韓國 컴퓨터情報學會 論文誌
第8卷 第2號, 2003. 6.

2003-8-2-1-3

ODMG 객체 모델 기반의 XML 문서 저장 관리 시스템에 관한 연구

박 준 범*, 박 경 우**, 오 수 열**

A Study on XML Document Repository Management System using ODMG Object Model

Joon-Bum Park*, Kyung-Woo Park**, Soo-Lyul Oh***

요 약

XML 문서의 저장 및 관리를 위해, 기존의 관계형 DBMS를 이용해왔다. 그러나 관계형 DBMS를 이용할 경우, XML 문서 구조를 관계형 데이터베이스로 매핑할 때, 정보 손실이 발생할 수 있으며, XML의 특성을 완벽히 반영하는 새로운 저장 관리 시스템을 새로이 구현하는 것은 많은 비용이 소요된다. 본 논문에서는 이러한 문제점을 극복하고, XML 문서의 구조적 특성을 충분히 반영하기 위해, 기존의 객체지향형 DBMS인 O2 DBMS를 기본 저장 구조로 이용하면서, XML 문서가 가지고 있는 다양한 특징을 수용할 수 있도록 하는 XML 문서 저장 관리 시스템을 설계하고 구현하였다. 구현한 XML 문서 관리 시스템 기능들로서는 전자 문서에 대한 라이브러리 서비스 기능(체크인/체크아웃, 버전제어, 사용자 액세스 관리 등), 동적 색인 및 구조 정보 검색 기능, 스타일시트를 이용한 출판 기능을 하는 XML 문서 저장 관리 시스템이다.

Abstract

In most organizations, the relational DBMS was used for store and manage XML documents due to the pre-established relational DBMS. But, relational DBMS has some problems. They has the possibilities of informational loses in the process of mapping the structure of XML documents to RDB, and they require expensive cost to reflect all XML properties. Thus, this paper was intended to design and implement XML document management system which utilize O2 DBMS-object oriented DBMS in order to overcome the existing problem and reflect all XML properties. The XML document management system purposed in this thesis has multiple function, such as the library service function for XML documents-check-in/check - out, versioning, user access management-, dynamic indexing and retrieval, and publishing function using style sheet.

* 동아인재대학 인터넷정보계열 교수, ** 목포대학교 정보공학부 컴퓨터공학전공 교수

Ⅰ. 서론

인터넷의 발달과 더불어 대량의 문서 유통이 가능해지고, 이들 문서 사이의 표준화된 표현 형태로 XML (eXtensible Markup Language) 문서 구조[1]에 대한 관심이 크게 증가하였다. 그러나, 이들 문서를 RDBMS (Relational Database Management System)나 ODBMS(Object-oriented Database Management System)와 같은 상용 데이터베이스를 활용하여 저장하기 위해서는 XML의 문서 구조를 엄격한 스키마를 가지는 기존 데이터베이스를 매핑하는 과정에서 정보의 손실이 발생할 수 있다.

본 논문에서는 여러 분야에서 활용되고 있는 XML 문서를 효율적으로 저장하고 관리할 수 있으며, 여러 사용자 환경에서 문서를 공유할 수 있는 ODMG(Object Database Management Group) 객체 모델을 이용한 XML 문서 저장 관리 시스템에 대해 기술한다.

DBMS 기반 시스템 중에서 RDBMS를 이용하는 방법은 몇 가지 문제점이 있다. 문서 저장 시 엘리먼트들 간의 관계를 유지하기 위해 많은 테이블과 추가적인 매핑코드들이 필요하며, 검색 시 고비용의 조인 연산을 수행함으로써 검색 결과를 추출하는데 많은 노력이 필요하다[2]. 가변길이 데이터 타입에 대한 지원이 제한적이므로 가변길이 XML 데이터를 저장할 때 정보 손실이 발생할 수 있으며, 관계형 모델은 특정 엘리먼트에만 락(Lock)을 수행할 때 공동저작 기능을 지원하기 위해 문서 전체나 일부분을 보호하는 락 기능이 미흡하다.

웹 기반 XML 문서 저장 관리 시스템은 클라이언트-서버(Client-Server)의 구조를 가져야한다. 그러나 클라이언트-서버의 전통적인 "one fork per client request" 방식은 클라이언트의 요청이 있을 때마다 자식 프로세스를 fork한다. 이 방식의 문제점은 웹 서버와 같이 하루에 수만개의 연결을 관리하는 환경에서는 자식 프로세스를 fork하는 비용 때문에 병목 현상이 발생된다.

본 논문에서는 "pre-fork with parent passing socket descriptor to child" 방식을 적용함으로써 이와 같은 병목 현상을 해결하였다.

본 논문에서는 ODMG 데이터 모델 기반의 XML 문서 저장 관리 시스템을 구현하는데 필요한 여러 가지 문제점과 해결 방안 및 시스템의 설계와 구현에 대해 기술한다.

Ⅱ. 관련연구

1. XML 문서 저장 방법

XML 문서의 구조 정보를 데이터베이스에 저장하는 방법은 분할 저장(Decomposition) 기법과 가상 분할 (Virtual fragmentation) 기법으로 나눌 수 있다[3, 4].

분할 저장 기법은 XML 문서를 엘리먼트 단위로 쪼개어 저장하고, 검색시에는 구조 정보를 참조하여 해당 엘리먼트나 하위 엘리먼트의 조합을 생성하여 처리하는 방법이다. 이 기법은 문서를 DTD의 엘리먼트 단위로 나누어 저장하기 때문에 검색 과정이 비교적 복잡하고 검색 시간이 많이 걸린다.

가상 분할 기법은 XML 문서 전체를 하나의 BLOB (Binary Large OBject) 형태로 DB에 저장하고 문서내의 단말 노드에 대해 실제 문서상에서 가지는 시작 offset과 종료 offset을 저장하는 방법이다. 이 기법은 문서를 한꺼번에 저장하기 때문에 단말노드에 대한 검색에서 위치 정보를 이용한 구조적 검색이 가능하고 검색 효율이 우수하다. 그러나, XML의 특정 엘리먼트에 대해서 추가/삭제 등의 연산에서 문서 내의 다른 엘리먼트들에 대한 위치 정보를 모두 수정해야 한다. 따라서 이 기법은 주로 검색을 많이 하는 시스템에 적합한 방법이다[5].

본 논문에서 구현한 XML 문서 저장 관리 시스템은 가상 분할 기법을 이용하여 문서를 저장한다.

2. XML 데이터 모델

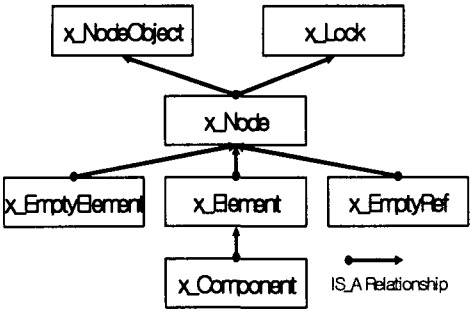

그림 1. XML 트리 모델의 클래스 구조
Fig. 1. Class Architecture of XML tree model

DTD 의존 모델의 단점을 극복하기 위해 하나의 스키마로 여러 DTD들을 수용할 수 있는 일반화 데이터 모델이 필요하다. 다음 〈그림〉은 일반화 데이터 모델의 XML 트리 모델의 클래스 구조를 나타낸다.

x_NodeObject 클래스는 XML 트리 모델의 슈퍼 클래스이며, 노드 유형에 대한 정보를 보관하는 클래스이다. XML 파스 트리의 위치 정보와 노드 이름/데이터에 대한 정보를 관리하는 클래스이다. XML의 PCDATA, PI(Processing Instruction), Comment 등은 파스 트리의 단말 노드에 위치하며, 별도의 클래스로 정의하지 않고 x_Node 클래스에 값을 설정한다. XML ELEMENT 노드는 x_Element 클래스에 값이 설정된다. x_Element 클래스는 파스 트리의 비단말 노드이므로 자식을 가질 수 있다. 또한, 속성 인스턴스를 리스트로 가질 수 있다. XML EMPTY ELEMENT는 자식 노드를 가지지 않고 속성 인스턴스만을 가지며 트리의 단말 노드에 위치한다. x_EmptyElement 클래스는 EMPTY ELEMENT에 대한 정보를 보관하는 클래스이다. x_EntityRef 클래스는 엔티티 참조에 대한 정보를 보관한다. 엔티티 이름은 x_Node 클래스에 설정되며, 엔티티 유형에 대한 정보를 저장한다. x_Component 클래스는 x_Element로부터 상속받아 정의되며, 별도의 속성을 갖지 않는다. 이 클래스는 XML 파스 트리에서 자식 노드들을 별도로 저장하지 않고 PCDATA로 합쳐서 저장하는데 사용된다.

Ⅲ. 시스템 설계 및 구현

1. XML 문서 저장 관리 서버 시스템

문서 저장 관리 시스템에서 소켓을 통한 클라이언트의 요청은 부모 프로세스가 관리한다. 부모 프로세스는 시스템 초기 구동시에 환경설정 파일에 기재된 클라이언트의 개수만큼 자식 프로세스들을 생성한다. 자식 프로세스를 관리하기 위해 시스템에서는 아래와 같이 Child Structure 구조체를 사용하여 자식 프로세스 풀(pool)을 생성한다.

```
// Child structure is used to keep a pre-forked
child process' information
typedef struct {
  pid_t    child_pid;    // process ID
  int      child_pipefd; // parent's stream pipe
to/from child
  int      child_status; // 0 = ready
  long     child_count;  // #connections handled
} Child;
```

구현한 문서 저장 관리 시스템은 효율적인 다수의 사용자 처리를 위해 TCP Preforked 서버로 구현한다. 또한 STEER와의 통신은 파이프를 통해 이루어진다. 다음 〈그림 2〉는 서버 시스템의 전체적인 구조를 나타내며, STEER와의 통신은 STEER에서 제공되는 bridge application을 통해 IPC(Inter Processor Channel) 설비인 파이프를 통해 데이터를 주고 받는다.

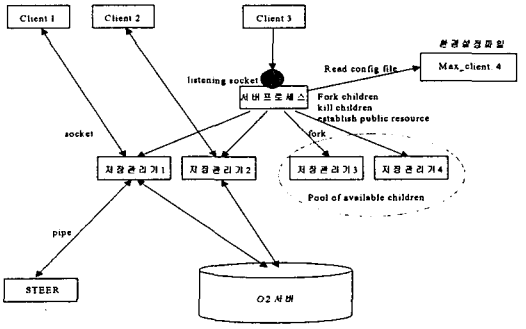

그림 2. 서버 시스템 구조
Fig. 2. Server system architecture

클라이언트-서버의 전통적인 one fork per client request 방식은 클라이언트의 요청이 있을 때마다 자식 프로세스를 fork한다. 이 방식의 문제점은 웹 서버와 같이 하루에 수 만개의 연결을 관리하는 환경에서는 자식 프로세스를 fork하는 비용 때문에 병목 현상이 발생된다. 이와 같은 병목 현상은 본 시스템에서 사용하는 prefork with parent passing socket descriptor to child 방식으로 해결 할 수 있다.

2. XML 문서 저장 관리 시스템 구조

본 논문의 XML 문서 저장 관리 시스템은 Solaris 2.5 환경에서 SPARCompiler C++4.1과 객체지향형 DBMS인 O2 ODBMS 5.0을 사용하여 구현하였으며, XML 문서 저장 관리 시스템의 구조는 〈그림 3〉과 같다.

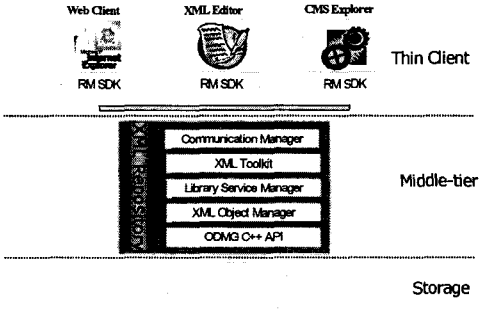

그림 3. XML 문서 저장 관리 시스템 구조
Fig. 3. XML document repository management system architecture

XML 문서 저장 시스템의 하부 저장시스템으로는 O2 ODBMS을 사용하였으며, TCP/IP를 이용한 연결 지향형 클라이언트-서버 구조로 구성하였다. O2는 페이지 서버 아키텍쳐를 갖는 객체 지향 데이터베이스 시스템으로 ODMG에 따른 C++ 바인딩과 OQL(Object Query Language)를 지원하며 상속과 다형성 등의 객체 지향 개념 등을 제공한다. 〈그림 3〉은 효율적인 다수의 사용자 지원을 위한 3-tier(Thin Client, Middle-Tier, Storage) 아키텍쳐 형태의 XML 문서 저장 관리 시스템 구조를 나타낸다.

XML 문서 저장 관리 시스템은 다음과 같은 구성요소를 가지고 있다.

(1) 통신 관리자(Communication Manager)

사용자와 시스템간의 연결을 제공하고 시스템 내부의 메시지 및 작업을 제어한다.

(2) XML 툴킷(XML Toolkit)

XML 파서의 출력 포맷은 ESIS(Element Structure Information Set) 표현으로 된 간단한 텍스트이다. 하지만 어떤 XML 표현은 완전하게 제공하고 있지 못하기 때문에 James Clark의 SP에 기반한 인-메모리 트리 구조를 제공하는 인텔리전트 XML 툴킷을 만들어 사용하였다. DTD의 파싱 결과인 룰 트리 뿐만 아니라, XML 인스턴스의 파싱 결과인 인스턴스도 만든다.

(3) 라이브러리 서비스 관리자(Library Service Manager)

구조화된 XML 문서를 XML 툴킷을 사용하여 DB 저장소에 변환 후 저장한다.

(4) XML/SGML 객체 관리자(XML/SGML Object Manager)

저장소 객체 관리자는 라이브러리 서비스 관리자에 의해 저장소 객체를 관리한다. 다 사용자 환경에서 문서 일부분이나 전체를 하나로 통합, 편집하기 위한 체크인/체크아웃 기능을 제공하고 문서의 히스토리 관리를 위한 버전 관리 기능, 사용자의 권한에 따른 액세스 제어 등을 제공한다.

3. 사용자 정의 Granularity 지원을 위한 컴포넌트 구성

사용자 정의 컴포넌트는 엘리먼트가 노드로 구성된 트리 형태의 XML 문서의 구조를 사용자가 지정한 논리적 단위로 그룹핑한 노드이다. 기존 XML 문서를 저장 관리 시스템에 트리 형태로 저장하거나 복원하는 경우에는 파스 트리 형태로 매핑하여 저장하기 때문에 삽입 시간과 복원 시간이 증가하는 단점이 있다. 그러므로 사용자가 기존 문서의 트리를 단순화시켜 축소한 형태로 시스템이 저장하고 복원하는 경우에 저장 및 복원 시간이 감소하고, 사용자에게 의미 있는 축소된 트리 형태로 뷰를 제공할 수 있는 장점이 있다.

사용자 정의 컴포넌트를 지원하기 위해서는 XML DTD의 트리 구조를 그룹 핑 시키는 명세 파일(Specification file)을 작성해야 한다. 이 경우 기존 구조화된 문서의 트리를 그룹 핑 시키는 위해 다음의 방법들을 고려할 수 있다.

(1) 그룹핑 방법 1

그룹핑해야 하는 트리의 context를 지정하지 않고 엘리먼트의 이름만 명시한다.

(2) 그룹핑 방법 2

그룹핑해야 하는 트리의 context를 지정하여 엘리먼트의 이름 명시한다.

(3) 그룹핑 방법 3

필요에 따라 트리의 context를 명시하거나 또는 명시하지 않고, 그룹핑 해야하는 엘리먼트의 이름만 명시한다.

DTD의 TO, FROM, BODY, P, CLOSE에 대한 그룹핑 구성 문서는 다음과 같이 작성될 수 있다.

```
<?xml version="1.0">
<!DOCTYPE cspec SYSTEM cspec.dtd>
<cspec>
<apply-doctype name=MEMO/>
<component match=TO/>
<component match=FROM/>
<component match=BODY/>
<component match=BODY/P/>
<component match=CLOSE/>
<cspec>
```

위와 같이 구성 파일이 작성되는 경우, P 엘리먼트의 하위트리는 트리상의 노드로 저장되지 않고, 텍스트 노드로 합쳐져서 저장된다. 합쳐진 엘리먼트가 애트리뷰트를 가지고 있는 경우에도 텍스트로 저장된다. 그러므로 P 엘리먼트의 자식 노드들 중에 애트리뷰트, 엔티티 참조 등이 포함되어 있는 경우, 그룹핑하여 저장하는 방법은 파스 트리 형태로 저장하는 방법에 비해 다음의 정보들은 텍스트로 저장된다.

- 그룹핑된 하향성(Descendent)의 트리 구조
- 그룹핑된 하향성의 엘리먼트 노드에 선언된 애트리뷰트
- 그룹핑된 하향성의 엔티티 참조

그러나 컴포넌트 저장 방식에서 중첩 관계는 무시된다. 즉, BODY 엘리먼트의 부엘리먼트인 P를 컴포넌트 엘리먼트로 지정했을 경우, BODY 엘리먼트의 자식노드들은 텍스트로 저장되기 때문에 P 엘리먼트는 컴포넌트 노드로 저장되지 않는다. 아래의 <그림 4>는 DTD와 규칙 파일에 따라 작성된 문서가 컴포넌트 방식으로 저장된 후의 트리 구조를 보여주고 있다.

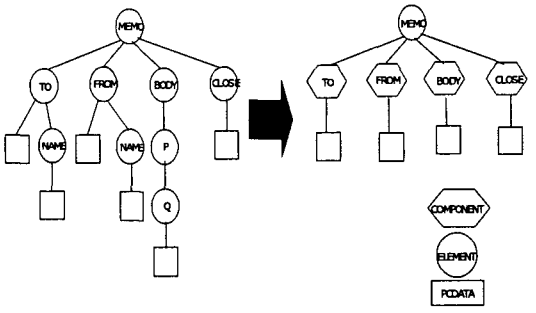

그림 4. 컴포넌트 방식의 트리 구조
Fig. 4. Tree of component mathod

4. XML 문서 저장 알고리즘

XML 파일은 다음의 단계를 거쳐 저장소에 저장된다. 클라이언트의 입력 argument는 저장될 폴더의 객체 식별자, 파일 이름, import type, indexing mode, 주석 등이다. import type은 저장 방식을 결정하는 변수로서, full pared tree 형태로 저장되는 ODM_IMPORT_XML, 사용자 정의 컴포넌트 트리 형태로 저장되는 ODM_IMPORT_XML_COMPONENT, BLOB 형태로 저장되는 ODM_IMPORT_BLOB이 있다. 사용자 정의 컴포넌트 트리 형태의 저장 방식은 DTD를 가진 Valid XML 파일만 가능하며, DTD에 트리를 그룹핑하는 규칙 파일은 메타 필터가 지정되어 있어야 한다.

```
// file import type
typedef enum {
        ODM_IMPORT_XML,
        ODM_IMPORT_XML_COMPONENT,
        ODM_IMPORT_BLOB
} ODM_IMPORT_TYPE;
```

인덱싱 모드(Indexing mode)는 XML 파일에 대해서만 지정 가능하며, 저장되는 XML 파일이 색인될 지의 여부를 지정한다. 색인을 수행하기 위해서는 ODM_INDEX_OK를 지정해야 하며, DTD를 가진 Valid XML 파일이고 DTD에 메타 필터가 지정된 경우에 한해서만 필드 단위 색인을 수행할 수 있다. Well-formed XML 파일과 DTD에 메타 필터가 지정되지 않은 XML 파일은 문서 전체에 대해서 색인을 수행한다.

```
// file indexing mode
typedef enum {
  ODM_INDEX_NO,  // 인덱싱을 수행함.
  ODM_INDEX_OK  // 인덱싱을 수행하지 않음.
} ODM_INDEX_MODE;
```

위의 입력 파라미터를 클라이언트로부터 전송 받은 후, 서버는 XML 파일의 Valid 또는 Well-formed 파일인지의 여부를 검사한다. 검사하는 방법은 XML 파일에 "DOCTYPE" 선언이 있으면 Valid XML로 간주한다. 사용자는 파일이 저장되는 폴더에 대해 쓰기 권한을 갖고 있어야 하며, 폴더 내에 동일한 이름을 갖는 파일이 없어야 한다.

두번째 단계에서는 XML 파일을 파싱하여 메모리 자료구조를 파서로부터 반환받는다.

세번째 단계에서는 x_SgmlFile 객체를 생성하여 사용자/그룹, 생성일, 버전 등의 속성 정보를 설정한 후, 재귀적으로 파스 트리를 순회하며 지속성 객체 트리를 생성한다. 이때 컴포넌트 트리 방식으로 저장하도록 마킹 된 노드는 자식 노드들을 텍스트로 합쳐서 저장한다.

본 논문에서 구현한 XML 문서 저장 관리 시스템은 구조적인 문서를 모델링 하는데 적합한 객체지향형 데이터베이스인 O2 ODBMS를 기반으로 시스템을 구성하였다. 그리고 ODMG 객체 모델에 따른 XML 일반화 데이터 모델을 사용함으로써 DTD에 의존하지 않고 다양한 DTD를 수용하여 모든 XML 문서를 저장한다. XML 문서를 트리 형태로 저장 및 복원할 때 사용자 정의 컴포넌트 트리를 구성할 수 있도록 함으로써, 사용자가 기존 문서의 트리를 단순화시켜 축소한 형태로 시스템이 저장하고 복원하는 경우에 저장 및 복원 시간이 감소하고, 의미 있는 축소된 트리 형태로 뷰를 제공한다. 구조화된 XML 문서뿐만 아니라 다양한 형태의 문서 포맷도 저장, 관리하고 저장소에 트리 형태로 저장된 문서에 대해 다수의 사용자가 동시에 편집할 수 있는 체크인/체크아웃, 버전 제어 기능을 제공한다.

Ⅳ. 실험 및 평가

본 논문에서 구현한 시스템은 논문지에 수록된 논문 280편을 대상으로 실험하였다. 논문의 크기 및 구성 특성은 다음과 같다.

표 1. 논문 테스트 셋 규격
Table 1. Standard of paper test set

구분	종합	평균
XML파일 개수	280 개	1 개
External Entity 파일 개수	4,108 개	14.67 개
XML+Entity 파일개수	4,338 개	15.67 개
XML 파일크기	13,278,473 bytes	47,423.1 bytes
External Entity 파일크기	33,518,258 bytes	119,708 bytes
XML+Entity 파일크기	46,796,731 bytes	167,131 bytes

표 2. 논문 테스트 셋을 파싱한 결과
Table 2. Paper test set parsing a result

규격	종합	평균
XML 노드 개수	350,430 개	1251.54 개
XML Attribute 개수	79,903 개	285.368 개
XML 문서에 포함된 노드+Attribute 개수	430,333 개	1596.908 개

실험에서 사용된 시스템은 SUN Enterprise 3000으로 CPU 168MHz에, 운영체제는 Solaris 2.5.1를 사용하였으며, 로컬 호스트에서 batch로 실험을 수행하였다. 실험 수행결과는 다음과 같다.

표 3. 저장 시간
Table 3. Store time

(단위 : 초)

구분	real time	user time	system time
논문	4.66	2.7756	0.2389

저장시간은 DB의 open, close 시간과 문서 파싱, XML 문서의 파스 트리와 애트리뷰트를 저장하는 시간과 외부 엔티티 파일들을 저장하는 시간, 파서 초기화 시간 등을 합한 것이다. 논문 테스트 셋의 real time를 기준으로 한 1개 노드에 대한 저장 시간은 0.0037초(4.66초/1251개 = 0.00372초)였다.

XML 문서 저장 관리 시스템의 성능 평가를 위해, [6]과 [7]에서 구현한 시스템과 비교하였다. [6]과 [7]에서는 셰익스피어 희곡 37편과 증권정보, DBLP database를 사용하였다. 아래 표 4는 [6]에서 사용한 DTD에 따른 저장시간과 본 시스템을 비교한 것으로, 본 논문의 시스템은 DTD에 독립적인 모델이다.

표 4. DTD에 따른 저장 시간과 비교
Table 4. Compare to store time on DTD

(단위 : 초)

문서량	DTD독립[6]	DTD의존[6]	본 시스템
100K	8.64	10.56	7.97
200K	21.32	26.12	17.69
300K	31.92	39.02	27.86

Edge 저장 기법을 사용한 RDBMS를 사용한 [7] 및 오디세우스 ORDBMS를 이용한 [6]과의 O2 ODBMS를 사용한 본 시스템과 비교 결과는 다음과 같다.

표 5. 저장 모델 및 저장 방법에 따른 시간 비교
Table 5. Compare to time on repository model and method

문서량	[7]	[6]	본 시스템
420KB	82sec	51.23sec	48.98

V. 결론

본 논문에서는 제안된 XML 데이터 모델을 이용한 XML 문서 저장 및 관리 시스템의 설계 및 구현에 대해 기술하였다.

XML 문서의 저장 및 색인과 검색과 같은 기본적인 기능 이외에도 워드 및 HTML과 같은 다양한 형태의 문서 포맷도 저장하고 관리할 수 있도록 하였으며, 다양한 DTD를 수용할 수 있고, 특정한 단위로 저장되어 있는 구조화된 문서를 동적으로 저장소로부터 읽어서 원형의 문서로 복원할 수 있는 기능도 제공한다.

실험을 통해, 기존의 저장 방법에 비해 대량의 문서를 정확하고 빠르게 저장할 수 있음을 보였다. 또한, 키워드에 의한 검색뿐만 아니라 구조 및 속성에 기반한 XML 문서의 검색이 가능하도록 함으로써 사용자의 다양한 질의를 수용할 수 있도록 하였다.

참고문헌

[1] W3C XML Working Group, Extensible Markup Language(XML), http://www.w3c.org/XML

[2] J. Shanmugasundaram, K. Tufte, C. Zhang, G. he, D. J. DeWitt and J. F. Naughton, "Relational Databases for Querying XML Documents:Limitation and Opportunities", Proceedings of 25th International Conference on VLDB, pp.302-314, 1999.

[3] P. Francois, Generalized SGML Repositories : Requirements and Modeling, Computer Standard & Interface, 18 : 11-24, 1996.

[4] 김훈, 한상웅, 홍의경, XML 문서 저장 시스템, 데이터베이스 연구회지, 16권 2호, pp. 29-34, 2000.

[5] S. Malaika, Using XML in Relational DB Application, 15th International Conference on Data Engineering, 1997.

[6] 박민경, 홍의경, XML 문서 저장 시스템의 성능 평가, 정보과학회 2001년 추계학술대회, VOL. 28NO.2 pp.0082-0084 2001.

[7] F. Tian, D. J. DeWitt, J. Chen, and C. Zhang, The Design and Performance Evaluation of Alternative XML Storage Strategies, Technical report, CS Dept., University of Wisconsin, 2000.

저 자 소 개

박 준 범

1990 목포대학교 전산통계학과(이학사)
1994 목포대학교 대학원 전산통계학과(이학석사)
2003 목포대학교 대학원 컴퓨터공학과(공학박사)
1994년 ~ 현재 동아인재대학 인터넷정보계열 부교수
〈관심분야〉 데이터베이스, XML, 운영체제

박 경 우

1986 전남대학교 전산통계학과(이학사)
1988 전남대학교 대학원 전산통계학과(이학석사)
1994 전남대학교 대학원 전산통계학과(이학박사)
1995년 ~ 현재 목포대학교 정보공학부 컴퓨터공학전공 부교수
〈관심분야〉 분산시스템, 시스템소프트웨어, 운영체제

오수열

1981 전남대학교 공과대학 재료공학과(공학사)
1985 조선대학교 대학원 전산학과(공학석사)
2002 전남대학교 대학원 전산통계학과(박사수료)
1988년 ~ 현재 목포대학교 정보공학부 컴퓨터공학전공 교수
〈관심분야〉 멀티미디어시스템, 객체지향기술, 컴포넌트소프트웨어