한국지능정보시스템학회논문지 제10권 제3호 2004년 12월(pp.27~37)

XML 문서의 효율적인 검색과 재사용성을 지원하는 데이터 모델

김은영
안산공과대학 디지털미디어학과
(key@act.ac.kr)

천세학
한림대학교 경영대학 경영학과
(shchun@hallym.ac.kr)

인터넷상에서 데이터를 표현하고 데이터를 서로 교환하기 위한 문서 표준으로 XML이 제시되고 있다. XML은 또한 웹상에 산재되어 있는 문서에 대한 쉬운 검색 및 재사용을 지원하는 문서 표준으로도 부각되고 있다. XML 콘텐츠 관리 시스템을 구현할 때 Semi-structured 데이터를 얼마나 효율적이고 효과적으로 검색 및 관리할 수 있는 가하는 점과 XML의 특징인 재사용성을 얼마나 지원해 줄 수 있는가를 고려해서 XML 데이터를 모델링 해야 한다. 또한 모델링한 데이터를 어떻게 실질적으로 저장해야 할 것인 가도 고려해야 한다. 본 논문에서는 XML 문서의 데이터를 데이터 시스템에 저장하고 검색하기 위한 새로운 데이터 모델을 제안한다. 제안하는 데이터 모델은 XML 문서의 데이터 뷰와 구조 뷰를 모두 표현하며 XML 문서를 위한 새로운 데이터 시스템이나 기존의 관계형 시스템 모두를 고려한다.

키워드 : XML 문서관리, 관계형데이터베이스, 객체 데이터베이스, 다중형식 정보복구

논문접수일 : 2004년 8월 게재확정일 : 2004년 12월 교신저자 : 천세학

1. 서론

XML(eXtensible Markup Language)은 SGML에 기반을 둔 간단하고 유연성 있는 마크업 언어로써 사용자들이 원하는 정보를 검색하는 방법을 제공하며, XML의 풍부한 데이터 표현 방식은 지능적인 메커니즘을 만들어서 웹상에서 사업을 할 수 있도록 지원하기 때문에 전자상거래분야에서 가장 각광받고 있으며 그 외에도 온라인 뱅킹, 푸쉬 기술, 검색 엔진 그리고 웹 기반 제어시스템과 에이전트등 웹의 거의 모든 분야에서 활용되고 있고 그 적용 범위는 급속도로 확산되고 있다. 또한 XML 문서의 재 사용성은 검색 엔진을 통해서 결과로 반환된 XML 문서나 기존의 또 다른 XML 문서, 또는 자신에게 전송된 XML 문서로부터 원하는 데이터만을 추출하여 자신만의 데이터 구조로 가공하여 저장할 수 있도록 만들었다.

XML 문서의 데이터를 관리하기 위한 접근은 XML 문서를 위한 새로운 데이터베이스 시스템을 이용하는 것과 관계형 데이터 시스템과 같은 기존의 데이터베이스 시스템을 이용하는 것으로 분류할 수 있다. XML 문서를 위한 새로운 데이터베이스 시스템은 Semi-structured 형식의 XML 데이터를 표현하는 새로운 데이터 모델을 기반으로 한다. 그리고 기존의 관계형 시스템이나 객체형 시스템을 고려한 경우에는 관계형이나 객

체형 시스템에 잘 매핑될 수 있는 매핑모델이 요구된다.

XML 문서의 데이터 모델에 대한 기존 연구에서는 XML 문서 종류에 따라서 개별적인 데이터 구조를 정의했다. 따라서 새로운 구조와 데이터를 가진 XML 문서를 위해 새로운 데이터 구조를 정의해야 하므로 도메인을 확대하기가 힘들고 적용하는 대상에 따라서 데이터 구조를 매번 다르게 정의해야 하며 정보에 대한 검색은 Top-down 접근만이 가능하다는 문제점이 있다. 또한 기존의 관계형 시스템이나 객체형 시스템으로 매핑하기 위한 데이터 모델에서는 XML 원본 문서의 데이터 뷰와 구조 뷰를 모두 잃어버리게 되어 저장된 데이터로부터 XML 문서를 다시 생성해 낼 수 없다. XML 문서를 위한 데이터 모델은 XML 문서의 데이터 및 구조와 관계없이 유일한 데이터 구조를 가져야 응용의 범위를 쉽게 확장할 수 있다. 또한 XML 문서의 정보에 대한 검색은 Top-down 접근뿐만이 아니라 Bottom-up 접근이나 Left-Right 접근의 검색도 가능해야 하고 XML 문서 구조에 대한 사전 지식이 없어도 검색이 가능해야 한다.

본 논문에서는 이러한 요구사항을 모두 지원하며 XML 문서를 위한 새로운 데이터베이스와 기존의 관계형 데이터 시스템 모두에 적용 가능하고 XML 문서의 데이터 뷰와 구조 뷰를 모두 표현하는 데이터 모델을 제안한다.

2. XML 문서를 위한 데이터 모델

2.1 데이터 모델의 필요성

XML 문서는 데이터와 구조를 기술한다. 데이터를 기술한다는 이유 하나로 XML 문서를 하나의 데이터베이스라고 볼 수는 없다. XML 문서가 데이터를 포함하고 있기는 하지만 데이터를 처리하는 추가적인 소프트웨어 없이는 스스로는 어떠한 작업도 할 수 없는 일반적인 텍스트 파일일 뿐이기 때문이다. 그러나 XML 문서와 관련된 XML 도구, 그리고 XML 제반 기술 모두를 통합한다면 하나의 데이터베이스라고 볼 수도 있는데, 여기에는 데이터베이스 시스템의 요소인 저장소, 스키마, 질의어, 프로그래밍 인터페이스 등이 포함되기 때문이다. 그렇지만 데이터베이스 시스템에서 지원하는 효율적인 저장소, 인덱스, 보안, 트랜잭션과 데이터 완전성, 다수-사용자 접근, 트리거, 다중 문서에 대한 질의 등은 지원하지 못한다. 따라서 데이터가 많지 않으면서 사용자가 적고 일반적인 성능만이 요구되는 환경이라면 XML 문서를 데이터베이스로서 사용할 수 있지만 사용자 수가 많으면서 데이터의 완전성과 높은 성능을 요구하는 환경에서는 XML 문서를 데이터베이스로서 사용할 수 없다. 이 경우에는 XML 데이터와 구조를 저장할 데이터베이스가 필요하게 된다. 현재 XML 문서의 데이터를 저장하기 위한 방법으로는 XML 문서만을 위한 새로운 데이터베이스 시스템에 새로운 데이터 모델로 저장하는 방법과 기존의 관계형이나 객체형 시스템에 XML 문서의 데이터를 매핑하여 저장하는 방법이 있다.

XML 문서만을 위한 새로운 데이터 시스템은 XML 문서의 데이터를 효과적으로 저장하고 표현 및 질의가 가능한 새로운 모델을 요구하며 기존의 관계형 시스템이나 객체형 시스템에서의 데이터 모델은 이들 데이터 시스템 모델과 자연스럽게 매핑될 수 있는 데이터 모델이 요구된다.

2.2 기존 데이터 모델

XML 문서의 새로운 데이터 시스템을 고려한 대표적인 데이터 모델로는 스텐포드 대학의 XML 문서를 위해 확장된 OEM 모델[1]과 펜실베니아 대학의 Edge Labeled Graph 모델[2,3], 그리고 AT&T의 리스트 모델[4]이 있다.

LORE 시스템의 OEM모델을 XML 문서의 데이터를 표현하기 위해 확장한 모델에서는[5] XML Element를 (eid, value)로 표현하는데 eid는 Element의 유일한 ID이고 Value는 원자(atomic) 값인 텍스트 문자열이거나 XML 태그, 속성 리스트, 참조하는 Element 리스트 등을 포함하는 복합 값이다.

Edge Labeled Graph[3]모델에서의 기본 정보는 레이블에 의해 표현하며 관계형 데이터베이스를 인코딩하는 기능이 강력하다. 레이블로써 사용될 수 있는 데이터 유형은 String, Integer, Real 등이며 이러한 유형은 릴레이션에서 값으로 사용된다.

리스트 모델[4]은 XSLT를 위한 데이터 모델[6]을 기본으로 하여 참조 노드를 추가하였으며 속성과 Element노드를 병합했다. 모델의 기본 유형은 Node이며 각 Node는 텍스트, Element, 또는 참조중의 하나이다. 속성은 문자 에트(@)를 이름 앞에 추가할 뿐 Element와 동일한 방법으로 취급하며 자식은 항상 하나의 리스트로 표현한다.

관계형 매핑 모델인 X-Ray[7]에서는 XML의 DTD를 관계형 시스템으로 매핑한다. DTD의 Element는 Element 유형과 반복횟수, 그리고 속성이 포함되어 있는지에 따라서 관계형 시스템에서 하나의 릴레이션이나 릴레이션의 한 속성으로 매핑하며 DTD에서의 속성은 관계형 시스템에서도 릴레이션의 속성으로 매핑된다. 관계형 시스템을 위한 모델중의 하나인 Inlining 기법[8]은 하나의 Element를 포함된 자식 Element와 함께 가능하면 한 릴레이션에서 표현하며 DTD에서 Element는 ER-Model의 entity 또는 attribute로 대응시킨다.

3. 제안하는 데이터모델

3.1 모델 개요

본 논문에서는 XML만을 위한 새로운 XML-native 데이터베이스 시스템을 설계할 때, 그리고 기존의 관계형 시스템에 적용할 수 있는 데이터 모델을 제안한다. 제안하는 데이터 모델은 Lore의 XML 데이터 모델과 Edge Labeled Graph 모델의 기본 구조를 기반으로 다음과 같은 요구사항들을 지원하도록 확장되고 개선되었다.

1) 글로벌 도메인에 있는 문서를 대상으로 한다.
2) 데이터 모델은 선택되는 데이터시스템의 종류에는 의존하지 않는다.
3) 문서의 구조와 데이터를 모두 표현한다.
4) 문서의 구조 및 데이터 변경을 융통성 있게 적용한다.
5) 데이터 모델로부터 문서를 역으로 매핑 할 때 문서내의 Element 순서를 모두 유지한다.
6) 질의는 Top-down, Bottom-up, Left-right, Right-left 탐색을 모두 가능하도록 한다.

3.2 기본 모델

XML 문서의 Element 구조는 트리로 표현가능하고 여기에 Element간의 상호 참조가 추가되어

XML 문서는 그래프로 표현하는 것이 가장 자연스럽다. XML 문서 데이터의 기본 구조를 그래프로 표현할 때 XML 문서의 각 Element는 자신의 태그 명을 가지고 있는 노드를 뿌리(root)로 하는 그래프에 해당한다. 따라서 Element에 포함되는 또 다른 Element는 자신을 포함하는 Element의 그래프 안에서 서브 그래프로 표현되며 노드와 노드는 간선에 의해 연결되고 계층 의미를 갖거나 참조 의미를 표현한다.

기본 모델에서는 그래프상의 모든 노드와 간선에 레이블이 있고(labeled), 노드 또는 간선사이에 순서가 있으며(ordered), 간선은 방향이 있는(directed) 그래프로 XM L문서를 표현한다.

XML 문서를 그래프 $G=(V,E,A)$로 표현할 때 V는 정점 즉 노드의 셋을 의미하고 E는 간선(edge)의 셋을 의미한다. 그리고 A는 Element의 시작 태그에 정의된 속성(Attributes) 셋을 의미한다. XML 문서에서의 한 Element는 해당하는 시작 태그부터 시작하여 시작 태그와 관련된 종료태그까지의 내용 모두를 의미하는데 제안하는 모델의 그래프에서 XML 문서의 각 Element는 해당 Element의 태그를 뿌리(root) 노드로 하는 문서의 서브 트리/그래프로 표현된다. 노드 V의 레이블(label)은 중간 노드인 경우에는 태그 값이고 단말(leaf) 노드인 경우에는 값(NULL과 EMPTY 포함)에 대응한다. 노드 V_i로부터 나오는 간선 E_i의 레이블은 노드 V_i와 도착하는 자식 노드 V_j의 관계에 대응한다. 자식 노드 V_j가 또 다른 Element라면 노드 V_i의 자식 Element 정보를 나타내는 *"CHILD"*를 레이블로 갖고 자식 노드 V_j가 Element의 값이라면 노드 V_i의 값 정보를 나타내는 *"VALUE"*를 레이블로 갖는다. 따라서 그래프상의 노드 셋인 V와 간선 E는 다음과 같이 표현할 수 있다.

$$V = V_{tag} \cup V_{value}$$
$$E = E_{child} \cup E_{value}$$

가. 기본 속성

(1) 노드의 기본 속성

$V(G)$가 그래프상의 노드 셋이고 v가 $v \in V(G)$를 만족할 때 노드 v의 기본적인 속성은 다음과 같다.

- *label(v)* : 뿌리 또는 중간 노드인 경우에는 태그 이름이 반환되고 단말(leaf) 노드인 경우에는 태그사이의 값을 반환한다. 공백을 갖는 Element의 두 가지 경우를 구분하기 위해 시작 태그와 종료 태그 사이의 값이 없는 경우에는 *NULL*을 반환하고 하나의 태그 안에서 Element를 기술한 경우에는 *EMPTY*를 반환한다.
- *level(v)* : 문서의 뿌리 노드를 1로 시작하여 문서 전체 그래프 상에서의 해당 노드의 계층 레벨을 가진다. 부모가 다른 노드인 경우에도 동일한 레벨을 가질 수 있다.

(2) 속성 셋의 기본 속성

$A(G)$가 문서상의 모든 속성 셋이고 속성 a가 $a \in A(G)$를 만족할 때 속성 a로부터 구할 수 있는 값은 다음과 같다.

- *name(a)* : 속성 a의 속성 이름을 반환한다.
- *value(a)* : 속성 a의 값을 반환한다.
- *node(a)* : 속성 a가 정의된 노드를 반환한다.
- *type(a)* : 속성의 유형을 반환한다.

(3) 간선의 기본 속성

*E(G)*가 그래프상의 간선 셋이고 간선 *e*가 $e \in E(G)$를 만족할 때 간선 *e*의 기본적인 속성은 다음과 같다.

- *from(e)* : 간선 *e*가 출발하는 노드를 반환한다.
- *to(e)* : 간선 *e*가 도착하는 노드를 반환한다.
- *relation(e)*: 간선이 출발하는 노드와 도착하는 노드의 관계 (*CHILD* 또는 *VALUE*)를 반환한다.
- *order(e)* : 동일한 부모 노드로부터 출발한 간선들 간의 순서를 반환한다.

나. 추가되는 속성

(1) 노드의 추가 속성

*V(G)*가 그래프상의 노드 셋이고 *v*가 $v \in V(G)$를 만족할 때 노드와 간선 $e(e \in E(G))$의 기본적인 속성으로부터 유도되는 노드 *v*의 추가 속성들은 다음과 같다.

- *child(v)* : 노드 *v*의 자식 노드를 반환한다.
  $child(v) \equiv \{to(e) \mid \exists e \in E: from(e)=v\}$
- *parent(v)* : 노드 *v*의 부모 노드를 반환한다.
  $parent(v) \equiv \{from(e) \mid \exists e \in E: to(e)=v\}$
- *ancestor(v)* : 노드 v의 조상 노드의 셋을 반환한다.

```
while ({e| ∃e∈E(G):to(e)=v}≠{}) {
    ancestorlist := ancestorlist ∪ from(e) ;
v:=from(e);
} end while;
return ancestorlist;
```

- *descendent(v)* : 노드 *v*의 자손이 되는 노드의 셋을 반환한다.

```
void descendent(node v) {
    for (each {e| ∃e∈E(G):from(e)=v} ) {
        node u := to(e);
        descendentlist := descendentlist ∪ u;
        descendent(u) ; } }
```

- *root(v)* : 노드 *v*의 뿌리 노드를 반환한다.

```
while ({e| ∃e∈E(G):to(e)=v} ≠ {} ) {
    rootnode :=from(e);
    v:=from(e);
} end while;
return rootnode;
```

- *sibling(v)* : 노드 *v*와 부모 노드가 같은 형제 노드의 셋을 반환한다.

$sibling(v) \equiv \{to(e) \mid \exists e \in E(G): from(e)= \{from(e) \mid \exists e \in E(G): to(e)=v\}\} - v$
$\equiv \{to(e) \mid \exists e \in E(G): from(e)= parent(v)\} - v$

- *identity(v)* : 노드 *v*와 동일한 부모 노드를 가진 형제 노드 중에서 노드 *v*와 동일한 레이블을 가진 형제 노드만을 반환한다.

$identity(v) \equiv \{to(e) \mid \exists e \in E(G): from(e)=\{from(e) \mid \exists e \in E(G): to(e)=v\} \wedge label(to(e))=label(v)\} - v$

속성 $a(a \in A(G))$의 기본적인 속성으로부터 유도되는 노드 *v*의 부가적인 속성들은 다음과 같다.

- *attributeS(v)* : 노드 *v*가 뿌리 또는 중간 노드인 경우에는 시작 태그에 정의한 속성과 속성값의 셋을 반환한다.
  $attributeS(v) \equiv \{name(a), value(a) \mid \exists a \in A(G): node(a)=v\}$
- *@att_name(v)* : 노드 *v*에 정의된 특정 이름을 가진 속성의 값을 반환한다.
  $@att\_name(v) \equiv \{value(a) \mid \exists a \in attributeS(v) : name(a)= \text{“att\_name”} \}$

(2) 간선의 추가 속성

노드 $v(v \in V(G))$와 간선 $e(e \in E(G))$의 기본적인 속성으로부터 유도되는 간선 e의 추가 속성은 다음과 같다.

· *successor(e)/predecessor(e)* : 간선 e와 동일한 출발 노드를 갖는 간선 중에서 간선 e 의 다음 또는 앞에 있는 간선 셋을 반환한다.

$successor(e) \equiv \{e_2 | \exists e_2 \in E(G): from(e_2) = from(e) \wedge order(e_2) > order(e)\}$

$predecessor(e) \equiv \{e_2 | \exists e_2 \in E(G): from(e_2) = from(e) \wedge order(e_2) < order(e)\}$

· *after(e)/before(e)* : 간선 e와 동일한 출발 노드를 갖는 간선 중에서 간선 e의 다음 또는 이전 간선을 반환한다.

$after(e) \equiv \{e_2 | \exists e_2 \in E(G): from(e_2) = from(e) \wedge (order(e_2) = order(e)+1)\}$

$before(e) \equiv \{e_2 | \exists e_2 \in E(G): from(e_2) = from(e) \wedge (order(e_2) = order(e)-1)\}$

· *first(e)/last(e)* : 간선 e와 동일한 출발 노드를 갖는 간선 중에서 처음/마지막 간선을 반환한다.

$first(e) \equiv \{e_2 | \exists e_2 \in E(G): from(e_2) = from(e) \wedge order(e_2) = 1\}$

$last(e) \equiv \{e_2 | \exists e_2 \in E(G): from(e_2) = from(e) \wedge (after(e_2) = \{\})\}$

(3) 노드의 두 번째 추가 속성

간선 $e(e \in E(G))$의 추가 속성으로부터 유도되는 노드 v의 두 번째 추가 속성은 다음과 같다.

· *next(v)/previous(v)* : 노드 v의 형제 노드 중에서 노드 v의 다음/이전 노드를 반환한다.

$next(v) \equiv \{ to(after(e)) | \exists e \in E(G) : to(e)=v\}$

$previous(v) \equiv \{to(before(e)) | \exists e \in E(G) : to(e)=v\}$

· *older(v)/younger(v)* : 노드 v의 형제 노드 중에서 노드 v의 앞 또는 뒤에 위치한 노드 셋을 반환한다.

$younger(v) \equiv \{to(successor(e)) | \exists e \in E(G) : to(e)=v\}$

$older(v) \equiv \{to(predecessor(e)) | \exists e \in E(G) : to(e)=v\}$

· *oldest(v)/youngest(v)* : 노드 v의 형제 노드 중에서 가장 처음 또는 마지막에 위치한 노드를 반환한다.

$oldest(v) \equiv \{to(first(e)) | \exists e \in E(G) : to(e)=v\}$

$youngest(v) \equiv \{to(last(e)) | \exists e \in E(G) : to(e)=v\}$

3.3 DTD를 고려한 확장 모델

DTD는 XML 문서 교환 시에 전송되는 XML 문서에 대한 상호간의 규약으로서 의미를 가진다. XML 문서가 DTD를 수반하는 경우에는 DTD 문서의 데이터로부터 Element와 Element 간의 참조관련 정보를 얻을 수 있다.

XML 문서의 그래프 상에 추가되는 모든 참조간선 셋을 $R(G)$라고 할 때 XML 문서를 위한 그래프 G는 다음과 같이 확장된다.

$G=(V, E, R, A)$

(1) 참조 간선의 기본 속성

$R(G)$*이 그래프상의 참조간선 셋이고* r*이* $r \in R(G)$*를 만족할 때 참조간선* r*의 기본적인 속성은 다음과 같다.*

- *refTo(r)* : 참조 간선이 도착하는 노드를 반환한다.
- *refFrom(r)* : 참조 간선이 출발하는 노드를 반환한다.
- *retAttr(r)*: 다른 Element를 참조하는 Element의 속성이름을 반환한다.

(2) 노드의 추가 속성

참조간선 $r(r \in R(G)$의 기본 속성으로부터 유도되는 노드 $v(v \in V(G))$의 추가 속성들은 다음과 같다.

- *references(v)* : 노드 v로부터 출발한 참조 간선들이 도착하는 노드 셋을 반환한다.
  $references(v) \equiv \{refTo(r) \mid \exists r \in R(G) : refFrom(r)=v\}$
- *referredBy(v)* : 노드 v로 도착하는 참조 간선들이 출발하는 노드 셋을 반환한다.
  $referredBy(v) \equiv \{refFrom(r) \mid \exists r \in R(G) : refTo(r)=v\}$

3.4 데이터 모델의 4가지 객체

제안하는 데이터 모델에서 XML문서는 하나의 그래프로 표현된다. 그래프는 노드, 간선, 속성, 참조간선 등으로 구성되며 각 구성 요소들은 하나의 객체로서 고유의 속성을 가진다.

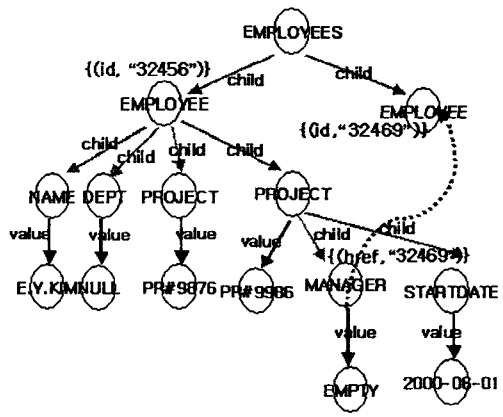
그래프 G의 각 구성 객체 V, E, A, R을 각각 포함하고 있는 속성으로 튜플을 표현하면 다음과 같다.

Vertex Object =(void, label, level)
Edge Object = (from, to, relation, order)
Attribute Object = (node, name, value, type)
Reference Edge Object =(refFrom, refTo, refAttr)

이 튜플 구조는 XML 문서의 데이터 및 구조와 관계없는 유일한 구조를 제공한다. [그림 1]은 XML 문서를 데이터 모델의 그래프로 표현한 것이다.

```
<EMPLOYEES>
 <EMPLOYEE id="32456">
   <NAME> E.Y. KIM </NAME>
        <DEPT></DEPT>
        <PROJECT>PR#9876</PROJECT>
        <PROJECT>PR#9986
          <MANAGER href="32469"/>
     <STARTDATE>2000-06-01
</STARTDATE>
   </PROJECT>
  </EMPLOYEE>
  <EMPLOYEE id="32469">
   ...
  </EMPLOYEE>
</EMPLOYEES>
```

(A)



(B)

[그림 1] (A). XML문서 (B). 데이터 모델의 그래프로 표현한 XML 문서

4. 검색

본 논문에서 제안한 데이터 모델에서 XML 문서에 대한 검색은 XML 문서의 구조 정보와 데이터 정보를 조건으로 탐색할 수 있으며 정확한 데이터 구조에 대한 사전 지식이 없어도 검색이 가능하다.

4.1 데이터 정보에 의한 탐색

특정 데이터를 포함하는 XML 문서에 대한 탐색은 Vertex Object의 *label*을 스캔하며 특정 속성 값에 의한 검색은 Attribute Object의 *name*과 *value*를 스캔한다.

AT 속성의 값이 **VAL**로 정의된 태그이름이 **A**인 Element를 포함하는 XML 문서에 대한 검색은 먼저, Attribute Object의 *name*과 *value* 영역을 스캔하여 해당 노드 정보를 구한 다음 이 정보를 이용하여 매치 되는 Vertex Object를 검색하고 *label*이 **A**인가를 확인한다.

A 태그 이름을 가진 Element 값이 **AVAL**을 만족하는 문서에 대한 검색은 먼저, Vertex Object의 *label*을 스캔하여 **A**인 오브젝트를 탐색한 다음 이 정보를 Edge Object에서 *from*과 매치시키고 *to* 영역을 스캔하여 그 정보를 가지고 다시 Vertex Object의 *label*을 스캔하여 **AVAL**인가를 확인한다.

4.2 구조 정보에 의한 탐색

특정 계층 구조를 가진 Element에 대한 검색은 Vertex Object의 *label*과 Edge Object의 *from*과 *to*를 스캔하며 동일 레벨에서 특정 위치에 있는 Element에 대한 검색은 Edge Object의 *from*과 *to*, 그리고 *order*를 스캔한다. 그리고 특정 Element를 참조하는 Element에 대한 검색은 Reference Edge Object의 *refFrom*과 *refTo*를 탐색한다.

A 태그 이름을 가진 Element가 *k*번째 Child인 **B** 태그 이름을 가진 모든 Element를 검색하는 경우에는 Bottom-up 스캔이 효율적이며, 먼저 Vertex Object의 *label* 부분을 스캔하여 **A**인 Object를 검색한 다음 검색된 Object를 Edge Object의 *to*와 매치 시키고 *from* 데이터를 이용하여 *k*번째 부모 노드를 찾아 나간 다음 탐색된 *from* 데이터를 이용하여 Vertex Object와 매치시켜서 탐색한 객체의 label이 **B**인 가를 확인한다. 이 유형의 탐색 방법은 **B** 태그 이름을 가진 Element가 *k*번째 Parent인 **A** 태그 이름을 가진 모든 Element를 탐색하는 경우에도 적용할 수 있다.

만일 **B** 태그 이름을 가진 Element의 자손인 **A** 태그 이름을 가진 모든 Element를 검색하는 경우나 **B** 태그 이름을 가진 Element가 조상인 **A** 태그 이름을 가진 모든 Element를 검색하는 경우에는 앞의 탐색 작업에서 *k*번째 부모 노드를 찾거나 *k*번째 자식 노드를 찾는 작업 대신 *root* 노드를 만나거나 자식 노드를 찾는 작업을 재귀적으로 호출하는 작업으로 대신한다.

R 태그 이름을 가진 Element가 *k*번째 오른쪽 형제인 **L** 태그 이름을 가진 모든 Element를 검색하는 경우에는 Left-right 스캔으로 접근되며, 먼저 Vertex Object의 *label*을 스캔하여 **L**인 Object를 검색한 다음 검색된 Object를 Edge Object의 *to*와 매치 시키고 *order*값을 저장한 다음 *from* 데이터와 (*저장한 order 값*+ *k*)인 조건을 이용하여 다른 Edge Object를 탐색한다, 그리고 찾은 Object의 *to* 정보를 가지고 Vertex Object와 매치 시켜서 탐색한 객체의 label이 **R**인 가를 확인한다.

5. 결론

본 논문은 인터넷상에서 데이터를 표현하고 데이터를 서로 교환하기 위한 문서 표준으로 제안되고 있는 XML 문서를 위한 새로운 데이터 모델을 제시하였다. 본 논문에서 제안한 데이터 모델은 XML 문서를 위한 새로운 데이터베이스 시스템을 설계하는 경우나 관계형 데이터 시스템과 같은 기존의 데이터베이스 시스템을 이용하는 경우에 있어서 모두 적용 가능한 모델이다.

XML 문서의 새로운 데이터 모델에 대한 기존 연구에서는 XML 문서 종류에 따라서 개별적인 데이터 구조를 정의했다. 데이터 모델을 몇 가지 이내의 XML 문서 유형만을 가지고 있는 도서관, 전자 상거래, 한 기업 또는 한 부서에 적용하는 경우에는 관계없으나 XML 문서 유형의 종류를 장담할 수 없고 도메인을 확대할 경우에는 모든 XML 문서 형태에 따른 데이터 구조가 각각 정의되어야 하는데, 최악의 경우에는 해당하는 유형의 XML 문서가 한 개일지라도 그 문서를 위한 새로운 데이터 구조를 정의해야 하므로 도메인을 확대하기가 힘들고 적용하는 대상에 따라서 데이터 구조를 매번 다르게 정의해야 한다는 문제점이 있다. 본 논문에서 제안하는 데이터 모델은 XML 문서의 데이터 및 구조와 관계없이 유일한 데이터 구조로 표현된다. 데이터 구조를 XML 문서의 형태나 구조로부터 독립시킴으로써 XML 문서의 가장 커다란 특징으로 꼽는 표현의 유연성과 확장성을 어떠한 조건 없이 지원하며 XML 문서 구조가 변경되면 데이터 구조를 변경해야하는 기존의 데이터 모델들의 문제점을 해결하였다. 또한 태그 이름과 같은 XML 문서 구조에 대한 사전 지식이 있어야만 검색이 가능하다는 기존 모델의 문제점도 해결하였다.

또한 기존의 관계형 시스템이나 객체형 시스템으로 매핑하기 위한 데이터 모델에서는 XML 원본 문서의 데이터 뷰와 구조 뷰를 모두 잃어버리게 되어 저장된 데이터로부터 XML 문서를 다시 생성해 낼 수 없을 뿐만 아니라 검색 결과로 Element에 해당하는 XML 문서의 서브 그래프를 반환할 수 없다는 문제점이 있었으나 제안하는 모델에서는 XML 원본 문서의 데이터 뷰와 구조 뷰를 모두 저장하므로 이 문제 역시 해결하였다. 따라서 제안하는 모델에서는 XML 문서의 검색 결과를 가지고 새로운 XML 문서를 생성할 수 있기 때문에 XML 문서의 또 하나의 커다란 특징인 재사용성을 지원한다.

제안하는 모델에서의 XML 문서에 대한 검색은 XML 문서에 포함된 모든 요소를 조건으로 하여 검색이 가능하며 Top-down 탐색뿐만 아니라 Bottom-up 탐색도 가능하다. 또한 동일한 부모를 가진 자식 Element간의 순서 정보도 가지고 있기 때문에 Left-Right 탐색도 가능하며 XML 문서의 Elementrks의 계층 구조뿐만 아니라 동일 레벨의 순서를 그대로 유지한다.

감사의 글

이 논문은 한림대학교의 학술연구지원사업 지원연구비에 의하여 연구되었음.

참고 문헌

[1] Mary Fernanadez, Jerome Simeon, Philip Wadler. "An Algebra for XML Query," In Foundations of Software Technology and

Theoretical Computer Science (FSTTCS 2000), New Delhi, December 2000.

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

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

[4] Fernndez, M., Simon, J., Suciu, C. and Wadler, P. "A Data Model and Algebra for XML Query", Draft Manuscript (1999) Available at http://www.cs.bell-labs.com/~wadler/topics/xml.html#algebra

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

[6] P. Wadler. "A formal semantics of patterns in XSLT," Markup Technologies, December 1999. 16

[7] Gerti Kappel, Elisabeth Kapsammer, Wemer Retschitzegger, "X-Ray-Towards Intergrating XML and Relational Database Systems, Technical Report," July 2000.

[8] 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.

Abstract

New Data Model for Efficient Search and Reusability of XML Documents

Eun-Young Kim* · Se-Hak Chun**

XML has been proposed as a document standard for the representation and exchange of data on the WWW, and also becoming a standard for the search and reuse of scattered documents. When implementing a XML contents management system, special consideration should be imposed on how to model data and how to store the modelled data for effective and efficient management of the semi-structured data. In this paper, we proposed a new data model for the storage and search of XML document data. This proposed data model could represent both of data and structure views of XML documents, and be applied to the new data system for XML documents as well as the existing data systems.

Key words : XML Document Management, Relational Database, Objective Database, Multi-format information retrieval

* Ansan College of Technology
** Hallym University