碩士學位論文

ETID를 이용한 XML 기반의 계층적 RDB 스키마 모델

濟州大學校 大學院

컴퓨터工學科

權　　勳

2005年 6月

ETID를 이용한 XML 기반의 계층적 RDB 스키마 모델

指導敎授 郭 鎬 榮

權 勳

이 論文을 工學 碩士學位 論文으로 提出함

2005 年 6 月

權勳의 工學 碩士學位 論文은 認准함

審査 委員長 _____ 印

委 員 _____ 印

委 員 _____ 印

濟州大學校 大學院

2005 年 6 月

Hierarchical RDB Schema Model based on XML using ETID

Hoon Kwon
(Supervised by professor Ho-Young Kawk)

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF MASTER OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING
GRADUATE SCHOOL
CHEJU NATIONAL UNIVERSITY

2005. 6.

목 차

그 림 목 차

표 목 차

Summary

Hierarchical RDB Schema Model based on XML using ETID

Hoon Kwon
Dept. of Information Engineering
The Graduate School
Cheju National University

XML instances for purpose of information exchange are normally stored in the legacy relational database. Therefore, integrations with relational database are required for effective XML applications. To support these requirements, virtual decomposition storage or decomposition storage methods and Element Type ID(ETID) Model which save separates structures of instances to relational database have researched. However, these storage methods contain different level information of instance structure and Schema structure layers which has caused difficulties to process query during search operation as well as

increased overheads due to duplicate savings for separate storages.

Therefore, in this thesis, additional field of "Eltype" has introduced to previous database schema structure to integrate instance and instance structure using ETID model, provide consistent information of layers and propose storage structure to map each field to schema field of relational database. As results, in this thesis, propose storage structure as hierarchical RDB Schema model based on XML using ETID model becomes XML instance and structures can be stored together to minimize overheads and required storage-space. Also, synchronized storage layer structure provides easier processing of search query.

Ⅰ. 서 론

최근 인터넷 사용으로 인한 정보 양의 급증에 따른 웹기반의 정보들을 보다 효과적으로 이용하고자 하는 연구가 활발히 이루어지고 있다[1]. 그러나 HTML(HyperText Markup Language)은 문서의 구조보다는 표현에 중점을 두고 있어서 특정 응용 분야의 정보 재활용에는 기능이 부족하다는 단점을 갖고 있다[1, 2, 6, 7]. 그래서 W3C(World Wide Web Consortium)에서는 HTML의 편리성과 SGML(Standard Generalized Markup Language)의 확장성 등의 장점을 취합하여 이기종간의 시스템에서 작성된 문서의 상호교환과 다양한 문서형식들의 구조에 따른 일관성을 유지[3, 4]할 수 있는 차세대 웹 문서의 표준인 XML(eXtensible Markup language)을 제안하였다[1, 2, 3].

그리고 이를 반영하듯 웹 문서뿐만 아니라 전자출판, 의학, 경영, 법률, 판매 자동화, EDI, 전자 상거래(EC), 지식 관리 시스템, E-비지니스, 음성인식 시스템(VoiceML), 디지털 전자도서관 등 다양한 분야에서 XML을 활용하고자 폭 넓은 연구가 진행 되고 있으며, 또한 그 기능을 확장해 오고 있다[2, 5, 6, 7]. 이러한 이유는 기존의 EDI는 한번 만들어진 문서 포맷의 수정이 쉽지 않고 기업들마다 거래하기 전의 품목과 데이터를 서로 맞춰야 하는 불편이 있지만, XML은 확장 언어로 태그를 바꾸거나 다른 방법으로 매핑하는 등 일정하지 않은 문서 포맷으로도 데이터를 주고받을 수 있는 정점이 있기 때문이다. 따라서 전사적 자원 관리(ERP, Enterprise Resource Planning)와 조달 시스템 그리고 온라인 주문 관리 시스템간의 데이터 교환, 그룹웨어 등에 XML 기반 애플리케이션 활용이 확대될 것이며 특히 전자상거래 분야에서 XML을 활용한 문서 교환이 증가하고 있다[6. 7].

하지만, 현재 문서의 교환을 위한 XML 데이터는 eXoelon과 tamino와 같은 XML 전용 데이터베이스 및 Legacy 데이터베이스를 이용하여 저장과 검색이 이루어지고 있다. Legacy 데이터베이스 측면에서 살펴보면, XML 저장구조와 Legacy 저장구조가 서로 상이하여 문서의 재활용에 많은 어려움이 존재하고 있다. 그러므로 XML 문서를 기존 데이터베이스 기반에서 저장, 검색할 수 있는 구조로 변경해주는 모델링 연구가 필요하다[8, 9].

따라서 본 연구는 ETID 모델[10, 11]을 이용하여 XML 기반의 계층적 RDB 스키마 모델을 제안하고, XML 저장에 따른 저장 공간과 검색 효율성을 나타낼 수 있도록 하였다. 제안하는 스키마 구조는 XML 스키마에 종속적인 구조를 가지며, 인스턴스 및 인스턴스 구조의 효율적인 저장과 검색을 위하여, 기존의 분리되어 저장되던 방식을 하나로 통합하여 저장하는 특성을 갖는다. ETID 모델을 이용하여 엘리먼트와 속성에 따른 구조를 표현하고, 인라인 기법을 통한 저장 방식으로 계층적 데이터베이스 스키마 모델을 제시하였다. 계층적 데이터베이스 스키마 모델을 위해, 기존 스키마 모델에 “SDICHK” 필드를 추가하여, 스키마와 DTD, 인스턴스를 구분하고, “Eltype” 필드를 통해 스키마 정의시 반복되는 엘리먼트의 형식인 ComplexType과 SimpleType를 축약하여 CLOB(Character Large OBject)의 형태로 저장이 가능하도록 제시하였다.

따라서, 기존 연구들이 처리하지 못 하였던, 인라인 저장 방식[11]에 따른 레벨 정보를 동일하게 저장하고, 반복되는 엘리먼트를 “Eltype” 필드를 이용하여 처리함으로써, 저장 공간과 검색에서의 효율성을 가져오게 되었다.

본 연구의 구성은 이미 살펴본 것과 같이 Ⅰ장 서론에서는 XML의 연구 동향 및 연구의 필요성에 대해 간략히 서술하였으며, Ⅱ장에서는 XML 문서에 대한 이론과 배경을 살펴보고, 또한 구조정보를 모델링하고, 저장하는 방법에 대한 기존 연구들을 살펴보겠다. Ⅲ장에서는 XML 문서의 저장 공간 및 검색 효율을 위한 RDB 스키마 저장모델에 대한 기본 저장구조를 제시하며, 실제 XML 문서의 저장 방법을 설명하고, Ⅳ장에서는 기존의 XML 문서 저장

모델 구조와 제안 모델 구조를 적용한 결과를 보이고, 이를 통한 비교 분석을 하였으며, 마지막 Ⅴ장은 결론으로 제안 모델에 대한 향후 연구 과제와 방향에 대해 서술하였다.

Ⅱ. 관련 연구

1. XML 문서 분석

XML은 플랫폼에 대해 독립적이며, 확장이 가능하고, 복합 구조(Complex Structures)와 검증(Validation) 등 SGML의 기본 특징을 그대로 지원하면서 간략화하여 구현하기 쉬운 장점을 갖는다. 또한, 사용자가 원하는 대로 태그와 속성을 지정할 수 있고, 새로운 태그를 만들 수도 있는 언어적 성격이 강하다. 즉 HTML이 태그집합체이었다면, XML은 확장이 가능하고, Java script나 VB script와도 연동이 가능한 개발 도구라는 특징을 가지고 있다.

이러한 XML의 문서 구조는 정형화되어 있으며, 일반적으로 Fig. 1과 같이 크게 3부분으로 구성된다.

Fig. 1 Structure of XML document

XML 문서는 크게 XML 선언부, XML 문서 타입 정의부, XML 몸체부로 나뉘게 된다.

XML 선언은 작성된 문서가 XML 문법을 따르는 문서라는 것을 나타낸다. XML 선언부는 반드시 태그 시작과 끝에 〈? ~ ?〉 태그를 붙여야 하며, 이는 처리지시(Processing Instruction) 요소 형식을 갖는다. 최소화 형태의 XML 선언은 예약어와 XML 스펙 버전 번호로 구성된다. XML 문서 표현상의 인코딩에 관한 특별한 사항이 존재한다면, 인코딩에 따르는 속성 정보를 가질 수 있다. 이 선언 앞에는 공백문자를 포함한 어떤 문자도 올 수 없으며, 반드시 제일 앞에 선언되어야 한다. 이러한 XML 선언을 위한 부분이 XML 선언부이다.

XML 문서 타입 정의부는 XML 문서의 정의 타입에 따르는 유효성을 검증하기 위한 부분으로써, DTD(Document Type Definition), XML 스키마(Schema) 선언, 네임스페이스(Namespace) 선언이 올 수 있다. 선택부분이기 때문에 생략이 가능하다.

1) DTD(Document Type Definition)

DTD는 문서의 형태를 정의하는 규칙을 말하는 것으로, EBNF 문법에 따라, XML 문서의 구조를 명시적으로 선언하는 역할을 하며, XML 문서가 잘 만들어진 유효한 문서인지를 확인하기 위해 사용한다. 따라서, DTD를 사용하여 요소와 요소의 내용, 속성과 속성의 내용, 그리고 요소의 순서나 반복성 등을 미리 정의해 놓게 된다.

2) XML Schema

XML Schema는 XML 문법에 따라, DTD와 동일하게 문서 구조와 내용을 제한하는 방법을 나타내며, DTD에 비해서 문서에 대해 더욱 강력한 제한을 허용하고, 기존의 데이터 타입의 범위, 길이와 같은 특정 속성을 제한하여 유도한 사용자 정의 데이터 타입을 정의할 수 있다. 또한 한번 정의한 XML Schema의 확장성과 재사용성이 높고, XML Namespace를 지원한다. 이에 따라, 점차 XML 문서정의 타입형식으로 XML Schema의 사용이 증가하는

추세이다.

3) XML Namespace

XML에서 태그를 자유롭게 정의할 수 있는 점은 강력한 장점중 하나이다. 그러나, 이러한 장점에도 불구하고, 태그의 중복사용이라는 문제를 야기하게 된다. 이에 따라, 중복된 태그로 빚어질 수 있는 이름 충돌을 해결하기 위한 개념이 XML Namespace이며, 이는 각 요소명과 속성명을 정확히 인식할 수 있는 개념을 제공한다.

DTD와 XML Schema는 각각의 장단점을 지니고 있다. Table 1.은 DTD와 XML Schema를 비교한 것이다.

Table 1. DTD와 XML Schema의 비교

	DTD	XML Schema
문 법	EBNF	XML 문법
자료형	문자열	다양한 자료형
파 서	DTD 전용 파서 필요	XML 파서 활용
네임스페이스	지원 불가	지원
모 델	닫힌 모델	열린 모델

XML 몸체부는 XML 문서 타입 정의부에서 정의된 요소들과 속성들에 대한 내용을 나타낸다. XML 태그와 컨텐츠로 이루어진 몸체(body)가 있다. 몸체는 요소(Element)와 속성(Attribute)으로 구성된다. 요소는 일반적으로 시작 태그(Start Tag)와 끝 태그(End Tag)로 구성되어 있으며, 그 사이에는 문자 데이터, 엔티티, 주석, CDATA 섹션 등이 올 수 있다. 요소의 구성은 시작 태그와 끝 태그 쌍, 그 사이에 들어 있는 컨텐츠인 데이터를 통칭하는 것이다.

속성은 XML 파서가 애플리케이션에게 보내는 값들을 뜻하며, 요소의 컨텐츠 값과는 서로 구별되는 값이다. 일반적으로 요소에 의미를 부여해 주는 형태로 많이 사용되어진다. 속성은 요소의 시작태그 안에서 선언되며, 해당 애플리케이션에 보내는 값을 지정할 수 있으며, 유의할 점은 속성 값은 항상 따옴표 안에 둘러싸여 있어야 한다.

2. XML 저장을 위한 데이터베이스

XML 문서를 저장, 관리, 검색하려면 XML 특성에 맞는 문서 저장/관리 시스템이 필요하다. XML 데이터를 위한 데이터베이스 구현 기술은 크게 4가지로 나눌 수 있다.

1) 파일 관리형 데이터베이스

파일 관리형 데이터베이스는 각 XML 데이터를 파일 시스템 내부의 파일로 보관하며, XML 파일을 디렉토리로 관리한다. 이는 구조정보를 가지지 않은 정형식 XML 문서관리에 적합하며, 인덱스를 통한 빠른 검색이 가능하다. 그러나, XML 문서 추출시마다 XML 파싱(Parsing)이 필요하다는 단점이 있다. 보통은 XML 문서를 CLOB(Character Large OBject)나 BLOB(Binary Large OBject)로 관리하는 XML 데이터베이스도 파일 관리형으로 취급한다.

2) 관계형 데이터베이스

관계형 데이터베이스(RDB)는 XML 문서를 매핑해서 저장 관리하는 방식이다. 기존의 관계 데이터베이스를 재사용할 수 있는 장점이 있고, 조작이 간편하고 SQL문 등을 사용할 수 있어 가장 일반적으로 구현하는 방식이다.

이미 많은 사용자를 확보하고 있기 때문에 XML 문서를 관계형 데이터베이스에 저장하기 위한 방법들에 대해서 보다 많은 관심을 받고 있다. XML의 계층구조를 관계형 데이터베이스의 관계(relation)로 표현하며, XML 각 요소(element) 값과 속성(attribute)을 테이블 필드로 표현하기 때문에, 구조를 정의하고 있는 문서를 적용하는데 좋다. 실제, XML 문서구조는 인덱스를 지니고 있으나, 저장 단위가 관계이기 때문에 데이터를 출력할 때는 조인(join)연산이 필요하다.

3) 객체지향형 데이터베이스

객체지향형(Object Oriented) 데이터베이스는 XML 데이터의 계층 구조를 객체지향의 클래스 계층에 매핑하여 구현하는 방식이다. 객체지향 개념을 이용할 수 있기 때문에 상속과 같은 객체지향 특성을 이용할 수 있으며, 엘리먼트 간의 전후종속 관계를 클래스에 기반한 객체들간의 링크로 나타낼 수 있다. 관계 데이터베이스보다 데이터 모델이 XML과 유사하여 문서 정의 타입으로부터 객체 스키마를 생성하는 문제와 XML 질의를 OQL(Object Query Language)로 변환하는 과정이 쉽고 가변 필드의 사용에 제한이 없어서 대용량 데이터 형식(built-in data type)으로 지원되기 때문에 XML의 순서정보 표현이 간단하다.

4) 네이티브 XML 데이터베이스

네이티브(native) XML 데이테베이스는 XML의 계층 구조를 계층형 데이터베이스 구조로 관리하며 항목의 데이터나 속성을 어디까지나 XML 데이터로 관리하므로 문서 관리에 적합하다고 볼 수 있다. 또한, 구조를 정의하고 있는 XML 데이터에 독자적인 인덱스를 부여, 고속의 트랜잭션 처리를 지원한다. 이러한 네이티브 XML 데이터베이스는 데이터 모델 변환이나 데이터 저장에서 부담이 적지만, 데이터 검색을 위하여 여러 가지 인덱스를 참조해야 하므로 대규모 XML 데이터 처리에서 성능이 낮아진다는 단점이 있다.

네이티브 XML 데이터베이스는 문서를 저장하는 모델에 따라 텍스트 기반과 모델기반으로 나눌 수 있다. 텍스트 기반은 XML 문서를 텍스트 형태로 통째로 저장하고 Access하기 위한 몇 가지 기능을 제공하며, 모델 기반은 DOM 트리처럼 XML 문서의 바이너리 모델을 지원한다. 이를 지원하는 데이터베이스로는 소프트웨어 에이지(Software AG)사의 타미노(tamino)나 엑셀론사의 Sonic/eXcelon XIS(eXtensible Information Server)등이 있다.

XML 저장을 위한 데이터베이스를 Table 2에 비교하여 나타내었다.

Table 2. XML 저장 데이터베이스의 비교

	파일관리형	관계형	객체지향형	네이티브
저장방식	File	테이블 매핑	클래스 매핑	독자적 인덱스
사용빈도	저	고	저	중
장 점	정형식 문서 관리 적합	데이터 재이용 가능 조작 간편	데이터모델 다양	쉬운 인터페이스 고속 트랜잭션 처리
단 점	문서 추출시 파싱 필요	조인 연산 필요	사용빈도와 정보 재이용율 낮음	검색시 인덱스의 빈번한 참조

3. XML 저장 모델

웹에서의 정보들을 활용하기 위한 방법으로 사용되고 있는 XML 문서에 의한 정보 교환이 점차 늘어남에 따라, 보다 효율적으로 XML 문서가 가지고 있는 정보를 저장하는 방법에 대한 연구들이 진행되어 지고 있다. 우선, XML 저장관리 시스템을 개발하기 위해 선행되어야 하는 것이 데이터 모델링이다. 데이터 모델링이란 데이터와 데이터들 간의 관계를 기술하는 개념적 도구로서, 데이터베이스의 논리적 구조를 명시하는 과정이다. 이러한 데이터 모델링은 DBMS 활용과 문서 정의 형식에 따라 저장하는 방법들이 연구되어

지고 있다. 첫째, DBMS 활용에 따라 기존의 데이터베이스(RDB, OODB)를 이용하거나, 별도의 전용 데이터베이스를 이용하여 XML 문서를 저장하는 방법으로 나눌 수 있으며, 두 번째, 문서 정의 형식에 따라 저장하는 연구는 관계형 데이터베이스 스키마를 생성한 후 XML 문서를 저장하는 방법과 문서 정의 형식에 상관없이 XML 문서를 저장하는 방법들이 연구되어지고 있다.

이러한 저장 방식 중 일반적으로 계층적 XML 문서 구조정보를 관계형 데이터베이스에 저장하는 모델로써, XML 문서를 엘리먼트 단위로 쪼개어 저장하는 분할 저장 모델과 문서 전체를 한꺼번에 저장하는 비분할 저장 모델로 나뉘어 진다.

1) 분할 저장 모델

XML 문서에 대해서, 엘리먼트 단위로 나누어서 저장하는 방식으로 문서의 실제 내용을 가지고 있는 각각의 단말 엘리먼트 안에 문서의 내용이 나뉘어 저장되고, 검색 시 구조 정보를 참조하여 해당 엘리먼트나 하위 엘리먼트의 조합을 생성하여 처리하는 방식이다. 따라서 문서의 구조적 정보나 일부 내용들이 수정되었을 때 관계 있는 엘리먼트만 수정하면 되므로 문서의 버전 관리가 쉽고, 동일한 내용을 갖는 노드들을 공유할 수 있다는 장점이 있다. 그러나 문서를 검색하여 내용을 추출해야 하는 경우에는 각 엘러먼트의 내용을 조합하여 결과를 구성해야 하므로 검색 과정이 복잡하고 검색 시간이 오래 걸린다는 단점과 문서 타입 정의에 의존적인 데이터 모델을 사용하므로 DTD나 스키마 구조가 갱신되어지면, 이와 관련된 모든 데이터들이 재구성되어야 한다는 문제점이 발생한다.

2) 비분할 저장 모델

XML 문서 전체를 저장하는 방식이다. XML 저장 관리 시스템에서는 전체문서뿐만 아니라, 엘리먼트 단위의 검색까지 지원해야 한다. 이때 엘리먼트들은 문서 안에서 계층상의 구조 정보를 가지고 있고 이로 인해서 모든

레벨에서 수평적 관계 및 수직적인 관계의 표현이 가능하다. 각 엘리먼트에 대한 구조 정보 및 위치 정보를 유지하기 위해서 별도의 테이블을 만들고, 엘리먼트가 문서 안에서 나타난 시작 오프셋과 끝 오프셋에 관한 정보를 엘리먼트 구분자와 함께 기록한다. 트리상의 각각의 노드는 실제 문서의 논리적인 구조만을 기술하고 내용부분을 다른 영역에서 관리하여 각각의 노드가 이 영역에 대한 위치정보와 길이 등을 가지고 내용을 표시하는 비분할 방법이다. 문서 전체는 한꺼번에 저장하므로, 전체 문서에 대해서는 빠른 검색이 가능하며, 각 엘리먼트에 대한 정보를 유지하므로 엘리먼트 단위의 검색도 지원한다.

문서를 한꺼번에 저장하기 때문에 분할 저장 방법과 달리 통합 과정이 필요하지 않으므로, 단말 노드에 대한 검색의 경우 위치 정보를 이용한 구조적 검색이 가능하고, 검색 효율이 상대적으로 우수하다. 그러나 XML 문서의 일부분에 대하여 엘리먼트의 추가, 삭제 등의 변경이 발생하는 경우에는 문서 내의 변경된 부분만을 효율적으로 반영하지 못하고, 문서 전체를 다시 저장해야 하는 단점이 있으며, 이에 따른 오버헤드와 데이터베이스 일관성을 유지해야 하는 문제가 있다.

분할 저장 모델과 비분할 저장 모델을 Table 3에 비교하여 나타내었다.

Table 3. 분할 저장 모델과 비분할 저장 모델의 비교

	분할 저장 모델	비분할 저장 모델
저장 방식	문서를 나누어 저장	전문(Full-Text) 전체저장
검색 방법	해당 노드를 하나씩 직접 가져온다.	오프셋, 길이를 이용해 문서를 읽어 온다.
테이블 구성	하나의 독립 테이블 구성	엘리먼트와 정보테이블로 구성
검색 시 취합여부	모든 리프 노드를 순회하여 취합	필요 없음

4. XML 구조 정보 표현 모델

XML을 이용하여 구조화된 정보로 표현하기 위한 여러 가지 모델의 연구가 이루어지고 있다. 현재 구조 정보를 표현하기 위한 모델로는 SCL 모델, K-ary 완전 트리 모델과 ETID 모델 등이 있다.

1) SCL(Simple Concordance List) 모델

Fig. 2와 같이 구조 문서의 계층적 관계보다는 포함 관계를 이용한 표현 방법으로서 SC-List라는 데이터 타입을 통해 중첩된 정보를 허용하므로 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 표현 할 수 없다는 단점이 있다. 또한, SCL 모델을 응용하여 각종 계층적 포함관계를 이용한 버전 부여 기법들이 연구되어지고 있다.

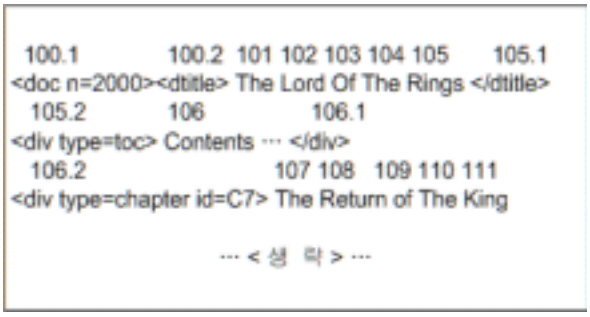


Fig. 2 Simple Concordance List model

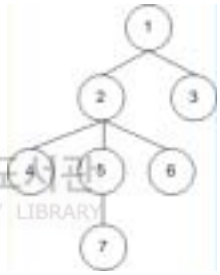
2) K-ary 완전 트리 모델

문서에 대한 트리로부터 이들 노드 중 가장 큰 차수 K를 구하여 K-ary

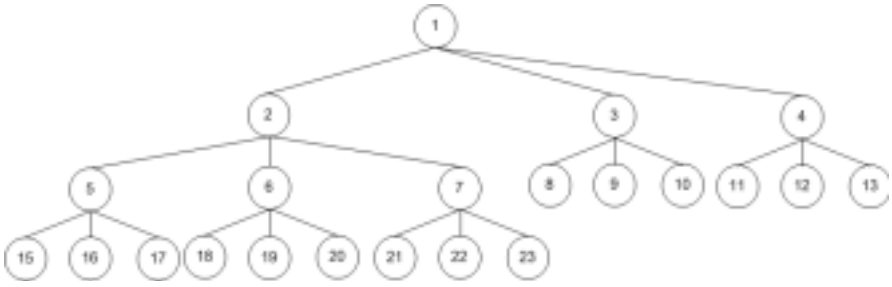
완전 트리로 재구성한 후, 문서 트리를 매핑하여 각 노드에 대해 모든 번호를 부여한다. Fig. 3은 K-ary 완전 트리 모델의 예이다.

이 모델은 문서 구조 사이의 계층 관계를 간단한 공식을 통해 쉽게 구할 수 있다는 장점이 있는 반면, Fig. 3의 (a)와 같은 파싱 트리에 (b)와 같이 노드값을 부여하게 되면, 현 파싱 트리에 대한 구조를 나타낼 수 있게 된다. 그러나, 이는 노드값만을 부여한 것이기 때문에, 노드 깊이나, 노드의 위치를 알 수 없다. 따라서 (c)와 같이 K-ary의 완전 트리 모델로 매핑 과정을 수행함에 있어 그림(c)와 같이 불필요한 Null 노드가 증가되어질 수 있고, 노드의 깊이가 깊어질수록 노드 변화가 커진다는 단점이 있다.

(a) 파싱 트리



(b) 노드값을 부여한 파싱 트리



(c) K-ary Tree (K=3)

Fig. 3 K-ary tree model

3) ETID(Element Type ID) 모델

ETID는 같은 엘리먼트들 간의 계층 정보와 동일 부모 엘리먼트를 갖는 자식 엘리먼트들의 순서 정보, 그리고 동일한 부모 엘리먼트를 갖는 자식들 중 동일한 타입의 엘리먼트들에 대한 순서 정보를 통해 구조 문서를 표현한다. Fig. 4는 ETID 모델의 예이다. Fig. 4에서와 같이 DTD의 구조를 분석하여 ETID를 모두 분석한다. 부모노드의 ETID를 상속받아 자식노드의 ETID 값이 구성된다. 이 방법은 내부의 노드 표현이 선택적으로 되어있을 때, 트리를 구성하기에 많은 문제점이 존재한다. 따라서, 이 ETID를 변형시킨 방법이 있다. 먼저 DTD의 EID를 추출하고, XML 문서를 순회하면서 ETID를 부여하는 방법이다. 이 방법은 DTD에서 계층 정보를 만드는것이 아니라, 단순히 EID만 추출하며, 이를 통해 XML 문서에서 ETID를 구성한다. 이를 통해 DTD 내부의 엘리먼트 순환에 따르는 문제를 해결할 수 있다. 이러한 ETID 모델은 기준 엘리먼트로부터 특정 엘리먼트에 대한 계층 정보와 순서 정보를 간단한 문자열 조작만으로 쉽게 구할 수 있다는 장점이 있는데 반해 트리의 깊이가 깊어질수록 각 노드를 표현하기 위한 공간이 무한대로 늘어난다는 단점이 있다.

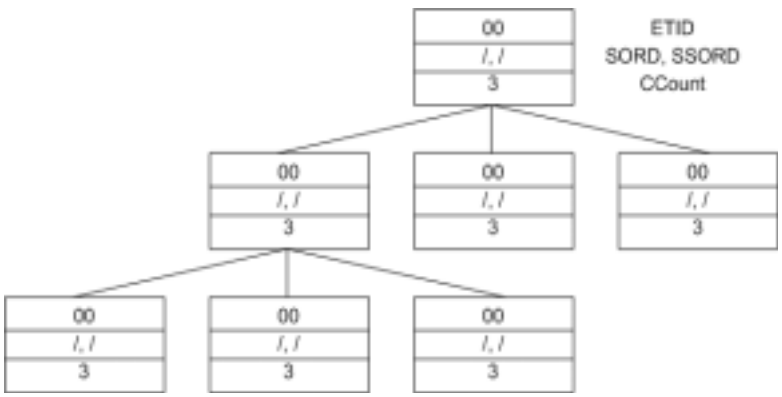


Fig. 4 Element Type ID model

5. 관계형 데이터베이스를 위한 XML 저장 방법

계층적 구조를 가진 XML 문서를 평평한 구조의 관계형 데이터베이스에 저장하는 작업은 쉽지 않다. 무엇보다도 관계형 테이블로는 계층 구조를 표현하기 어렵기 때문이다. 대표적으로 기존의 준구조적(semistructured) 데이터를 저장히기 위해 많이 사용되던 간선저장 방식과 일정한 속성을 공유하는 엘리먼트들을 하나의 테이블에 "인라인" 시키는 방식으로 연구되어지고 있다.

1) 간선(Edge) 저장 방식

XML 문서를 RDBMS를 사용하여 저장하고자 할 때, 가장 단순한 방식이라 할 수 있는 간선 저장 방식은 기본적으로 XML 문서를 표현한 그래프 상에서의 모든 간선들에 대한 정보를 간선(Edge)이라 불리는 하나의 테이블에 저장한다. 이러한 간선 테이블은 그래프 상에서의 각 간선의 소스(source)와 타겟(target)노드의 노드 식별자 값을 저장하기 위한 source 애트리뷰트와 target 애트리뷰트, 간선의 이름을 저장하기 위한 name 애트리뷰트, 간선이 가리키는 노드가 내부 노드 (internal node) 인지 리프 노드인지를 표현하기 위한 flag 애트리뷰트, 간선의 순서정보를 저장하기 위한 ordinal 애트리뷰트로 구성된다.

그러나, 이런 간선 저장 방식은 많은 조인 연산의 필요성으로 인하여 성능이 저하되는 큰 단점을 가진다.

2) 인라인 저장 방식

인라인 저장 방식[11]은 일정한 속성을 공유하는 엘리먼트들을 하나의 테이블에 "인라인" 시키는 방법으로, DTD를 관계형 스키마로 표현하기 위하여 세가지 관계형-변형 알고리즘인 Basic, Shared, Hybrid기법을 이용하고 있다. Basic 기법은 엘리먼트 그래프를 깊이 우선 순회하면서 방문하여 엘리

먼트들을 하나의 테이블에 인라인 시키는 방법을 사용하였다. Shared 기법은 DTD 그래프를 순회하면서 루트 노드에 해당하는 엘리먼트에서만 테이블을 생성한다. 자식 엘리먼트 노드들을 방문하면서 엘리먼트들을 인라인 시키는 것은 Basic과 같지만, 추가적으로 다른 테이블과 공유되어질 수 있는 엘리먼트들을 만나게 되면 독립적인 테이블로 처리하는 기법이다. Hybrid 기법은 앞의 두 기법을 혼합한 것으로, 가장 좋은 성능 평가를 나타낸다. 이 기법은 DTD를 이용하여 가능한 적은 수의 테이블로 XML 데이터를 사상시킨다.

현재 DTD를 이용한 인라인 기법은 제안되어있으나, DTD를 대체하기 위해 W3C에서 새로이 제정된 XML Schema를 위한 인라인 기법에 대한 연구는 미비한 실정이다.

Ⅲ. 제안 시스템 및 DB 스키마 모델

1. 제안 시스템 모델

Fig. 5는 제안 시스템 모델이다. 제안 시스템 모델은 크게 3개의 Layer로 나뉘어 처리 된다. XML 문서가 들어오면 Mapping Layer에서 XML 문서가 정형화 되었는가를 분석을 한 후, 문서를 순차적으로 읽어 내려가면서 요소들의 연관 관계를 추출한다. 분류되어진 요소들은 매핑 테이블로 옮겨지고 데이터베이스 스키마를 생성한다. 생성된 스키마는 인덱스 관리자로 이동하고 각각의 엘리먼트, 속성, 컨텐츠에 인덱스를 부여한다.

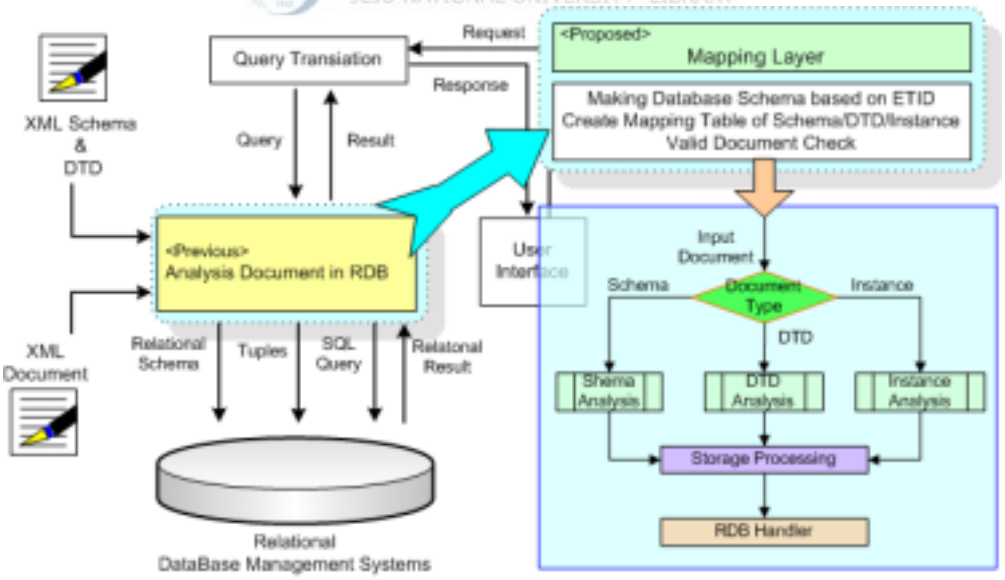


Fig. 5 Proposed System Model

요소들과 인덱스들은 데이터베이스에 저장이 된다. 구조 정보를 검색하기 위해서는 사용자 인터페이스 통해 질의를 하고, 질의 처리기는 사용자가 질의한 키워드를 기반으로 구조적인 정보를 수집하여 원하는 구조 정보를 검색하여 사용자에게 다시 전달을 한다. 본 연구에서는 Mapping Layer에서의 처리 방법에 대해 중점을 두었다.

1) Input Layer

Input Layer에서는 XML Schema, DTD 또는 XML 문서의 입력을 받고, 이를 Mapping Layer에 전달하는 기능을 한다.

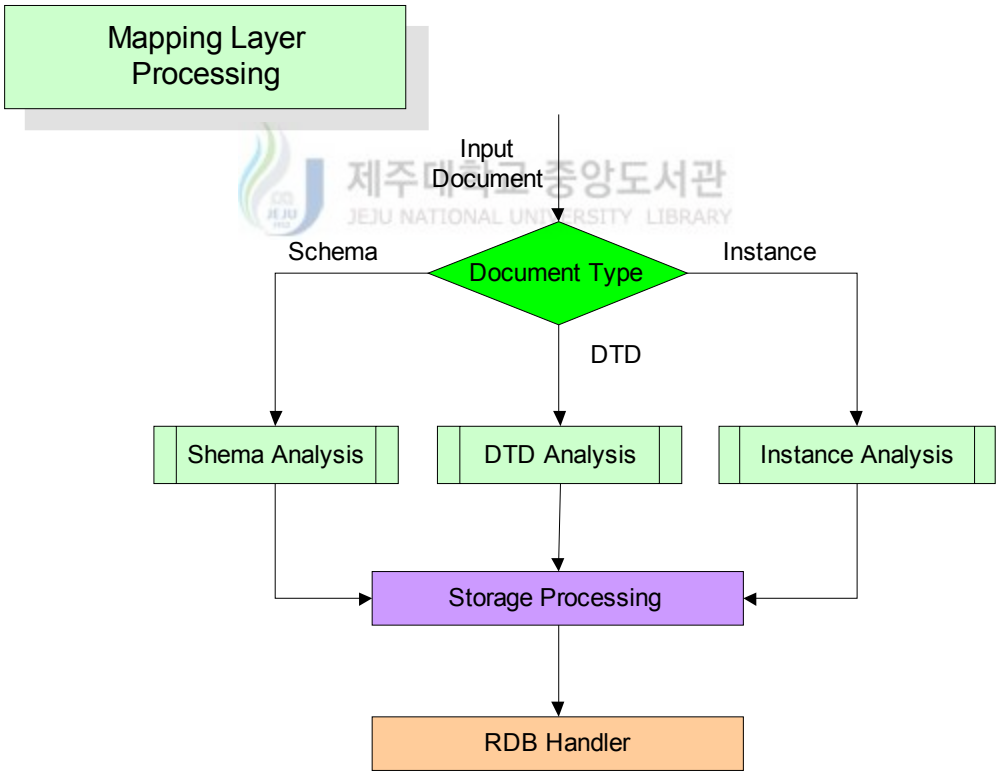


Fig. 6 Mapping Layer Processing

2) Mapping Layer

Mapping Layer에서는 제안된 데이터베이스 스키마에 따라 입력받은 XML 문서에 대해 Fig. 6과 같은 처리(Processing)과정을 수행한다. Fig. 6에서 볼 수 있듯이 XML 문서가 들어오게 되면, 일단, 들어온 문서가 Schema인지, DTD 인지, 인스턴스인지를 구분하게 되며, 각각의 해당되는 경우의 분석(Analysis)과정을 거치게 된다. 문서 구조에 대한 분석과정이 끝나면, 제안된 데이터베이스 스키마에 매핑 테이블을 구성한다. 그 후, RDB Handler를 통해 기존 RDB에 저장한다.

(1) Schema Analysis

스키마의 입력이 들어오게 되면, 인라인 방식으로 각각의 엘리먼트를 읽어온다. Schema는 엘리먼트명을 정의한 다음에는 반드시 엘리먼트 문서타입을 정의하게 되어있다. 따라서, 엘리먼트 문서타입부분을 먼저 읽어들여 계층정보에 바로 저장하지 않고, "Eltype" 필드에 CLOB의 형식으로 약속된 문자열로 축약하여 저장하는 과정이 필요하며, 이에 따른 계층정보 역시 수정되어져야 한다. 또한, 엘리먼트 선언에 따른 제약사항을 분류하여, 각각의 정의된 필드에 분류하는 작업이 수행되어진다.

(2) DTD Analysis

DTD의 입력이 들어오면, Schema Analysis와 같이 인라인 방식으로 엘리먼트 정의를 읽어오게 되며, 이때, 엘리먼트의 포함관계에 따라 계층정보를 나뉘어 저장하게 된다. 이 과정에서는 Schema Analysis와 달리 DTD는 문서타입정의에 따른 저장방법은 필요하지 않다.

(3) Instance Analysis

Instance의 입력이 들어오면, 인라인 방식으로 엘리먼트를 순서대로 읽어오며, 읽어온 구조를 통해, 계층정보를 부여하고, 이때, 모든 Instance은 정보를 읽게 되면, 기존에 저장되어진 RDB 상의 Schema구조와 비교하여 해당 Schema를 찾아 검증(Validation)을 수행하게 된다. 이때, 검증이 된 Instance 문서에 대해서 데이터베이스 스키마에 저장되게 된다.

3) User Interface Layer

실제 사용자가 인터페이스를 통해 접근하여 결과를 확인하는 Layer이다. 여기서 사용자는 제안 방식에 따라 저장된 결과를 확인하고, 검색에 대한 질의를 할 수 있다. 검색 질의시, Query Transaction에 질의를 보내 처리된 결과를 User Interface Layer를 통해 사용자에게 보여주게 된다.

2. Database Schema Structure

본 연구에서 다루는 가장 중요한 부분인 Mapping Layer에서는 입력받은 XML에 대한 구조분석을 통한 기존 RDB의 저장을 위한 DB 스키마를 구성하게 되는데, Fig. 7은 기존 연구에서의 DB 스키마 구조와 제안 시스템 모델에서의 DB 스키마 구조를 나타낸다.

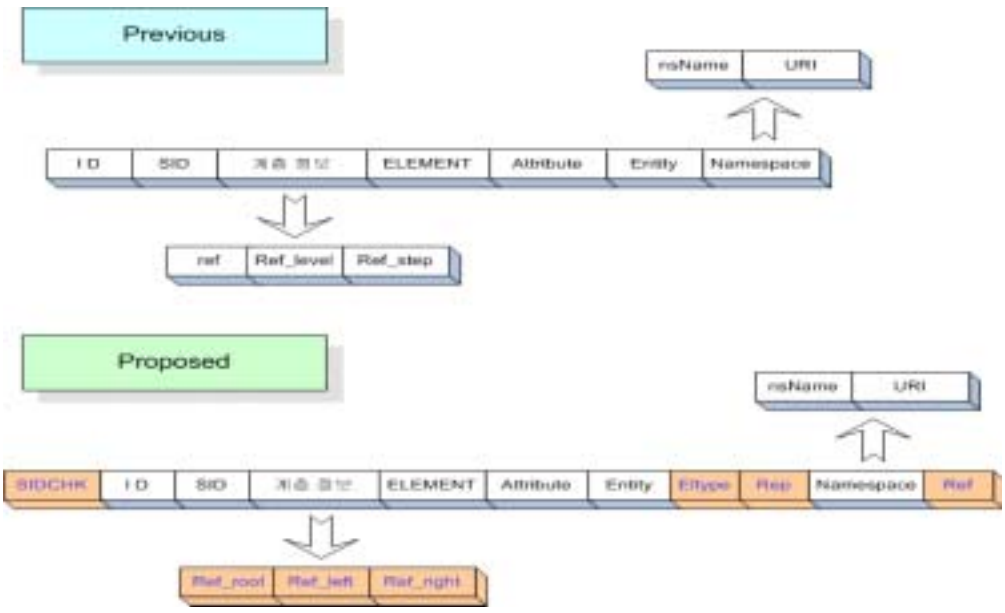


Fig. 7 Existing and Proposed Database Schema in Mapping Layer

1) 기존 데이터베이스 스키마

기존의 데이터베이스 스키마에서의 각 필드에 해당하는 내용을 정리해 보면 Table 4과 같다.

Table 4. Information of Field in Existing Database Schema

필드명	의 미
ID	엘리먼트를 구분하기 위한 식별자
SID	XML Schema 와 Instance의 적용 구분자
ref	상위 엘리먼트
ref_level	엘리먼트 등급
ref_step	엘리먼트의 순서
element	엘리먼트 이름
attribute	속성 이름과 값
entity	엘리먼트의 값
Namespace	네임스페이스를 처리하기 위해 접두사 nsname 과 경로 URI 로 표현

이와 같은 Database Schema를 기반으로 하여 XML을 기존 RDB에 저장하게 되면, Schema와 Instance에 따르는 두 개의 정적 테이블이 만들어지고 모든 XML 구조를 인라인 방식으로 해석하여, 엘리먼트 단위로 저장함으로써, 계층정보를 표현하는 ref, ref_level, ref_step간의 정보가 Schema와 Instance 간에 서로 불일치하게 된다. 이로 인한 검색에 대한 질의가 복잡하게 된다.

2) 제안 데이터베이스 스키마

본 연구에서 제안하는 데이터베이스 스키마에서는 기존 데이터베이스 스

키마에 Fig 7에 나타낸 것처럼 SDICHK, Eltype, Rep, Ref 의 4개의 필드를 새로 추가하고, 계층정보를 나타내는 기존의 방법을 보안하여, XML 트리 구조의 효율적인 탐색을 위한 Ref_root, Ref_left, Ref_right로 바꾸었으며, 이에 대한 내용은 Table 5와 같다.

Table 5. Information of Field in Proposed Database Schema

필드명		의 미
SDICHK		XML Schema 와 Instance의 식별자 Schema : S , DTD : D , Instance : I
Level Info	ref_root	자신의 상위 엘리먼트의 위치
	ref_left	자신의 왼쪽 형제 엘리먼트의 위치
	ref_right	자신의 오른쪽 형제 엘리먼트의 위치
Eltype		Schema에서의 element 타입을 정의 Complex Sequence : CS , Complex All : CA Complex Choise : CC , Group Sequence : GS SimpleType : S
Rep		엘리먼트의 재사용 횟수를 기억
Ref		Eltype의 재사용을 위한 참조 값을 기억

Table 5의 필드들 중 LevelInfo 필드는 3부분으로 나뉘어, 실제 Element에 대한 위치정보를 나타내게 된다. 이 정보는 자신의 상위를 나타내는 Root, 왼쪽 형제를 나타내는 Left, 오른쪽 형제를 나타내는 Right로 세분화되어 있다. Eltype 필드는 Element 정의 타입형식으로 스키마에서 주어지는 형식을 CLOB(Charactor Large OBject)기법을 이용하여 축약시켜 가상 분할 저장하였다. Rep 필드는 엘리먼트의 반복 횟수를 지정할 수 있도록 하였으며, 마지막으로 Ref 필드는 문서내의 Element 정의형식에 따른 참조를 나타내는 값으로 구성된다.

3. Database Schema Modeling

제안 구조에 따른 모델링을 위한 XML 문서는 Table 6와 같이 크게 두 개의 예제 문서로 구분된다. 서로 다른 각각의 예제 문서를 Fig. 7에서의 제안 구조를 통해 저장해 보면 Table 7과 같은 구조로 RDB에 모델링 될 수 있다.

Table 6. Examples of XML document

	Example 1
XML schema	〈?xml version="1.0" ?〉 〈xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"〉 〈xs:element name="Order" maxOccurs="unbounded"〉 〈xs:complexType〉 〈xs:sequence〉 〈xs:element name="Name" type="xs:string"/〉 〈xs:element name="Su" type="xs:string"/〉 〈/xs:sequence〉 〈/xs:compleType〉 〈/xs:element〉 〈/xs:schema〉
Instance	〈?xml version="1.0" ?〉 〈Order〉 〈Name〉DB book〈/Name〉 〈Su〉3〈/Su〉 〈/Order〉
	Example 2
XML schema	〈?xml version="1.0" ?〉 〈xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"〉 〈xs:complexType name="Namec"〉 〈xs:sequence〉 〈xs:element name="FirstName" type="xs:string" /〉 〈xs:element name="Middle" type="xs:string" /〉 〈xs:element name="LastName" type="xs:string" /〉 〈/xs:sequence〉 〈xs:attribute name="customerID" type="integer" /〉 〈/xs:complexType〉 〈xs:element name="Customer" type="NameC" maxOccurs="3"〉 〈/xs:schema〉
Instance	〈?xml version="1.0" ?〉 〈Customer customerID="24332"〉 〈FirstName〉Ray〈/FirstName〉 〈MiddleInitial〉G〈/MiddleInitial〉 〈LastName〉Bay〈/LastName〉 〈/Customer〉

Table 7. Modeling table

SIDCHK	ID/SID	LevelInfo	Element	Attribute	Entity	Eltype	Rep	Namespace	Ref
S	0/1	0/0/0	xml	version="1.0"				xs / http://....	
S	1/1	0/0/0	element	name="Order"		CS	*	xs / http://....	
S	2/1	1/0/3	element	name="Name" type="xs:string"				xs / http://....	
S	3/1	1/2/0	element	name="Su" type="xs:string"				xs / http://....	
				... ...					
I	0/1	0/0/0	xml	version="1.0"					
I	1/1	0/0/0	Order				1		
I	2/1	1/0/3	Name		DB book				
I	3/1	1/2/0	Su		3				
				... ...					
S	0/2	0/0/0	xml	version="1.0"				xs / http://....	
S	1/2	0/0/0	element	name="Customer"		CS	3	xs / http://....	Namec
S	2/2	1/0/3	element	name="FirstName" type="xs:string"				xs / http://....	
S	3/2	1/2/4	element	name="Middle" type="xs:string"				xs / http://....	
				... ...					
S	1A/2	0/0/0	attribute	name="customerID" type="xs:integer"				xs / http://....	
I	0/2	0/0/0	xml	version="1.0"					
I	1/2	0/0/0	Customer	CustomerID="24332"			1		
I	2/2	1/0/3	FirstName		Ray				
I	3/2	1/2/4	Middle		G				
				... ...					

제안 구조에 따른 모델링은 SIDCHK 필드를 통한 통합 테이블 저장 방식을 나타내고 있으나, SIDCHK 필드를 사용하지 않고 스키마와 인스턴스를 따로 분리하여 저장하여도 실제 저장공간과 검색시간에 따른 현저한 차이는 나타나지 않았으며, 이에 따라 본 논문에서는 통합 테이블 저장 방식으로 인한 모델링을 나타내었다.

Ⅳ. 구현 결과 및 성능 평가

1. 사용자 인터페이스 구현

사용자 인터페이스는 Fig. 8에 보여지는 것과 같이 크게, XML 문서를 입력하는 파트와 실제 RDB에 모델링된 RDB 내용을 보여주는 파트, 마지막으로 문서 내용에 따른 검색 파트로 나뉘어진다.

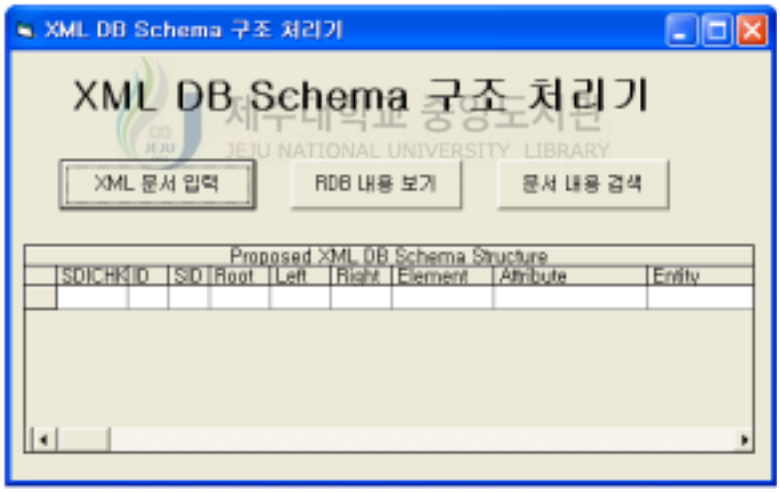


Fig. 8 User Interface

1) XML Document Input Part

XML 문서 입력 파트에서는 스키마와 DTD 그리고 XML Document를 확장자를 통해 구분하여 입력을 받는다. Mapping Layer의 분석기는 입력 받은 파일을 분석하여 그 결과를 RDB에 저장을 하게 된다. 만약, 문서가 저장된 문서형식에 유효하지 않은 문서가 입력되었을 때에는 RDB에 저장을 하지 않

고 걸러내게 된다. Fig. 9는 실제 XML 문서를 입력받는 과정을 나타내며, Fig. 10은 Not Valid 한 문서가 입력되었을 때 경고를 보여주고 있다.

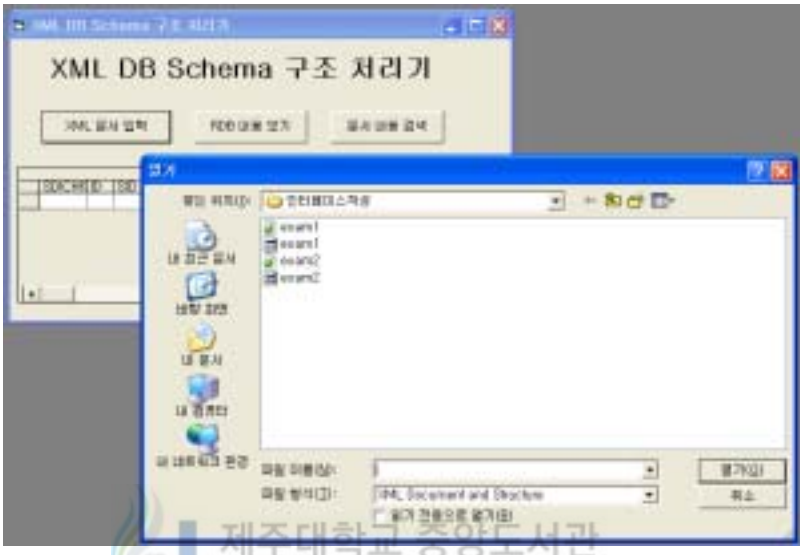


Fig. 9 XML Document Input

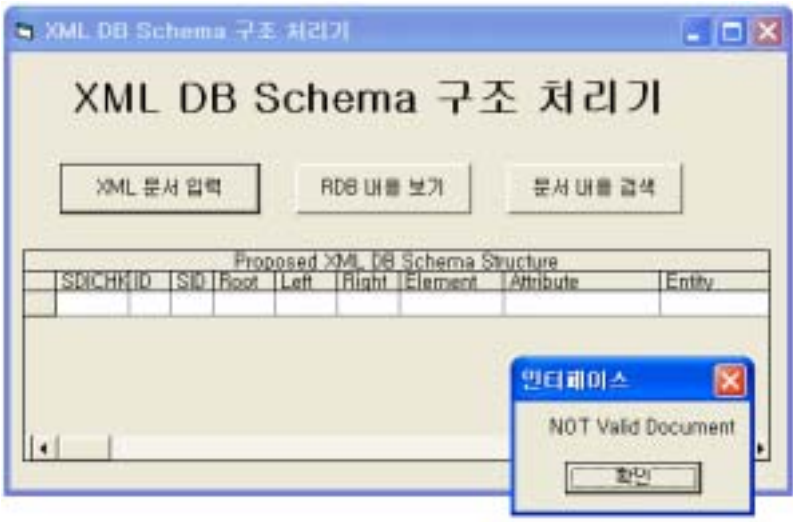


Fig. 10 XML Document Input (Not valid)

2) RDB Contents View Part

ODBC를 통하여 Database Schema의 내용을 출력하는 부분이다. Fig. 11은 실제 RDB를 연결하여 내용을 보여주는 것을 나타내고 있다.

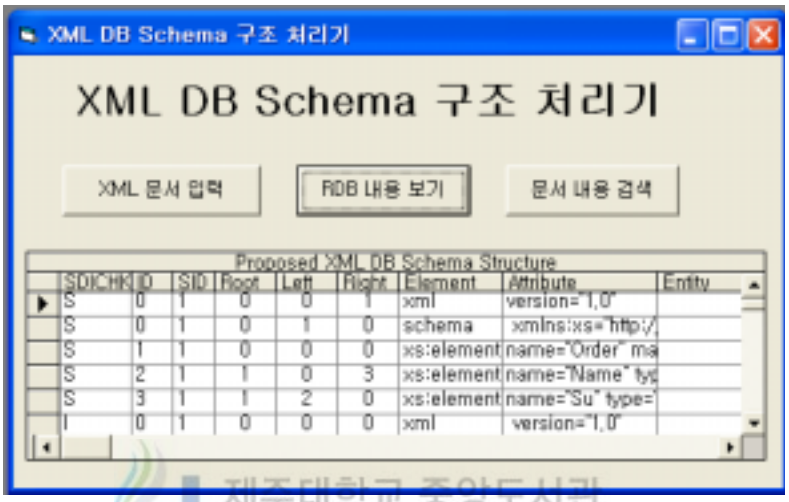


Fig. 11 RDB Contents View Part

3) RDB Contents Search Part

RDB에 저장된 데이터를 검색하는 부분이다. 데이터를 검색하는 방법은 컨텐츠 값에 의한 데이터 검색, 계층 정보에 의한 데이터 검색, ID 값에 의한 데이터 검색 등 3가지로 나뉘어 진다.

(1) 컨텐츠 값에 의한 데이터 검색

데이터베이스 안에 저장된 ENTITY 필드를 기준으로 실제 XML 문서의 컨텐츠 값을 직접 검색하는 방법이다. 이는 단어위주의 데이터 검색 및 "*"를 사용하여 모르는 값에 대한 검색범위를 지원하고 있다.

(2) 계층 정보에 의한 데이터 검색

계층 정보에 의한 데이터 검색은 XML 문서에 따른 구조를 이용한

검색 방법이다. 이는 데이터간의 구조를 이용하여 종속적 관계 및 수평적 관계를 계층 구조를 통해 해당 데이터를 유추할 수 있는 장점이 있다.

(3) ID 값에 의한 데이터 검색

해당 데이터의 엘리먼트 ID에 의한 검색방법이다. 이는 엘리먼트 단위의 순차적인 ID 값을 이용하여, 해당 데이터의 엘리먼트 위치를 검색할 수 있다.

2. 성능 평가

성능 평가를 위한 검증데이터는 두 그룹으로 나뉘어 각각 10회 실시하였으며, 이에 따른 저장 공간 및 저장에 따른 시간, 그리고 검색에 따른 시간을 평가하였다.

제안 시스템 검증을 위한 데이터로 학생들에 대한 이력서를 토대로 작성된 XML데이터를 사용하였다. 학생들에 대한 이력데이터는 1명의 이력만 가지고 있는 데이터(A)와 40명의 이력을 가지고 있는 데이터(B)로 나뉘어 구성되었다. 이력 데이터를 따르는 스키마는 Fig. 12, 인스턴스는 Fig. 13과 같다.

1) RDB 저장 공간 및 저장에 따른 시간 측정

데이터 (A)와 데이터 (B)에 따른 RDB 저장 공간 및 저장에 따른 시간은 기존 시스템과 제안시스템에서 각각 Table 8과 Table 9과 같은 결과를 얻을 수 있었다.

```
<?xml version="1.0" encoding="euc-kr" ?>
<schema xmlns="http://www.w3.org/2001/XMLSchema">
  <element name="이력서">
    <complexType>
        <sequence>
          <element name="이름" type="string" />
          <element name="주민번호" type="string" />
          <element name="생년월일" type="date" />
          <element name="주소" type="string" />
          <element name="연락처" type="string" />
          <element name="학력사항">
            <element name="사항" type="string" minOccurs="1" maxOccurs="unbounded">
          <complexType>
                    <sequence>
                          <element name="날짜" type="string" />
                          <element name="내용" type="string" />
                        </sequence>
                  </complexType>
                </element>
          </element>
                  ...
  </element>
 </schema>
```

Fig. 12 Schema document


```
<?xml version="1.0" encoding="euc-kr" ?>
- <이력서>
  <이름>김미나</이름>
  <주민번호>850111-xxxxxxx</주민번호>
  <생년월일>1985년 1월 11일(만 19세)</생년월일>
  <주소>제주도 서귀포시 서호동</주소>
  <연락처>010-7179-xxxx</연락처>
- <학력사항>
- <사항>
    <날짜>2000년 3월 3일</날짜>
    <내용>삼성여자고등학교 입학</내용>
    <발행처 />
  </사항>
- <사항>
    <날짜>2003년 2월 7일</날짜>
    <내용>삼성여자고등학교 졸업</내용>
    <발행처 />
  </사항>
- <사항>
    <날짜>2003년3월3일</날짜>
    <내용>제주한라대학 정보통신계열 입학</내용>
    <발행처 />
  </사항>
- <사항>
    <날짜>2005년2월15일</날짜>
    <내용>제주한라대학 정보통신계열 졸업예정</내용>
    <발행처 />
  </사항>
 </학력사항>
- <경력사항>
- <사항>
    <날짜>2004년 09년 29일</날짜>
    <내용>유리스 1개월째</내용>
    <발행처>산학협력체</발행처>
  </사항>
 </경력사항>
</이력서>
```

Fig. 13 Instance document in group A

(1) 기존 시스템

Table 8. Result storage in existing system

	데이터 (A)		데이터 (B)	
	스키마	인스턴스	스키마	인스턴스
저장공간	21 Rec	20 Rec	24 Rec	800 Rec
저장시간	0.923 Sec	0.922 Sec	0.923 Sec	1.682 Sec

(2) 제안 시스템

Table 9. Result storage in proposed system

	데이터 (A)		데이터 (B)	
	스키마	인스턴스	스키마	인스턴스
저장공간	14 Rec	20 Rec	15 Rec	781 Rec
저장시간	0.906 Sec	0.922 Sec	0.907 Sec	1.641 Sec

위에서 나타낸 바와 같이, 일정 데이터의 저장 공간과 저장에 따른 시간을 비교해 보면, 저장 공간에서 특히 스키마에 효율적으로 동작함을 알 수 있다. 이는 중복 정의되는 스키마의 정의형식을 Eltype 필드를 이용해 데이터를 축약시켜 저장하였기 때문이다. 실제 다양하고 복잡한 형태의 스키마 구조를 가진 실 전자상거래의 문서들은 제안 시스템에 의해 저장 공간이 많이 줄어들 것이다. 인스턴스상의 저장 공간에 큰 차이가 없는 것은, 인스턴스의 모든 데이터가 정보로 활용되고 있기 때문이다. 그리고, 저장 시간은 XML 문서가 나뉘어진 상태로 DB Schema에 저장되는 시간을 측정한 것이므로, 저장 시간의 차이는 나뉘어진 데이터양에 의한 차이에 따라 차이가 나타나는 것이며, 일반적으로 ODBC를 이용한 RDB 접근 시간이 저장 시간에 약 0.5초 정도 유지되는 것으로 검증되었다. Fig. 14는 스키마와 인스턴스 데이터에 따른 저장 공간의 차이를 그래프로 나타내고 있다.

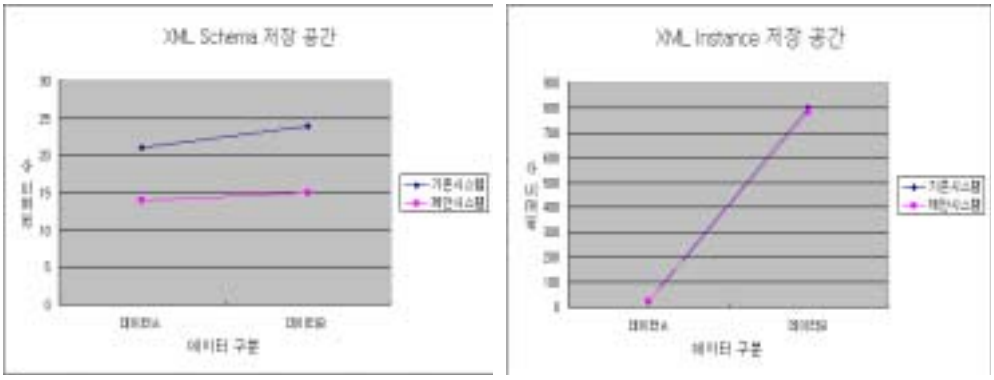


Fig. 14 데이터에 따른 저장 공간

2) 검색에 따른 시간 측정

Table 10. Searching conditional

검 색 유 형	검 색 조 건
컨텐츠 값에 의한 검색	1) 김* or 홍* 2) 홍길동
계층 정보에 의한 검색	1) 2/0/4
ID에 의한 검색	1) 9* 2) 9

검색에 따른 시간 측정은 데이터 (A)와 데이터 (B)에 각각 Table 9와 같은 검색 유형과 검색 조건을 가지고 측정하였다. 검색 조건에 따른 측정을 10회 반복하여 나온 결과수치에 대한 평균을 기존 시스템과 제안 시스템에 따라 알아보았다.

(1) 기존 시스템

기존 시스템에서는 XML 스키마와 인스턴스간의 계층 구조가 서로 상이하여 계층 정보에 의한 검색을 지원하지 않았다. 검색에 의한 산출 시간은 데

이터 컨텐츠에 의한 검색 시간보다 ID 검색에 따른 검색 시간이 상대적으로 빠른 결과를 나타내었으며, 이러한 결과는 이미 정렬 되어진 ID 값을 이용한 탐색시간의 차이에 따른 것이라 할 수 있다. Table 11은 기존 시스템에서의 검색 결과를 나타낸다.

Table 11. Result search in existing system

검색 유형	데이터 (A)	데이터 (B)
컨텐츠 검색	0.104 Sec	0.127 Sec
	0.098 Sec	0.119 Sec
계층정보 검색	-	-
ID 검색	0.084 Sec	0.085 Sec
	0.092 Sec	0.098 Sec

(2) 제안 시스템

제안 시스템에서의 Table 10의 검색 유형을 참고로 질의를 처리한 결과 Table 12과 같은 결과를 얻을 수 있었다. 기존 시스템에서와 마찬가지로 컨텐츠 검색 보다 ID검색의 유형에서 검색시간이 빠르게 나타남을 확인할 수 있으며, 기존 시스템보다 모든 검색유형에서 조금 향상된 결과를 나타내었다. 이러한 결과는 인스턴스 저장 공간의 감소로 인한 레코드 대비 탐색시간의 축소, 그리고 여러 테이블에 걸친 조인 연산이 아닌 하나의 독립 테이블에서의 질의 처리로 인한 조인 연산의 감소에 의한 것이라 할 수 있다.

Table 12. Result search in proposed system

검색 유형	데이터 (A)	데이터 (B)
컨텐츠 검색	0.102 Sec	0.116 Sec
	0.095 Sec	0.109 Sec
계층정보 검색	0.086 Sec	0.088 Sec
ID 검색	0.084 Sec	0.084 Sec
	0.091 Sec	0.096 Sec

Fig. 15은 검색 유형에 따른 그래프를 나타내고 있다.

여기서 데이터 A, B 라고 하는 것은 검증 데이터 A와 B를 의미하며, 1, 2라고 하는 것은 Table 10에서 검색유형에 따른 검색 조건 1)과 2) 를 나타낸다.

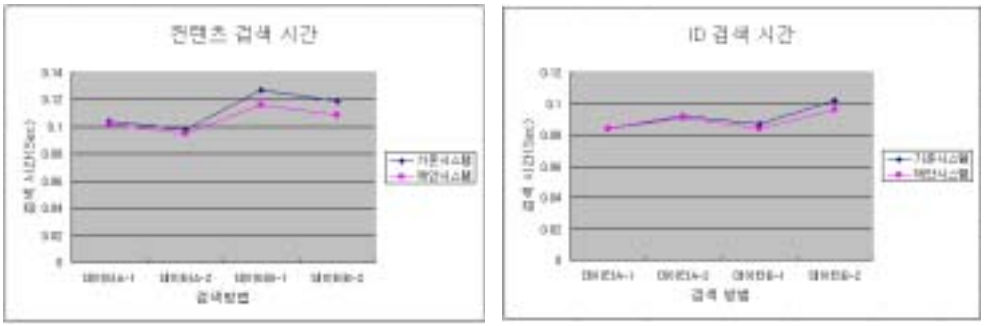


Fig. 15 검색 유형에 따른 검색 시간

3) 측정 결과에 따른 평가

데이터 (A)와 데이터 (B)에 따른 RDB 저장 공간 및 검색 시간에 대한 효율을 나타내면 Fig. 16와 Fig. 17와 같다.

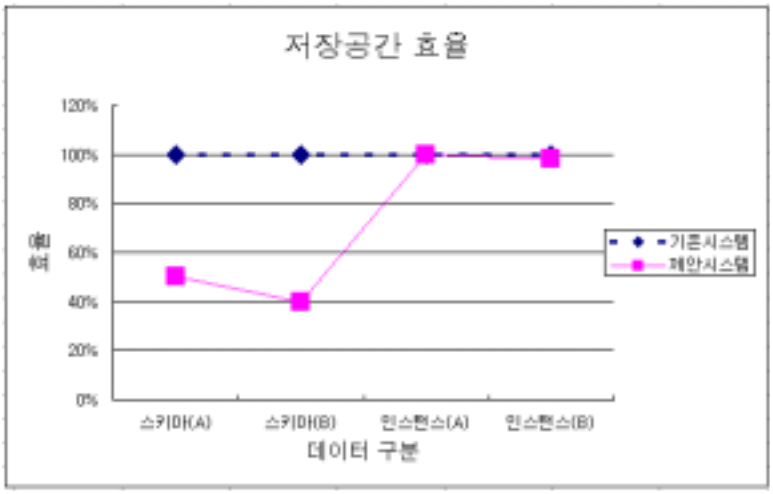


Fig. 16 데이터에 따른 저장 공간 효율

Fig. 16에서 보듯이, 데이터 저장 공간에 대한 효율은 기존시스템을 100%의 효율이라고 가정했을 때, 스키마에 대한 데이터(A) 는 50 %, 데이터 (B)는 60%에 가까운 감소를 나타내었다. 이는 스키마 구조가 복잡 할수록 저장 효율이 좋아짐을 나타낸다. 그러나 인스턴스에 대해서는 기존시스템과 제안시스템간의 차이가 거의 나타내지 않는다.

Fig. 17 검색 유형에 따른 검색 효율

Fig. 17에서 보듯이, 데이터 검색유형에 따른 검색 효율은 기존 시스템을 100%의 효율로 가정했을 때, 모든 검색 유형에 대해, 기존 시스템보다 나은 효율을 보였으며, 데이터 (A) 보다 데이터 (B)가 효율이 더 좋은 것은 데이터의 양이 증가 할수록 검색 시간 효율이 낫다는 것을 나타낸다. 이것은 기존 시스템에 비해 데이터의 양이 줄고, 여러 테이블에서의 조인 검색 시간 감소에 따른 제안 시스템의 효율 증가를 나타낸다.

또한, 데이터의 계층정보를 알고 있는 경우는, 일반적으로 데이터의 양에 관계없이 거의 비슷한 검색 속도를 가져옴으로써, 계층정보를 활용한 데이터 검색의 효율성도 기존 시스템의 검색방법들과는 대비되는 새로운 검색방법으로 활용이 가능하다.

Ⅴ. 결론 및 향후 연구

XML 문서를 관계형 데이터베이스에 저장하기 위한 기존 모델링 방법은 XML 정의형식인 DTD와 인스턴스를 분리 저장하였고, 또한, 가상 또는 분할 저장방식 기반위에 데이터베이스 스키마는 여러 구조로 연구되어 왔다.

이러한 저장방법은 DTD를 따르는 인스턴스의 엘리먼트를 기준으로 ID를 부여하여 데이터베이스 스키마에 저장함으로써, 엘리먼트 정의형식에 따른 중복저장 비율의 증가로 인한 구조정보의 오버헤드가 높아지게 되었고, XML 정의형식과 인스턴스간의 계층정보가 불일치되어 검색 시 질의 처리를 어렵게 했다.

그러나, 본 연구에서는 ETID 모델을 이용하여 XML 기반의 계층적 RDB 스키마 모델을 제안하여, 기존의 스키마 모델 적용에 따른 상이한 계층정보를 동일화하고, 기존의 분리 저장방식을 통합할 수 있도록 관계형 데이터베이스 스키마를 개선하고, XML 정의형식으로 DTD의 제한사항을 극복한 XML 스키마를 적용한 저장 시스템을 제안 및 구현하였다. 그 결과 제안 시스템에서는 ETID 모델을 이용하여 해당 엘리먼트 정보의 위치를 계층적으로 표현하고, 기존 관계형 데이터베이스 스키마 구조의 개선을 통해 XML 스키마와 인스턴스간 동일한 계층정보를 유지하게 되었으며, 저장 공간 활용 및 비율에 따른 효율성이 높아졌고, 계층구조 상이에 따른 손상을 최소한으로 유지하게 되어 별도의 연산시간 없이 문서간의 검색질의가 수월하게 되었다. 그러나, 관계형 데이터베이스 내에 저장되어 있는 XML 문서에 대한 데이터에 대한 갱신이 발생하였을 때, 제안 시스템 내에서의 갱신처리가 불가능하다.

향후 연구 과제로는 이미 저장되어 있는 관계형 데이터베이스 내의 데이터 갱신에 따른 처리에 대한 연구가 필요하다.

〔참고문헌〕

[1] Tim Bray, Jean Paoli et. al (ed.). Extensible Markup Language(XML) 1.0(Third Edition), World Wide Web Consortium(W3C), 2004. (http://www.w3c.org/TR/2004/REC-xml-20040204)

[2] 연제원, 조정수, 이강찬, 이규철, "XML 문서 구조검색을 위한 저장 시스템 설계",『한국정보과학회 학술 발표논문집(B)』, 제26권 제1호, 1999, pp. 3-5.

[3] D. W. Shin, "BUS:An Effective Indexing and Retrieval Schema in Structured Documents", in Proc. Digitial Libraries, 1998.

[4] Brian Lowe, Justin Zobel, Ron Sacks-Davis, "A Formal model for Databases of Structured Text", Proceedings of the Fourth International Conference on Database Systems for Advanced Applications(DASFAA '95), 1995, pp. 449-456.

[5] Toung Dao, "An Indexing Model for Structured Document to Support Querues on Content, Structured and Attributes", Proceedings of ADL '98, 1998, pp. 88-97.

[6] 박종관 외, "XML 문서의 효율적인 구조 검색을 위한 색인 모델",『한국정보과학회 논문지』, 제 8-D권, 2001, pp451-460.

[7] 홍석건 외 2명, "검색과 저장의 효율성을 위한 정적 테이블 기반의 XML 저장구조 설계",『 한국정보과학회 학술발표논문집(B)』, 제30권 2호, 2003, pp205-207.

[8] Daniela Florescu and Donald Kossmann, "Storing and Querying XML Data using an RDBMS", IEEE Data Engineering Bulletin, 22(3), PP 27-34, 1999

[9] E. Damiani, S. Vimercati, S. Paraboachk and p. Samarati, "XML Access Control System: A Component-Based Approach", In Proc. IFIP WG11.3 Working Conference on Database security, The Netherlands, 2000. 8

[10] 정태선 외 3명 "XML 데이터를 위한 객체지향 데이터베이스 스키마 및 질의 처리", 한국정보과학회 논문지 : 데이터베이스, 29(2), 2002

[11] J. Shanmugasundaram, H. Gang, K. Tufte, C. Zhang, D. DeWitt and J. Naughton, "Relational databases for Querying XML Documents : Limitations and Opportunities", 25th VLDB Conference, Edinburgh, Scotland, 1999

[12] S-Y. Chien, V.J Tsotras, and C. Zaniolo, "Version Management of XML Documents", WebDB 2000 Workshop, Dallas. TX, 2000

[13] Albrecht Schmidt, Florian Waas, Martin Kersten, Michael J. Carey, Ioana Manolescu and Ralph Busse, "Xmark : A Benchmark for XML Data Management", Proc. VLDB, Hong Kong, China, 2002

[14] 민영수 외 5명, "XML 문서를 위한 구조정보 추출기의 설계 및 구현", 한국정보과학회, '99 가을 학술발표논문집(Ⅰ), pp, 81-83, 1999

[15] A. Gabillon and E. Bruno, "Regulation Access to XML Documents", In Proc. IFIP WG11.3 Working Conference on Database Security, 2001

감사의 글

정신없이 지내온 2년이라는 대학원 생활, 길지도 짧지도 않은 그 시간 이었지만, 무엇보다도 제겐 소중한 시간들이었던 것 같습니다.

항상 부족한 것이 많은 저에게 보살핌과 가르침을 주셨던 지도교수님이신 곽호영 교수님이 계셨기에 무사히 졸업을 할 수 있게 되었습니다. 그래서, 지도교수님께 감사의 말씀을 다시 한번 올리지 않을 수 없었습니다.
교수님 감사드립니다.

그리고 이 논문을 위해 심사위원으로 많은 심사와 지도를 위한 열정을 아끼지 않으셨던, 안기중 교수님, 김도현 교수님께도 감사의 말씀을 드립니다. 또한, 김장형 교수님, 변상용 교수님, 이상준 교수님, 송왕철 교수님 그리고 변영철 교수님께도 보다 학술적인 접근을 바라는 말씀과 충고를 아끼지 않아 주셨음에 감사드립니다.

이 논문을 완성하기까지 수많은 분들로부터 도움을 많이 받았습니다. 우선, 같은 연구실에서 함께 공부하며, 생활한 이들이 있었기에 정말 소중한 열정을 쏟아낼 수 있었던 것 같습니다. 언제나 부족한 부분을 채워주고, 따끔한 충고도 서슴치 않았던 정희 선배와 경복 선배, 또한 많은 시간을 같이 하며, 조언을 하여 줬던 인석 선배, 석건. 더 나은 미래를 위해 고민하던 봉남. 그리고, 성철 형님과 혁준 선배에게도 이 지면을 빌어 감사의 마음을 전하고자 합니다.

그리고, 연구실은 다르지만 옆에서 힘이 되어준, 송재경, 윤현주, 양동호, 변태보, 허지완 선배, 경진, 정윤, 그리고, 이정하, 정은경 조교 선생님, 제

주산업정보대학 컴퓨터 정보계열 교수님들, 한라대학교 인터넷 전자상거래과 교수님들과 양은정 조교 선생님에게도 그동안의 배려와 격려에 대해 감사드립니다. 또한, 항상 만나면 편한 "피사모"의 회원님들, 김경표 원장님 이하 강용익, 한정헌, 김경미, 오주연 선생님, 연합서클 제로하나 선배, 동기, 후배들과 고정철, 이혜선, 강길봉 선생님에게도 고맙다는 말을 전하고 싶습니다.

마지막으로, 늘 묵묵히 옆에서 지켜봐 주신 아버님, 어머님, 그리고 말을 굳이 하지 않더라도 든든한 힘이 되어주는 형님, 그리고 형수님, 조카들. 이 모든 이들의 관심과 격려, 그리고 배려가 있었기에 결실을 맺게 되었습니다. 저를 아는 모든 분들이 항상 건강하고, 행복한 일만 가득하길 바랍니다.

2005年 6月 연구실에서

권 훈