갱신 효율을 위한 엘리먼트 ID 상속 기반의 XML 저장

강 인 석* · 권 훈* · 김 정 희* · 곽 호 영**

XML Store for Updating-efficiency based on the Element ID Inheritance

In-Suk Kang* Hoon Kwon* Jeong-Hee Kim* Ho-Young Kwak**

ABSTRACT

In this thesis, ID inheritance storage system modelbase on XML document and information of schema element structure has proposed. proposed system parse XML document and read in DOM type then assign ID to element by using analysis module.at this time, ID assigned to child element is inherited from parent element ID and assigned ID will be saved to database as form of defined schema structure proposed for efficient search and updates of document. As results, when specific element has inserted or updated, location information are not required to configured again.

Key Word: XML, SCL, K-ary, EDIT, DOM

Ⅰ. 서 론

빠른 속도로 정보화 사회가 되면서, 컴퓨터를 이용한 문서처리의 중요성이 증가되었다[1]. 한편, 웹 문서의 전자적 처리가 요구되면서, 이를 위한 워드프로세서, 전자출판 시스템과 같은 전자 문서처리 시스템이 현실화되고 있다[2,3].

하지만 이러한 시스템으로 작성된 문서는 각기 독자적 문서 구조와 다양한 내용 정보를 갖고 있으므로, 서로 다른 시스템 및 장치(device)를 갖는 문서 처리환경에서 이들 문서의 교환 및 공유를 위한 표준 문서구조 모델이 필요하게 되었다[2]. 그래서 W3C는 HTML의 편리성과 SGML의 확장성을 결합한 웹 문서 표준인 XML(eXtensible Markup lannguage)을 1996년 제안하였고, 현재까지 기능이 계속 확장되고 있

* 제주대학교 대학원
Graduate School, Cheju Nat,l Univ.
** 제주대학교 통신?컴퓨터공학부, 첨단기술연구소
faculty of Telecommunication & Computer Eng., Res. Insti. Advanced. Tech., Cheju Nat,l Univ

다[1-3]. 새로운 웹 표준이라 할 수 있는 XML 문서에 대한 저장과 검색을 효율적으로 지원하는 관리 시스템의 필요성이 대두되었다[8,9].

이런 연구들을 살펴보면 XML 문서를 관계 또는 객체지향 데이터 모델로 변환하여 저장하는 모델링 연구와 효율적인 검색을 위한 인덱스 및 검색 구조 설계연구로 나누어진다[10].

하지만, 이런 연구들은 XML 문서구조 갱신이 발생하는 경우, 이에 따르는 모든 구조 정보가 수정되어야 한다. 그리고 순차적 탐색 기반으로 리스트 형태의 정보를 데이터베이스에 저장하므로, 검색 시 효율이 떨어지거나 특정 위치에 대한 엘리먼트 탐색 시 문제점이 발생한다[10, 11].

이에 본 연구에서는 기존 연구들의 장점을 최대한 수용하여 동적 환경에 적합한 효율적 문서 검색 및 변경을 위한 엘리먼트 ID 상속 기반의 문서저장 구조를 제안한다.

본 연구의 구성은 먼저, Ⅰ장 서론에서는 XML 문서의 연구 동향에 대해 서술을 하였고, Ⅱ장에서는 XML 문서에 대한 기존 연구방법 중 저장 및 검색에 대해 살펴보겠다. Ⅲ장에서는 본 연구가 제안하는 XML 문서 저장구조를 설명하고 Ⅳ장에서는 모델링 검증 및 구현하며, Ⅴ장에서는 결론 및 향후연구에 대해 언급을 한다.

Ⅱ. 관 련 연 구

2.1 XML 저장 방법

XML 문서의 저장의 형태는 크게 DTD와 같은 스키마 정보가 있는 경우와 그러지 않은 경우로 나누어 진다. 먼저, 스키마 정보를 추출해 내어 관계형 테이블들을 생성하는 방법이 있고, 두 번째 방법은 XML 문서를 구성하는 요소들, 즉 노드와 간선(edge)들만을 저장하는 방법이 있다. 첫째의 경우는 STORED[3]에 해당하고, 둘째는 반구조적인 데이터를 저장하기 위한 방법[7,13]과 간선(edge) 방식이다[5,6].

2.2 XML 문서 색인 방법

구조화된 XML 문서를 검색하기 위해서는 키워드에 의한 문서 단위의 내용 검색뿐만 아니라 엘리먼트를 기본단위로 하는 구조 검색 및 속성에 대한 검색도 지원되어야 한다[8,9]. 이를 위해 구조 정보를 효율적으로 표현하기 위한 연구가 선행되어야 한다. 현재 구조 정보를 표현하기 위한 모델로는 SCL모델, K-ary 완전 트리 모델, 그리고 ETID모델 등이 있다.

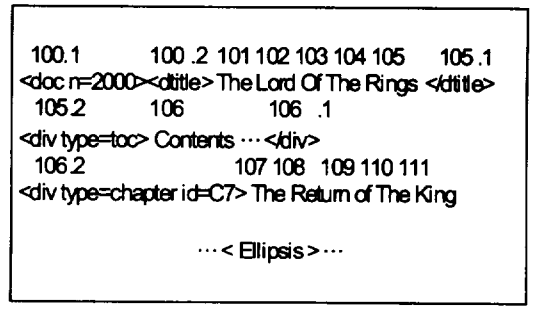

Fig. 1 Simple concordance list model

SCL(Simple Concordance List) 모델은Fig. 1과 같이 구조 문서의 계층적 관계보다는 포함 관계를 이용한 표현 방법이다[9]. 이런 표현 방법은 SC-list라는 데이터 타입을 통해 중첩된 정보를 허용하므로 리스트의 리스트와 같은 순환 구조를 다룰 수 있다는 장점이 있다. 이 모델은

텍스트와 마크업에 대해 색인 넘버를 부여한 후, 불용어를 제외한 텍스트 어휘들을 텍스트 인덱스에 색인 넘버로 저장하고, 마크업은 시작 태그와 종료 태그의 쌍으로 마크업 인덱스에 저장한다. 그러나 SCL 구조는 트리의 깊이를 표현할 수 없으므로 조상이나 형제 엘리먼트를 검색할 수 없다는 단점이 있다[11,12]

K-ary 완전 트리 모델[9]은 문서에 대한 트리로부터 이들 노드 중 가장 큰 차수 K를 구하여 트리로 재구성한 후, 여기에 문서 트리를 매핑하여 각 노드에 모든 번호를 부여한다. 이 모델은 문서 구조 사이의 계층 관계를 간단한 공식을 통해 쉽게 구할 수 있다는 장점이 있지만,

매핑 과정에서 Null 노드가 많아질 수 있고 노드의 깊이가 깊어질수록 노드 변화가 커진다는 단점이 있다[8,10].

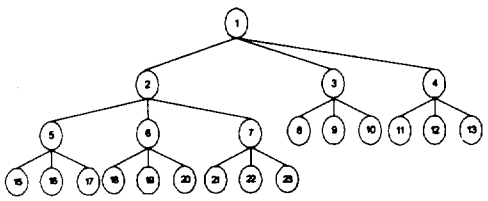

Fig. 2 K-ary tree model

III. XML 문서 저장 구조

3.1 문서 저장 구조

본 연구에서 가상 분할 저장이 가지고 있는 문서에 대한 효율적 검색과 분할 저장이 가지고 있는 문서 변경 시 변경된 부분만 동적으로 변화시킬 수 있는 장점을 이용하여 저장구조 시스템 Fig. 3을 설계하였다.

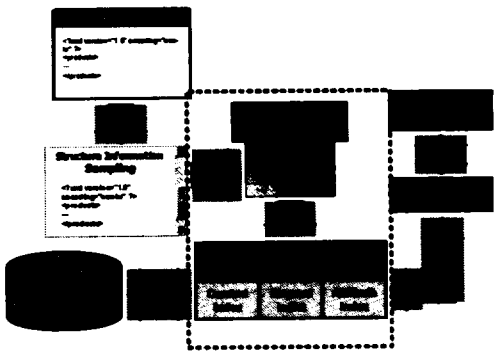

Fig. 3 Stored structure system

Fig. 4는 정형화된 XML 문서이다. XML문서는 내용정보와 구조정보를 동시에 포함하고 있고, 문서의 구성은 세 부분으로 나누어진다. 첫 번째 부분은 선언부, 두 번째 부분은 스키마와 네임스페이스를 선언하지만, 이 부분은 생략이 가능하다. 그리고 마지막 부분은 몸체를 이루는 태그와 컨텐츠로 구성된다. Fig. 4는 XML 문서를 크게 선언부와 몸체로 나누고, 몸체를 엘리먼트로 나눈다. 이렇게 분리된 정보들은 DOM으로 파싱되어 계층 구조의 형태 Fig. 5와 같이 표현된다.

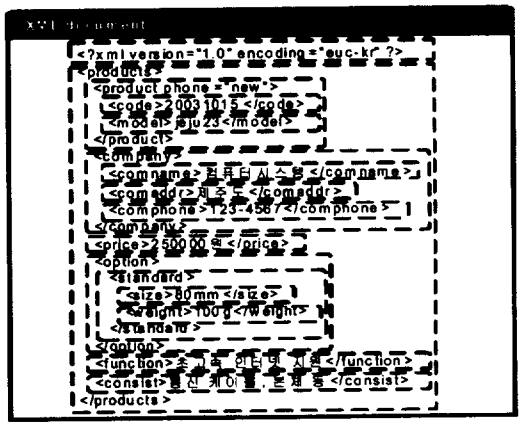

Fig. 4 XML document

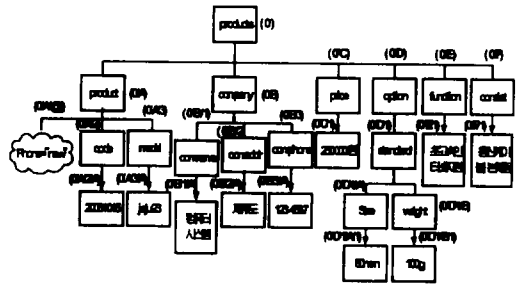

Fig. 5 Class structure

3.2 ID 부여 방식

본 연구에서는 문서 구조정보를 표현하기 위해 SCL 모델의 단점인 문서의 단계를 표현하지 못하는 단점과 K-ary 모델의 단점인 노드의 깊이가 길어질수록 노드의 변화가 커진다는 단점을 보완한 ID 부여 방식을 제안하였다.

1) 엘리먼트 ID부여 방식

엘리먼트 ID 부여 방식은 루트 엘리먼트의 값을 '0'으로 시작하여, 1 단계(Depth)는 영문으로, 2 단계는 숫자, 3 단계는 다시 영문을 부여하는 반복 패턴방식을 사용하였다. 반복 패턴방식을 사용하여 엘리먼트의 값을 가지고 엘리먼트의 단계를 알 수가 있으므로 별도로 단계를 저장하기 않고도 엘리먼트의 단계를 알 수가 있다. 자식 엘리먼트 ID 값은 부모 엘리먼트 ID 값과 자신의 위치 정보 값을 조합하여 생성한다.

2) 속성 ID부여

속성 ID 값은 속성과 속성의 값으로 나누어 저장은 한다. 속성 ID의 부여는 해당하는 엘리먼트 ID 값을 자신의 위치 정보와 조합하여 생성하고, 속성 ID에 '@'을 조합하여 속성값을 생성한다.

Table 1 Before information table

단계	PID	CID	NodeID	Element	Attribute	Value
0		A/B/C/D/E	0	Time-o-gram		
1	0	@1	0/A		pri	Impot
1	0	1	0/B	to		
1	0	1	0/C	subject		
1	0	1/2/3/4/5	0/D	message		
1	0	1	0/E	from		
2	0/B		0/B/1			Seoh
2	0/C		0/C/1			Reminder
2	0/D		0/D/1			Don't ···
2	0/E		0/E/1			The Doctor
···	···	···	···	···	···	···

3) 사전 정보 테이블

Table 1은 본 연구에서 제안한 ID 부여 방식에 의해 얻어진 결과를 데이터베이스에 저장한 결과이다. 예를 들면, 'subject' 엘리먼트는 부모 노드의 ID 정보와 자신의 ID 정보를 결합한 '0/C'로 나타낸다.

3.3 정의된 테이블 스키마 구조

XML 문서를 분석한 후 생성된 사전 정보 테이블정보는 파일의 형태를 저장하는 테이블(FT), 문서를 저장하는 테이블(DT), 엘리먼트의 목록을 저장하는 테이블(ELT), 엘리먼트의 위치를 저장하는 테이블(NT), 엘리먼트를 저장하는 엘리먼트 테이블(ET), 속성을 저장하는 테이블(AT), 문서의 내용을 저장하는 테이블(CT)로 이동되며 이를 저장하기 위한 정의된 테이블 스키마 구조는 다음과 같다.

1) File_Table(FT)

DocID	FileName	Description

2) Document_Table(DT)

DocID	Value	PNID	CNImfor

3) Element_List_Table(ELT)

DocID	Element	PathID

4) Node_Table(NT)

DocID	Position	PathID

5) Element_Table(ET)

PathID	PNId	CNId

6) Attribute_Table(AT)

DocID	PathID	Attribute	Value	PNId	CNId

7) Content_Table(CT)

DocID	PathID	Value	PNid	CNid

Ⅳ. 모델링 결과 및 검증

4.1 XML 문서 저장 결과

저장될 XML 문서(Fig. 1)의 파싱한 형태(Fig. 2)와 이를 이용한 사전 정보 테이블, 그리고 최종적으로 생성된 정보를 정의된 테이블 스키마 구조에 저장한 형태는 다음과 같다.

1) File_Table

DocID	FileName	Description
1	Letter.xml	Description

2) Document_Table

DocID	Position	PathID
1	0	1
1	0/A	2
1	0/B	3
···	···	···

3) Element_List_Table

DocID	Element	PathID
1	time···	1
1	pi=···	3
1	to	4
···	···	···

4) Node_Table

DocID	Position	PathID
1	0	1
1	0/A	2
1	0/B	3
···	···	···

5) Element_Table

PathID	PNID	CNID
1	0	A/B/C/D/E
2	0/B	1
3	0/C	1
···	--	···

6) Attribute_Table

DocID	PathID	Attribute	Value	PNId	CNId
1		pi	imp···	PNId	CNId

7) Content_Table

DocID	PathID	Value	PNID	CNID
1	7	Ser···	0/B	A/B/C/D/E
1	8	Rem···	0/C	1
1	9	The···	0/E	1
···	···	···	···	···

4.2 검색 처리

질의문 : /time-o-gram//to

위의 질의 의미는 "time-o-gram 엘리먼트 아래의 to 엘리먼트 값을 찾아라."라는 의미이다. 질의문을 처리하기 위해, 먼저 ELT에서 to 엘리먼트의 Path_ID의 값을 알아보고 ET에서 P_N_id의 값 "0/B" C_N_id의 값 "1"은 자식 엘리먼트가 하나가 존재한다는 의미이다. 그리고 값을 결합을 하면 "0/B/1" 값이 되어 CT에 있는 테이블을 참조 하면서 Path_ID가 7인 값인 "Sarah"를 찾을 수가 있다.

4.3 갱신 처리

질의문 : /time-o-gram//to, Sarah→Tom

위의 질의 의미는 "time-o-gram 엘리먼트 아래의 to 엘리먼트 값을 Sarah→Tom 갱신하라" 라는 의미이다. 검색 처리에서 했던 방법과 동일한 처리로 CT에 있는 테이블을 참조한다. Path_ID가 7인 값을 갱신하고 NT 테이블에서 Position 값 "0/B/1"값을 찾아서 갱신을 하면서 마친다.

4.4 삭제 처리

질의문 : /time-o-gram//to, Sarah→√

위의 질의 의미는 "time-o-gram 엘리먼트 아래의 to 엘리먼트 값을 Sarah→√ 삭제하라"라는 의미이다. 검색 처리와 동일한 방법으로 Path_ID 값이 찾아서, NT, ET, CT 테이블의 Path_ID가 7인 값을 삭제하고, P_N_id와 C_N_id값을 조합하여 DT 테이블에 있는 P_N_id의 값와 동일 값을 찾아서 삭제를 하면서 마친다.

4.5 속성값과 자식원소 출력

Fig. 6 은 속성값과 자식 원소를 출력하기 위한 XML 문서에 대한 노드에 타입을 얻고, 문자열을 출력한다.

```
public void print(Node node) {
  int type = node.getNodeType();
    switch(type) {
        case Node.DOCUMENT_NODE:
        System.out.println("DOC: "+node.getNodeName());
        Document d = (Document)node;
        print(d.getDocumentElement());
    break;
    case Node.TEXT_NODE:
      if ( n == 0) {
        System.out.println("TEXT: "+node.getNodeValue());
        }
    break;
    case Node.ELEMENT_NODE:
        System.out.println("ELEMENT: "+node.getNodeName());
        // 속성 값 출력하기
        NamedNodeMap attrs = node.getAttributes();
          int len = attrs.getLength();
          for(int i = 0; i < len; i++) {
            print (attrs.item(i));
          }

        // 자식 원소 출력하기
        NodeList children = node.getChildNodes();
          if (children != null) {
                int n = children.getLength();
                for(int j = 0 ; j < n ; j++) {
                        print(children.item(j));
                }
          }
    break;
```

Fig. 6 Attribute and child element

V. 결론 및 향후 연구

본 연구에서는 XML 문서를 효율적으로 검색하고 변경을 지원하는 저장 구조를 제안하였다. 이를 위하여 저장될 XML 문서를 분석하여 제안된 7개의 테이블 스키마에 저장하였으며, 또한 사전 테이블을 생성하여 엘리먼트의 부모-자식간의 상속 ID를 부여토록 하였다. 그 결과 엘리먼트, 속성, 구조, 내용 색인 및 추가 및 갱신이 가능함을 보였으며, 특히 엘리먼트 추가나 삭제시 위치 정보 변경 발생을 피하기 위하여 제안된 부모-자식간의 엘리먼트의 ID 상속 값을 사용하

여 위치 정보 변경 없이 갱신이 가능함을 보였다.

향후 연구 과제로는 Path_ID를 가변적으로 부여함으로 인해 기존에 부여되었던 ID는 재사용이 불가능하다. 따라서, 이를 재사용 측면을 고려한 유동적인 부여 방식에 대한 추가 연구가 필요하다고 본다.

참 고 문 헌

1) 정희경 외 2인 공저, SGML 가이드, 사이버출판사, 1997

2) 정희경, WWW 문서 작성을 위한 자세대 언어 XML 가이드, 그린, 1998

4) A. Deutsch, M. Fernandez, D. Suciu, Storing semistructured data with STORED, SIGMOD 99, pp 431-442, 1999

5) Danielo Florescu, Donald Kossmann, Storing and Querying XML Data Using an RDBMS, Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, Vol .22, pp 27-34, 1999

6) J. McHugh, S. Abiteboul, R. Goldman, D. Quass, and J. Widom, Lore : A database management system for semistructured data, ACM SIGMOD Record, Vol. 26, No. 3, pp 54-66, 1997

7) V. Christophides. et al, From Structured Documents to Novel Query Facilities, ACM SIGMOD, pp. 313-324, Minesota, USA 1998

8) 이종설 외 7, 구조 정보 검색을 위한 XML 저장관리시스템 설계 및 구현, Journal of the Research Institute for Computer and Information Communi-cation Vol. 7, No. 2, 1999. 11

9) Toung Dao An Indexing Model for Structured Documents to Support Queries on Content, Structure and Attributes, Proceedings of ADL'98, pp. 88-97, 1998

10) T Dao, R. Sacks-Davis and J. A. Thom An indexing scheme for structured documents and its implementation, In Proceedings of the 5th International Conference on Databases Systems for Advanced Applications, pp.125-134, Melbourne, Australia, April. 1997

11) C. Kanne, G. Moerkotte, Efficient Storage of XML data, ICDE 00, p. 198, 2000