 대해 효율적으로 질의할 수 있는 시스템의 필요성이 점차 부각되었으며, Quilt[2], XQL[3], XML-QL[4], XPath[5], XQuery[6] 등과 같은 다양한 XML 질의 언어들이 등장하였다. 이 중에서 XPath와 XQuery

가 가장 널리 사용되고 있는 XML 질의 언어이다.

XPath는 XML 문서를 요소(element)와 속성(attribute)들로 이루어진 트리로 모델링하고, 트리 상에서 노드가 되는 요소와 속성들을 검색할 수 있는 표준 질의 언어이다[5]. XQuery는 XPath를 포함하는 질의 언어로서 현재 World Wide Web Consortium(W3C)[7]에서 표준화가 진행중이다. XQuery는 XPath를 이용하여 XML 문서의 요소와 속성들에 접근(access)할 수 있을 뿐만 아니라, 여러 XML 문서들을 조인하거나 새로운 XML 문서를 생성할 수 있는 기능을 제공한다. 특히 XQuery는 SQL의 select-from-where 문과 유사한 기능을 가지는 FLWR 표현식을 제공한다. FLWR 표현식은 for 절, let 절, where 절, 그리고 return 절로 구성되며, 각 절 안에 또 다른 FLWR 표현식이 중첩될 수 있는 특징이 있다.

그러나 중첩된 FLWR 표현식을 포함하는 XQuery 질의는 중첩된 루프문의 실행을 야기하기 때문에, 실제 시스템에서 처리될 때는 비효율적인 문제가 있다. 따라서 중첩된 XQuery 질의를 동일한 의미를 가지면서 보다 효율적으로 실행될 수 있는 질의로 변환하는 정규화 규칙들이 필요하다.

XQuery 질의의 정규화에 관한 연구로는 참고 문헌 [8] 등의 연구가 있다. 참고 문헌 [8]에서는 복잡한 XQuery 질의를 단순화하고, 중첩이 있는 XQuery 질의에서 중첩을 제거하는 규칙들을 제안하였다. 그러나 제안된 XQuery 질의의 정규화 규칙들 중에서 질의를 구성하는 주요 표현식인 FLWR 표현식의 중첩을 제거하는 규칙은 매우 제한적이라는 문제점이 있다. 즉, FLWR 표현식의 where 절에 중첩이 있는 경우에 대해 고려하지 않았으며, for 절과 return 절에 중첩이 있는 경우에도 매우 제한적인 형태에 대해서만 정규화 규칙을 제안하였다.

본 논문에서는 SQL 질의의 정규화 규칙들을 확장하여 XQuery 질의를 구성하는 FLWR 표현식의 모든 절에 나타나는 중첩을 제거하는 정규화 규칙들을 제안한다. 본 논문의 주요 공헌은 다음과 같다. 첫째, 상관과 집계의 유무에 따라 XQuery 질의의 중첩 유형을 분류하고, 각 유형 별로 정규화 규칙을 제안한다. 둘째, 중첩된 XQuery 질의에 정규화 규칙들을 적용하는 세부 알고리즘들을 제안한다.

본 논문의 구성은 다음과 같다. 제2절에서는 관련 연구로서 XQuery 질의 언어에 대해 소개하고, 기존의 XQuery 질의의 정규화 방법과 SQL 질의의 정규화 방법에 대해 각각 설명한다. 제3절에서는 확장된 XQuery 질의의 정규화 규칙들에 대해 설명한다. 제4절에서는 제3절에서의 설계한 규칙들을 바탕으로 XQuery 질의의 정규화의 구현에 대해 설명한다. 마지막으로 제5절에서는 결론을 내린다.

2. 관련 연구

본 절에서는 관련 연구로서 XQuery 질의 언어, XQuery 질의의 정규화, 그리고 SQL 질의의 정규화에 대해서 설명한다. 제2.1절에서는 XQuery 질의 언어에 대해 설명하고, 제2.2절에서는 XQuery 질의를 정규화하는 규칙들에 대해 설명한다. 제2.3절에서는 SQL 질의를 정규화하는 규칙들에 대해 설명한다.

2.1 XQuery

XQuery는 현재 W3C에서 표준화가 진행 중인 XML 질의 언어이다. XQuery 질의를 구성하는 기본 단위는 표현식(expression)이며, XQuery 질의 언어가 제공하는 기본 표현식으로는 경로 표현식(path expression), FLWR 표현식(FLWR expression), 요소 생성자(element constructor), 비교 표현식(comparison expression) 등이 있다. 그리고 각 표현식은 다른 표현식 안에 중첩될 수 있는 특성이 있다[6].

먼저 경로 표현식은 XPath의 형식을 따르는 표현식으로서, 질의 1은 경로 표현식을 이용하여 이름이 "John"인 직원을 검색하는 질의이다. 경로 표현식에서 대괄호 안에 나타나는 표현식은 술어(predicate)를 의미한다.

질의 1. 경로 표현식을 이용하여 이름이 "John"인 직원을 구하라.

```
document("Departments.xml")/Departments/Dept/Emp[EName="John"]
```

질의 1의 경로 표현식은 XML 문서 "Departments.xml"에서 루트 요소(root element)인 Departments의 자식 요소(child element) 중에서 Dept를 검색하고, 다시 Dept의 자식 요소 중에서 Emp를 검색하여 Emp의 자식 요소인 EName의 값이 "John"이면 검색한 Emp를 결과로서 반환한다.

질의 1에서 사용한 XML 문서 "Departments.xml"는 회사의 부서 정보를 나타내는 XML 문서로서, 그림 1과 같다. XML 문서 "Departments.xml"의 루트 요소인 Departments는 부서를 의미하는 요소인 Dept로 구성되며 각 Dept 요소는 부서의 식별 번호를 의미하는 요소인 DNO, 부서의 이름을 의미하는 요소인 DName, 부서가 있는 위치를 의미하는 요소인 DLoc, 그리고 그 부서에서 근무하는 직원의 정보를 가진 요소인 Emp로 구성된다. 그리고 각 Emp 요소는 고유 번호를 의미하는 요소인 SSN과 이름을 의미하는 요소인 EName으로 구성된다. 그림 1의 XML 문서에 질의 1을 수행하면 결과

1을 얻을 수 있다.

```
<Departments>
  <Dept>
    <DNO> D2 </DNO>
    <DName> Research </DName>
    <DLoc> Bellaire </DLoc>
    <DLoc> Houston </DLoc>
    <Emp>
      <SSN> 123456789 </SSN>
      <EName> John </EName>
    </Emp>
    <Emp>
      <SSN> 333445555 </SSN>
      <EName> Franklin </EName>
    </Emp>
  </Dept>
</Departments>
```

그림 1 XML 문서 "Departments.xml"

결과 1

```
<Emp>
  <SSN> 123456789 </SSN>
  <EName> John </EName>
</Emp>
```

다음으로 FLWR 표현식은 XQuery 질의를 구성하는 주요 표현식으로서, for 절, let 절, where 절, 그리고 return 절로 구성된다. for 절, where 절, return 절은 각각 SQL의 from 절, where 절, select 절과 유사한 의미를 가지며, let 절은 표현식을 하나의 변수로 치환하는 의미를 가진다.

질의 2는 FLWR 표현식을 이용하여 "Research" 부서의 위치를 구하는 질의이다. 질의 2의 for 절은 변수 $x에 document("Departments.xml")/Departments/Dept가 돌려주는 요소들을 하나씩 바인딩한다. 다음으로 where 절은 $x에 바인딩된 요소들 중에서 자식 요소인 DName의 값이 "Research"인 것을 선택한다. 마지막으로 return 절에서 $x에 바인딩된 요소의 자식 요소들 중에서 DLoc 요소들을 반환한다. 그림 1의 XML 문서에 질의 2를 수행하면 결과 2를 얻을 수 있다.

질의 2. FLWR 표현식을 이용하여 "Research"부서의 위치를 구하라.

```
for $x in document("Departments.xml")/Departments/Dept
where $x/DName = "Research"
return $x/DLoc
```

결과 2

```
<DLoc> Bellaire </DLoc>
<DLoc> Houston </DLoc>
```

다음으로 요소 생성자는 XQuery 질의 내에서 새로운 요소를 생성할 수 있도록 해주는 표현식이다. 질의 3은 요소 생성자를 이용하여 "Research" 부서의 위치 정보를 가지는 요소인 "Research_Locations"를 생성하는 질의이다. 그림 1의 XML 문서에 이 질의를 수행하면 결과 3을 얻을 수 있다.

질의 3. "Research"부서의 위치 정보를 가진 요소를 생성하라.

```
<Research_Locations>
 {
  for    $x in document("Departments.xml")/Departments/Dept
  where $x/DName = "Research"
  return $x/DLoc
 }
</Research_Locations>
```

결과 3

```
<Research_Locations>
  <DLoc> Bellaire </DLoc>
  <DLoc> Houston </DLoc>
</Research_Locations>
```

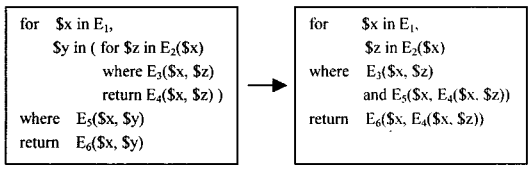
마지막으로 비교 표현식은 두 값을 비교하는 연산을 나타낸다. SQL과 같은 질의 언어들과는 달리, XQuery에서의 비교 연산은 리스트 간의 비교 연산을 허용한다. 즉, 리스트 L_1과 L_2에 대한 비교 연산 L_1 op L_2가 있을 때 $x \in L_1$, $y \in L_2$인 x, y에 대해 x op y의 값이 참이 되는 (x, y)의 쌍이 하나라도 존재하면 비교 연산의 값은 참이 되고, 존재하지 않으면 비교 연산의 값은 거짓이 된다. 예를 들어, 비교 연산(1, 2, 3) = (1, 4)은 참 값을 가지며, (1, 2) = (3, 4)은 거짓 값을 갖는다.

2.2 XQuery 질의의 정규화

실세계에서 대부분의 데이타는 관계형 데이타 형태로 저장되어 있으나, XML 문서가 정보 교환의 표준으로 많이 사용됨에 따라 관계형 데이타를 XML 문서 형태로 접근(access)하고자 하는 요구가 발생하였다. 이를 위해 기존의 관계형 데이타를 가상의 XML 문서로 표현하고, 이 가상의 XML 문서에 대해 XQuery 질의를 수행하고자 하는 연구[8]가 이루어졌다. 참고 문헌 [8]에서는 XQuery 질의를 SQL 질의로 변환하기 쉬운 형태로 바꾸기 위해, XQuery 질의를 단순화하고 질의의 중첩을 제거하는 정규화 규칙들을 제안하였다. 즉, 중첩이 있는 XQuery 질의에서 중첩을 제거하는 규칙들을 제안하였다.

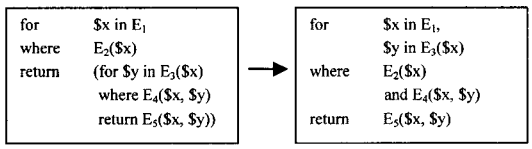
제안된 XQuery 질의의 정규화 규칙들 중에서 본 논문과 관련이 있는 FLWR 표현식의 중첩을 제거하는 규칙에 대해 설명한다. 먼저 각 규칙에 사용되는 기호의 의미는 다음과 같다. 각 규칙에서 $x, $y는 변수를 의미하며, E_n은 표현식을 의미한다. E_n($x)는 표현식 E_n에 변수 $x가 나타남을 의미한다.

규칙 1은 for 절에서 FLWR 표현식의 중첩을 제거하는 규칙이다. FLWR 표현식의 for 절에 또 다른 FLWR 표현식이 중첩된 경우에는 먼저 두 FLWR 표현식의 for 절을 하나로 합치고, 두 FLWR 표현식의 where 절을 and 연산으로 연결한다. 그리고 중첩된 FLWR 표현식이 바인딩된 변수를 중첩된 FLWR 표현식의 return 절에 있는 표현식으로 치환한다.



규칙 1 for 절에서 FLWR 표현식의 중첩을 제거하는 규칙

규칙 2는 return 절에서 FLWR 표현식의 중첩을 제거하는 규칙이다. FLWR 표현식의 return 절에 또 다른 FLWR 표현식이 중첩된 경우에는 먼저 두 FLWR 표현식의 for 절을 하나로 합치고, 두 FLWR 표현식의 where 절을 and 연산으로 연결한다. 그리고 중첩된 FLWR 표현식을 중첩된 FLWR 표현식의 return 절에 있는 표현식으로 치환한다.



규칙 2 return 절에서 FLWR 표현식의 중첩을 제거하는 규칙

참고문헌 [8]은 복잡한 XQuery 질의들을 단순화하고, 중첩된 XQuery 질의에서 중첩을 제거하는 규칙들을 제안함으로써 XQuery 질의의 정규화에 관한 연구의 효시가 되었다. 그러나 이 논문에서는 XQuery 질의를 SQL 질의로 변환하기 쉬운 형태로 바꾸기 위해 정규화 규칙들을 제안한 것이기 때문에 제한적인 형태의 중첩 유형에 대해서만 정규화 규칙들을 제안하였다. 즉, SQL 질의에서는 where 절의 중첩이 허용되므로 FLWR 표현식이 where 절에 중첩된 경우에 대해서는 정규화 규칙을 제안하지 않고, for 절과 return 절에 중첩된 경우에 대해서만 정규화 규칙을 제안하였다. 그런데 for 절과 return 절에 중첩된 경우들 중에서도 FLWR 표현식이 집계 함수의 인자로 중첩된 경우에 대해서는 정규화 규칙을 제안하지 않았다.

2.3 SQL 질의의 정규화

SQL(Structured Query Language)은 현재 가장 많이 쓰이고 있는 표준 관계형 데이타베이스 질의 언어로서 질의를 중첩하여 표현할 수 있는 특징이 있다. 그러나 데이타베이스 시스템에서 중첩된 질의를 처리하는 것은 중첩된 루프문의 실행을 야기하기 때문에 비효율적이다. 따라서 중첩된 질의를 중첩되지 않은 질의로 변환하여 처리함으로써 질의 처리 성능을 향상시키고자 하는 연구(9-13)가 진행되었다.

SQL 질의의 기본 구조는 select 절, from 절, where 절로 구성된 질의 블록으로 표현할 수 있다. 하나의 질의 블록 안에는 또 다른 질의 블록이 중첩될 수 있는데, 이때 중첩을 포함하는 질의 블록을 주 질의 블록이라고 하고 중첩된 질의 블록을 하위 질의 블록이라고 한다. 상관은 하위 질의 블록의 where 절에 주 질의 블록의 릴레이션(relation)를 참조하는 부분이 있는 것이며, 집계는 질의 블록의 select 절에 집계 함수가 있는 것이다. 참고 문헌 [9]에서는 중첩된 SQL 질의를 하위 질의 블록의 상관과 집계 유무에 따라 Type-A, Type-N, Type-J, Type-JA, 그리고 Type-D 유형으로 분류하고, 각 유형 별로 질의를 정규화하는 규칙을 제안하였다.

Type-A 유형의 중첩 질의는 하위 질의 블록에 상관은 없고, 집계만 있는 경우이다. Type-A 유형 중첩 질의의 경우, 하위 질의 블록이 주 질의 블록과 독립적으로 수행될 수 있기 때문에 하위 질의 블록을 먼저 수행하고 결과로 얻은 값을 하위 질의 블록과 치환함으로써 중첩을 제거할 수 있다. Type-A 유형 중첩 질의의 예를 설명하기 전에 먼저 예제에 쓰인 데이타베이스 스키마를 설명하고, 다음으로 질의의 예를 설명한다. 예제에 쓰인 데이타베이스 스키마는 다음과 같다.

Dept(DNO, DName, DLoc)
Proj(PNO, PName, PLoc, DNO)
Emp(SSN, EName, DNO)
WorksOn(SSN, PNO, Hours)

예제 데이타베이스 스키마는 부서 정보를 제공하는 릴레이션인 Dept, 프로젝트 정보를 제공하는 릴레이션인 Proj, 직원 정보를 제공하는 릴레이션인 Emp, 그리고 직원의 근무 정보를 제공하는 릴레이션인 WorksOn으로 구성된다. 릴레이션 Dept는 부서의 식별 번호를 의미하는 속성인 DNO를 주 키로 하고, 부서의 이름과 위치를 의미하는 속성인 DName과 DLoc를 갖는다. 릴레이션 Proj는 프로젝트의 식별 번호를 의미하는 속성인 PNO를 주 키로 하고, 프로젝트 이름과 진행지를 의미하는 속성인 PName과 PLoc, 그리고 외래키로 프로젝트가 속한 부서의 식별 번호를 의미하는 속성인 DNO를 갖는다. 릴레이션 Emp는 직원의 식별 번호를 의미하는 속성인 SSN을 주 키로 하고, 직원의 이름을 의미

하는 속성인 EName과 외래키로 직원이 속한 부서의 식별 번호를 의미하는 속성인 DNO를 갖는다. 릴레이션 WorksOn은 직원의 식별번호를 의미하는 속성인 SSN과 프로젝트의 식별번호를 의미하는 속성인 PNO를 주키로 하고, 직원이 해당 프로젝트에서 근무한 시간을 의미하는 속성인 Hours를 갖는다.

질의 4는 위의 데이타베이스 스키마를 이용한 Type-A 유형 중첩 질의의 예이다. 질의 4에서 where 절에 중첩된 질의 블록을 미리 계산하여 임시 변수 T에 저장함으로써 질의 4-1과 같이 정규화할 수 있다.

질의 4. 릴레이션 WorksOn에서 근무 시간이 가장 긴 직원의 SSN을 구하라.

```
select  W.SSN
from    WorksOn W
where   W.Hours = (select max(Hours)
          from WorksOn)
```

질의 4-1. 질의 4를 정규화함

```
T := (select max(Hours)
      from WorksOn)

select W.SSN
from   WorksOn W
where W.Hours = T
```

다음으로 Type-N 유형의 중첩 질의는 하위 질의 블록에 상관과 집계가 모두 없는 경우이고, Type-J 유형의 중첩 질의는 하위 질의 블록에 상관이 있고, 집계는 없는 경우이다. Type-N 유형과 Type-J 유형의 중첩 질의는 주 질의 블록의 릴레이션과 하위 질의 블록의 릴레이션을 조인함으로써 중첩이 없는 동일한 의미의 질의로 변환 할 수 있다. Type-N 유형과 Type-J 유형의 중첩 질의를 정규화하는 NEST-N-J 규칙은 규칙 3과 같다. 질의 5는 Type-J 유형 중첩 질의의 예로서, NEST-N-J 규칙을 적용하여 질의 5-1과 같이 정규화 할 수 있다.

1. 하위 질의 블록의 from 절에 있는 릴레이션들을 주 질의 블록의 from 절에 추가한다.
2. 하위 질의 블록의 where 절을 주 질의 블록의 where 절에 and로 연결한다.
3. 주 질의 블록의 where 절에 있는 하위 질의 블록을 하위 질의 블록의 select 절로 치환한다. 그리고 하위 질의 블록을 피연산자로 가지는 연산의 연산자가 in인 경우 =로 치환하고, in이 아닌 경우 원래 연산자를 그대로 둔다.
4. 주 질의 블록의 select 절은 그대로 둔다.

규칙 3 NEST-N-J 규칙

질의 5. 부서가 위치한 곳에 진행되는 프로젝트를 하나 이상 가지는 부서를 구하라.

```
select   D.DName
from     Dept D
where    D.DLoc in (select P.PLoc
           from Proj P
           where P.DNO = D.DNO)
```

질의 5-1. 질의 5를 정규화함

```
select   D.DName
from     Dept D, Proj P
where    D.DLoc = P.PLoc and P.DNO = D.DNO
```

Type-JA 유형의 중첩 질의는 하위 질의 블록에 상관과 집계가 모두 있는 경우이다. Type-JA 유형의 중첩 질의는 하위 질의 블록에서 각 상관 값에 대한 집계 함수의 값들을 미리 계산하여 임시 릴레이션으로 생성하고, 이 임시 릴레이션을 하위 질의 블록의 릴레이션과 치환함으로써 Type-J 유형의 중첩 질의로 변환할 수 있다. Type-JA 유형의 중첩 질의를 정규화하는 NEST-JA 규칙은 규칙 4와 같다.

1. 상관 값과의 조인에 참여하는 속성들로 하위 질의 블록의 릴레이션을 group by하여 집계 함수의 값을 계산한 임시 릴레이션을 생성한다.
2. 하위 질의 블록의 릴레이션을 1에서 생성한 임시 릴레이션으로 치환한다.
3. 하위 질의 블록의 select 절에 있는 집계 함수 부분을 1에서 미리 계산된 집계 함수의 값을 가지는 속성으로 치환하여 Type-J 유형의 중첩 질의로 변환한다.
4. NEST-N-J 규칙을 적용한다.

규칙 4 NEST-JA 규칙

질의 6은 Type-JA 유형 중첩 질의의 예이다. NEST-JA 규칙에 따라 질의 6을 Type-J 유형의 중첩 질의인 질의 6-1과 같이 변환하고, NEST-N-J 규칙을 적용하여 최종적으로 질의 6-2와 같이 정규화 할 수 있다.

질의 6. 2개 이상의 프로젝트를 가지는 부서의 이름을 구하라.

```
select   D.DName
from     Dept D
where    2 =< (select count(P.PNO)
                from Proj P
                where P.DNO = D.DNO)
```

질의 6-1. 질의 6을 Type-J 유형의 중첩 질의로 변환함.

```
T := select DNO, count(PNO) as NUM_OF_PROJS
     from Proj
     group by DNO

select D.DName
from Dept D
where 2 =< (select T.NUM_OF_PROJS
              from T
              where T.DNO = D.DNO)
```

질의 6-2. 질의 6-1을 정규화함

```
select   D.DName
from     Dept D, T
where    2 =< T.NUM_OF_PROJS and T.DNO = D.DNO
```

그러나 NEST-JA 규칙에는 NEST-JA 규칙을 적용하여 정규화한 질의의 결과가 정규화하지 않은 질의의 결과와 달라지는 문제인 Count bug[10]가 존재하여 아우터 조인(outer join)을 이용하여 이를 해결하기 위한 연구들(11-13)이 진행되어왔다.

마지막으로 Type-D 유형의 중첩 질의는 주 질의 블록의 where 절에 두 개의 하위 질의 블록을 피연산자로 갖는 연산이 있고, 이 중 하나 이상의 하위 질의 블록에 상관이 있는 경우를 말한다. 질의 7은 Type-D 유형 중첩 질의의 예로서, 자신이 속한 부서가 가진 모든 프로젝트에 참여하는 직원의 이름을 구하는 질의이다. 질의 7에서 where 절은 직원이 근무하는 모든 프로젝트의 리스트에 이 직원이 속한 부서가 가진 모든 프로젝트의 리스트가 포함되면 참이 되는데, 이는 division 연산을 이용하여 처리할 수 있다. Type-D 유형 중첩 질의의 정규화 규칙은 규칙 5와 같다.

질의 7. 자신이 속한 부서가 가진 모든 프로젝트에 참여하는 직원의 이름을 구하라.

```
select   E.EName
from     Emp E
where    (select W.PNO
          from WorksOn W
          where W.SSN = E.SSN)
          =
          (select P.PNO
          from Proj P
          where P.DNO = E.DNO)
```

1. 두 하위 절의 블록에서 상관 값과의 조인에 참여하는 속성들과 select 절의 속성들을 선택하여 각각의 임시 릴레이션을 생성한다.
2. 1에서 생성된 두 임시 릴레이션을 division하여 새로운 임시 릴레이션을 생성한다.
3. 주 질의 블록의 릴레이션과 2에서 생성한 임시 릴레이션을 조인한다.

규칙 5 NEST-D 규칙

3. XQuery 질의의 정규화 규칙

본 절에서는 확장된 XQuery 질의의 정규화 규칙들에 대해 설명한다. 제3.1절에서는 중첩된 XQuery 질의의 유형을 분류하고, 제3.2절에서는 중첩 유형에 따라 XQuery 질의를 정규화하는 규칙들에 대해 설명한다. 제3.3절에서는 XQuery 질의를 정규화 할 수 없는 경우에 대해 설명한다.

3.1 중첩 질의의 유형 분류

XQuery 질의는 SQL 질의와 마찬가지로 상관과 집계의 유무에 따라 중첩된 질의의 유형을 분류할 수 있으며, XQuery 질의에서의 상관과 집계는 SQL 질의에서와 유사한 의미를 가진다. 즉, 중첩된 XQuery 질의에서 중첩을 포함하는 FLWR 표현식을 주 FLWR 표현식이라고 하고 중첩된 FLWR 표현식을 하위 FLWR 표현식이라고 할 때, XQuery 질의에서의 상관은 하위 FLWR 표현식의 where 절에 주 FLWR 표현식의 for 절 변수가 나타나는 것이며, 집계는 FLWR 표현식이 집계 함수의 인자로 쓰인 것을 의미한다.

중첩된 XQuery 질의는 SQL 질의와 마찬가지로 Type-A, Type-N, Type-J, Type-JA, 그리고 Type-D 유형으로 분류할 수 있다. 먼저 Type-A, Type-N, Type-J, Type-JA 유형에 대해 설명하고, 다음으로 Type-D 유형에 대해 설명하도록 한다.

Type-A 유형의 중첩 질의는 하위 FLWR 표현식에 상관이 없고, 집계만 있는 경우이고, Type-N 유형의 중첩 질의는 하위 FLWR 표현식에 상관과 집계가 모두 없는 경우이다. Type-J 유형의 중첩 질의는 하위 FLWR 표현식에 상관만 있고, 집계는 없는 경우이고, Type-JA 유형의 중첩 질의는 하위 FLWR 표현식에 상관과 집계가 모두 있는 경우이다.

이와 같이 XQuery 질의의 Type-A, Type-N, Type-J, Type-JA 중첩 유형이 SQL 질의의 중첩 유형과 유사함에도 불구하고 SQL 질의의 정규화 규칙을 그대로 적용할 수는 없다. 왜냐하면 SQL 질의의 정규화에서는 where 절에 중첩된 질의에 대해서만 유형을 분류하여 정규화하였으나, XQuery 질의를 구성하는 주요 표현식인 FLWR 표현식에서는 for 절, where 절, return 절에 모두 중첩이 가능하기 때문이다. 따라서 중첩된 XQuery 질의는 각 유형 별로, 그리고 각 유형 내에서 다시 각 절 별로 나누어 정규화 규칙을 적용해야 한다.

다음으로 Type-D 유형의 중첩 질의는 주 FLWR 표현식의 where 절에 두 개의 하위 FLWR 표현식을 피연산자로 갖는 연산이 있고, 이 중 하나 이상의 하위 FLWR 표현식에 상관이 있는 경우를 말한다. XQuery의 Type-D 유형 중첩 질의는 SQL 질의에서와 그 의미가 다르다. SQL의 Type-D 유형 중첩 질의에서는 where 절 연산이 두 하위 질의 블록 간의 포함 관계를 묻는 연산을 의미한다. 반면, XQuery의 Type-D 유형 중첩 질의에서는 where 절 연산이 두 하위 FLWR 표현식 간의 비교 연산을 의미한다. 따라서 SQL의 Type-D 유형 중첩 질의의 정규화 규칙과는 다른 새로운 정규화 규칙이 필요하다.

결론적으로 FLWR 표현식에서 가능한 중첩 유형을 표 형태로 정리하면 표 1과 같다. 여기서 흰색 원으로 표시된 칸은 기존 연구 [8]에서 제안된 XQuery 질의의 정규화 규칙이 있는 경우이다. 그리고 검은색 원으로 표시된 칸은 기존 연구 [8]에서 정규화 규칙을 제안하지 않았기 때문에 본 논문에서 새로운 정규화 규칙을 제안한 경우이다.

표 1 FLWR 표현식에서 가능한 중첩 유형의 분류

유형 \ 절	for	where	return
Type-A	●	●	●
Type-N	○	●	○
Type-J	○	●	○
Type-JA	●	●	●
Type-D	N/A	●	N/A

○ : 기존 연구 [8]에서 제안한 정규화 규칙
● : 본 논문에서 제안한 정규화 규칙

3.2 정규화 규칙

본 절에서는 XQuery 질의의 정규화 규칙에 대해 설명한다. 제3.2.1절에서는 Type-A 유형의 중첩 질의를 정규화 하는 규칙에 대해 설명하고, 제3.2.2절에서는 Type-N 유형의 중첩 질의를 정규화 하는 규칙에 대해 설명한다. 제3.2.3절에서는 Type-J 유형의 중첩 질의를 정규화하는 규칙에 대해 설명하고, 제3.2.4절에서는 Type-JA 유형의 중첩 질의를 정규화하는 규칙에 대해 설명한다. 마지막으로 제3.2.5절에서는 Type-D 유형의 중첩 질의를 정규화 하는 규칙에 대해 설명한다.

3.2.1 Type-A 중첩을 처리하는 규칙

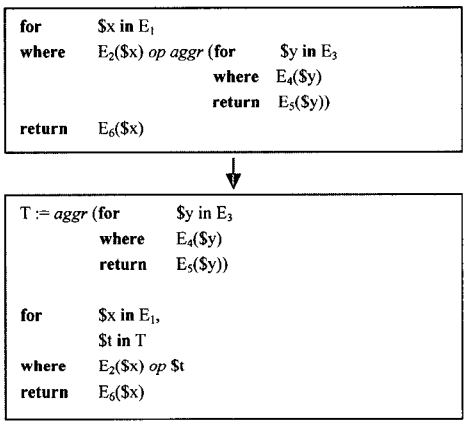
주 FLWR 표현식의 for 절에 Type-A 유형의 중첩이 있는 경우에는 하위 FLWR 표현식이 독립적으로 한 번만 수행되기 때문에 정규화 규칙을 적용할 필요가 없다. 주 FLWR 표현식의 where 절과 return 절에 Type-A 유형의 중첩이 있는 경우에는 참고 문헌 [8]에서 제안한 정규화 규칙이 없기 때문에 참고 문헌 [9]에서 제안한 SQL의 Type-A 유형 중첩 질의의 정규화 규칙을 확장하여 적용한다. 즉, SQL 문법으로 표현된 정규화 규칙을 동일한 의미를 가지는 XQuery 문법의 규칙으로 변환한다.

자세히 말하자면, XQuery의 Type-A 유형 중첩 질의는 SQL의 Type-A 유형 중첩 질의와 마찬가지로 하위 FLWR 표현식을 주 FLWR 표현식과 독립적으로 수행할 수 있다. 따라서 하위 FLWR 표현식을 인자로 하는 집계 함수를 미리 계산할 수 있으며, 계산된 값을 사용하여 주 FLWR 표현식을 수행할 수 있다. 즉, for 절에 미리 계산한 집계 함수의 값을 바인딩하는 임시 변수를 생성하고, where 절이나 return 절에 나타나는 집계 함수를 새로 생성한 for 절의 임시 변수로 치환하여 정규화한다.

Type-A 유형의 중첩 질의를 정규화 하는 규칙은 규칙 6과 같다. 그림에서는 정규화 규칙이 XQuery 질의로 표현되어 있으나, 실제 구현은 제4.2절에서 설명할 질의 그래프 상에서 이루어진다.

3.2.2 Type-N 중첩을 처리하는 규칙

주 FLWR 표현식의 for 절과 return 절에 Type-N



규칙 6 Type-A 유형 중첩 질의의 정규화 규칙

유형의 중첩이 있는 경우는 제2.2절에서 설명한 참고 문헌 [8]의 정규화 규칙 1과 규칙 2를 이용하여 중첩이 없는 질의로 변환한다. 주 FLWR 표현식의 where 절에 Type-N 유형의 중첩이 있는 경우에는 참고 문헌 [8]에서 제안한 정규화 규칙이 없기 때문에 참고 문헌 [9]의 NEST-N-J 규칙을 XQuery 문법에 맞게 확장하여 적용한다.

그러나 SQL 질의의 NEST-N-J 규칙은 하위 질의 블록의 결과에 중복된 값(duplicate)이 존재하지 않는다고 가정하였기 때문에, 하위 질의 블록의 결과에 중복된 값이 존재하는 경우에는 정규화된 질의의 결과가 정규화되지 않은 질의의 결과와 달라지는 문제가 있다. 이러한 문제는 SQL의 Type-N 유형 중첩 질의와 XQuery의 Type-N 유형 중첩 질의에서 동일하게 발생할 수 있다.

예를 들어, 그림 1의 XML 문서 "Departments.xml"와 그림 2의 XML 문서 "Projects.xml"에 대하여 Type-N 유형 중첩 질의인 질의 8을 수행하면, 결과 8을 얻는다.

질의 8. 프로젝트를 1개 이상 가지는 부서의 이름을 구하라.

```
for $x in document("Departments.xml")/Departments/Dept
where $x/DNO = (for $y in document("Projects.xml")/
Projects/Proj
return $y/DNO)
return $x/DName
```

```
<Projects>
  <Proj>
    <PNO> P1 </PNO>
    <PName> Newbenefits </PName>
    <PLoc> Houston </PLoc>
    <DNO> D2 </DNO>
  </Proj>
  <Proj>
    <PNO> P2 </PNO>
    <PName> Reorganization </PName>
    <PLoc> Bellaire </PLoc>
    <DNO> D2 </DNO>
  </Proj>
</Projects>
```

그림 2 XML 문서 "Projects.xml"

결과 8
<DName> Research </DName>

그러나 질의 8에 NEST-N-J 규칙을 그대로 적용하여 정규화하면 질의 8-1과 같고, 질의 결과 8-1은 본래의 질의 결과 8과 다르게 됨을 볼 수 있다. 이렇게 질의 결과가 달라지는 이유는

질의 8에서는 하위 FLWR 표현식의 결과인 (D2, D2)의 리스트에 $x/DNO의 값, 즉 D2가 있으면 $x/DName을 반환하므로 결과가 한번만 반환되는데, 질의 8-1에서는 $x/DNO를 $y/DNO와 비교하여 같은 값을 가지면 $x/DName을 반환하므로 DNO 값이 같은 Proj 요소의 수만큼 결과가 반복해서 반환되기 때문이다.

질의 8-1. 질의 8을 NEST-N-J 규칙을 이용하여 정규화함

```
for $x in document("Departments.xml")/Departments/Dept,
$y in document("Projects.xml")/Projects/Proj
where $x/DNO = $y/DNO
return $x/DName
```

결과 8-1
<DName> Research </DName>
<DName> Research </DName>

이러한 문제는 하위 FLWR 표현식을 먼저 수행하여 얻은 질의 결과에서 중복된 값들을 제거한 후, 그 결과를 주 FLWR 표현식과 조인하여 정규화 함으로써 해결할 수 있다. 문제점을 해결한 where 절의 Type-N 유형 중첩 질의의 정규화 규칙은 규칙 7과 같다. 질의 8에 규칙 7을 적용하면 질의 8-2와 같이 정규화할 수 있으며, 질의 결과는 정규화하기 전과 동일하다.

질의 8-2. 질의 8을 규칙 7의 정규화 규칙을 이용하여 정규화함

```
T := distinct(for $y in document("Projects.xml")/Projects/Proj
              return $y/DNO)

for $x in document("Departments.xml")/Departments/Dept,
    $t in T
where $x/DNO = $t
return $x/DName
```

```
for     $x in E1
where   E2($x) op (for     $y in E3
                   where   E4($y)
                   return  E5($y))
return  E6($x)
```

↓

```
T := distinct (for     $y in E3
               where   E4($y)
               return  E5($y))

for     $x in E1,
        $t in T
where   E2($x) op $t
return  E6($x)
```

규칙 7 Type-N 유형 중첩 질의의 정규화 규칙

3.2.3 Type-J 중첩을 처리하는 규칙

주 FLWR 표현식의 for 절과 return 절에 Type-J 유형의 중첩이 있는 경우에는 참고 문헌 [8]에서 제안한 정규화 규칙 1과 규칙 2를 이용하여 중첩이 없는 질의로 변환한다. 주 FLWR 표현식의 where 절에 Type-J 유형의 중첩이 있는 경우에는 참고 문헌 [8]에서 제안한 정규화 규칙이 없기 때문에 참고 문헌 [9]의 NEST-N-J 규칙을 XQuery 문법에 맞게 확장하여 적용한다.

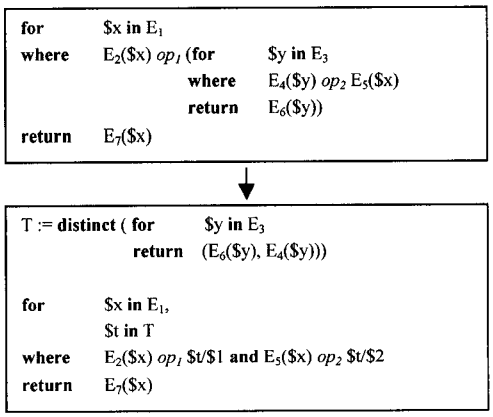
Type-N 유형인 경우와 마찬가지로 Type-J 유형의 중첩인 경우에도 NEST-N-J 규칙을 그대로 적용하여 정규화하면 정규화된 질의의 결과가 정규화되지 않은 질의의 결과와 달라지는 문제가 있다. 이러한 문제는 SQL의 Type-J 유형 중첩 질의와 XQuery의 Type-J 유형 중첩 질의에서 동일하게 발생할 수 있다.

주 FLWR 표현식의 where 절에 Type-J 유형의 중첩이 있는 경우에는 주 FLWR 표현식의 where 절에 Type-N 유형의 중첩이 있는 경우와 마찬가지로 하위 FLWR 표현식을 먼저 수행하여 얻은 질의 결과에서 중복된 값들을 제거하고, 주 FLWR 표현식과 조인해야 한다. 그러나, Type-J 유형의 중첩인 경우에는 하위 FLWR 표현식에 상관이 존재하기 때문에 하위 FLWR 표현식을 독립적으로 먼저 수행하여 질의 결과를 얻는 것이 불가능하다.

따라서 Type-J 유형의 중첩이 있는 질의는 먼저 하위 FLWR 표현식의 where 절에서 상관이 있는 조인 조건을 제거하고, 상관 값과의 조인에 참여하는 표현식을 return 절에 추가하여 반환하도록 하위 FLWR 표현식을 수정한다. 다음으로 수정된 하위 FLWR 표현식을 수행하여 얻은 결과에서 중복을 제거한 후, 그 결과를 이용하여 주 FLWR 표현식과 조인한다. 이때 앞에서 제거한 조인 조건을 사용하여 조인을 수행한다. Type-J 유형 중첩 질의의 정규화 규칙은 규칙 8과 같다. 규칙 8

에서 $1은 $t에 바인딩된 값의 쌍에서 첫번째 값을 의미하고, $2는 두 번째 값을 의미한다.

질의 9는 Type-J 유형 중첩 질의의 예이다. 질의 9에서 하위 FLWR 표현식은 상관 값인 $x/DNO가 주어질 때마다 한번씩 수행되어야 하므로 주 FLWR 표현식과 독립적으로 수행될 수 없다. 따라서 하위 FLWR 표현식에서 상관이 있는 조인 조건인 $y/DNO = $x/DNO를 제거하고 조인에 참여하는 $y/DNO를 return 절에 추가한다. 다음으로 하위 FLWR 표현식을 독립적으로 수행하여 중복이 없는 $y/PLoc와 $y/DNO의 쌍들을 구한다. 마지막으로 그 결과를 주 FLWR 표현식과 조인한다. 질의 9를 규칙 8의 Type-J 유형 중첩 질의의 정규화 규칙을 이용하여 정규화하면 질의 9-1과 같다.



규칙 8 Type-J 유형 중첩 질의의 정규화 규칙

질의 9. 부서가 위치한 곳에 진행되는 프로젝트를 하나 이상 가지는 부서를 구하라.

```
for $x in document("Departments.xml")/Departments/Dept
where $x/DLoc = (for $y in document("Projects.xml")/
Projects/Proj
                 where $y/DNO = $x/DNO
                 return $y/PLoc)
return $x/DName
```

질의 9-1. 질의 9를 정규화함

```
T := distinct(for $y in document("Projects.xml")/Projects
/Proj
              return ($y/PLoc, $y/DNO))
for $x in document("Departments.xml")/Departments/Dept,
    $t in T
where $x/DLoc = $t/$1 and $x/DNO = $t/$2
return $x/DName
```

3.2.4 Type-JA 중첩을 처리하는 규칙

주 FLWR 표현식의 for 절, where 절, 그리고 return 절에 Type-JA 유형의 중첩이 있는 경우에는 참고 문헌 [8]에서 제안한 정규화 규칙이 없기 때문에 참고 문헌 [9]의 NEST-JA 규칙을 XQuery 문법에 맞게 확장하여 적용한다. 즉, 하위 FLWR 표현식의 결과를 상관 값과의 조인에 참여하는 값으로 그룹화하여 집계 함수의 값들을 미리 계산하고, 그 결과를 주 FLWR 표현식과 조인하여 정규화 한다.

SQL 질의에서와 마찬가지로 XQuery의 Type-JA 유형의 중첩 질의도 그룹화만을 이용하여 정규화하면 Count bug가 존재할 수 있다. 본 논문에서는 Count bug를 해결하기 위한 방법들 중에서 가장 최근에 제안된 참고 문헌 [13]의 Magic Decorrelation 방법을 응용한다. 즉, Count bug가 발생할 수 있는 경우에는 Magic Decorrelation 방법을 응용하여 그룹화와 아우터 조인을 이용한 정규화 규칙을 적용한다.

XQuery에는 그룹화와 아우터 조인을 명시적으로 나타낼 수 있는 표현식이 존재하지 않기 때문에, Type-JA 유형의 중첩 질의를 정규화하는 규칙은 동일한 의미의 XQuery 질의로 표현하기 어렵다. 따라서 Type-JA 유형의 중첩 질의를 정규화하는 규칙은 제 4.3절에서 질의 그래프를 사용하여 설명하도록 하겠다.

3.2.5 Type-D 중첩을 처리하는 규칙

주 FLWR 표현식에 Type-D 유형의 중첩이 있는 경우에는 참고 문헌 [8]에서 제안한 정규화 규칙이 없고, 참고 문헌 [9]에서 정의한 SQL의 Type-D 유형 중첩과 의미가 다르기 때문에 새로운 정규화 규칙을 제안한다.

XQuery의 Type-D 유형 중첩 질의는 주 FLWR 표현식의 where 절에 두 개의 하위 FLWR 표현식을 피연산자로 갖는 연산이 있고, 이 중에서 하나 이상의 하위 FLWR 표현식에 상관이 있는 경우를 말한다. 여기서 두 하위 FLWR 표현식 간에는 상관이 존재하지 않기 때문에, 각 하위 FLWR 표현식을 독립적으로 수행하여 처리할 수 있다. 따라서 XQuery의 Type-D 유형 중첩 질의는 각 하위 FLWR 표현식의 중첩 유형에 따라 해당하는 정규화 규칙을 차례로 적용함으로써 정규화 할 수 있다.

질의 10은 Type-D 유형 중첩 질의의 예이다. 예에서 사용된 "Departments.xml" 문서와 "Projects.xml" 문서는 각각 앞에서 설명한 그림 1과 그림 2의 XML 문서이며 "WorksOn.xml" 문서는 직원의 근무 정보를 기록하는 XML 문서로서 그림 3과 같다. "WorksOn.xml" 문서의 각 WorksOn 요소는 직원의 식별 번호를 나타내는 요소인 SSN, 직원이 참여하는 프로젝트의 식별 번호를 나타내는 요소인 PNO, 그리고 직원이 프로젝트에 대해 근무한 시간을 나타내는 요소인 Hours로 구성된다.

질의 10. "Houston"에서 진행되는 프로젝트에 참여하고 있는 직원들의 이름을 구하라.

```
for $x in document("Departments.xml")/Departments
/Dept/Emp
where (for $y in document("WorksOn.xml")/Works-
OnList/ WorksOn
        where $y/SSN = $x/SSN
        return $y/PNO)
        =
       (for $z in document("Projects.xml")/Projects/Proj
        where $z/PLoc = "Houston"
        return $z/PNO)
return $x/EName
```

```
<WorksOnList>
  <WorksOn>
     <SSN> 123456789 </SSN>
     <PNO> P1 </PNO>
     <Hours> 32.5 </Hours>
  </WorksOn>
  <WorksOn>
     <SSN> 333445555 </SSN>
     <PNO> P2 </PNO>
     <Hours> 40.0 </Hours>
  </WorksOn>
</WorksOnList>
```

그림 3 XML 문서 "WorksOn.xml"

이 세 XML 문서에 대한 질의 10은 각 하위 FLWR 표현식의 중첩 유형에 따라 정규화 규칙을 차례로 적용하여 질의 10-1와 같이 정규화할 수 있다. 그러나 질의 10에서 두 하위 FLWR 표현식의 결과 간에 같은 값이 2개 이상 존재하면 정규화된 질의의 결과가 정규화되지 않은 질의의 결과와 달라질 수 있다. 예를 들어, 질의 10의 where 절에서 하나의 Emp 요소에 대해 좌측의 하위 FLWR 표현식의 결과가 (P1, P2)이고 우측의 하위 FLWR 표현식의 결과가 (P1, P2)이면 $x/EName이 한번 반환된다. 반면, 질의 10-1과 같이 정규화하게 되면 $x/EName이 두 번 반환된다.

질의 10-1. 질의 10을 정규화함

```
T1 := distinct(for $z in document("Projects.xml")/
Projects/Proj
                where $z/PLoc = "Houston"
                return $z/PNO)
T2 := distinct(for $y in document("WorksOn.xml")/
WorksOnList/WorksOn
                return ($y/SSN, $y/PNO))
for $x in document("Departments.xml")/Departments
/Dept/Emp,
    $t1 in T1,
    $t2 in T2
where $x/SSN = $t2/$1 and $t1 = $t2/$2
return $x/EName
```

이러한 문제를 해결하기 위해서는 각 정규화 규칙에 따라 미리 수행된 하위 FLWR 표현식의 결과를 주 FLWR 표현식과 직접 조인하지 않고, 중복된 값들을 먼저 제거한 다음에 조인해야 한다. 따라서 질의 10은 질의 10-2와 같이 정규화할 수 있다. 문제점을 해결한 Type-D 유형 중첩 질의의 정규화 규칙은 규칙 9와 같다.

질의 10-2. 질의 10-1의 문제점을 해결하여 질의 10을 정규화함

```
T1 := distinct(for $z in document("Projects.xml")/
Projects/Proj
                where $z/PLoc = "Houston"
                return $z/PNO)

T2 := distinct(for $y in document("WorksOn.xml")/
WorksOnList/WorksOn
                return ($y/SSN, $y/PNO))

T3 := distinct(for $t1 in T1,
                   $t2 in T2
               where $t1 = $t2/$2
               return $t2/$1)

for $x in document("Departments.xml")/Depart-
ments/Dept/Emp,
    $t3 in T3
where $x/SSN = $t3
return $x/EName
```

1. 중첩된 두 하위 FLWR 표현식 중에서 하나의 하위 FLWR 표현식이 주 FLWR 표현식과 이루는 중첩 유형에 따라 하위 FLWR 표현식을 미리 수행하여 임시 릴레이션을 생성한다.
2. 1에서 정규화하지 않은 다른 하위 FLWR 표현식이 주 FLWR 표현식과 이루는 중첩 유형에 따라 하위 FLWR 표현식을 미리 수행하여 임시 릴레이션을 생성한다.
3. 1과 2에서 생성된 임시 릴레이션들을 조인한 후에 주 FLWR 표현식과의 조인에 참여하는 값을 선택하여 중복을 제거한 임시 릴레이션을 생성한다.
4. 주 FLWR 표현식과 3에서 생성된 임시 릴레이션을 조인한다.

규칙 9 Type-D 유형 중첩 질의의 정규화 규칙

3.3 정규화의 한계

XQuery 질의의 정규화는 중첩된 질의가 반복하여 수행되지 않도록 하기위한 것이다. 그러나 중첩된 질의가 반드시 구체화(materialize)된 XML 형태의 중간 결과를 생성해야 하는 경우에는 본 논문에서 제안한 정규화 규칙을 적용하여 중첩을 제거하면 질의의 결과가 달라지는 문제점이 있다.

예를 들어, 질의 11은 부서의 이름과 그 부서에 속하는 프로젝트의 이름들을 Project_List 요소로 생성하여 돌려주는 질의이다. 이와 같이 하위 FLWR 표현식이 요소 생성자 안에 쓰인 경우에는 주 FLWR 표현식의 for 절 변수에 바인딩되는 Dept 요소 하나마다 하나의 Project_List 요소를 결과로 반환해 주어야 한다. 따라

서 질의 11의 결과는 결과 11과 같다.

질의 11. 부서별로 부서의 이름과 그 부서에 속하는 프로젝트의 리스트를 구하라.

```
for $x in document("Departments.xml")/Departments/Dept
return <Project_List>
         { $x/DName }
         <Projects>
           {
             for $y in document("Projects.xml")/Projects/Proj
             where $y/DNO = $x/DNO
             return $y/PName
           }
         </Projects>
       </Project_List>
```

결과 11

```
<Project_List>
  <DName> Research </DName>
  <Projects>
    <PName> Newbenefits </PName>
    <PName> Reorganization </PName>
  </Projects>
</Project_List>
```

그러나 요소 생성자를 무시하고 정규화 규칙을 적용하여 질의 11-1과 같이 정규화하면 for 절 변수에 바인딩되는 Dept 요소 하나마다 같은 DNO를 가지는 Proj 요소들의 수만큼 Project_List 요소가 생성되어 결과로 반환되므로 질의 결과는 결과 11-1과 같고, 정규화 되지 않은 질의의 결과인 결과 11과 달라진다.

질의 11-1. 질의 11을 정규화함.

```
for $x in document("Departments.xml")/Departments/Dept,
    $y in document("Projects.xml")/Projects/Proj
where $y/DNO = $x/DNO
return <Project_List>
         { $x/DName }
         <Projects>
           { $y/PName }
         </Projects>
       </Project_List>
```

결과 11-1

```
<Project_List>
  <DName> Research </DName>
  <Projects>
    <PName> Newbenefits </PName>
  </Projects>
</Project_List>
<Project_List>
  <DName> Research </DName>
  <Projects>
    <PName> Reorganization </PName>
  </Projects>
</Project_List>
```

4. XQuery 질의의 정규화 구현

본 절에서는 제 3절의 설계를 바탕으로 XQuery 질의의 정규화의 구현에 대해 설명한다. 제 4.1절에서는 정규화의 과정에 대해 설명하고, 제 4.2절에서는 정규화를 위해 사용되는 내부 자료 구조 및 질의 그래프에 대해 설명한다. 제 4.3절에서는 내부 자료 구조 및 질의 그래프 상에서의 정규화 알고리즘에 대해 설명한다.

4.1 정규화 과정

XQuery 질의는 그림 4와 같은 단계를 거쳐 정규화된다. 먼저, 사용자로부터 질의를 입력 받아서 어휘 분석(lexical analysis) 및 구문 분석(syntactic analysis)을 수행하여 구문 오류(syntax error)를 검사한다. 구문 오류가 없으면 요약 구문 트리를 생성하고 의미 분석(semantic analysis)을 수행하여 의미 오류를 검사한다. 의미 오류가 없으면 정규화에 필요한 내부 자료 구조를 생성한다.

다음으로 생성된 내부 자료 구조를 사용하여 정규화 규칙들을 적용한다. 그런데 입력된 질의의 구조를 그대로 반영하는 내부 자료 구조만으로는 상관과 집계의 유무를 파악하기가 어렵다는 문제점이 있다. 그러므로 본 논문에서는 정규화 과정을 참고 문헌 [8]의 정규화 규칙들을 적용하는 1차 정규화 과정과 본 논문에서 제안한 정규화 규칙들을 적용하는 2차 정규화 과정으로 분리한다. 즉, 상관과 집계의 유무에 따라 XQuery 질의의 중첩 유형을 구분하지 않는 참고문헌 [8]의 정규화 규칙들

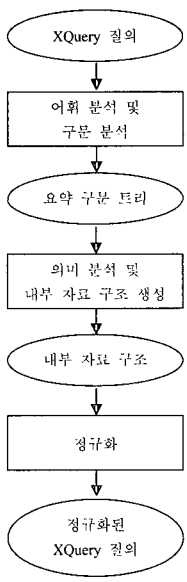


그림 4 정규화 과정

을 1차 정규화 과정에서 내부 자료 구조만을 사용하여 먼저 처리한다. 그런 다음, 2차 정규화 과정에서 정규화된 내부 자료 구조를 상관과 집계의 유무의 파악이 쉬운 질의 그래프로 변환하고, 질의 그래프를 이용하여 상관과 집계의 유무에 따라 각 중첩 유형 별로 제 3.2절에서 설계한 정규화 규칙들을 적용한다.

마지막으로 정규화된 XQuery 질의를 반환한다. 정규화된 XQuery 질의는 내부 자료 구조와 질의 그래프로 표현되어 있다.

4.2 내부 자료 구조 및 질의 그래프

XQuery 질의의 정규화 과정에서는 내부 자료 구조(internal data structures)와 질의 그래프(query graph)를 이용하여 질의를 정규화한다. 내부 자료 구조는 입력된 질의의 구조를 그대로 반영한 자료 구조로서 경로 표현식을 나타내기 위한 자료 구조, FLWR 표현식을 나타내기 위한 자료 구조, 요소 생성자를 나타내기 위한 자료 구조, 집계 함수를 나타내기 위한 자료 구조 등이 있다.

질의 그래프는 QGM(Query Graph Model)[14]을 확장한 것으로서, 내부 자료 구조상에서 알기 어려운 FLWR 표현식 간의 상관 유무를 알기 쉽게 표현해 주는 자료 구조이다. 여기서 QGM은 IBM DB2 DBMS에서 질의 처리와 질의 최적화를 위해 SQL 질의를 내부적으로 표현하는데 사용하는 자료 구조이다.

질의 그래프는 FLWR 표현식을 나타내는 질의 상자들로 구성되며, 각 질의 상자는 헤드(head)와 바디(body)로 구성된다. 먼저 헤드는 FLWR 표현식의 return 절에 대응되는 것으로서 바디에서 반환되어야 할 질의의 결과를 나타낸다. 그리고 바디는 FLWR 표현식의 for 절과 where 절에 대응되는 것으로서 for 절에 대응되는 노드(node)와 where 절에 대응되는 에지(edge)로 구성된다. 바디에서 동일한 노드에 대한 에지는 실렉션 조건을 의미하며, 서로 다른 노드 간의 에지는 조인 조건을 의미한다. 특히, 서로 다른 질의 상자에 속한 노드 간의 에지는 각 질의 상자가 나타내는 FLWR 표현식 간에 상관이 있음을 의미한다. 마지막으로 각 질의 상자는 distinct를 나타내는 플래그와, group by와 order by에 대한 정보를 부가적으로 가질 수 있다.

예를 들어 질의 12와 같은 Type-JA 유형의 중첩 질의를 질의 그래프로 나타내면 그림 5와 같다. 질의에서 주 FLWR 표현식과 하위 FLWR 표현식은 각각 질의 상자 QB_1과 QB_2에 해당된다. 주 FLWR 표현식의 for 절은 질의 상자 QB_1에서 Dept 요소들이 바인딩되는 노드 $x로 표현된다. 그리고 where 절은 실렉션 조건을 나타내는 에지로 표현되는데, 실렉션 조건은 피연산자로 하위 FLWR 표현식을 나타내는 질의 상자 QB_2을 갖는다. 마지막으로 return 절은 헤드로 표현된다. 하위 FLWR 표현식의 for 절은 질의 상자 QB_2에서 Proj 요소들이 바인딩되는 노드 $y로 표현된다. 그리고 where 절은 조인 조건을 나타내는 노드 $x와 노드 $y 간의 에지로 표현되는데, 이 에지는 서로 다른 질의 상자에 속하는 노드 간의 에지이므로 두 질의 상자 간에 상관이 존재함을 나타낸다. 마지막으로 return 절은 헤드로 표현된다.

질의 12. 2개 이상의 프로젝트를 가지는 부서의 이름을 구하라.

```
for $x in document("Departments.xml")/Departments/Dept
where 2 =< count(for $y in document("Projects.xml")/Projects/Proj
                 where $y/DNO = $x/DNO
                 return $y/PNO)
return $x/DName
```

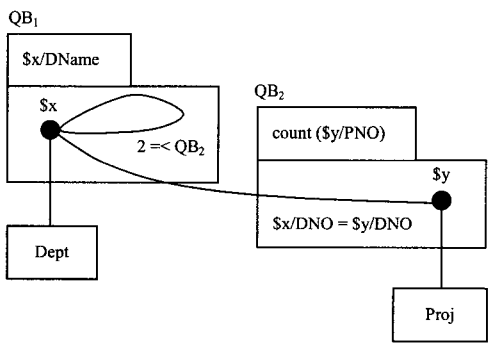


그림 5 질의 그래프의 예

4.3 내부 자료 구조 및 질의 그래프 상에서의 정규화 알고리즘

내부 자료 구조를 이용한 1차 정규화 과정의 알고리즘은 그림 6과 같다. 중첩이 있는 질의의 내부 자료 구조는 중첩을 포함하는 표현식의 내부 자료 구조를 부모 노드로 하고, 중첩된 표현식의 내부 자료 구조를 자식 노드로 하는 트리 구조로 볼 수 있다. Normalize_GDS 프로시저는 입력 받은 내부 자료 구조를 루트(root) 노드로 하는 트리를 깊이 우선 탐색으로 순회하면서 정규화 규칙을 적용할 수 있는 패턴을 검색하여 상향식(bottom-up)으로 정규화 한다. 즉, 입력된 내부 자료 구조 E_1을 스택에 저장하고(라인 4), E_1의 각 자식 노드인 내부 자료 구조 E_2에 대해 E_2가 리프(leaf) 노드가 아니면 E_2를 인자로 하여 Normalize_GDS 프로시저를 순환적으로 호출한다(라인 7~10). 만일 내부 자료 구조 E_2가 리프 노드이면 스택에서 원소가 하나 남을 때까지

원소를 2개씩 꺼내어 정규화 규칙을 적용할 수 있는 패턴이면 해당되는 정규화 규칙을 적용한다(라인 11~24).

```
1:  Procedure Normalize_GDS(내부 자료 구조 E1)
2:  /* S는 내부 자료 구조를 저장하는 스택 */
3:  begin
4:    S.push(E1)
5:    for each E1이 자식 노드로 갖는 내부 자료 구조 E2
6:    begin
7:      if (E2가 리프 노드가 아님)
8:      then begin
9:        Normalize_GDS(E2)
10:     end
11:     else
12:       while (S의 원소의 개수가 2개 이상임)
13:       begin
14:         inner := S.pop()
15:         outer := S.pop()
16:         if((inner, outer)의 쌍이 정규화 규칙을 적용해야 하는 패턴임)
17:         then begin
18:           해당하는 정규화 규칙을 적용
19:           S.push((inner, outer)의 쌍을 정규화한 내부 자료 구조)
20:         end
21:         else
22:           S.push(outer)
23:         end
24:       end
25:     end
26:   end
27: end
```

그림 6 내부 자료 구조상에서의 정규화 알고리즘

2차 정규화 과정에서는 먼저 1차 정규화 과정에서 정규화된 내부 자료 구조를 질의 그래프로 변환한다. 질의 그래프에서는 질의 상자 간의 조인 에지를 보고 상관의 유무를 파악할 수 있으며, 질의 상자의 헤드를 보고 집계의 유무를 파악할 수 있다.

2차 정규화 과정에서는 상관과 집계의 유무에 따라 각 중첩 유형 별로 정규화 규칙을 적용한다. 질의 그래프를 이용한 2차 정규화 과정의 알고리즘은 그림 7, 그림 8와 같다. 그림 7의 Normalize_QG 프로시저는 입력 받은 질의 상자가 나타내는 FLWR 표현식의 각 절에 중첩이 있으면 ApplyNormalizationRules 프로시저를 호출하는 모듈이고, 그림 8의 ApplyNormalizationRules 프로시저는 중첩 유형 별로 정규화 규칙을 적용하는 모듈이다.

Normalize_QG 프로시저의 라인 3~6에서는 FLWR 표현식의 for 절 중첩을 제거한다. 즉, 입력 받은 질의 상자 QB_1의 노드에 다른 질의 상자 QB_2가 바인딩되어 있으면(라인 3) 질의 상자 QB_2를 인자로 하여 ApplyNormalizationRules 프로시저를 호출한다(라인 5). 라인 7~17에서는 FLWR 표현식의 where 절 중첩을 제거한다. 즉, 입력 받은 질의 상자 QB_1의 실렉션 조건에 하나 이상의 다른 질의 상자가 포함되어 있을 때(라인 7), 조건의 피연산자 중에서 하나만 질의 상자인 경우에는

```
1:  Procedure Normalize_QG (질의 상자 QB1)
2:  begin
3:    if (QB1의 노드에 다른 질의 상자 QB2가 바인딩됨)
4:    then begin
5:      ApplyNormalizationRules(QB2)
6:    end
7:    if (QB1의 실렉션 조건에 하나 이상의 다른 질의 상자가 포함됨)
8:    then begin
9:      if (피연산자 중에서 하나만 질의 상자 QB2임)
10:     then begin
11:       ApplyNormalizationRules(QB2)
12:     end
13:     if (피연산자가 두 개 모두 질의 상자이고 상관이 존재함)
14:     then begin
15:       Type-D 유형의 중첩 질의를 정규화 하는 규칙 적용
16:     end
17:   end
18:   if (QB1의 헤드에 다른 질의 상자 QB2가 포함됨)
19:   then begin
20:     ApplyNormalizationRules(QB2)
21:   end
```

그림 7 질의 그래프 상에서의 정규화 알고리즘

```
1:  Procedure ApplyNormalizationRules(질의 상자 QB2)
2:  begin
3:    Normalize_QG(QB2)
4:    if (QB2의 조인 조건에 상관이 없음)
5:    then begin
6:      if (QB2의 헤드에 집계 함수가 있음)
7:      then begin
8:        Type-A 유형의 중첩 질의를 정규화 하는 규칙 적용
9:      end
10:     else
11:       Type-N 유형의 중첩 질의를 정규화 하는 규칙 적용
12:     end
13:   end
14:   else
15:     if (QB2의 헤드에 집계 함수가 없음)
16:     then begin
17:       Type-J 유형의 중첩 질의를 정규화 하는 규칙 적용
18:     end
19:     else
20:       Type-JA 유형의 중첩 질의를 정규화 하는 규칙 적용
21:     end
22:   end
23: end
```

그림 8 ApplyNormalizationRules 프로시저

중첩된 질의 상자 QB_2를 인자로 하여 ApplyNormalizationRules 프로시저를 호출한다(라인 9~12). 그리고 피연자가 두 개 모두 질의 상자이고 하나 이상의 질의 상자에 상관이 존재하는 경우에는 Type-D 유형의 중첩 질의를 정규화 하는 규칙을 적용한다(라인 13~16). 마지막으로 라인 18~21에서는 FLWR 표현식의 return 절 중첩을 제거한다. 즉, 입력 받은 질의 상자 QB_1의 헤드에 다른 질의 상자 QB_2가 포함되어 있으면(라인 18) 질의 상자 QB_2를 인자로 하여 ApplyNormalizationRules 프로시저를 호출한다(라인 20).

그림 8의 ApplyNormalizationRules 프로시저에서는 입력 받은 질의 상자 QB_2를 인자로 하여 Normalize_QG 프로시저를 호출함으로써 QB_2를 먼저 정규화한다(라인 3). 그런 다음 QB_2의 조인 조건에 QB_1에 대한 상관이 없고 헤드에 집계 함수가 있으면 Type-A 유형의 중첩 질의를 정규화 하는 규칙을 적용하고(라인 4~9), 조인 조건에 상관이 없고 헤드에 집계 함수가 없으면 Type-N 유형의 중첩 질의를 정규화 하는 규칙을 적용한다(라인 11). 그리고 QB_2의 조인 조건에 QB_1에 대한 상관이 있고 헤드에 집계 함수가 없으면 Type-J 유형의 중첩 질의를 정규화 하는 규칙을 적용하고(라인 14~18), 조인 조건에 상관이 있고 헤드에 집계 함수가 있으면 Type-JA 유형의 중첩 질의를 정규화 하는 규칙을 적용한다(라인 20).

예를 들어 그림 5와 같은 질의 그래프에 2차 정규화 과정의 알고리즘을 적용하면 그림 9와 같이 정규화된다. 그림에서 최종적으로 정규화된 질의 그래프에는 질의 상자 간의 상관을 나타내는 에지가 존재 하지 않음을 볼 수 있다. 그리고 이러한 질의 그래프가 실제 시스템에서 처리될 때는 질의 상자 QB_2가 먼저 수행되고 그 결과를 질의 처리에 이용하기 때문에, 질의 상자 QB_2가 반복하여 수행되지 않고 한 번만 수행된다.

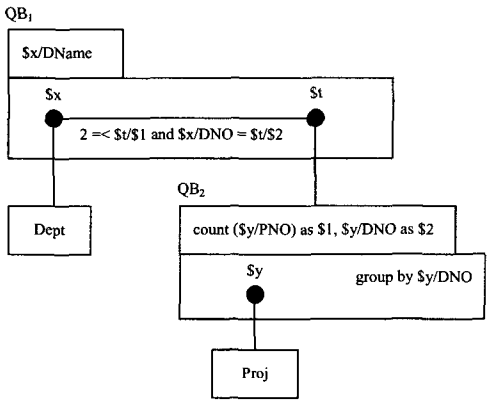


그림 9 질의 그래프의 정규화 예

5. 결 론

XQuery는 현재 W3C에서 표준화가 진행 중인 XML 질의 언어로서, SQL의 select-from-where 문과 유사한 기능을 가지는 FLWR 표현식을 제공한다. FLWR 표현식은 각 절 안에 또 다른 FLWR 표현식이 중첩될 수 있는 특징을 가지는데, 중첩된 FLWR 표현식은 중첩된 루프문의 실행을 야기하여 비효율적인 문제가 있다. 따라서 중첩된 XQuery 질의를 동일한 의미를 가지면서 보다 효율적으로 실행될 수 있는 질의로 변환하는 정규화 규칙들[8]이 제안되었다.

그러나 제안된 정규화 규칙들은 XQuery 질의를 SQL 질의로 변환하기 쉬운 형태로 바꾸기 위한 것이었기 때문에 매우 제한적이라는 문제점이 있다. 즉, FLWR 표현식의 where 절에 중첩이 있는 경우에 대해 고려하지 않았으며, for 절과 return 절에 중첩이 있는 경우에도 FLWR 표현식이 집계 함수의 인자로 중첩된 경우에 대해서는 정규화 규칙을 제안하지 않았다.

본 논문에서는 SQL 질의의 정규화 규칙들[9]을 확장하여 XQuery 질의를 구성하는 FLWR 표현식의 모든 절에 나타나는 중첩을 제거할 수 있는 정규화 규칙들을 제안하였다. 이를 위해 기존에 제안된 SQL 질의의 정규화 규칙들을 XQuery 문법에 맞게 변환하였으며, Type-N 유형과 Type-J 유형의 중첩 질의에서 중복된 값으로 인해 발생할 수 있는 문제를 해결하였다. 그리고, SQL 질의에서와 의미가 다른 Type-D 유형의 중첩 질의에 대해서는 새로운 규칙을 제안하였다.

본 논문의 주요 공헌은 다음과 같다. 첫째, 상관과 집계의 유무에 따라 XQuery 질의의 중첩 유형을 분류하고, 각 유형 내에서 다시 각 절 별로 나누어 정규화 규칙을 제안하였다. 둘째, 중첩된 XQuery 질의에 정규화 규칙들을 적용하는 세부 알고리즘들을 제안하였다. 이를 위해 먼저 XQuery 질의의 구조와 의미를 나타내는 내부 자료 구조와 질의 그래프를 제안하였으며, 내부 자료 구조와 질의 그래프에 정규화 규칙들을 적용하는 세부 알고리즘들을 제안하였다.

참 고 문 헌

[1] Simon, H., *Strategic Analysis of XML for Web Application Development*, Computer Research Corp., 2000.

[2] Chamberlin, D., Robie, J., and Florescu, D., "Quilt: An XML Query Language for Heterogeneous Data Sources," In *Proc. SIGMOD/PODS Workshop on the Web and Databases*, Dallas, Texas, pp. 53-62, May 2000.

[3] Robie, J., Lapp, J., and Schach, D., "XML Query Language (XQL)," In *Proc. QL'98*, Cambridge, Mass., Dec. 1998.

[4] Deutsch, A. et al., "XML-QL: A Query Language for XML," In *Proc. 8th Int'l Conf. on World Wide Web*, Toronto, May 1999.

[5] World Wide Web Consortium, XML Path Language (XPath) Version 1.0, W3C Recommendation, Nov. 1999 (availiable from http://www.w3.org/TR/xpath.html).

[6] World Wide Web Consortium, XQuery 1.0: An XML Query Language, W3C Working Draft, Aug.

2003 (availiable from http://www.w3.org/TR/xquery/).
[7] World Wide Web Consortium (availiable from http://www.w3.org/).
[8] Manolescu, I., Florescu, D., and Kossmann, D., "Answering XML Queries over Heterogeneous Data Sources," In *Proc. 27th Int'l Conf. on Very Large Data Bases*, Roma, Italy, pp. 241-250, Sept. 2001.
[9] Kim, W., "On Optimizing an SQL-like Nested Query," *ACM Trans. on Database Systems*, Vol. 7, No. 3, pp. 443-469, Sept. 1982.
[10] Kiessling, W., SQL-like and Quel-like Correlation Queries with Aggregates Revisited, UCB/ERL Memo 84/75, Electronics Research Laboratory, Univ California, Berkeley, Sept. 1984.
[11] Ganski, R. and Wong, H., "Optimization of Nested SQL Queries Revisited," In *Proc. Int'l Conf. on Management of Data*, ACM SIGMOD, San Francisco, USA, pp. 23-33, May 1987.
[12] Dayal, U., "Of Nests and Trees: A Unified Approach to Processing Queries that contain Nested Subqueries, Aggregates and Quantifiers," In *Proc. 13th Int'l Conf. on Very Large Data Bases*, Brighton, England, pp. 197-208, Sept. 1987.
[13] Seshadri, P., Pirahesh, H., and Leung, T., "Complex Query Decorrelation," In *Proc. the 17th Int'l Conf. on Data Engineering*, pp. 450-458, Feb. 1996.
[14] Pirahesh, H., Hellerstein, J.M., and Hasan, W., "Extensible/Rule Based Query Rewrite Optimization in Starburst," In *Proc. Int'l Conf. on Management of Data*, ACM SIGMOD, San Diego, California, pp. 39-48, June 1992.

김 서 영
2002년 2월 고려대학교 컴퓨터학과 컴퓨터학전공 학사. 2004년 2월 한국과학기술원 전자전산학과 전산학전공 석사. 2004년 2월~현재 삼성전자 기술총괄 소프트웨어센터 연구원. 관심분야는 데이타베이스 시스템, XML, XQuery

이 기 훈
2000년 2월 한국과학기술원 전자전산학과 전산학전공 학사. 2002년 8월 한국과학기술원 전자전산학과 전산학전공 석사 2002년 9월~현재 한국과학기술원 전자전산학과 전산학전공 박사 과정. 관심분야는 XML, IR, Compression, Query Optimization

황 규 영
정보과학회논문지 : 컴퓨팅의 실제
제 10 권 제 1 호 참조