경로 식별자를 이용한 다중 정규경로 처리기법
(Processing of Multiple Regular Path Expressions using PID)

김 종 익 † 정 태 선 † 김 형 주 ††
(Jongik Kim) (Tae-Sun Chung) (Hyoung-Joo Kim)

요 약 XML에 대한 질의는 데이타 그래프 내의 경로에 대한 질의를 기반으로 하며 그래프 내의 임의의 경로를 표시하기 위해 정규식을 사용한다. 일반적으로 의미있는 질의를 표현하기 위해서는 질의 내에 여러 개의 정규식을 포함하는 경우가 많다. 그럼에도 불구하고 기존의 연구는 주로 단일 정규식으로 이루어진 질의의 최적화 방법을 다루고 있다.

본 논문에서는 데이타 그래프를 탐색하지 않고 다중 정규 경로 질의를 처리할 수 있는 방법을 제시한다. 본 논문에서는 다중 정규 경로를 효율적으로 처리하기 위해서 데이타 그래프 상의 임의의 두 노드 사이에 경로가 존재하는지를 직접 알 수 있는 방법인 경로 식별자를 제안하고 경로 식별자를 만드는 방법을 제공한다. 또한 본 논문에서 제안된 방법을 구현하여 그 성능에 대한 결과를 제공한다.

키워드 : XML, 정규 경로식, 질의 처리, 경로 식별자

Abstract Queries on XML are based on paths in the data graph, which is represented as an edge labeled graph model. All proposed query languages for XML express queries using regular expressions to traverse arbitrary paths in the data graph. A meaningful query usually has several regular path expressions in it, but much of recent research is more concerned with optimizing a single path expression.

In this paper, we present an efficient technique to process multiple path expressions in a query. We developed a data structure named as the path identifier(PID) to identify whether two given nodes lie on the same path in the data graph or not, and utilized the PID for efficient processing of multiple path expressions. We implement our technique and present preliminary performance results.

Key words : XML, regular path, query processing, path identifier

1. 서 론

XML[4]이 인터넷 상의 문서교환의 표준으로 채택되면서 최근 XML에 대한 연구가 활발하게 진행되고 있다. XML 데이타는 고정된 스키마가 없고 구조정보가 데이타 내에 포함되어 있다는 것이 큰 특징이다. 이러한 특징은 XML 데이타가 비정형 데이타의 한 종류라는 것을 보여준다. 따라서 XML 데이타는 비정형 데이타를 표현하기 위해 사용되는 레이블이 있는 그래프 모델로 표현될 수 있다.

그림 1은 비정형 데이타 모델로 표현된 영화에 대한 XML 데이타를 보여준다. XML을 위한 질의 언어는 비정형 데이타에 대한 질의 언어와 마찬가지로 데이타 그래프내의 경로에 대한 질의를 기반으로 하고 있으며, 데이타 그래프 내의 임의의 경로를 표현하기 위해 정규식을 사용하여 경로를 표현하고 있다는 것이 특징이다. 예를 들어 다음 질의는 BradPitt가 출연한 영화를 모두 찾는 질의이다.

```
select m
from _*.movie m in root, _*.actor._*.name.
        BradPitt n in m
```

위의 질의는 질의 내에 _*.movie와 _*.actor._*.name.

· 본 연구는 두뇌한국21 사업에 의하여 지원 받았음.
† 비 회 원 : 서울대학교 컴퓨터공학부
jikim@oopsla.snu.ac.kr
tschung@oopsla.snu.ac.kr
†† 종신회원 : 서울대학교 컴퓨터공학부 교수
hjk@oopsla.snu.ac.kr
논문접수 : 2001년 7월 18일
심사완료 : 2002년 5월 9일

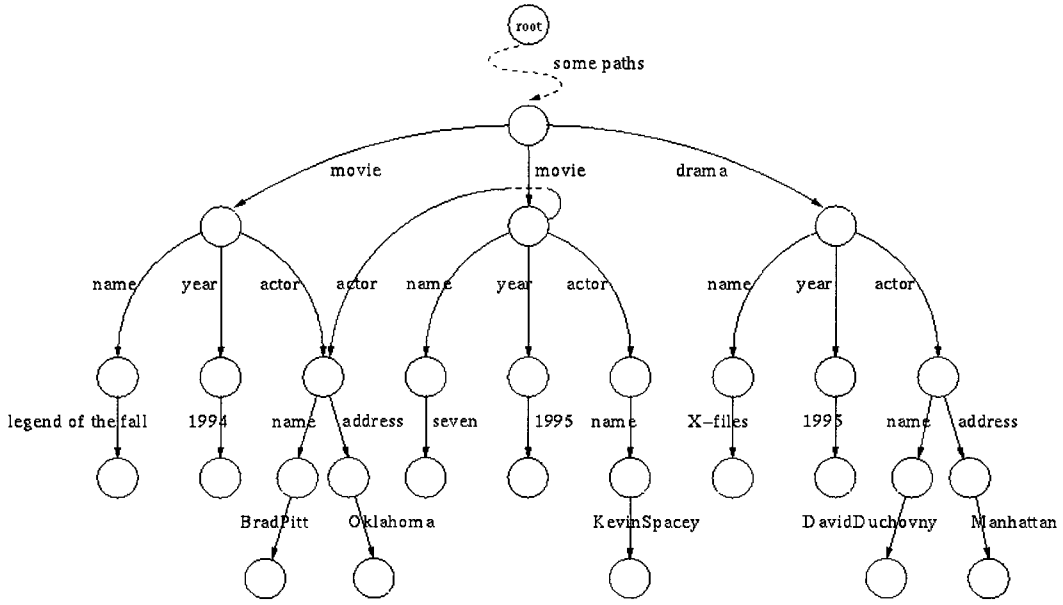


그림 1 영화 데이타 그래프

BradPitt의 두개의 정규경로식을 가지는 질의이다. 일반적으로 보다 의미 있는 질의를 구성하기 위해서는 위의 질의와 같이 하나의 질의 내에 여러 개의 정규경로가 포함되는 경우가 많다. 위의 질의를 처리하는 가장 간단한 방법은 데이타 그래프 전체를 탐색해 가면서 정규식을 만족하는 경로를 찾아내는 것이다. 하지만 이러한 방법은 매우 비효율적임을 쉽게 알 수 있다. 최근에 질의 내의 정규경로를 효율적으로 처리하기 위한 많은 연구가 수행되었다. 하지만 대부분의 연구는 한 개의 정규경로가 나타나는 질의의 최적화에 한정되어 있어 위와 같이 하나의 질의 내에 여러 개의 정규경로가 나타나는 질의에 적합하지 않다. 기존의 연구를 통해 위와 같이 두개 이상의 정규경로를 가지는 질의를 처리하는 방법은 다음과 같이 3가지로 나누어 볼 수 있다.

방법 1: 스키마 그래프 사용. 스키마 그래프(datagudes나 1-index 등)를 사용해 _*.movie에 해당하는 extent를 찾아낸다. 찾아낸 extent 의 각각의 노드들중에 _*.actor._*.name.BradPitt경로를 가지는 노드를 데이타그래프를 탐색하여 찾아낸다. 이 방법은 두 번째 이후에 나오는 경로를 처리하기 위해서 데이타 그래프를 탐색해야 하므로 효율적인 성능을 기대할 수 없다.

방법 2: 2-index사용. 2-index를 사용해 a _*.root._*.movie b 의 경로를 만족하는 (a,b)를 찾을 수 있다. 다시 c _*.actor._*.name d의 경로를 만족하는 (c,d)를 찾은 후 b=c의 조건으로 (a,b)와 (c,d)를 조인하여 (a, m, d)의 결과를 얻었을 때 m이 원하는 결과가 된다. 하지만 이 방법은 최악의 경우 색인 그래프의 크기가 데이타 그래프의 크기의 제곱 승으로 증가할 수 있고 많은 경우 색인 그래프의 크기가 커져 비효율적이라 할 수 있다.

방법 3: T-index사용. T-index를 사용하기 위해서는 질의 내에 포함되는 정규 경로식이 이미 색인으로 구축되어 있어야 한다. 만약 적절한 색인이 구축되어 있지 않은 경우 이 방법을 사용할 수 없다는 단점을 가지고 있다.

본 논문에서는 스키마 그래프를 사용하지만 데이타 그래프를 탐색하지 않아도 되는 효율적인 다중 경로의 질의처리 기법을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구를 간단히 살펴보고 3 장에서는 본 논문에서 사용되는 데이타 모델과 질의 언어를 정의하고 본 논문에서 스키마 그래프로 사용하는 *dataguides*에 대해 설명한다. 4 장에서는 데이타 그래프 내의 임의의 두 노드사이에 경로가 있는지를 확인할 수 있는 방법인 경로 식별자에 대해 설명하고 경로 식별자를 구성하는 알고리즘을 제공한다. 5장에서는 경로 식별자를 사용해 다중정규경로를

처리하는 방법에 대해 설명한다. 6장에서는 실험결과를 보여주고 7장에서 결론을 기술한다.

2. 관련연구

[19]에서 XML데이타와 비정형 데이타의 유사점과 차이점을 설명하고 있다. 비록 약간의 차이점이 있긴 하지만 XML데이타는 비정형 데이타의 한 종류 볼 수 있으며 XML데이타를 비정형 데이타에서 사용하는 레이블이 있는 그래프 모델[1,5]로 표현하는 것이 일반적이다.

XML을 위한 질의 언어는 [8, 10]에서 찾아볼 수 있으며 이들 질의 언어는 비정형 데이타에 대한 질의 언어[2,6]와 같이 정규경로식을 포함하고 있다. 특히 Query[8]는 XPath[21]를 사용하여 정규경로를 표현하고 있다는 것이 특징이다. 우리가 제안하는 기법은 이와 같이 정규 경로식을 기반으로 하는 모든 질의 언어에 적용될 수 있다.

비정형 데이타에 대한 구조요약(data guide)[1]은 비정형 데이타에 대한 스키마의 역할을 한다. 비정형 데이타에 대한 스키마의 역할을 하는 스키마 그래프 추출에 대한 여러 기법이 제안되었다[12,18,20]. 특히 [12]에서는 경로의 색인역할을 하는 스키마 그래프에 대해 설명하였다. 본 논문에서는 경로의 색인으로 스키마 그래프를 사용하여 질의를 최적화 하는 방법을 제공한다.

비정형 데이타의 질의처리 최적화에 관한 연구는 정규경로의 최적화에 초점을 맞추고 있다. 정규경로의 최적화를 위한 기법으로 [11,15,16]등이 제안되었다. [11]은 graph schemas[6]를 사용하는 질의 변환과 질의 prunning방법을 통해 정규경로를 최적화하였으며 graph shemas를 색인 그래프로 사용할 수 있음을 보여준다. [16]에서는 비정형 데이타에 대한 색인 구조로 1-index, 2-index, T-index를 제안하였다. 이중에서 1-index는 색인 역할을 하는 스키마그래프와 유사하지만 색인 그래프의 크기가 데이타 그래프의 크기에 선형적으로 비례하도록 만들었다는 것이 특징이며, 다중 정규경로를 가지는 질의를 처리하기 위한 방법으로 2-index와 T-index 가 제안되었으나 2-index는 색인의 크기가 데이타 그래프의 제곱 승으로 증가할 수 있다는 단점을 가지고 있고 T-index의 경우 처리해야할 질의가 구축되어 있는 색인을 만족하지 않는 경우 사용할 수 없다는 단점을 가지고 있다. [11]에서도 다중 정규경로를 가지는 질의의 prunning 방법을 제안하였지만 단순히 정규 경로식의 변환에만 초점을 맞추었을 뿐 두 번째 이후에 나오는 경로의 처리를 위해 데이타그래프를 탐색해야하는 비효율성을 가지고 있다.

[14]에서는 트리 내의 임의의 두 노드에 대해서 조상 후손의 관계를 알아낼 수 있는 Numbering Scheme을 제안하고 이를 관계형 데이타베이스의 색인으로 구축하여 정규경로 질의를 처리하는 방법을 제안하고 있다. 본 논문에서 제안하는 경로 식별자는 트리뿐만 아니라 임의의 그래프에 대해서 조상 후손 관계를 알아낼 수 있는 방법이다. 또한 본 논문에서는 경로 식별자와 스키마 그래프를 이용해 정규경로식이 두 개이상 포함된 질의를 효과적으로 처리하는 조인 방법에 대해 언급하고 있으며, [14]에서 제안하는 방법인 Numbering scheme을 사용해 관계형데이타 베이스에 XML의 색인을 만들고 이를 사용해 단일 정규경로식이 포함되어 있는 질의를 처리하는 방법과는 차이가 있다.

3. 배경지식

3.1 데이타 모델

비정형 데이타와 마찬가지로 XML은 레이블이 있는 그래프 모델로 표현될 수 있다[19]. O를 무한한 객체 식별자의 집합이라 하고, C를 O와 교집합을 가지지 않는 무한한 상수의 집합이라 할 때 XML데이타 모델을 다음과 같이 정의한다.

정의 3.1 데이타 그래프 DB=(V,E,R)은 루트 노드를 가진 그래프로 $V \subseteq O$는 노드의 집합을 나타내고, $E \subseteq V \times C \times V$는 유향 에지(directed edge)의 집합 그리고 $R \in V$는 루트노드를 의미한다.

XML 데이타에서 각 원소(*element*)를 노드 V에 대응시키고 원소(element)와 속성(attribute)간의 관계와 원소와 부 원소(subelement)간의 관계, 원소들 간의 참조 관계를 에지 E에 대응시킴으로써 정의 3.1의 데이타 모델로 표현할 수 있다.

3.2 질의 언어

XML 질의 언어는 정규 경로식을 기반으로 하고 있다. 본 논문에서는 데이타 그래프에 대해 정규 경로식을 사용하는 간단한 질의 언어 Q를 다음과 같이 정의하여 사용한다.

정의 3.2 질의 언어

Q := **select** V **from** C, ..., C
V := *var* | V, *var*
C := R *var* **in** *var*
R := label | _ | R.R | (R|R) | R^*

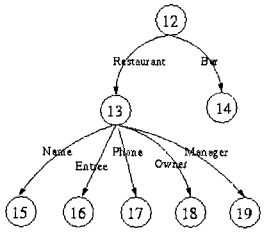
위의 질의에서 사용하는 변수들(*var*)은 데이타 그래프의 각 노드에 대응된다. 정규식에서 _(wild card)는 임

의의 레이블(label)이 될 수 있음을 의미한다.

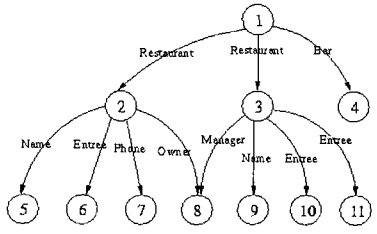
본 논문에서는 질의의 **from**절에 있는 정규경로식의 최적화 초점을 맞추며 **select** 절에 부질의(subquery)가 나타나는 경우에 대해 고려하지 않는다. R x **in** y의 조건은 y노드로부터 x노드까지 R을 만족하는 경로가 존재하는가를 나타낸다. 이때, 경로 R을 통해 연결되어 있는 x와 y의 쌍을 찾는 과정을 우리는 *정규경로조인*이라고 정의한다.

3.3 스키마 그래프

스키마 그래프는 데이타 그래프의 구조요약으로 사용자가 데이타 그래프의 구조를 쉽게 파악할 수 있도록 도와주는 역할을 하며[1], 이러한 목적 외에도 경로의 색인 역할을 수행할 수 있다. 본 논문에서는 편의상 경로의 색인 역할을 하는 스키마 그래프로 *dataguides* [12]를 사용하며 이후에 언급하는 스키마 그래프는 dataguides를 나타낸다. 하지만 본 논문은 특정 스키마 그래프에 의존적이지 않은 알고리즘을 제시하며 따라서 1-index[16]나 graph schemas[20]등과 같은 다른 스키마 그래프(경로 색인)를 사용하여도 같은 결과를 얻을 수 있다. 원시 데이타 그래프를 통해 dataguides를 생성하는 방법은 비결정적 오토마타를 결정적 오토마타로 변환하는 방법과 동일하게 진행된다[12,13,18]. 그림 2는 데이타 그래프와 해당 dataguides를 보여준다. 그림 2의 (a)에서 루트노드로부터 Restaurant인 레이블(label)을 가지는 에지들은 그림 2의 (b)에서와 같이 하나의 에지로 합쳐지게 된다. 마찬가지 방법으로 하위노드들에 대해서도 동일한 레이블을 가지는 에지들을 합침으로써 dataguides를 생성할 수 있다. 스키마 그래프가 데이타 그래프에 대한 경로의 색인 역할을 수행하기 위해서 스키마 그래프의 각 노드는 데이타 그래프 내에 루트 노드로부터 자신까지의 경로와 같은 경로를 가지는 노드들의 *extent*를 저장한다. 예를 들어 그림 2의 (b)에서 Restaurant.Entree가 가지는 extent는{6, 10, 11}이 된다.



(a) 데이타 그래프



(b) 스키마 그래프(dataguides)

그림 2 스키마 그래프

4. 경로 식별자

경로 식별자는 데이타 그래프내의 임의의 두 노드 사이에 경로가 있는지를 데이타 그래프를 탐색하지 않고 알아내기 위해 데이타 그래프의 각 노드에 저장되는 정보이다. 4.1에서 경로 식별자에 대해 설명하고 데이타 그래프의 각 노드에 경로 식별자를 생성하는 알고리즘을 제시한다 4.2에서는 경로식별자의 갱신방법에 대해서 설명한다.

4.1 경로 식별자 생성

경로 식별자의 설명을 위해 다음과 같은 몇가지 간단한 정의를 한다.

정의 4.1 **'범위'**는 두개의 숫자로 구성된다. 범위를 나타내는 두개의 숫자 중 첫 번째 숫자는 **'시작점'**이라 하고 두 번째 숫자는 **'끝점'**이라하며 '시작점 $\leq$끝점'의 관계가 성립한다.

정의 4.2 두 개의 범위 $r_1 = (s_1, e_1)$, $r_2 = (s_2, e_2)$ 가 있어서, $s_2 - e_1 = 1$인 경우 r_1, r_2는 **'연속적'**이라 하고 $s_1 \leq s_2$ 이고 $e_1 \geq s_2$이고 $e_2 \geq e_1$ 인 경우 **'겹친다'**고 하며 $s_1 \leq s_2$ 이고 $e_1 \geq e_2$ 인 경우 r_1은 r_2를 **'포함한다'**고 한다.

정의 4.3 범위를 원소로 가지는 두 집합 R_1, R_2가 있어서 R_2의 각각의 원소가 R_1에서 자신을 포함하는 원소를 찾을 수 있는 경우 R_2를 R_1의 **'경로부분집합'**이라 부른다.

정의 4.4 ***경로 식별자(PID : the Path Identifier)*** *는 다음 제약을 만족시키는 범위의 집합으로 구성된다*

경로 식별자 제약 데이타 그래프의 내의 모든 노드는 자식 노드들의 경로 식별자가 자신의 경로 식별자의 경로부분집합이 되어야 하며 자식 노드 이외의 다른 노드의 경로 식별자를 경로 부분집합으로 가질 수 없다.

위와 같이 정의된 경로 식별자를 사용하면, 데이타 그래프 상의 임의의 두 노드에 대해 만약 두 노드의 경로 식별자가 경로 부분집합의 관계를 가지면 두 노드는 같은 경로 상에 있다는 것을 쉽게 알 수 있다.

알고리즘 1은 데이타 그래프의 각 노드에 경로 식별자 제약사항을 만족시키는 경로 식별자를 구성해 주는 알고리즘이다. 이 알고리즘에서 사용되는 스콜렘 함수는 호출될 때마다 이전에 돌려주었던 숫자보다 1큰 숫자를 돌려주는 함수이다. 이 알고리즘은 깊이 우선 탐색의 방법으로 데이타 그래프를 탐색하면서 스콜렘 함수를 통해 데이타 그래프의 각 노드에 시작점과 끝점의 값이 동일한 범위 하나씩을 할당하게 된다. 즉 데이타 그래프의 각 노드는 하나의 범위를 가지는 경로 식별자를 가지게 된다. 단말 노드의 경로 식별자는 이렇게 할당받은 한개의 범위를 원소로 가지는 집합으로 구성되며, 중간노드의 경로 식별자는 자식 노드의 경로 식별자들과 자신의 경로 식별자를 합집합으로 만든 후에 연속적이거나 겹치거나 포함관계에 있는 원소들을 하나의 원소로 합침으로써 구성된다. 알고리즘 1은 데이타 그래프가 사이클이 존재하지 않는 유향 그래프(DAG)에 대한 경우만을 고려하고 있다. 만약 데이타 그래프가 사이클을 포함하고 있는 경우에는 데이타 그래프에 대한 Condensation[3]을 구성한 후, Condensation의 각 노드에 알고리즘 1을 통해 경로식별자를 구성한다. 이때 데이타 그래프의 strongly connected component내의 각 노드들은 Condensation내의 해당 노드의 경로 식별자를 가지게 된다. 즉 하나의 strongly connected component 내에 있는 모든 노드들은 동일한 경로 식별자를 가지게 된다. 이렇게 함으로써 알고리즘 1을 통해 임의의 그래프에 경로 식별자를 할당할 수 있다.

그림 3은 각각 데이타 그래프가 트리인 경우, DAG인 경우, 사이클이 있는 그래프의 경우에 대해 데이타 그래프 내의 각 노드가 가지는 경로 식별자를 나타낸다. 데이타 그래프 내에 사이클이 있는 경우 그림 3의 (c)와 같이 같은 stronly connected component 내의 모든 노드들은 같은 경로 식별자를 가지게 된다.

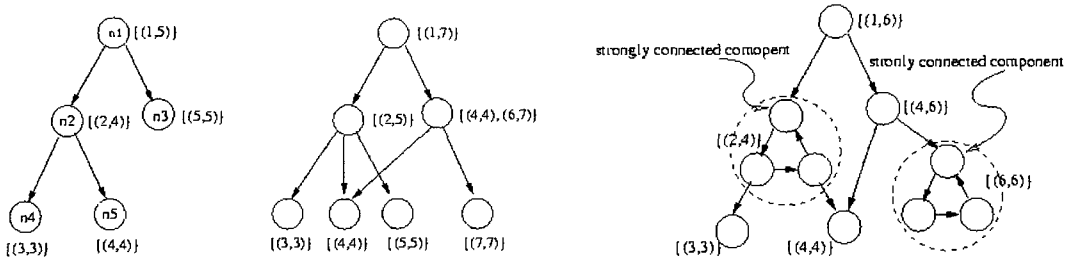
데이타 그래프의 중간노드의 자식노드 개수 중 최대

알고리즘 1 경로 식별자 생성

```
1:  //Input: target, the root oid of a source data
      graph
2:  //Effect: construct path id at each node in
      source data graph
3:  //Initially, PID at each node in data graph is
      empty
4:
5:  sf() : skolem function
6:
7:  PIDConst(target)
8:  {
9:     if(target.PID is not empty)
10:       return target.PID
11:
12:    start-point = end-point = sf()
13:    insert (start-point, end-point) into
              target.PID
14:
15:    if(target is non-leaf node){
16:       foreach(child node, c of target)
17:          target.PID = target.PID ∪
                          PIDConst(c)
18:       Merge elements in the target.PID
19:    }
20:    return target.PID
21: }
```

값을 m이라하고 데이타 그래프의 크기를 n이라고 할 때, 각 중간 노드는 경로 식별자 생성 시 각 원소를 합하기 위해 m^2의 비용을 가지게 된다. 또한 위의 알고리즘은 경로 식별자를 생성하기 위해 데이타 그래프의 각 노드를 한번씩만 방문하게 되므로 경로 식별자를 생성하는데 드는 비용은 $O(m^2n)$임을 알 수 있다. 하지만 m은 n에 비해 상당히 작은 수이므로 m을 상수로 생각한다면 위의 알고리즘은 데이타 그래프의 크기에 선형적인 시간 복잡도를 가지는 알고리즘이라 할 수 있다.

위와 같은 방법으로 생성된 경로 식별자는 그림 3에서 알 수 있듯이 일반적으로 적은 수의 원소의 개수를 가지



(a) 트리 (b) DAG (c) 사이클이 포함된 그래프

그림 3 경로 식별자

는 집합으로 구성된다. 많은 경우 XML데이타는 트리의 형태로 이루어져 있으며 데이타그래프가 트리인 경우 그림 3의 (a)와 같이 데이타 그래프의 각 노드의 경로 식별자는 원소를 하나만 가지는 집합으로 구성된다.

이렇게 구성된 경로 식별자는 스키마 그래프의 각 노드에 저장되는 extent 내에 객체의 oid와 함께 포함될 수 있다. 하지만 만약 경로 식별자의 크기가 큰 노드의 경우 extent에는 객체의 oid만을 저장하고 경로 식별자가 필요한 경우 직접 데이타 그래프의 해당 노드를 통해 경로 식별자를 얻어올 수 있도록 구성하면 된다.

4.2 경로식별자 갱신 방법

경로 식별자를 갱신하는 가장 단순한 방법은 데이타 그래프 내의 경로 식별자 전체를 갱신하는 것이지만 이 방법은 아주 비효율적임을 쉽게 알 수 있다. 경로식별자의 점진적 유지를 위해서는 새롭게 삽입되는 노드를 위해 범위 여유 값이 있어야 한다. 이를 위해 알고리즘 1의 스콜렘 함수가 돌려주는 값을 이전에 돌려주었던 값보다 n큰 값을 돌려주도록 수정한다. 알고리즘이 이렇게 수정되면 정의 4.1은 $s_2 - s_1 = n$인 경우 '*연속적*'으로 변경되어야 한다. 즉 연속적인 두 노드 사이에 새로운 노드가 삽입되는 경우를 위해 n만큼의 공간을 비워두는 것이다.

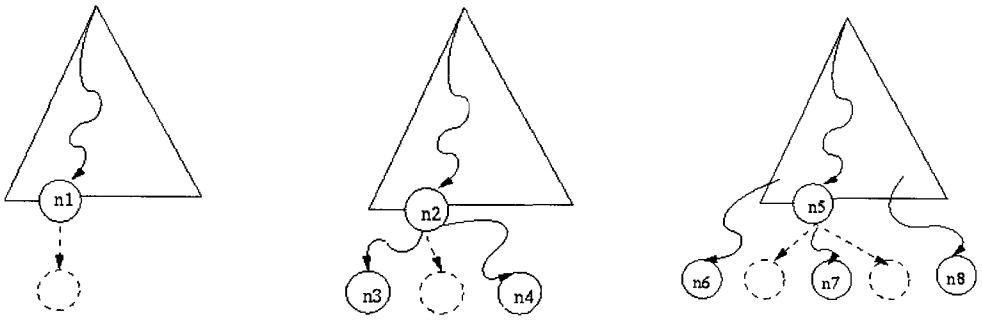
새로운 노드가 삽입되는 경우는 그림 4와 같이 세 가지로 나누어진다. 그림 4의 (a)와 같이 단말 노드 n_1의 자식 노드로 새로운 노드가 삽입되는 경우 새로 삽입되는 노드의 경로 식별자를 n_1과 같게 만들어 주면된다. 그림 4의 (b)와 같이 두 단말 노드 n_3, n_4의 사이에 삽입되는 경우 n_4의 범위의 시작점에서 n_3의 범위의 끝점을 뺀 값의 중간 값을 이용해 새로 삽입되는 단말 노드의 경로 식별자를 할당해 주면된다. 그림 4의 (c)는 두 가지 경우에 대해서 설명하고 있다. 첫 번째 경우는 노드 n_5의 왼쪽 첫 번째 자식 노드로 새로운 노드가 삽입 되는 경우이다. 이때 n_6와 n_7을 이용해 (b)의 경우와 마찬가지로 경로 식별자를 할당하면 된다. 하지만 n_6가 존재하지 않는 경우 n_7의 경로 식별자의 시작점보다 n작은 값을 통해 경로 식별자를 할당하면 된다. 이때 새롭게 삽입된 노드의 경로 식별자가 n_5의 경로 식별자와 경로부분집합 관계를 만족하도록 n_5의 경로 식별자를 갱신해 주어야 하며 n_5의 조상들도 마찬가지로 수정해 주어야 한다. 두 번째 경우는 노드 n_5의 오른쪽 마지막 자식 노드로 새로운 노드가 삽입되는 경우이며 첫 번째 경우와 대칭적으로 생각해 보면 된다. 위와 같이 경로 식별자를 갱신하더라도 새롭게 삽입되려는 노드의 양 옆에 있는 노드들(예를 들면 그림 4의 (b)에서 n_3와 n_4) 사이의 경로식별자의 범위의 차이가 1이 되면 삽입이 불가능해 진다. 이러한 경우가 생길 때마다 데이타 그래프 전체의 경로 식별자를 주기적으로 갱신해 주게 된다.

5. 다중 정규경로 조인

본 논문에서 다루는 다중정규경로 질의는 다음과 같이 정의된다.

select $x_1, \ldots, x_k$
from $P_1\ y_1$ **in** *root*, $P_2\ y_2$ **in** $y_1, \ldots, P_n\ y_n$
in $y_{n-1}\ (x_i \in \{y_1, \ldots, y_n\})$

다른 형태의 다중정규경로 질의는 쉽게 위의 형태의 질의로 변환될 수 있으며 본 논문에서는 다루지 않는다. 본 논문에서는 우선 정규경로를 두개 포함하는 질의에 대해서 설명한 후 일반화된 알고리즘을 제시한다.



(a) 경우 1 (b) 경우 2 (c) 경우 3

그림 4 경로 식별자의 갱신

P_1 x P_2 y와 같은 경로식을 갖는 질의의 처리는 데이타 그래프 내의 P_1에 해당하는 각각의 노드가 P_2에 해당하는 경로를 가지고 있는지를 검사하는 것과 동일하다.

우선 P_1과 P_2가 정규식으로 표현되지 않은 경로인 경우에 대해서 생각해 보기로 한다. 이 경우 스키마 그래프 내에 P_1에 해당하는 노드와 $P_1.P_2$에 해당하는 노드는 각각 한 개씩[1]이 되며 본 논문에서는 다음과 같은 방법으로 처리한다. 우선 스키마 그래프를 통해 P_1에 해당하는 extent e_{p_1}과 $P_1.P_2$에 해당하는 extent e_{p_2}찾아 낸 후 e_{p_2}내의 어떤 객체 o_2의 경로 식별자가 e_{p_1}내의 어떤 객체 o_1의 경로식별자의 경로 부분집합이 되면 (o_1, o_2)를 두 extent의 조인 결과에 추가한다. 즉 e_{p_1}내의 객체의 경로 식별자와 e_{p_2}내의 객체의 경로 식별자가 경로부분집합 관계를 가지는지가 두 extent의 조인조건이 된다. 본 논문에서는 위의 과정을 다음과 같이 정렬합병조인과 비슷한 방법을 통해 수행한다. 우선 e_{p_1}과 e_{p_2}를 정렬한다. extent를 정렬하는 방법은 다음과 같다. extent내의 각 원소 e=(oid, {r_1, ..., r_n})을 (oid, r_1), ..., (oid, r_n)와 같이 분해한다. 예를 들어 extent {(o_1,{(2,4),(7,8))}, (o_2,{(2,6),(9,10))})는 {(o_1,(2,4)), (o_1,(7,8)), (o_2,(2,6)), (o_2,(9,10))}과 같이 바뀐다. 분해된 extent를 각 원소가 가지고 있는 범위의 시작점에 의해 오름차순으로 정렬한다. 만약 시작점이 같은 경우 끝점을 통해 오름차순으로 정렬한다. 앞의 예제를 정렬하면 {(o_1,(2,4)), (o_2,(2,6)), (o_1,(7,8)), (o_2,(9,10))} 과 같이 된다. 정렬된 e_{p_1}과 e_{p_2}를 다음의 과정을 통해 조인한다.

1. 정렬된 e_{p_1}에서 첫 번째 원소의 범위를 r = (sp, ep)이라 할 때, 정렬된 e_{p_2}에서 끝점의 값이 r.sp보다 작은 범위를 가지는 모든 원소를 삭제한다.

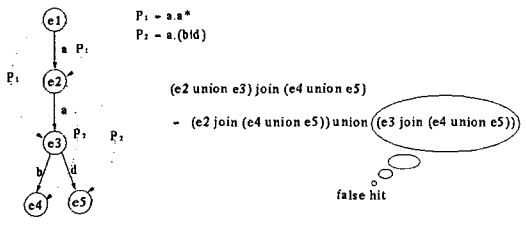


그림 5 거짓 결과 생성

2. 정렬된 e_{p_2}에서 시작점의 값이 r.ep보다 작거나 같은 범위를 가지는 모든 원소에 대해 r과의 포함관계를 조사한다. 만약 포함관계가 있다면 두 원소를 결과에 추가한다.(이때, 각 원소의 분해되기 전의 PID를 이용하여 포함관계를 조사한다.)

3. 정렬된 e_{p_1}에서 첫 번째 원소를 삭제한 후 1부터 다시 반복한다.

위의 과정은 정렬된 e_{p_1}이 공집합이 되거나 정렬된 e_{p_2}가 공집합이 될 때까지 반복된다.

P_1과 P_2가 정규식으로 표현된 경로인 경우 스키마 그래프 내에 P_1을 만족하는 노드가 여러 개 존재할 수 있다. 즉 여러 개의 extent가 생기게 된다. $P_1.P_2$의 경우도 마찬가지이다. 이때, P_1에 해당하는 extent들의 합집합과 $P_1.P_2$에 해당하는 extent들의 합집합을 위와 같은 방법으로 조인할 수 있으나 이 경우 거짓 결과(false hit)를 생성할 수 있다. 예를 들어 그림 5에서와 같이 a.a* x a.(b|d) y와 같은 경로식을 가지는 질의는 거짓 결과를 생성할 수 있다. 그림 5의 그래프는 스키마 그래프로 그래프의 각 노드에 있는 $e_1, \ldots, e_5$는 각 노드가 가지는 extent를 나타낸다. 그림 5에서 노드 e_3로부터 a.(b|d)의 경로를 만족하는 노드가 존재하지 않지만 노드 e_2로부터 a.(b|d)의 경로를 만족하는 노드 e_4와 e_5가 우연히 노드 e_3와 같은 경로를 가지게 됨으로 인해 e_3가 $e_4 \bigcup e_5$와 조인되어 거짓결과를 생성하게 된다.

이러한 거짓 결과(false hit)를 방지하기 위해 다음과 같은 방법으로 질의를 처리하면 된다. P_1에 해당하는 각각의 extent e_i로부터 P_2 에 해당하는 노드들을 스키마 그래프를 사용해 찾고 찾아낸 노드들의 extent 의 합집합과 e_i를 조인한 후에 각각의 조인 결과를 합친다. 예를 들어 그림 5의 경우 P_1을 만족하는 노드는 e_2와 e_3이다. 우선 e_2로부터 P_2를 만족하는 노드 e_4와 e_5를 찾아 $e_2 \bowtie (e_4 \bigcup e_5)$와 같이 조인하고 e_3로부터 P_2를 만족하는 노드(해당 노드가 없으므로 extent는 공집합이 된다.)를 찾아 $e_3 \bowtie \varnothing$을 한 후 두 결과를 합치면 거짓 결과가 없는 정확한 결과를 얻을 수 있다.

알고리즘 2는 정규경로가 두개 이상인 경우에 대해 조인 처리를 하는 방법에 대한 알고리즘이다. processingJoin은 각각의 정규경로에 대해 재귀적으로 동작하면서 각각의 정규경로를 처리한다. 이 함수의 9번째 줄에서 병합정렬을 하면서 결과 집합을 합하면 10번째 줄의 결과 집합을 정렬하는 부분은 필요 없게 된다.

1) 1-index의 경우 비결정적(nodeterministic) 성질 때문에 하나 이상의 extent가 생성될 수 있다. 여기서 스키마 그래프는 dataguides를 말하며, 알고리즘 2에서는 특정 스키마 그래프에 의존적이지 않은 조인 알고리즘을 제공한다.

알고리즘 2 다중 정규경로 조인 알고리즘

```
1: // Input: root node of the schema graph and 1
2: // Output: result set of regular path join query
3:
4: processingJoin(Node N, i)
5: {
6:    result is empty set
7:
8:    foreach(node n that satisfies the regular path P1 from the
      node N)
9:       result = result ∪ processingJoin(n, i+1)
10:   sort the result by first element of each tuple in
         the result}
11:   return extentJoin(result, extent in the node N)
12: }
13:
14: extentJoin(ext, tuple set)
15: {
16:   result is empty set
17:   s ext = sort the ext
18:
19:   if(tuple set is empty){
20:      foreach(element e in s ext)
21:          add 1-ary tuple (e) to result
22:       return result
23:   }
24:
25:   while(tuple set != empty && s ext != empty){
26:      t = first tuple in tuple set}
27:      e1 = first element of s_ext
28:      e2 = first element of t
29:      while(e1.sp > e2.ep){
30:            delete t from tuple set
31:            t = first tuple in tuple set
32:            e2 = first element of t
33:      }
34:        while(e1.ep ≥ e2.sp){
35:          if(e2의 PID가 e1의 PID의 경로 부분집합인경우){
36:                add e1 in front of t
37:                insert t into result
38:          }
39:          t = next tuple in tuple set
40:          e2 = first element of t
41:      }
42:      delete e1 from s_ext
43:   }
44:   return result
45: }
```

processingJoin에서 호출하는 extentJoin 함수는 두개의 extent를 경로 식별자를 이용해 조인하는 것과 같은 역할을 수행하는 함수이다. extentJoin의 첫 번째 인자는 현재 처리하고자 하는 노드이고 두 번째 인자는 그 노드의 후손들을 조인한 결과이다. 두 번째 인자는 {((oid_{11}, $range_{11}$), ...,(oid_{1n}, $range_{1n}$)), ...,((oid_{m1}, $range_{m1}$), ..., (oid_{mk}, $range_{mk}$))}와 같이 구성된다. 이때 extentJoin함수는 첫 번째 인자와 두 번째 인자에 포함된 각 투플의 첫 번째 원소 (oid_{i1}, $range_{i1}$)를 앞에서 설명한 방법으로 조인한다.

extent를 미리 정렬하여 스키마 그래프의 각 노드에 저장해 놓을 수 있으며 많은 경우 XML데이타는 트리와 비슷한 형태로 만들어져 데이타 그래프 내의 대부분의 노드가 가지는 경로 식별자는 하나의 원소만을 가지는 집합으로 구성된다. 따라서 위의 알고리즘은 정렬합병조인과 같이 조인에 참여하는 객체의 개수에 선형적인 시간 복잡도를 가진다고 할 수 있다.

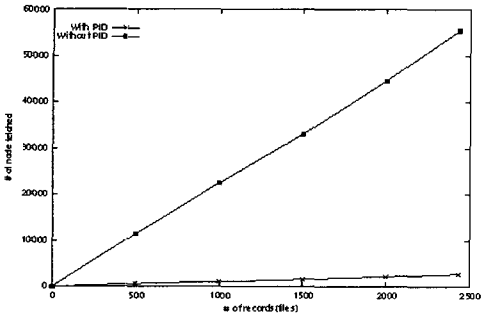
6. 실험결과

본 논문의 실험은 우리가 제안한 방법과 서론에서 설명한 방법 1과의 성능비교를 통해 제안한 방법의 우수성을 보인다. 서론에서 설명한 방법 2(2-index사용)와는 다음과 같은 이유로 성능비교를 하지 않는다. 1-index나 2-index는 색인 자체가 반 구조데이타이므로 색인의 크기가 색인의 성능과 밀접한 연관이 있다. 1-index는 2-index보다 항상 작은 크기로 생성되며[16] 따라서 2-index의 색인 그래프를 탐색하는 것은 항상 1-index의 색인그래프를 탐색하는 것보다 비용이 커지게 된다. 본 논문에서는 데이타 그래프를 탐색하지 않고 스키마 그래프만을 탐색하여 질의처리를 하며 주어진 정규 경로식을 단 한번씩만 탐색하므로 스키마 그래프로 1-index 를 사용한다면, 2-index의 색인그래프를 탐색하는 것보다 항상 좋은 성능을 보인다는 것을 보장한다.

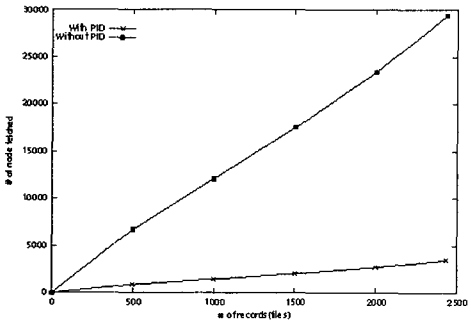
실험에서는 NASA의 천문학데이타[17], 세익스피어의 희곡 데이타[7]중 햄릿데이타와 인터넷 영화데이타[9] 그리고 임의로 만들어진 메이져리그 데이타베이스를 사용했다. 천문학 데이타 및 인터넷 영화 데이타는 많은 XML파일들로 구성되어 있으며 실험을 위해 이 파일들을 하나로 병합하여 사용하였다. 특히 천문학 데이타는 파일의 개수를 0에서 2434개까지 증가시켜 가면서 데이타의 크기가 증가하는 경우에 대해 실험하였다.

실험에서는 메모리로 가져온 객체의 개수로 성능을 평가하였으며 대부분의 XML데이타와 같이 실험에서 사용한 데이타도 거의 트리와 같은 형태를 가지고 있어 경로 식별자의 크기는 객체의 크기보다 훨씬 작게 구성되었고 따라서 경로 식별자의 크기를 성능평가 기준으로 고려하지 않았다.

두개의 정규경로를 가지는 질의 여섯 개를 사용해 실



(a) 질의 6.1



(b) 질의 6.2

그림 6 실험 결과 - 질의 6.1, 6.2

표 1 실험 결과 - 질의 6.3, 6.4, 6.5

	데이타 내의 총 객체수	경로식별자 사용	경로식별자 미사용
질의 6.3	12158	34	6611
질의 6.4	70521	373	40971
질의 6.5	3843	56	1284

험하였으며 서론에서 설명한 방법 1과 본 논문에서 제안하는 방법을 통해 질의를 처리하는 방법의 성능을 비교하였다. 본 실험에서 사용한 질의는 다음과 같다.

질의 6.1 **select** x,y
from _*.reference._*.(journal|other) x in root, _*.year y in x

질의 6.2 **select** x,y **from** _*.history x in root, _*.creator y in x

질의 6.3 **select** x
from PLAY._*.SCENE x in root, _*.SPEAKER.FRANCISCO y in x

질의 6.4 **select** x,y
from _*.Movie x in root, _*.Year.(1975|1980) y in x

질의 6.5 **select** x,y
from MLB._*.EAST.player x in root, _*.nickname y in x

질의 6.1과 6.2는 NASA의 천문학데이타에 대한 질의이며 질의 6.3은 햄릿 데이타에 대한 질의이다. 질의 6.4는 인터넷 영화 데이타베이스에 대한 질의이며 질의 6.5는 메이져리그 데이타베이스에 대한 질의이다.

그림 6은 질의 6.1과 6.2에 대한 실험 결과를 보여주며, 표 1은 질의 6.3, 6.4, 6.5에 대한 결과를 보여준다. 그림 6에서 보듯이 경로 식별자를 사용하지 않는 방법은 두 번째 정규 경로식을 처리하기 위해서 데이타 그래프를 탐색하므로 데이타의 크기가 커질수록 가져오는 객체수가 현저하게 증가되지만, 경로 식별자를 사용하는 방법은 질의를 만족하는 결과 객체수가 증가함에 따른 객체 수 증가만을 보이고 있다. 표 1에서 경로 식별자를 사용하는 방법과 사용하지 않는 방법 사이에 가져오는 객체의 수가 큰 차이를 보이는 것은 실험에 사용된 데이타가 반복되는 구조로 이루어져 있어 스키마 그래프가 데이타 그래프에 비해 아주 작게 구성되기 때문이며, 표 1의 질의 6.3 의 경우 질의를 만족하는 객체의 수가 제한되어 있기 때문에 경로 식별자를 사용하는 방법과 그렇지 않은 방법 사이에 큰 차이를 보인다.

본 논문에서 제안하는 방법은 스키마 그래프에 의존적이며, 1-index의 경우 스키마 그래프가 최악의 경우 데이타 그래프의 크기에 선형적으로 증가할 수 있고 dataguides의 경우 스키마 그래프가 최악의 경우 데이타 그래프의 크기의 지수승으로 증가할 수 있으므로 이러한 경우에 좋은 성능을 보장할 수 없다. 예를 들어 XML데이타내의 모든 원소(element)들이 서로 다른 tag들로 구성되어 있는 경우 스키마 그래프와 데이타 그래프의 크기가 같아지며, 본 논문에서 제안하는 방법은 데이타 그래프를 직접 탐색하는 것과 동일한 성능을 가져오게 된다. 하지만 위의 실험에서도 알 수 있듯이 대부분의 XML데이타는 반복되는 원소가 많은 데이타로 구성되어 있어 스키마 그래프가 데이타 그래프에 비해 아주 작은 크기로 만들어지게 된다. 본 논문의 방법은 데이타 그래프를 탐색하지 않으므로 질의 처리 시 (스키마 그래프의 크기 + 결과 객체의 개수)개 이하의 객체를 가져오게 되는 것을 보장하며 따라서 스키마 그래프의 크기가 데이타 그래프의 크기보다 작게 구성되는 경우 본 논문의 방법은 항상 우수한 성능을 보인다는 것을 알 수 있다.

7. 결 론

정규경로는 구조가 고정되지 않은 데이타에 대한 질의에 유용하고도 필수적인 도구라 할 수 있다. 특히 여러 정규식이 포함되는 질의(조인 질의)는 관계형 데이타베이스의 조인 질의만큼이나 중요하지만 아직 조인 질의 최적화에 대한 연구는 아주 적다. 본 논문에서는 경로 식별자를 통해 데이타 그래프 내의 임의의 두 노드 사이에 경로가 존재하는지를 쉽게 알 수 있는 방법을 제공했으며 경로 식별자와 그래프 스키마를 이용해 여러 개의 정규식이 포함된 질의를 데이타 그래프를 탐색하지 않고 효율적으로 처리할 수 있는 방법을 제공했다. 그리고 실험을 통하여 본 논문에서 제안하는 알고리즘의 우수성을 보였다.

본 논문에서 제공하는 방법은 다른 연구들과 함께 사용되어 보다 효율적인 질의 처리 방법을 제공할 수 있다. 예를 들어 [11,12]등의 방법을 통해 질의 처리 이전 단계에서 정규 경로식을 변환한 후에 질의 처리를 함으로써 더 성능을 향상시킬 수 있다.

본 논문에서는 데이타 그래프가 갱신되는 경우 경로 식별자를 갱신하는 간단한 방법을 소개하였다. 하지만 데이타의 삽입이 어느 한곳으로 몰리게 되는(skewed) 경우 데이타 그래프 전체에 대한 경로식별자의 갱신 기간이 짧아지게 된다. 따라서 데이타가 변화하는 환경에서 사용하기 위해서는 경로 식별자를 점진적으로 유지하는 좀 더 효과적인 방법에 대한 연구가 필요하다.

참 고 문 헌

[1] Serge Abiteboul, "Querying semi-structured data," In Proceedings of the International Conference on Database Theory, 1997.

[2] Serge Abiteboul, Dallan Quass, Jason McHugh, Jennifer Widom, and Janet Wiener, "The lorel query language for semistructured data," International Journal on Digital Libraries, 1996.

[3] Sara Baase, "Computer Algorithms. Introduction to Design and Analysis," Addison-Wesley Publishing Company, 1988.

[4] T.Bray, J.Paoli, and C.Sperberg-McQueen, "Extensible markup language(XML) 1.0," Technical report, W3C Recommandation, 1998.

[5] Peter Buneman, "Semi-structured data," In Proceedings of ACM Symposium on Principles of Database Systems, 1997.

[6] Peter Buneman, Susan Davidson, Gerd Hillebrand, and Dan Suciu, "A query language and optimization techniques for unstructured data," In Proceedings of the ACM SIGMOD International Conference on the Management of Data, 1996.

[7] Robin Cover, "The XML Cover Pages," http://xml.coverpages.org.

[8] D.Chamberlin, D.Florescu, J.Robie, J.Simeon, and M.Stefanescu, "XQuery: A Query Language for XML," Technical report, W3C Working Draft, February 2001.

[9] "The Internet Movie Database," http://www.imdb.com.

[10] A.Deutsch, M.Fernandez, D.Florescu, A.Levy, and D.Suciu, "Query language for XML," In Proceedings of Eighth International World Wide Web Conference, 1999

[11] Mary Fernandez and Dan Suciu, "Optimizing regular path expressions using graph schemas," In IEEE Internaltional Conference on Data Engineering, 1998.

[12] Roy Goldman and Jennifer Widom, "DataGuides: enabling query formulation and optimization in semistructured databases," In Proceedings of the Conference on Very Large Data Bases, 1997.

[13] John E. Hopcroft and Jeffery D. Ullman, "Introduction to automata theory, languages, and computation," Addison-Wesley Publishing Company, 1979.

[14] Quanzhong Li and Bongki Moon, "Indexing and Querying XML Data for Regular Path Expressions," In Proceedings of the Conference on Very Large Data Bases, 2001.

[15] J. McHugh and J. Widom, "Compile-Time Path Expansion in Lore," In Proceedings the Workshop on Query Processing for Semistructured Data and Non-Standard Data Formats, 1999.

[16] Tova Milo and Dan Suciu, "Index structure for path expressions," In Proceedings of the International Conference on Database Theory, 1999.

[17] "Astronomical Data Center," http://tarantella.gsfc.nasa.gov/xml/.

[18] Svetlozar Nestorov, Jeffery D. Ullman, Janet Wiener, and Sudarshan Chawathe, "Representative objects: concise representations of semistructured, hierarchical data," In IEEE International Conference on Data Engineering, 1997.

[19] Dan Suciu, "Semistructured data and XML," In Proceedings of International Confernece on Foundataions of Data Organization, 1998.

[20] Dan Suciu, Mary Fernandez, Susan Davidson, and Peter Buneman, "Adding structure to unstructured data," In Proceedings of the International Conference on Database Theory, 1997.

[21] W3C, "XML Path Language(XPath) 1.0," In W3C Recommendation, 1999.

김 종 익

1998년 한국과학기술원 전산학과 학사. 2000년 한국과학기술원 전산학과 석사. 2000년 ~ 현재 서울대학교 컴퓨터공학부 박사과정. 관심분야는 XML, 데이터베이스

정 태 선

1995년 한국과학기술원 전산학과 학사(B.S.). 1997년 서울대학교 전산과학과 석사(M.S.). 1997년 ~ 현재 서울대학교 전산과학과 박사과정

김 형 주

1982년 서울대학교 전자계산학과(학사). 1985년 Univ. of Texas at Austin(석사). 1988년 Univ. of Texas at Austin (박사). 1988년 5월 ~ 1988년 9월 Univ. of Texas at Austin. Post-Doc. 1988년 9월 ~ 1990년 12월 Georgia Institute of Technology(부교수). 1991년 1월 ~ 현재 서울대학교 컴퓨터공학부 교수.