논문 01-26-4A-18 한국통신학회논문지 '01-4 Vol.26 No.4A

SVG 웹 그래픽을 포함하는 XML 문서를 위한 하이브리드 데이터 모델

정회원 김 은 영*, 오 해 석**

Hybrid Data Model for XML Documents including SVG (Scalable Vector Graphics) Web Graphic

Eun-Young Kim*, Hae-Seok Oh** *Regular Members*

요 약

웹 상에 산재되어 있는 데이터를 표현하고 문서에 대한 쉬운 검색 및 재사용을 지원하는 문서 표준으로 XML이 제시되고 있고 XML에서 2차원 그래픽을 구사하기 위한 그래픽 기술 언어인 SVG는 새로운 웹 그래픽 표준으로 등장하였다. XML문서를 데이터베이스에 저장하거나 역으로 데이터베이스에 저장된 값으로부터 XML문서를 생성하고 또한 검색하기 위한 새로운 데이터 모델들이 요구되고 있다. 또한 새로운 데이터 모델은 웹 그래픽 기술 언어인 SVG를 포함하는 XML문서를 지원할 수 있어야 하며 top-down 검색뿐만이 아니라 bottom-up 검색이 가능해야 하고 최적화 되어야 한다. 본 논문에서는 XML문서가 지원하는 웹 그래픽 포맷인 SVG의 데이터를 XML문서와 함께 데이터베이스에 저장하고 검색하기 위한 하이브리드 데이터 모델을 제안한다. 제안하는 하이브리드 데이터 모델은 XML 문서의 데이터 뷰와 구조 뷰, 그리고 SVG 구조 뷰를 모두 표현하며 텍스트를 포함하는 SVG에 대한 효율적인 검색을 지원한다.

ABSTRACT

XML is emerging as a dominant standard for representing data and searching document and supporting reuse of document on the WWW. And emerging SVG that a technical graphics language as new web-graphics standard to represent the two dimension of graphics in XML. New models for generating and searching XML documents from save database or saved values in database are needed. Also new model is must support XML documents include SVG that a technical web-graphics language and must can not only top-down but also bottom-up searching. The hybrid data model proposed in this paper is a model for saving data of SVG that web-graphics format to database with XML documents and searching. The hybrid data model represent all that data view, structure view of XML document and SVG structure view and support efficient searching function for SVG include text.

I. 서 론

최근 XML(eXtensible Markup Language)은 SGML에 기반을 둔 간단하고 유연성 있는 마크업 언어로써 사용자들이 원하는 정보를 검색하는 방법을 제공하고, 웹에서 사용할 수 있는 XML 메타미디어의 유용성으로 인해 웹상에서 데이터를 표현하는 표준으로 제안되고 있다. XML은 또한 자신이 나타내는 컨텐트의 포맷 방식보다는 컨텐트를 정확히 표현하는 데 중점을 둔다. 또한 웹 상에서 새로운 그래픽 표준으로 제안되고 있는 SVG(Scalable Vectors Graphic)는 XML로 작성된 언어의 한 종류

* 숭실대학교 전자계산원(eunyung@soongsil.or.kr)
** 숭실대학교 컴퓨터학부(oh@computing.soongsil.ac.kr)
논문번호 : K01068-0210, 접수일자 : 2001년 2월 10일

로써 2차원 그래픽을 기술한다. SVG는 벡터 그래픽 도형과 이미지, 그리고 텍스트의 3가지 그래픽 객체를 지원하는데, 그래픽 객체들을 그룹화 시킬 수 있으며 스타일을 설정하고 변형시킬 수도 있다[1]. 또한 이전에 랜더링한 객체에 새로운 그래픽 객체를 포함시킬 수도 있고 클리핑 기능과 알파 마스크, 그리고 필터 효과를 적용할 수 있을 뿐만 아니라 애니메이션 시킬 수도 있다. SVG는 XML문서의 서브 트리로 기술되거나 독립적인 SVG 문서로 저장될 수 있으며 SVG안에 또 다른 SVG를 포함할 수도 있다. 다른 웹 문서와 마찬가지로 직접 스크립팅을 할 수 있기 때문에 SVG를 포함하는 XML 문서와 함께 동시에 편집이 가능하다. SVG는 XML로 작성됐기 때문에 텍스트와 같은 단위로 취급할 수 있어서 보다 높은 수준의 통합이 가능하고 XML의 장점을 그대로 수용하고 있어서 로고나 라벨 같은 텍스트 파생 컨텐츠를 포함하고 있는 이미지를 검색하고 재 사용하거나 변형하기 쉽다. 또한 SVG는 그래픽 표기법에서 그래픽 정보를 부호화 한다.

XML이 인터넷상에서의 문서 표준으로 제시되면서 원격 응용들은 XML문서를 조작하고 해석하는 기능을 강화하기 시작하였다. 데이터베이스 관점에서 보면 XML은 다양하고 흥미로운 가능성들을 제기하였는데 여기에는 데이터를 XML 문서로 저장하고 이들 문서 내용에 대한 질의가 포함된다[2]. XML 문서에 대한 질의는 XML문서 셋으로부터 컨텐츠를 분석하고 종합하며 추출할 수 있어야 한다. 현재까지 제안된 질의어에는 Lorel/ Lorel for XML, XML-QL과 XQL등이 있으며 XML 문서에 대한 최상의 질의 성능을 제공하기 위한 여러 가지 데이터 모델들도 연구되고 있다[3]. Semi-structured 데이터를 모델 화한 Lore 시스템에서는 Directed, Unordered, Labeled Graph 형태의 OEM 모델을 기반으로 하고 있으며 Lorel이라는 질의어를 XML문서에 확장시킨 Lorel for XML이라는 질의어를 제공한다. 또한 관계형 데이터베이스 환경을 기반으로 Semi-Structured 텍스트를 표현하기 위한 Tree-Structured 모델[2]에서는 label이 지정된 노드들로 구성된 트리를 구성한다. 그러나 모델상의 각 노드들에는 root로부터의 순서만 있을 뿐 동일한 레벨간의 순서가 없고 특정 노드에 대한 접근은 root로부터 계층구조를 따라서만 가능하고 직접 접근할 수 없다.

본 논문에서는 XML문서를 위한 하이브리드 데이터 모델을 제안한다. 제안하는 모델에서는 순수 XML 문서의 데이터와 구조에 대한 뷰를 표현하고 다양한 검색을 지원할 뿐만 아니라 독립적인 SVG 문서와 SVG를 Element로 가진 XML문서에 대해서도 모든 기능을 지원한다.

Ⅱ. XML과 SVG

1. XML 특징

SGML은 마크업 Element를 기술하기 위해 DTD (Document Type Definition)를 사용하지만 XML은 DTD없이 디자인할 수 있다. DTD없는 XML인 경우에는 Element의 존재와 위치만으로 문서 구조에 대한 정보를 얻을 수밖에 없는데 이를 위해서 Element의 시작과 끝을 파악할 수 있도록 모든 Element는 식 (1)과 같이 시작한 태그에 대한 종료 태그가 있는 형식으로 기술하거나 종료태그가 없는 경우에는 식 (2)와 같이 태그 안에서 "/"로 끝내야 한다. 식 (1)과 (2)에서 ρ는 XML 문서의 Element를 표현하며 c_i는 Element안에 포함된 자식 Element를 의미한다. 그리고 a_i는 속성 이름을, A_i는 속성 값을 나타낸다. 만일 $m=0$인 경우에는 식 (1)과 식 (2)의 표현은 같으며 어떤 형식도 사용할 수 있다.

$$\langle \rho\ a_1 = " A_1 " \ldots a_n = " A_n " \rangle\ c_1 \ldots c_m \quad \text{for } n \geq 0, m \geq 0 \tag{1}$$

$$\langle \rho\ a_1 = " A_1 " \ldots a_n = " A_n " / \rangle \quad \text{for } n \geq 0 \tag{2}$$

또한 태그 안에 또 다른 태그를 포함하는 경우 태그를 겹쳐서 사용할 수 없다. XML 문서의 모든 Element들이 모두 시작 태그와 종료 태그를 가지고 있고 겹쳐서 사용한 태그가 없는 경우의 문서를 'well-formed'문서라고 한다. 또한 모든 속성 값은 이중 부호를 사용하며 속성 값에 이중 부호를 포함한 경우에는 단일 부호를 앞에 붙여서 구분하고 한 Element에서 다른 Element를 참조하는 경우에는 속성 값으로 정의한다.

XML문서는 Semi-structured 데이터의 가장 대표적이다. XML문서 데이터의 특징[4]을 살펴보면 다음과 같다.

1) 문서 데이터는 계층적 구조를 갖는다.
2) 각 문서는 다양한 문서 구조를 가진다.
3) 일반적인 구조에 대한 정의와 정형적인 데이터

유형만을 가진다.

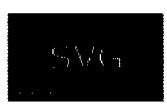
4) 데이터 Element의 크기와 데이터 Element의 성분은 매우 다양하다.
5) 문서 구조에 대해서 사전에 알고 있지 않다.
6) 문서 데이터는 XML 자신에 의해서 기술(Self-describing)된다. 즉 데이터를 기술하지 않으면 데이터를 사용할 수 없다.
7) 데이터는 대부분 텍스트 유형이며 대용량이고 길이를 지정할 수 없고 일반 단어들로 구성되어 있기 때문에 복합적인 의미를 담고 있어서 매치시키기가 어렵다.

2. SVG 개요

SVG는 XML로 작성된 언어의 한 종류로써 2차원 그래픽을 기술하는 백터 마크업 언어이다. SVG 문서는 DTD를 반드시 필요로 하며 그림 1(A)와 같이 별도의 SVG 문서(.svg)로 작성하거나 그림 2와 같이 XML 문서의 서브 트리 형식으로 <svg:svg>와 <svg:svg>태그 안에서 SVG 이미지를 포함시킨다. 또한 SVG 포맷을 지원하는 상용 그래픽 소프트웨어를 이용하여 이미지를 만든 다음 SVG 포맷으로 저장하면 자동으로 SVG 파일이 생성된다.

```
<?xml version="1.0"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20001102//EN"
  "http://www.w3.org/TR/2000/CR-SVG-20001102/DTD/svg-20001102.dtd">
<svg width="3cm" height="2cm" viewBox="0 0 200 120">
<title>Example text SVG</title>
<rect x="1" y="1" width="198" height="118" style="fill:#888888; stroke:blue"/>
  <g>
   <g>
      <path style="fill:none; stroke:#D90000; stroke-width:10"
         d="M50,90 C0,90 0,30 50,30 L150,30 C200,30 200,90 150,90 z"/>
      <path style="fill:#D90000"
         d="M60,80 C30,80 30,40 60,40 L140,40 C170,40 170,80 140,80 z"/>
      <g style="fill:#FFFFFF; stroke:black; font-size:45; font-family:Verdana">
        <text x="52" y="76">SVG</text>
      </g>
   </g>
  </g>
</svg>
```

(A) 독립적인 SVG문서



(B) (A)에 의해서 만들어진 SVG 이미지

그림 1.

SVG 문서에 포함될 Element와 속성들은 이미 정의되어 있으며 text Element에 대한 DTD인 그림 3에서 보여주는 것처럼 각 Element의 속성과 속성 유형도 정의되어 있다. SVG Element에는 그래픽

```
<?xml version="1.0" standalone="yes"?>
<parent xmlns="http://someplace.org"
        xmlns:svg="http://www.w3.org/2000/svg">
   <!-- parent contents here -->
   <svg:svg width="4cm" height="8cm">
     <svg:ellipse cx="2cm" cy="4cm" rx="2cm" ry="1cm" />
   </svg:svg>
   <!-- ... -->
</parent>
```

그림 2. XML문서의 Element로 포함된 SVG

Element, 그래픽 Element나 또 다른 컨테이너 Element를 자식 Element로 가지는 컨테이너 Element, 그리고 다른 Element에 의해서 참조되는 Element 유형이 있다. 또한 그래픽 Element의 경우에는 위치를 설정하기 위한 속성 x, y, 그리고 크기를 정의하는 속성 width와 height등이 있으며 그래픽 객체들을 그룹화 시킬 수 있고 스타일을 설정하고 또한 변형시킬 수도 있으며 필터링을 적용할 수 있다. 또한 크기를 제어하고 SVG 이미지의 특정 부분을 확대할 수 있으며 애니메이션과 이벤트를 제어할 수 있다. 그리고 비트맵 이미지와는 다르게 SVG 이미지안의 텍스트는 선택 및 탐색이 가능하다. 예를 들면 특정 도시 이름을 포함하는 지도 이미지를 찾을 수 있다.

SVG에서의 렌더링 순서는 SVG 문서의 첫 번째 Element가 그려진 다음 기술된 순서대로 처음 그려진 그림의 왼쪽 모서리를 기준으로 하여 그려지게 되고 렌더링 하는 순서가 곧 레이어 순서로 결정되므로 각 Element의 순서는 매우 중요하다.

```
<!Element text (#PCDATA|desc|title|metadata|
                tspan|tref|textPath|altGlyph|a|animate|set|
          animateMotion|animateColor|animateTransform
                %geExt;%textExt;)* >
<!ATTLIST text
  %stdAttrs;
  %testAttrs;
  %langSpaceAttrs;
  externalResourcesRequired %Boolean; #IMPLIED
 .... >
```

그림 3. SVG의 text Element를 위한 DTD

많은 응용에 의해 언어표현을 이해할 수 있도록 하는 SVG의 정보와 압축성은 강력한 웹 응용을 위한 가능성을 열어 놓았는데 서로 다른 데이터베이스간에 그래픽 표현 데이터를 전달하거나 LAN 또

는 웹 응용을 위해 관련 그래픽 및 텍스트를 간략하게 결합하는 것을 가능하게 한다.

Ⅲ. 하이브리드 데이터모델

이 장에서는 본 논문에서 제안하는 하이브리드 데이터모델에서의 데이터 구조와 질의에 대해서 설명한다.

1. 모델 개요

XML 컨텐트 관리 시스템은 Semi-structured 데이터를 얼마나 효과적이고 강력하게 관리할 수 있는가 라는 관점에서 접근되는데 여기에는 두 가지 주요 이슈가 있다. 한가지는 시스템이 XML 데이터를 어떻게 모델 화를 하느냐 하는 것이고 다른 하나는 모델 화된 XML 데이터를 어떻게 실제적으로 구현하느냐 하는 것이다. 데이터 모델에서는 "무엇"을 정의하는 단계이며 실제적으로 구현하는 단계는 "어떻게"를 정의하는 단계이므로 데이터 모델링하는 단계에서 구현하는 단계를 고려하지 않아도 되지만 XML 데이터 모델이 구현될 시스템을 고려하여 정의되는 접근 방식은 어떤 데이터베이스 시스템을 배경으로 하는가에 따라서 기본 골격이 결정된다. XML 컨텐트 관리 시스템을 구현하기 위해 선택되는 데이터베이스 시스템[4]에는 기존의 관계형 시스템과 객체 지향 시스템, 그리고 XML만을 위한 XML-native 데이터베이스 시스템이 있는데 본 논문에서는 현재 보편화되어 있는 관계형 시스템으로의 매핑을 고려한다. 하이브리드 데이터 모델은 Lore의 새로운 XML 데이터 모델[5]의 directed, labeled 그래프와 펜실베니아 대학 데이터베이스 팀에서 Unstructured 데이터를 표현하기 위해 제안한 labeled 그래프 모델[6]의 기본 구조를 기반으로 다음과 같은 요구사항들을 지원하도록 확장되고 개선되었다. 다음 요구 사항에서의 문서는 SVG를 포함하지 않은 XML, SVG를 포함하는 XML, 독립적인 SVG 모두를 통칭한다.

1) 문서의 구조와 데이터를 모두 표현한다.
2) 문서의 구조 및 데이터 변경을 융통성 있게 적용한다.
3) 데이터 모델로부터 문서를 역으로 매핑할 때 문서내의 Element 순서를 모두 유지한다.
4) 질의는 Top-down, Bottom-up, Left-right, Right-left 탐색을 모두 가능하도록 한다.
5) 데이터베이스에 저장된 데이터로부터 데이터 모델을 역으로 매핑하고 따라서 문서로의 매핑까지 지원한다.

```
<p>
  <q id="x1">1st </q>
  <q id="x2">2nd </q>
  <q href="x1">3rd </q>
</p>
```

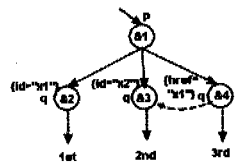


그림 4. (A) XML 문서 (B) (A)문서의 그래프

가장 기본적인 XML 문서의 각 Element는 하나의 데이터 객체로 대응하며 다음과 같은 구조로 정의한다.

Data Object = (OID, LABEL, POID, ROID, COID, DATA, AttributeS)

XML 문서의 기본 구조는 트리이고 여기에 노드간의 상호 참조가 추가되어 XML 문서는 그래프로 표현하는 것이 가장 자연스럽다. 하이브리드 데이터 모델에서는 labeled, ordered, directed 그래프로 XML문서를 표현한다.

Attribute = (OID, AID, Att-Name, Value)

표 1.에서는 그림 4(B)의 그래프의 각 Element 노드가 대응된 Data Object의 튜플 데이터를 보여준다. Attributes는 표 2와 같은 또 다른 튜플 데이터를 가지게 된다.

표 1. 그림 4(B)의 Data Object 튜플 데이터

Element	OID	LABEL	POID	ROID	COID	DATA
p	&1	p	..	..	&2	null
q	&2	q	&1	&3	null	1st
q	&3	q	&1	&4	null	2nd
q	&4	q	&1	null	null	3rd

표 2. 그림 4(B)의 Attribute 튜플 데이터

Element	OID	AID	Att-Name	Value
q	&2	&&1	id	x1
q	&3	&&1	id	x2
q	&4	&&1	href	x1

그림 5는 그림 1(A)의 SVG문서를 그래프로 표현한 것이다.

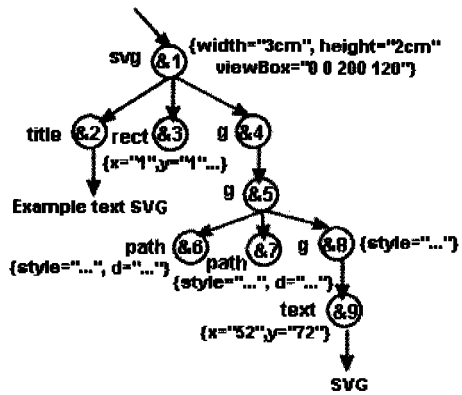


그림 5. 독립적인 SVG문서(그림 1(A))를 위한 그래프

표 3과 표 4는 그림 5의 그래프의 각 Element 노드가 대응된 Data Object와 Attributes 의 튜플 데이터이다.

표 3. 그림 5의 Data Object 튜플 데이터

Element	OID	LABEL	POID	ROID	COID	DATA
svg	&1	svg	..	..	&2	null
title	&2	title	&1	&3	null	Exa...
rect	&3	rect	&1	&4	null	null
g	&4	g	&1	null	&5	null
g	&5	g	&4	null	&6	null
path	&6	path	&5	&7	null	null
path	&7	path	&5	&8	null	null
g	&8	g	&5	null	&9	null
text	&9	text	&8	null	null	SVG

표 4. 그림 5의 Attribute 튜플 데이터

Element	OID	AID	Att-Name	Value
svg	&1	&&1	width	3cm
svg	&1	&&2	height	2cm
svg	&1	&&3	viewBox	0,0,200,120
rect	&3	&&1	x	1
rect	&3	&&2	y	1
rect	&3	&&3	width	198
rect	&3	&&4	height	118
rect	&3	&&5	style	fill:#...
...	...	...	...	...

SVG 문서를 XML문서의 Element로 포함하는 경우에 대해서는 그림 5의 그래프를 그림 4(B)의 서브 트리 형식으로 표현하면 된다.

앞에서의 하이브리드 데이터 모델의 도메인은 하나의 문서였다. 따라서 검색은 하나의 문서를 대상으로 한다. 검색을 문서 레벨로 하려면 예를 들어서, 특정 텍스트를 포함하는 SVG문서(또는 SVG 문서를 포함하는 XML 문서)를 검색하려면 도메인을 다중 문서로 확장해야한다. 이를 위해 Element 튜플 구조에 문서의 유일한 ID를 갖는 DID를 추가할 수 있다.

Data Object = (DID, OID, LABEL, POID, ROID, COID, DATA, AttributeS)

이렇게 하면 *OID*는 해당하는 문서에 대해서만 유일한 값이 되므로 Attribute의 튜플 구조에도 다음과 같이 DID를 포함한다.

Attribute = (DID, OID, AID, Att-Name, Value)

2. 검색

Semi-structured 데이터에 대한 질의어는 객체 데이터 시스템을 위한 ODMG 모델의 질의어인 OQL을 확장시킨 형태[7]와 XML 문서의 트리 모델을 위한 질의어[8]들이 제안되었는데 이 절에서는 Semi-structured 데이터 모델의 대표적인 프로젝트인 스탠포드 대학의 Lore시스템에서 사용한 질의어 LoreL (Lore Language)[9]을 XML로 확장한 질의어[5]와 W3C에서 XML을 위한 질의어로 제안한 XML-QL[10]을 이용하여 하이브리드 데이터 모델에서의 검색을 보여준다.

특정 텍스트를 포함하는 SVG 이미지에 대한 검색은 LABEL이 "svg"인 Element가 포함된 계층 구조상에서 LABEL이 "text"인 노드의 DATA 부분을 스캔한다. 이것은 특정 Element가 포함된 XML 문서에서 특정 태그 사이에 특정 문자열을 포함하는 문서에 대한 검색과 동일하다. 또한 독립적인 SVG 문서와 일반적인 XML문서 안에 Element로 포함된 SVG 모두를 검색한다. 검색 결과는 문서 계층상에서 svg Element를 root로 하는 서브 계층이 되므로 결국에는 SVG 이미지가 검색 결과가 된다.

LoreL의 경우에는 최상위 객체가 필요하므로 그래프 상의 최상위 객체를 root라고 가정한다.

XML을 위한 LoreL의 경우 :

```
( SELECT X
FROM root.#.svg X
WHERE X.#.text LIKE "%korea%" )
UNION
( SELECT Y
FROM  root.#.svg:svg Y
WHERE Y.#.svg:text LIKE "%korea%" )
```

XML-QL의 경우:

```
WHERE
<$*><(svg|svg:svg)><$*><(text|svg:text)>*korea*
        </></></></></>
      IN "XML데이터소스를 표현하는 URI" ,
CONSTRUCT <(svg|svg:svg)> $a </>
```

특정 Attribute가 주어진 조건을 만족하는 텍스트를 포함하는 SVG 이미지에 대한 검색은 LABEL이 "svg"인 Element가 포함된 계층 구조상에서 LABEL이 "text"인 DATA 부분을 스캔한 다음 Attibute의 Att-Name과 Value 영역을 매치 시킨다. 이것은 특정 Element가 포함된 XML 문서에서 주어진 속성과 값을 가지고 있는 특정 태그 사이의 특정 문자열을 포함하는 문서에 대한 검색과 동일하다.

XML을 위한 LoreL의 경우 :

```
( SELECT X
FROM #.svg X
WHERE X.#.text LIKE "%korea%" AND
        X.#.text.@x > "10" AND
        X.#.text.@y > "30" )
UNION
( SELECT Y
FROM  #.svg:svg Y
WHERE Y.#.svg:text LIKE "%korea%"
        Y.#.svg:text.@x > "10" AND
        Y.#.svg:text.@y > "30" )
```

XML-QL의 경우 :

```
WHERE <$*><$e1><$*><$e2 x=$x y=$y>*korea*
        </></></></></>
      IN "XML데이터소스를 표현하는 URI" ,
      $e1 IN {svg, svg:svg}
      $e2 IN {text, svg:text}
        $x > "10"
      $y > "30"
CONSTRUCT <$e1> $a </>
```

3. 오퍼레이션

XML을 위한 Lore 모델[5]과 펜실베니아 대학의 labeled 그래프 모델[6]은 Top-down 경로 정보만을 가지고 있기 때문에 같은 부모를 가지고 있는 동일 레벨의 Element에 대하여 순서 있는 조건으로 검색할 수 없다. SVG 문서에서는 동일 레벨의 Element 기술 순서가 레이어의 개념이 되므로 이 순서는 매우 중요한데 따라서 이들 모델에서는 특정 순서로 그래픽 Element들을 포함하는 SVG 이미지를 검색할 수 없다. 또한 질의시 시작 경로를 명시해야하며 추출된 문서의 구조는 원본 문서의 계층과 순서를 보장할 수 없다. 따라서 앞에서 보여준 검색은 기존의 데이터 모델을 기반으로 한 질의어를 이용하여 Top-down 접근의 검색 예만 보여주었다.

하이브리드 데이터 모델에서는 문서의 데이터와 구조를 그대로 유지하고 있으므로 어떤 접근의 검색도 가능하다. 앞 절에서 검색을 위하여 선택한 질의어들은 하이브리드 모델에서의 다양한 탐색을 지원하지 못하고 있는데 하이브리드 모델에서의 모든 가능한 탐색을 위해 질의어에 포함해야할 추가적인 오퍼레이션들은 다음과 같다.

구조 및 내용을 통합한 오퍼레이션은 태그 이름을 포함하는 LABEL과 문서의 구조 정보를 담고 있는 POID, ROID, COID를 대상으로 구조 탐색을 하며 내용 탐색은 DATA와 Attribute를 대상으로 한다. 이들 오퍼레이션은 다음과 같은 유형의 접근을 허용한다.

유형 1) C 태그 이름을 가진 Element가 k번째 Child인 P 태그 이름을 가진 모든 Element.

이 유형의 탐색은 Bottom-up 스캔이 효율적이며, 먼저 LABEL을 스캔하여 C인 Object를 검색한 다음 POID를 참조하여 k번째 부모 노드를 찾아서 LABEL이 P인 Element를 반환한다. 즉, 결과는 부모 Element가 반환된다. 이 유형의 탐색 방법은 P 태그 이름을 가진 Element가 k번째 Parent인 C 태그 이름을 가진 모든 Element를 탐색하는 경우에도

적용할 수 있으며 이때는 LABEL이 C인 자식 Element가 반환된다.

Parent(P C k) 와 *Child(C P k)*

만일 C 태그 이름을 가진 Element가 자손인 P 태그 이름을 가진 모든 Element를 검색하는 경우나 P 태그 이름을 가진 Element가 조상인 C 태그 이름을 가진 모든 Element를 검색하는 경우에는 다음과 같은 오퍼레이션을 사용한다.

Ancestor(P C) 와 *Descendant(C P)*

P에 특정 속성 값을 가진 조건을 추가할 수 있도록 다음과 같이 정의한다. 이 경우에는 Attribute 영역을 스캔하게 된다.

P ::= identifier

*|identifier.@attname=value{,identifier.@attname=value}**

C는 자식 Element이므로 데이터 값을 가질 수 있다. 따라서 특정 데이터 값을 가진 조건을 추가할 수 있도록 다음과 같이 정의한다. 이 경우에는 DATA 영역을 스캔하게 된다.

C ::= P | identifier&String

유형 2) R 태그 이름을 가진 Element가 k번째 오른쪽 형제인 L 태그 이름을 가진 모든 Element.

이 유형의 탐색은 Left-right 스캔으로 접근되며, 먼저 LABEL을 스캔하여 L인 Object를 검색한 다음 ROID를 참조하여 k번째 형제 노드를 찾아서 LABEL이 R인가를 확인하여 LABEL이 L인 Element를 반환한다. 즉, 그래프 상에서 조건에 대해 상대적으로 왼쪽에 위치한 Element가 반환된다. 이 유형의 탐색 방법은 L 태그 이름을 가진 Element가 k번째 왼쪽 형제인 R 태그 이름을 가진 모든 Element를 탐색하는 경우에도 적용할 수 있으며 이때는 그래프 상에서 조건에 대해 상대적으로 오른쪽에 위치한 Element가 반환된다.

Lsibling(L R k) 와 *Rsibling(R L k)*

만일 k값을 지정할 수 없는 경우, 즉 R 태그 이름을 가진 Element가 오른쪽 형제인 L 태그 이름을 가진 모든 Element를 검색하는 경우와 L 태그 이름을 가진 Element가 왼쪽 형제인 R 태그 이름을 가진 모든 Element를 검색하는 경우에는 다음과 같은 오퍼레이션을 사용한다.

Psibling(R L) 와 *Fsibling(L R)*

L과 R이 특정 데이터 값을 가진 조건과 특정 속성 값을 가진 조건을 추가할 수 있도록 다음과 같이 정의한다.

L,R ::= C

구조 및 내용을 통합한 오퍼레이션의 예를 들면 다음과 같다.

예 1) 태그 명이 "TEXT"인 Element가 "SVG"값을 가지고 있고 이 Element를 3번째 Child로 가진 "G" 태그 이름을 가진 모든 Element를 포함하는 SVG 이미지를 검색한다.

Ancestor(SVG (Parent(G TEXT& "SVG" 3))

예 2) 태그 명이 "RECT"인 Element의 속성 width가 10이고 "SVG"값을 가지고 있는 태그 명이 "TEXT"인 Element를 오른쪽 형제로 가지며 부모 중에 "G" 태그 이름을 가진 Element를 포함하는 SVG 이미지를 검색한다.

Ancestor(SVG(Ancestor
(G Psibling(RECT.@width=10 TEXT&"SVG"))))

Ⅳ. 결 론

하이브리드 모델에서, XML문서의 그래프의 각 노드는 고정된 컬럼을 갖는 Data Object 튜플로 매핑 되며 관계형 시스템에서는 하나의 릴레이션으로 매핑된다. 그리고 각 Data Object에 포함되는 Attributes는 별도의 튜플로 분리하였으며 마찬가지로 관계형 시스템에서 하나의 릴레이션으로 매핑된다. 본 논문에서는 문서에 대한 튜플은 별도로 구성하지 않았지만 문서에 대한 메타 정보를 위해 추가할 수 있다.

하이브리드 모델은 XML문서에 대한 데이터 뷰와 구조 뷰를 모두 표현하고 관계형 시스템으로의 매핑이 자연스럽게 유도될 수 있다. 또한 Top-down

검색뿐만이 아니라 Bottom-up과 Left-right, 그리고 특정 계층 레벨을 갖는 Element에 대한 검색 또한 가능하며 계층 순서 및 같은 레벨에서의 순서를 유지하기 때문에 검색 결과에 소스 문서의 구조가 그대로 표현된다.

본 논문에서는 문서 검색 시스템을 위한 질의어에서 지원해야 할 오퍼레이션들을 보여주었으며 이러한 오퍼레이션들은 XML 기반의 질의어에 포함시키거나 관계형 시스템의 스토어드 프로시저 또는 API로 구현될 수 있다.

참 고 문 헌

[1] J. Bowler, M. Capsimalis, R. Cohn, D. Dodds, A. Donoho, D. Duce..., "Scalable Vector Graphics (SVG) 1.0 Specification" http://www.w3.org/TR/2000/CR-SVG-20001102/, 2 November 2000.

[2] Jayavel Shanmugasundaram, H. Gang, Kristin Tufte, Chun Zhang, David J. DeWitt, and Jeffrey F. Naughton, "Relational databases for querying XML documents: Limitations and opportunities", In VLDB'99, In Proc. of 25th International Conference on Very Large Data Bases, Edinburgh, Scotland, pp. 302-304, 1999.

[3] M. Erdmann and R. Studer, "How to structure and access XML documents with ontologies," Data and Knowledge Engineering, 2000.

[4] T. Arnold-Moore, M. Fuller and R. Sacks-Davis, "System Architectures for Structured Document Data", Markup Languages, Theory and Practice, 2.1 pp. 15-44, 2000

[5] R. Goldman, J. McHugh, and J. Widom, "From Semistructured Data to XML: Migrating the Lore Data Model and Query Language", In Proc. of the 2nd International Workshop on the Web and Databases (WebDB '99), pp. 25-30, Philadelphia, Pennsylvania, June 1999.

[6] Peter Buneman, Susan Davidson, and Dan Suciu, "Programming constructs for unstructured data", In Proc. of 5th International Workshop on Database Programming Languages, Gubbio, Italy, September 1995.

[7] D. Quass, A. Rjaraman, Y. Sagiv, and J. Ullman, "Querying semistructured heterogeneous information", In Proc. of Fourth International Conference on Deductive and Object-oriented Databases, pp. 319-344, Dec. 1995.

[8] Peter Buneman, Susan Davidson, Gerd Hillebrand, and Dan Suciu, "A query language and optimization techniques for unstructured data", In Proc. of ACMSIGMOD International Conference on Management of Data, pp. 505--516, Montreal, Canada, June 1996.

[9] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. Wiener, "The Lorel Query Language for Semistructured Data", International Journal on Digital Libraries, 1(1), pp. 68-88, April 1997.

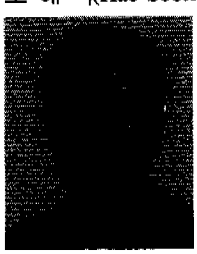
[10] Alin Dutsch, Mary Fernandez, Daniela Florescu, Alon Levy, Dan Suciu, "XML-QL:A Query Language for XML", http://www.w3.org/TR/1998/NOTE-xml-ql-1998

김 은 영(Eun-Young Kim) 정회원

1987년 : 숙명여자대학교 전산학과(이학사)
1993년 : 숙명여자대학교 전산학과(이학석사)
2000년 : 숭실대학교 컴퓨터학과 박사과정 수료
숭실대학교 전자계산원 전임

<주관심 분야> 인터넷 표준, 멀티미디어 데이터베이스

오 해 석(Hae-Seok Oh) 정회원



1975년 : 서울대학교 응용수학과 (공학사)
1981년 : 서울대학교 계산통계학과 (이학석사)
1989년 : 서울대학교 계산통계학과 (이학박사)

1983년~현재 : 숭실대학교 컴퓨터학과 교수
1996년~1999년 : 숭실대학교 부총장 역임
2000년 : 스탠포드 대학 객원 교수

<주관심 분야> 멀티미디어통신, 웨이블릿영상 코딩, 멀티 미디어응용