공학석사 학위논문

이력데이터를 이용한 효율적인 XML문서 버전관리 기법에 관한 연구

지도교수 조 우 현

이 論文을 工學碩士學位論文으로 提出함

2005년 2월

부 경 대 학 교 대 학 원

컴 퓨 터 공 학 과

김 성 록

김성록의 공학석사 학위논문을 인준함

2004년 12월

주 심 공학박사 서 경 룡 (인)

위 원 공학박사 송 하 주 (인)

위 원 공학박사 조 우 현 (인)

목 차

그 림 목 차

표 목 차

이력데이터를 이용한 효율적인 XML문서 버전관리 기법에 관한 연구

김 성 록

부 경 대 학 교 대 학 원 컴 퓨 터 공 학 과

요 약

최근 웹상에서 반구조화된 정보와 XML 문서들의 대중성으로 인해 효율적인 저장을 보증하는 문제는 중요하다. 버전 차이를 표현하기 위해 최단 편집 스크립트를 사용하고 텍스트라인 순서에 따라 문서들을 모델화 하는 RCS와 같은 전통적인 문서 버전 관리 시스템들은 원래 문서의 이력 데이터와 논리적인 구조를 보존하지 않기 때문에 XML 버전 관리 시스템으로 부적당하다. 본 논문에서는 XML 문서내의 각 element들에 대한 버전을 효율적으로 유지하기 위해 새로운 기법을 제안한다. XML 문서의 이력 데이터를 이용하여 새로운 버전 번호들을 생성하고 이것을 XML 문서의 버전닝과 재구성을 위해 활용한다. 또한 XML 문서 구조정보를 얻기 위해 dewey ordering 기법을 사용한다. 마지막으로 다양한 형태로 지원되는 질의들을 기술한다.

A Study on Effective Version Management of XML Document Using Historical Data

Seong Rok Kim

Dept. of Computer Eng., Graduate School,
Pukyong National University

Abstract

The problem of ensuring efficient storage and fast retrieval for version structured document is important because of the recent popularity of XML documents and semi-structured information on the web. Traditional document version control systems, e.g. RCS, which model a document as a sequence of lines of text and use the shortest edit script to represent version difference are inadequate since they do not preserve the logical structure and historical data of the original document. In this paper, we propose a new approach to efficiently keep the versions of each elements in an XML document. Using historical data of XML documents, new version numbers are created and utilized for reconstructing and versioning of XML documents. We also use the approach of Dewey ordering. Finally, we describe various types of queries to be supported.

1. 서론

인터넷 기술의 급속한 보급과 발전은 정보기술의 패러다임을 변화시키고 있다. 오늘날 많은 정보시스템들은 웹을 기반으로 구축되어 활용되고 있으며 다양한 전자 문서들을 이것을 통해 제공하고 있다. 이러한 변화의 중심에는 웹을 기반으로 XML (eXtensible Markup Language)기술을 이용하는 다양한 웹서비스(Web-Service)들이 있다. 웹서비스는 자기 기술적이며 퍼블리싱이 가능하며, 웹이나 개방 인터넷 표준을 기반으로 하는 로컬 네트워크라면 어떤 곳에서도 위치 할 수 있다. 즉, 단순 객체 접근 프로토콜(SOAP), 웹서비스 기술 언어(WSDL), 범용 기술 탐색 및 통합(UDDI) 등 XML 관련 공개 표준을 사용해 다양한 모습으로 웹에 산재해 있는 개별 웹 애플리케이션을 효율적으로 통합할 수 있는 방안을 제공하는 일련의 과정을 말한다. XML은 W3C(World Wide Web Consortium)에서 표준으로 제정, 발표한 대표적인 전자 문서 표준으로서 플랫폼에 독립적이고 문서가 HTML이나 SGML 문서처럼, 구조적인 태그를 사용해서 텍스트 문서를 표시하기 위한 메타언어이다. XML이 유행하게 된 것은 구조적인 마크업으로 문서의 포맷과 내용을 설명할 수 있기 때문이다. 이것은 '이식성 있는 데이터'를 표현할 수 있고, 자바의 '이식성 있는 코드'와 결합하면 막강한 힘을 발휘한다. 또한 문서가 가지는 정보를 손실 없이 전송 및 교환 하는 것이 용이 하고, 문서의 의미를 있는 그대로 나타낼 수 있는 언어이다. 문서의 구조와 내용을 구분해 관리함으로서 다양한 문서를 체계적으로 다룰 수 있고, 기존의 파일 형태의 정보에 비하여 의미적 정보의 단위를 구조로 표현 가능하며 이러한

구조 정보를 사용해서 문서의 관리 및 저장, 검색에 이용할 수 있는 장점이 있다. 또한 XML문서로 저장되고 교환되는 데이터양이 증가함에 따라 XML 문서에 대한 효율적 관리와 효율적 버전관리에 대한 많은 연구들이 이루어지고 있다. XML 버전관리 시스템은 지속적인 버전관리를 할 수 있는 기능을 제공하여야한다. XML 문서에 표현된 모든 정보는 시간이 경과됨에 따라 내용의 추가, 변경, 삭제 등으로 인해 계속적으로 변화하게 된다. 이렇게 계속적으로 변하는 XML문서의 각 엘리먼트들에 대한 유지관리가 필요하다. 따라서 본 논문에서는 버전을 부여하고 저장 관리함에 있어서 변경된 최신의 문서에 대한 정보뿐 아니라 변경 이전의 이력까지 검색이 가능하게 하는 새로운 버전관리 방법을 제안한다. 또한 제안된 방법에서는 사용자의 질의 유형에 따라 효과적으로 XML문서를 재구성하거나 특정 엘리먼트의 버전 변화 이력에 대한 정보까지도 제공해 준다.[1][2][10][14]

본 논문은 다음과 같이 구성된다. 2장에서는 관련연구를 중심으로 기술하고 3장에서는 제안한 버전관리 시스템에 대한 설명을 한다. 4장에서는 결론 및 향후 연구에 대해 기술한다.

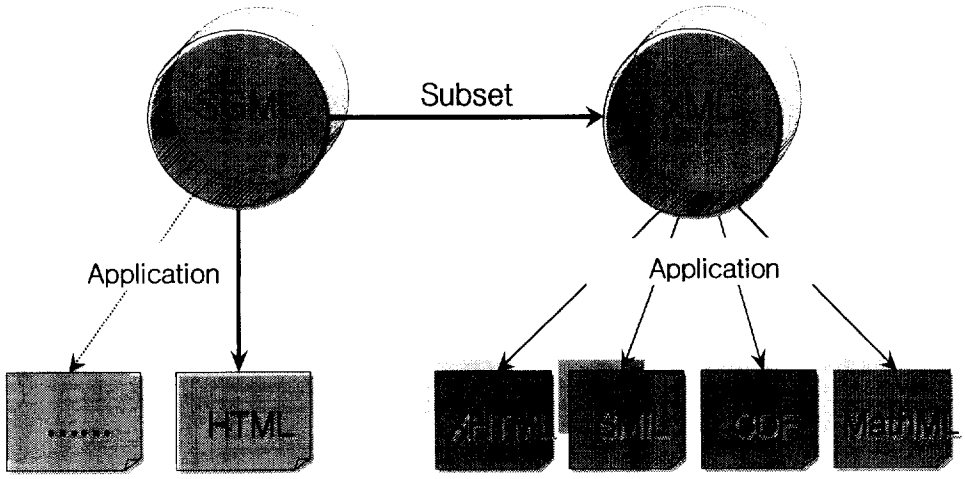
2. 관련 연구 및 기존 연구 소개

2.1 XML

XML은 1996년 W3C(World Wide Web Consortium)에서 제안한 것으로서, 웹상에서 구조화된 문서를 전송 가능하도록 설계된 표준화된 텍스트 형식이다. XML은 HTML과 SGML의 장점을 유지하면서 단점을 극복하는 Markup Language이다. XML은 SGML의 장점인 각 소프트웨어와 하드웨어에 독립적으로 어떠한 시스템간에도 상호교환과 공유가 가능하고 문서의 구조를 저장할 수 있기 때문에 문서 구조를 기반으로 한 다양한 응용(검색, 저장 등)에 사용할 수 있으며, 또 접근방식이 다양하여 효과적인 정보교환 방법을 받아들이고 거의 사용되지 않는 기능을 없애는 등 중요한 많은 기능을 그대로 두고 있다. 따라서 XML을 SGML의 부분집합(subset)이라고도 한다. 그러므로 SGML은 XML로 변환이 용이하고 XML을 모두 수정 없이 SGML의 응용에 사용할 수 있다는 장점이 있다. [3][4]

XML은 새로운 태그 세트와 속성을 정의 할 수 있는 확장성을 가지므로 웹 제작자들은 자신의 편의에 따라 혹은 자신의 데이터를 구분하기 위해 새로운 태그세트를 임으로 만들 수 있다. 그 예로 XML로 만들어진 웹문서는 동적목차(Dynamic Table of Contents)를 적용할 수 있어 HTML문서에 비해 정보를 카테고리별로 구분할 수 있으며 XML로 구축한 웹페이지는 인터넷 이용자들에게 정보를 정렬하거나 정보검색 작업을 손쉽게 해준다. 또한 XML은 HTML이 지원하지 않는 여러번의 중첩된 구조를 지원하기 때문에

객체 지향적인 문서 작업을 할 수 있다. 따라서 XML은 월드와일드웹, 인트라넷 등에서 데이터와 포맷 두가지 모두를 공유하려는 유용한 방법이며 현재 W3C로부터 웹을 좀 더 다양한 목적으로 이용하기 위한 도구로서 공식 추천되고 있다.[13]



(그림 1) SGML/HTML/XML 관계

2.2 XML 파서

XML파서는 XML문서가 XML의 문법에 맞게 작성되었는지 검증하는 기능을 제공한다. 그리고 XML문서를 조작할 수 있도록 API(Application Programming Interface)를 제공한다. 대표적인 파서 프로그램들을 살펴보면 다음과 같다.

◎ Oracle XML Developer's Kit

오라클 XML 파서는 Oracle XDK(XML Developer's Kits)의 일부분으로 존재한다. 그리고 Java, C/C++,PL/SQL별로 버전이 존재하여 각 언어로 작

성된 응용 프로그램에서 사용될 수 있다. 이 파서들은 DOM(Document Object Model) API와 SAX(Simple API for XML)API, XML namespaces를 지원하며 유효성을 검증하는 모드와 검증하지 않는 모드가 존재한다. 또한 XSL(Extensible Stylesheet Language)변환도 지원한다.

◎ XML Parser for java (XML4J)

IBM 사에서 제작한 XML Parser for Java는 100% 순수 java로 작성된 XML파서이다. 따라서 XML문서에 대한 유효성 검증과 XML문서를 파싱하고 생성 및 조작하며, 유효성을 검증하는데 사용하는 클래스와 메소드를 포함하고 있다.

◎ XML Mate

XML Mate는 Insight Technologies사의 제품으로 단지 사용자가 정의할 수 있는 프리젠테이션 엔진이 특징인 XML 유틸리티이다. 또한 XML Mate는 강력한 XML 에디터를 제공하고 개발자는 XML과 XML에 관련된 기술들인 XSL, XPath, XQL(XML Query Language)등의 장점을 취할 수 있다. XML Mate는 XML개발 공정을 단순화하며 다양한 데이터 소스를 XML로 변환함으로서 개발 시간을 단축 시켜준다.

◎ Xerces-J

Xerces-J는 아파치 그룹에서 제작한 Java로 구현된 XML파서이다. 이 파서는 XML 1.0 뿐만 아니라. DOM Level1, SAX version 1과 XML Schema, DM Level2 version 1.0, SAX version 2들의 확장된 기능도 지원

한다.

◎ JAXP (Java API for XML Processing)

JAXP는 DOM과 SAX를 사용하여 XML문서의 처리를 지원한다. JAXP를 사용하면 애플리케이션에서 특정 XML처리 구현과는 상관없이 XML 문서의 구문을 분석하고 XML문서를 변환할 수 있다. 애플리케이션 요건에 따라, 개발자는 애플리케이션 코드를 변경하지 않고도 XML 프로세서간에 교체 할 수 있는 유통성을 가지고 있다. 따라서 애플리케이션 및 도구 개발자는 전자 상거래, 애플리케이션 통합 및 동적 웹 출판용 JAVA 애플리케이션과 기타 애플리케이션에서 신속하고 쉽게 XML을 사용할 수 있도록 한다.[6][7]

2.3 SAX (Simple API for XML)

SAX는 사실상 W3C에서 만들어 졌다기 보다는 XML-DEV 메일링 리스트의 공동 프로젝트를 통해서 만들어진 상품이다. SAX는 노드의 트리 형태가 아닌 이벤트의 시퀀스로 XML 문서의 정보에 접근한다. 따라서 "Serial-access 혹은 event-driven 프로토콜(메커니즘) " 이라 부르기도 한다. 왜냐하면 이 기술은 핸들러를 SAX parser와 함께 등록한 후 파서가 XML문서를 파싱하다가 XML 태그를 만나면 그 태그에 대응하는 함수를 호출하는 구조를 가졌기 때문이다. 이것은 DOM 처럼 object model의 tree를 만드는 과정이 필요 없다. 이런 특징 때문에 XML문서를 읽거나 쓰는 속도가 빠르다. 그렇기 때문에 SAX를 사용하는 서블릿 등의 프로그램에서

XML 문서의 처리를 위한 XML 데이터의 전송과 수신에 있어서 DOM을 사용하는 것 보다 좋은 성능을 기대할 수 있다. 그 이유는 SAX는 현재의 XML 문서를 다루는 메커니즘 중에 제일 빠르고 가장 적게 메모리를 사용하기 때문이다.

◎ SAX는 다음과 같은 일들이 필요하다.

- 특정한 object model을 생성할 때
- 특정 SAX 이벤트를 생성해서 object model를 생성할 때

DOM에서는 이러한 과정이 필요치 않다. 왜냐하면 DOM은 이미 object model을 노드의 트리형태로 정보를 표현하기 때문이다. DOM의 경우, 파서가 거의 모든 일을 수행 한다. XML 문서를 읽어 들이고, java object model를 생성해서 이 object model(Document object)를 참조할 수 있도록 하여 이것의 조작을 도와준다. SAX를 Simple API for XML라고 하는 것도 이런 이유 때문이다. SAX는 파서에게 많은 것을 기대하지 않는다. 단지 SAX 파서는 XML 문서를 읽어 들여서 어떤 태그를 만나면 그에 따라 이벤트를 생성한다. 개발자의 입장에서는 XML document handler class를 만들어서 SAX 파서가 생성한 이벤트를 잡아내서 그것에 따른 처리를 해주기만 하면 되는 것이다. 즉, 모든 태그마다 이벤트를 생성하고 그에 따라 object model 내에 새로운 object가 만들어지는 것이다.

위에서도 언급했듯이, SAX는 object model이 단순하다면 런타임(Run-time)에는 매우 빠르다. 반면에 XML 데이터를 읽으면서 각각의 태그마다 파서가 이벤트를 생성해야 하기 때문에 DOM에 비해 보다 많은 프로

그래밍 작업이 필요하다.

SAX 파서에 의해서 발생되는 SAX이벤트의 종류를 보면 정말 단순한다. SAX는 모든 open tag, close tag, #PCDATA 부분, CDTAT 부분에 대해서 이벤트를 발생시킨다. 그러면 document handler는 발생된 이벤트를 잡아서 해석하고 특정용도의 custom object model을 생성한다. 여기서 이벤트의 해석과 각 이벤트의 발생순서는 매우 중요하다. [6][7]

◎ SAX model을 사용하는 것이 유용한 경우

- XML문서 전체가 아닌 어느 한 부분만 읽을 경우 (SAX는 메모리를 적게 차지하기 때문이다.)
- 한 개 혹은 몇 개의 엘리먼트만 추출 할 때
- 같은 에러의 처리
- 유효성 처리
- 이미 존재하는 데이터의 변환

2.4 DOM(Document Object Model)

XML 문서를 파싱하기 위해서는 DOM API를 사용한다. DOM은 HTML 문서 또는 XML 문서와 같은 인터넷 문서에 대하여 문서의 내용 및 구조를 객체로 표현하고 그 객체를 핸들링 할 수 있는 인터페이스이다. DOM의 가장 두드러진 특징은 XML 문서 안에 나타나는 정보 구조를 표현할 수 있다는 점이다. DOM(Document Object Model) Level Recommendation은 W3C에서 잘 구성된 XML이나 HTML 문서를 지원하기 위한 언어에 중립

적인 인터페이스로 고안되었다. DOM XML parser는 애플리케이션이나 서블릿 안에서 XML문서를 어떠한 자바 객체의 집합으로 변형시켜 메모리에 저장하는 자바 프로그램이다. 다시 말해서 XML문서를 파일 스트림이 아닌 객체로 다루는 것을 뜻한다. 한편 DOM 프로토콜(메커니즘)은 "Random Access" 프로토콜이라고도 알려져 있다. 왜냐하면 DOM을 사용하면 XML 데이터의 어떠한 부분에도 임의로 접근해서 수정, 제거, 새로운 데이터의 삽입을 할 수 있는 기능이 제공되기 때문이다. DOM은 트리 형태의 구조를 가지고 있고 이 트리 형태의 각 노드는 XML구조의 특정한 컴포넌트를 포함한다. 트리를 구성하는 가장 일반적인 노드의 타입으로는 엘리먼트(XML 데이터의 한 단위로 태그로 구분된다. 각 엘리먼트는 중첩이 가능하다.) 노드와 텍스트(각 태그 사이에 들어가는 text) 노드 두 가지가 있다. DOM은 XML 문서의 구조와 정보를 기반으로 노드를 구성하고 계층적인 object model의 형태로 접근이 가능하다. 즉, XML 문서를 수정할 때 직접 XML을 다룰 필요가 없고 메모리에 저장되어 있는 object model을 다루면 되는 것이다. 결국 object model을 표현한 노드와의 상호작용이 곧 XML 문서와의 상호작용이 된다. 그렇기 때문에 DOM에서 지원하는 메소드(APIs)를 사용하면 각 노드의 참조, 생성, 삭제, 수정과 그 내용의 변경을 쉽게 할 수 있다. 일단 XML 문서의 정보가 트리 형태로 표현되므로 일단 XML 문서의 정보가 트리 형태로 표현되므로 트리를 가시화 하기위해 GUI를 첨가하게 되면 사용자의 입장에서는 문서의 구조와 정보를 한 눈에 파악할 수 있고 그것의 수정 또한 쉽게 할 수 있다.

DOM은 XML 문서의 정보에 상관없이 Document object를 만들 때 XML 문서에 따라서 노드의 트리 형태로 만든다. 즉, DOM을 사용하면 XML 문

서의 정보에 접근하기 위해서 트리 모델을 사용할 수밖에 없다. 그런데 원래 XML 자체가 계층적인 구조를 가지고 있기 때문에 이러한 방식은 매우 자연스럽게 돌아간다. 이런 이유로 DOM은 통계적인 데이터이든 품목의 리스트이든 상관없이 모든 정보를 트리에 집어 넣는 것이다. 이러한 계층적인 구성은 파일 시스템과 유사하다. 파일 시스템의 계층 구조를 보면 폴더는 그 안에 파일을 포함할 수 있고 또 다른 폴더를 포함하기도 한다. 각 엘리먼트의 노드는 또 다른 노드 즉, 자식노드(child node)를 가지는 것을 볼 수 있다. 이 자식 노드는 텍스트 값이나 또 다른 노드를 가진다. 그렇기 때문에 만일 특정한 값에 대한 접근이 필요할 때는 엘리먼트 노드에 대한 검색은 불필요하다. 위에서 언급했듯이 엘리먼트는 텍스트 데이터와 다른 엘리먼트를 가질 수 있기 때문에, DOM을 사용할 때는 엘리먼트 노드의 값을 가져오기 위해서는 별도의 작업이 수반된다. 일반적으로 XML 문서가 순수한 데이터를 가지고 있으면 그것을 블록화해서 하나의 문자열(String)로 표현하는 것이 적당하고, DOM은 각 엘리먼트가 가진 값으로 그 문자열을 리턴 한다. 만약에 XML 문서에 저장되는 데이터가 문서일 경우 잘 동작하지 않을 것이다. 순수한 데이터, 이를테면 데이터베이스의 테이블과 같은 데이터에서 엘리먼트의 시퀀스는 그렇게 중요하지 않다. DOM은 XML 문서의 모든 데이터를 document로 간주하기 때문에 XML 문서를 읽어서 엘리먼트의 시퀀스를 보관해둔다. Document Object Model(DOM)이라고 불리는 이유도 여기에서 근거한다. DOM은 단지 W3C에서 정의한 자바 인터페이스에 지나지 않는다. 그리고 이 인터페이스의 구현은 제공되지 않기 때문에 XML 파서를 쓰기 위해서는 구현은 개발자 스스로 해야 한다. 왜냐하면 XML 파서는 플랫폼과 언어에 중립적이기 때문이다. 이러한 W3C의 정책은

보다 낮은 XML 파서의 구현이 나올 수 있도록 유도하는 것이고 이렇게 되면 XML 문서와 자바프로그램은 어떠한 파서에도 구애 받지 않는 것이다.

개발자가 DOM을 이용해서 XML 문서에 저장된 정보를 Java object model로 저장하려고 한다면 SAX는 전혀 고려를 할 필요가 없겠지만 DOM으로 처리하지 못할 경우에는 SAX 고려해 볼 수 있다. 만약 특정 용도에 쓰이는 "Custom object model"을 생성할 필요가 있는 개발이라면 DOM을 사용해도 무방하지만 SAX를 사용하는 것이 훨씬 유용하다.[6][7]

2.5 XML Order Encoding 방법

XML 문서를 관계형 데이터베이스에 엘리먼트별로 나누어 저장하고 질의하기위한 순서를 정하는 것이다. 일반적으로 전체적으로 들어오는 순서대로 노드를 읽어 들이는 과정에서의 순서를 적용하는 Global Order Encoding과 같은 레벨 상에서 순서를 적용하는 Local Order Encoding 방법이 있다. 여기에 좀 더 효과적으로 XML문서 구조의 특징과 순서를 한 번에 적용 가능한 Dewey Order Encoding에서 대해 설명한다.

◎ Global Order Encoding

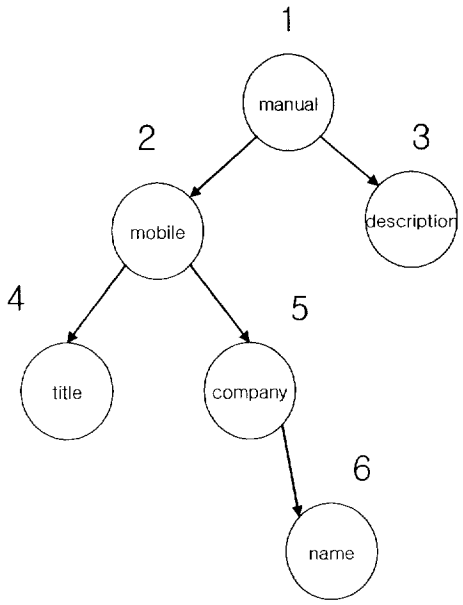
Global Order에서는 각 노드는 XML문서에서 노드들의 절대적인 위치를 표현하는 하나의 순서를 할당하는 것이다. 예를 들어 하나의 엘리먼트들의 위치는 시작하는 문서로부터 시작하는 태그의 바이트 오프셋으로서 인코드된다. 이로 인해서 문서의 순서로서 일정한 길이를 사용한다. (그림 2)의 기본 XML문서에 따른 (그림 3) 은 Global Order를 기술한 예제이다.

◎ Local(Sibling) Order Encoding

Local Order에서는 각 노드는 관계있는 형제(Sibling)사이에서 순서를 표현하는 것이다. Local Order는 update시 낮은 오버헤드를 갖게 하는 것이 특징이다. Local Order에 대한 예는 [그림 4]에서 잘 볼 수 있다.

```
<manual>
  <mobile>
    <title> 사용자 매뉴얼 </title>
    <company>
      <name> SK Telecom </name>
    </company>
  </mobile>
 </mobile>
 <description> </description>
</manual>
```

(그림 2) 기본 XML 문서



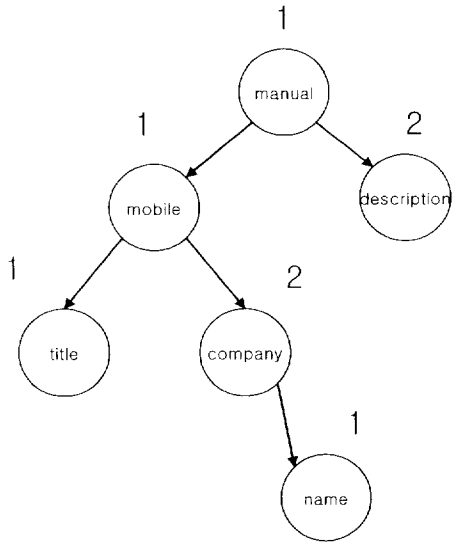
(그림 3) Global Order

◎ Dewey Order Encoding

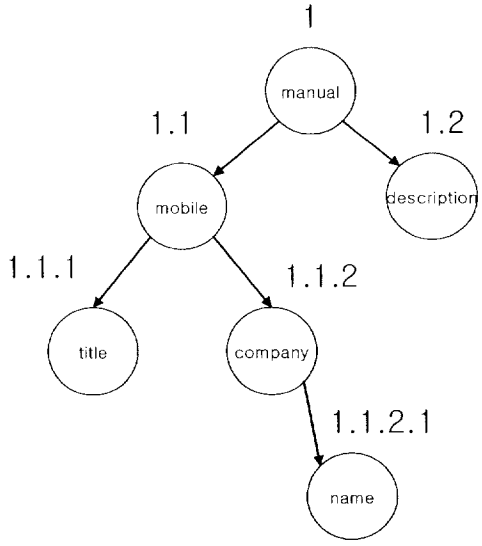
Dewey Order Encoding 방법은 dewey 10진 분류법을 기반으로 한다. Dewey Order는 문서의 루트에서 각 노드까지 경로를 나타내는 하나의 백터를 표현한 것이다. 경로를 표현하는 각 컴포넌트는 (그림 4) 처럼 조상 노드의 Local Order 표현을 나타내는 특징을 갖는다. Dewey Order에서 문서내의 노드의 절대적인 위치를 각 경로에서 유일하게 식별 할 수 있다.

Dewey 경로들은 Global Order를 제공하기 때문에 Global Order내의 쿼

리 처리와 비슷하다. 즉, Global Order와 Local Order를 결합한 Order Encoding인 Dewey Order는 Global Order와 Local Order의 장점을 가진다 즉, 새로 추가되는 노드에 대해 번호를 붙이는 것이 쉽고 또한 검색시 빠른 접근을 가지며 한 번에 문서의 높이와 넓이를 표현함으로서 문서의 구조를 잘 파악 할 수 있다. Dewey Order의 대한 예는 (그림 5)에서 자세히 볼 수 있다.[1][3]



(그림 4) Local Order



(그림5) Dewey Order

2.6 기존 버전관리 시스템

XML문서에 대한 효율적인 관리와 검색 및 저장을 지원하는 XML문서 버전 관리시스템은 반드시 정보 시스템 응용 분야에서 필요하다.

XML문서를 생성하고 개발하는 동안 문서내의 구성 요소는 변경된다. 이

러한 변경 작업으로 XML 문서의 구성 요소는 변경되며 한 버전들을 다음 버전으로 변환 시키는 일련의 과정을 버전관리라 한다. 기존의 버전관리 시스템들로는 SCCS(Source Code Control System), RCS(Revision Control System), UBCC(Usefulness Based Copy Control), 수평 및 수직 버전 중심의 XML 버전관리 시스템들이 있다.

◎ RCS(Revision Control System)

RCS는 파일의 수정정보를 버전으로 저장, 검색, 비교 및 추적하며 병합하는 여러 일들을 할 수가 있다. RCS의 특징은 TEXT 파일의 리버전을 저장하고 검색 할 수 있으며 저장된 임의의 버전을 선택해서 검색 할 수 있다. 또한 각 버전에 대한 특징을 기록할 수 있으며 Temp 파일을 저장할 필요가 없어 저장 공간을 절약한다. 하지만 이런 장점에도 불구하고 RCS는 문자 파일의 버전만이 제어가 가능하며 접근 제어가 원시적이고 가장 최신의 버전을 완전히 저장하는 반면 다른 모든 버전들은 역방향 편집 스크립트로서 저장하므로 최신 버전을 제외한 다른 버전을 위해 구버전을 생성하기 위해 역방향 편집 스크립트를 적용해야 하는 추가의 프로세싱을 필요로한다. 또한 RCS는 원본문서의 논리적 구조를 유지하지 않으므로 XML문서의 구조 검색을 어렵게 만들고 버전 재구성시 고비용이 드는 단점이 있다. [1][3][5][8]

◎ SCCS(Source Code Control System)

SCCS는 소스 코드 파일의 모든 버전을 저장하고 보호할 수 있는 기능이 가능하다. 즉, 파일의 갱신이 발생하면 갱신 과정를 기록하고 언제 어떻

게 일어났는지 관리 할 수 있도록 해준다. SCCS는 파일의 갱신 작업이 발생 할 경우 단지 한 사용자만 파일을 갱신할 수 있도록 잠금을 걸어두고 모든 변화를 히스토리 파일에 기록하여 버전 별 파일 저장 및 검색이 가능하다. 즉, 함수의 초기 버전을 생성 할 경우 SCCS 시스템을 이용하여 SCCS 형식의 파일로 변환해야 한다. SCCS형의 파일은 특수하게 저장되며 보통 방식으로는 편집되거나 번역되지 않는다. 이 파일은 생성된 시간과 생성한 사람에 대한 정보를 가지며 함수 수정이 발생할 경우 이전 버전으로부터 어떤 변경이 있었는지에 대한 정보를 포함한다. [1][3][5][8][9]

◎ UBCC(Usefulness Based Copy Control)

UBCC는 edit-based 편집에 바탕을 둔 방법을 향상시킨 기법이다. 이 방법은 주어진 버전의 유효한 객체들을 데이터 페이지에 군집화하는 페이지 유용성의 개념을 사용한다. 즉, 한 페이지에서 유효한 많은 객체들이 어떠한 임계값 이하이면 유효한 모든 페이지 객체를 다른 페이지로 복사되어진다. 유용한 페이지들이 액세스 될 때마다 하나의 버전이 구성되어진다. 특정 문서의 버전을 재구성하는 것은 버전을 포함하는 객체들을 찾아서 그것들을 정확한 논리적 순서로 생성하는 것과 같다. 그러나 버전들의 편집에 바탕을 둔 표현은 문서 내용을 재배열 및 재구성하는 변경에 대해서는 비효율적이다.[1][3][5][8][9]

◎ 수직 및 수평 버전 중심의 XML 버전관리

수직 및 수평 버전 중심의 XML 버전관리 기법은 손실 없는 XML문서의 저장 , 수정, 관리를 위한 수직 및 수평 버전들을 관리하는 기법이다. 기

존의 이 방법은 버전 번호 구분을 릴리스 버전번호와 변경 버전 번호로 나누어 버전번호를 유지한다. 즉, 수직 및 수평 버전 번호 할당 방법의 버전 계층은 트리로 구성되며 실질적인 버전 번호 부여는 한 문서가 버전닝이 되면 버전 번호 할당은 릴리스 버전 번호에 더불어 수평 버전 번호와 수직 버전 번호의 순서대로 변경 버전 번호의 두 번호를 조합하여 할당하며 각각의 버전 번호의 구별은 점로 구분한다. 여기서 하나의 릴리스 문서가 처음 변경되면 수평 변경 작업이 되고 이것은 이후에 같은 릴리스 문서에 또 다른 수평 변경 작업이 발생해도 기존의 버전번호와 독립적으로 수평 작업의 버전 번호가 부여 될 수 있는 장점을 가진다. [9]

수평 변경 작업시 버전 번호는 두 변경 버전 번호가 조합되어 부여 되는데, 수평 변경 번호만이 부여 되며 수직 변경 번호는 수직 변경 작업이 발생하지 않았음을 나타내는 0을 부여하는 방식으로 수평 변경임을 나타낸다. 그리고 수직 변경은 수직 번호가 하나 증가하여 할당 되며 수평 번호는 변경하지 않는 번호 부여 방식이다. 이 방법은 기존 버전들의 버전 번호는 변하지 않고 확장된 새로운 버전 번호추가 할 수 있고 버전 번호를 버전의 순서 결정이 가능하며 해당 버전의 직계 조상이나 후손을 알 수 있다는 장점들을 가지고 있다. 하지만 수직 및 수평 번호 할당 법만으로 XML 문서의 시간적인 이력을 자세히 알 수 없고 재구성시 엘리먼트의 순서와 엘리먼트 깊이에 대한 정보를 따로 유지해야 하기 때문에 비용 및 시간적인 측면에서 비효율적이다. [9]

본 논문에서는 기존의 논문들의 약점인 XML 문서의 시간적인 이력 정보

를 제공하지 못한다는 단점을 극복하고자 이력 데이터를 효율적으로 관리하기 위하여 버전번호와 시간적인 이력을 결합한 새로운 버전 번호 방식을 제안한다.

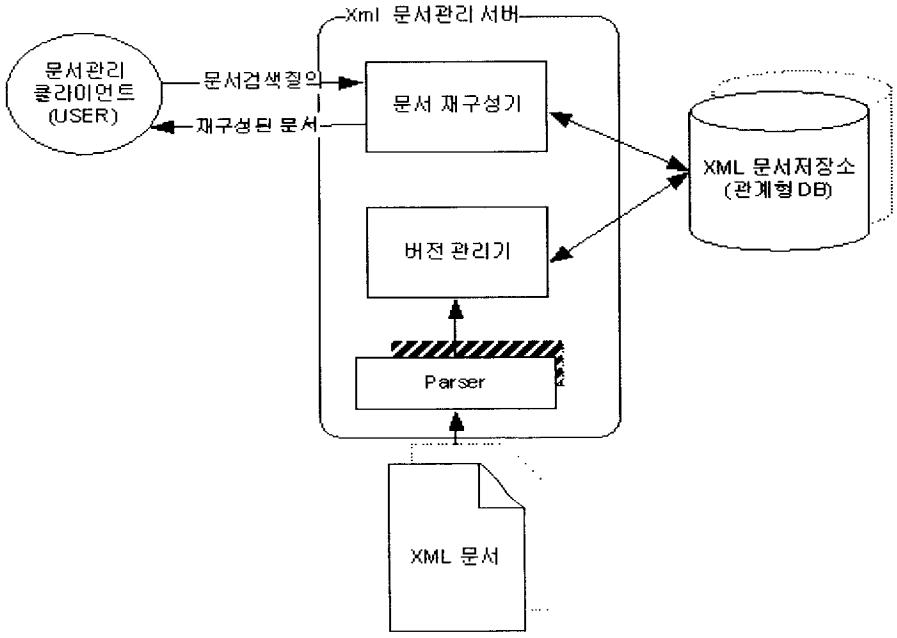
또한 XML 문서 재구성시 좀 더 효율적으로 관리 할 수 있도록 하기 위해 Ordering 방법으로 Dewey Order를 이용하여 XML 문서의 구조 및 엘리먼트의 순서를 함께 표현하여 더욱 빠른 검색 과 사용자로 하여금 XML 문서의 버전 변화를 쉽게 파악 할 수 있도록 한다.

3. 제안하는 XML 문서 버전 시스템

3.1 XML 문서 버전 관리 시스템 구조

본 논문에서 제안하고자하는 XML 문서 버전 관리 시스템은 XML 문서를 파싱하여 XML 문서의 구조정보와 시간정보를 얻는다. 이들 정보는 버전 관리기에 의해 부여된 버전번호와 함께 원격의 데이터베이스에 저장하도록 설계한다. 전체 시스템의 구성 요소 중 핵심이 되는 시스템은 버전관리기와 문서 재구성기가 된다.

전체 시스템 구성도는 (그림6)과 같다.



(그림 6) 전체 시스템 구성도

3.2 시스템의 구성 요소

본 논문에서 제안하는 시스템의 구성요소는 다음과 같다.

전체 시스템은 3-tier 구조로 되어 있으며 (그림6)처럼 세부분으로 나눌 수 있다.

◎ XML 문서 관리기

본 논문의 시스템에서 가장 핵심적인 부분인 XML 문서 관리기는 (그림 6)의 전체 시스템 구성도에서 볼 수 있듯이 기능적인 면에서 버전관리기와 문서 재구성기로 나누어 볼 수 있다.

먼저 파서로부터 각 element 단위로 문서를 읽어 들이면 각 element는 버전관리기를 처음 거치게 된다. 이때 element는 파서를 거치면서 dewey ordering을 첨부하면서 버전관리기에서 시간정보와 버전의 일련번호를 결합한 버전번호를 할당 받는다. 이렇게 해서 버전번호와 dewey ordering 및 XML 문서에서 엘리먼트의 초기 생성 시간인 start_time과 엘리먼트의 삭제 및 다른 버전으로 엘리먼트가 업데이트 될 때 이전의 버전의 종료시간인 end_time 등을 데이터베이스에 element 테이블과 버전 테이블에 저장을 하게 되는 것이다. 버전 관리기에서는 이들 각 element들의 정보 중 버전번호와 dewey order를 이용한 인덱스를 유지하면서 다음에 어떤 element나 text의 변경이 발생시 빠른 변경을 제공한다.

문서 재구성기는 다양한 사용자의 질의 조건에 따라 XML 문서를 재구성해주는 기능 제공하게 된다. 질의가 들어 왔을 경우 재구성기는 데이터베이스의 버전테이블에 저장된 값을 기준으로 원하는 정보를 추출하며 문서 재구성기는 알맞은 형태의 XML 문서로 질의 정보를 보여주는 것이다.

◎ 데이터베이스

모든 XML 문서의 버전정보 및 이력에 관한 정보를 저장해주는 곳이다. 여기서는 XML 문서관리기와 계속적인 연관 관계를 가지며 XML 문서가 파서를 통해 엘리먼트 단위로 분할 되어 버전관리기에 의해 이력정보가 첨부된 엘리먼트를 데이터베이스에 저장 시키고 이 저장된 정보를 문서 재구성기에서 질의가 왔을 경우 추출을 하여 USER에게 원하는 정보를 보여 줄 수 있도록 지속성을 유지 시켜주는 기능을 제공한다.

◎ USER

사용자는 언제 어디서나 자신이 원하는 정보를 질의를 통하여 XML 문서로 볼수 있다. 사용자는 시스템 내부의 내용을 알 필요가 없으며 원하는 문서의 이력정보 및 XML 문서의 변화과정을 질의를 함으로서 획득 할 수 있는 응용계층이다. 사용자는 문서 재구성기를 통해 데이터베이스의 접근을 하게 되며 문서 재구성기는 사용자의 질의를 분석한 후 데이터베이스의 정보를 사용자에게 직접적으로 XML 문서를 만들어서 제공해주면 된다.

위와 같이 본 논문의 시스템 구조에 따라 XML 문서의 버전의 변화 및 이력정보를 획득하는 과정이 USER 그리고 XML 문서 관리기, 데이터베이

스 연동을 이용하여 빠른 질의 처리와 더 빠른 XML 문서 재구성을 3-tire로 가능하게 하는 시스템을 구성한다.

3.3 이력 정보를 위한 데이터베이스 구성

이력정보를 위한 데이터베이스는 관계형 데이터베이스를 이용하여 구성한다. [2] 데이터베이스의 테이블 구성은 최초의 XML 문서를 지속적으로 유지하기위한 엘리먼트 테이블과 XML 문서의 계속적인 변화를 저장하고 검색을 위한 테이블인 버전 테이블로 구성한다. 두 개의 테이블들의 세부 구성은 (그림 7) 그리고 (그림 8)과 같이 구성 한다.

Xid	element	version	Xorder	Start_time	End_time	content	Flag

(그림 7) 버전 테이블

Xid	element	version	Xorder	content

(그림 8) 엘리먼트 테이블

이들 테이블에서 엘리먼트 테이블의 속성으로는 Xid, element, version, Xorder, content 로 이루어져 있다. 그리고 이력정보를 가지고 있는 버전 테이블에서는 Xid, element, version, Xorder, Start_time, End_time, content, Flag 속성으로 이루어져 있다. 먼저 엘리먼트 테이블의 Xid는 XML 문서에 부여되는 고유 식별자이다. element 속성은 XML 문서에 해당 엘리먼트의 이름이고 version은 버전관리기에 의해 부여되는 시간적 정보와 버전 번호

이다. Xorder는 dewey order 방법을 이용한 XML 문서 구조에 대한 정보를 나타내는 것이며 content는 해당 엘리먼트가 포함하고 있는 텍스트 정보를 나타낸다.

버전 테이블은 엘리먼트 테이블의 속성들과 함께 추가 속성으로 XML 문서에서 엘리먼트의 초기 생성 시간인 Start_time과 엘리먼트의 삭제 및 다른 버전으로 엘리먼트가 업데이트 될 때 이전 버전의 종료시간을 나타내는 End_time 속성를 가지며 그리고 엘리먼트들이 업데이트 또는 삭제 및 삽입이 발생 할 때마다 Flag 속성에는 변경 상태를 표시하기 위해 각각 이니셜을 따서 업데이트(Update)는 U로 표기하고, 삭제(Delete)가 발생될때 D로 표기하며 새로운 삽입(Insert)이 발생하면 I를 표기한다.

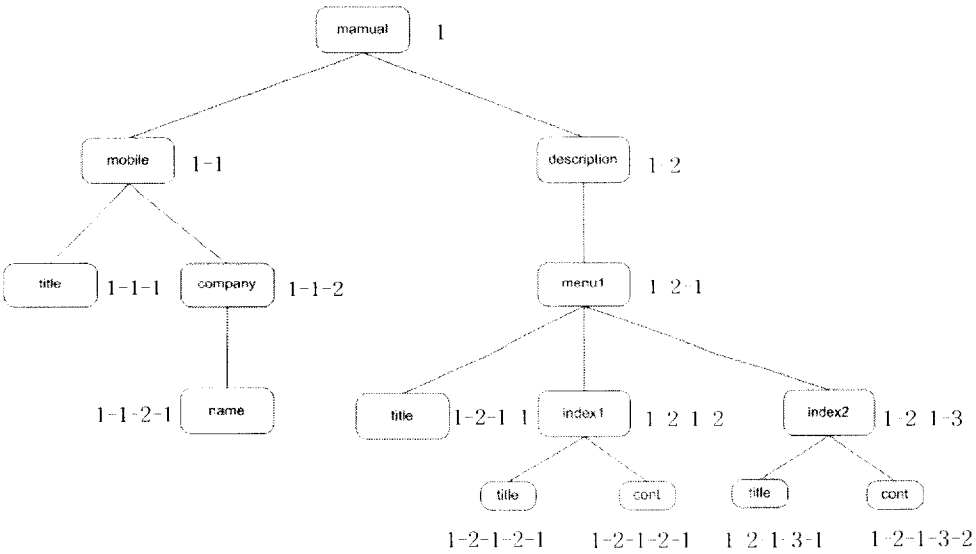
이와 같이 두 테이블인 엘리먼트 테이블과 버전 테이블를 이용해서 XML 문서의 지속적인 변화를 저장할 수 있다. 그리고 우리는 이런 테이블 내의 속성들을 이용하여 검색을 수행한다면 좀 더 효율적으로 XML 문서를 관리할 수 있다.

3.4 XML 문서 및 계층적 구조에 따른 Dewey Order 방법

본 논문에서는 XML 문서의 예제로 휴대폰 사용자 매뉴얼 문서의 버전변화를 위한 XML 문서 (그림 9)와 XML 문서가 파서를 통해 Dewey Order를 부여 받는 형태를 (그림 10)과 같이 표현 한다.

```
<manual>
  <mobile>
     <title>휴대폰 메뉴얼</title>
     <company>
        <name>회사이름</name>
      </company>
  </mobile>
  <description>
     <menu1>
        <title>기본기능</title>
        <index1>
          <title>휴대폰 on/off</title>
          <cont>stop버튼사용</cont>
        </index1>
        <index2>
          <title>전화 걸기</title>
          <cont>번호입력 후 send 버튼</cont>
        </index2>
     </menu1>
  </description>
</manual>
```

(그림 9) 휴대폰 메뉴얼 XML 문서



(그림 10) 휴대폰 메뉴얼 XML 문서에 적용된 Dewey Order

위 (그림 9)의 XML 문서는 (그림 10)처럼 Dewey Order 엘리먼트 노드의 순서를 부여받는다. 이 Dewey Order는 파서를 통해 XML 문서를 처리 시 순서적으로 루트에 영향을 받아 루트에서 떨어진 정도에 따라 "-" 구분자를 통해 형제인지 자손인지를 나타내어준다. 예를 들면 (그림 10)에서 루트의 엘리먼트 이름이 manual인 Dewey Order는 1이며 바로아래 아들 엘리먼트인 mobile 엘리먼트와 description 엘리먼트는 Preorder(전위순회)방법을 적용하여 mobile 엘리먼트의 Dewey Order는 1-1이고 형제간인 description 엘리먼트는 1-2를 부여받는다. 즉, "-" 구분자의 수에 따라 형제인지 아니면 손자인지 결정이 가능하며 노드의 순서 또한 알 수 있는 장점을 가지게 된다. 따라서 문서 재구성기를 통해 이 Dewey Order는 더 효율적인 재구성 방법으로 제공될 수 있다.

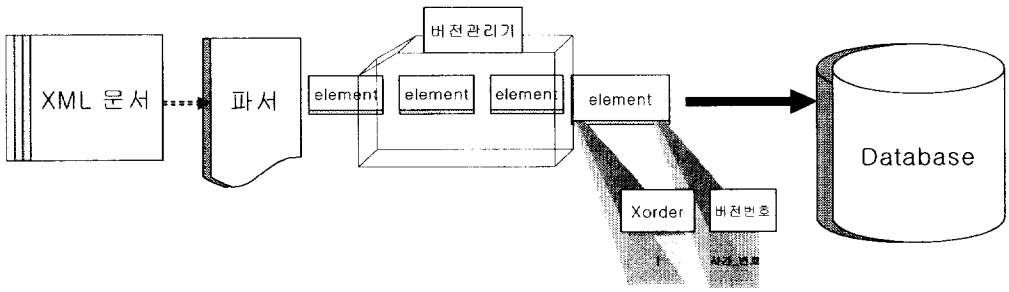
3.5 새로운 시간정보 버전번호 부여방법

본 논문에서는 제안하는 문서 버전할당 방법은 시간에 대한 이력정보가 포함된 버전 번호 할당 방법을 제안한다.

시간정보와 버전번호의 조합은 하나의 이력정보로 데이터베이스에 저장한다. 문서의 변화를 한 번에 볼 수 있다는 장점을 가진다.

본 논문에서 사용하는 버전번호 방식은 새로운 시간정보 버전번호 부여방법으로서 날짜 정보와 일련의 버전번호를 조합하여 버전 번호를 할당 한다. 이 버전 번호 할당 방법은 버전 번호만을 조회하는 것으로도 XML 문서내

의 각 엘리먼트들의 시간적인 버전 변화와 버전이 일어난 횟수를 알 수 있다. 예를 들어 최초로 생성된 XML 문서에서 엘리먼트들을 데이터베이스에 저장 할 때 각 엘리먼트는 자신의 첫 번째 버전임을 나타내기 위해 버전번호로 일련 번호 1을 부여받는다. 그리고 버전번호와 함께 시간적 정보를 제공하기 위해 생성 시점의 년도와 월을 같이 명시해 준다. 이들을 결합하는 방법은 예를 들어 문서 생성시점 시간이 2004년 08월 05일 에 최초로 생성된 엘리먼트라고 한다면 버전번호 부여는 0408_1로 버전번호를 부여한다. 이런 시간정보 버전번호는 버전관리기가 처리한다. (그림 11)은 새로운 시간정보 버전번호 부여방법에 대한 처리는 다음과 같은 그림으로 표현할 수 있다.



(그림 11) 새로운 시간정보 버전번호 부여 방법

3.6 버전 테이블에 변경된 XML 문서저장

XML 문서 예는 (그림 9)의 예제 XML 문서에 대한 최초 엘리먼트 테이블과 버전 테이블은 (표 1), (표 2)와 같다.

Xid	element	version	Xorder	content
1	manual	0407_1	1	
1	mobile	0407_1	1-1	
1	title	0407_1	1-1-1	휴대폰 메뉴얼
1	company	0407_1	1-1-2	
1	name	0407_1	1-1-2-1	회사이름
1	description	0408_1	1-2	
1	menu1	0408_1	1-2-1	
1	title	0408_1	1-2-1-1	기본기능
1	index1	0408_1	1-2-1-2	
1	title	0408_1	1-2-1 2-1	휴대폰on/off
1	cont	0408_1	1-2-1-2-2	stop 버튼사용
1	index2	0408_1	1-2-1-3	
1	title	0408_1	1-2-1-3-1	전화걸기
1	cont	0408_1	1-2-1-3-2	번호입력 후 send버튼

(표 1) 휴대폰 메뉴얼 XML 문서의 엘리먼트 테이블

Xid	element	version	Xorder	content	start_time	end_time	Flag
1	manual	0407_1	1		2004-07-09-11:11	now	
1	mobile	0407_1	1-1		2004-08-09-11:11	now	
1	title	0407_1	1-1-1	휴대폰 메뉴얼	2004-07-09-11:11	now	
1	company	0407_1	1-1-2		2004-07-09-11:11	now	
1	name	0407_1	1-1-2-1	회사이름	2004-07-09-11:11	now	
1	description	0408_1	1-2		2004-07-09-11:11	now	
1	menu1	0408_1	1-2-1		2004-07-09-11:11	now	
1	title	0408_1	1-2-1-1	기본기능	2004-07-09-11:11	now	
1	index1	0408_1	1-2-1-2		2004-07-09-11:11	now	
1	title	0408_1	1-2-1-2-1	휴대폰on/off	2004-07-09-11:11	now	
1	cont	0408_1	1-2-1-2-2	stop 버튼사용	2004-07-09-11:11	now	
1	index2	0408_1	1-2-1-3		2004-07-09-11:11	now	
1	title	0408_1	1-2-1-3-1	전화걸기	2004-07-09-11:11	now	
1	cont	0408_1	1-2-1-3-2	번호입력 후 send버튼	2004-07-09-11:11	now	

(표 2) 휴대폰 메뉴얼 XML 문서의 버전 테이블

만약 XML 문서의 삽입이나 삭제 및 수정이 일어난 경우 변경정보를 테

이블의 저장한 내용은 (표 3)과 같다. 또한 (표 3)에 의해 변경된 XML문서는 (그림 12)의 결과와 같다.

Xid	element	version	Xorder	content	start_time	end_time	Flag
1	manual	0407_1	1		2004-07-09-11:11	now	
1	mobile	0407_1	1-1		2004-08-09-11:11	now	
1	title	0407_1	1-1-1	휴대폰 메뉴얼	2004-07-09-11:11	now	
1	company	0407_1	1-1-2		2004-07-09-11:11	2004-08-15-17:00	
1	name	0407_1	1-1-2-1	회사이름	2004-07-09-11:11	2004-08-12-18:00	
1	description	0408_1	1-2		2004-07-09 11:11	now	
1	menu1	0408_1	1-2-1		2004-07-09-11:11	now	
1	title	0408_1	1-2-1 1	기본기능	2004-07-09-11:11	now	
1	index1	0408_1	1-2-1-2		2004-07-09-11:11	now	
1	title	0408_1	1 2 1-2-1	휴대폰on/off	2004-07-09-11:11	now	
1	cont	0408_1	1 2-1-2-2	stop 버튼사용	2004-07-09-11:11	now	
1	index2	0408 1	1-2-1-3		2004-07-09-11:11	now	
1	title	0408_1	1 2 1-3-1	전화걸기	2004-07-09-11:11	now	
1	cont	0408 1	1 2 1-3-2	번호입력 후 send버튼	2004-07-09-11:11	now	
1	menu2	0408_1	1-2-2		2004-08-01 10:00	now	I
1	title	0408_1	1-2-2-1	전화번호 메뉴	2004-08-01-10:00	now	I
1	name	0408 2	1-1-2-1		2004-08-12 18:00	Expire	D
1	company	0408_2	1-1-2	SK Telecom	2004-08-15-17:00	now	U

(표 3) 변경 작업 후 버전 테이블

최초 XML 문서에 대한 XML 문서 정보 저장은 XML 문서 관리기를 통해서 데이터베이스의 엘리먼트 테이블과 버전 테이블에 각각 저장 한다. 따라서 문서가 변경이 발생 되면 (표 3)과 같은 버전 테이블의 변경 작업이 발생하는데 이 변경작업은 다음과 같은 변경을 말한다. (표 3)에서 변경된 내용은 아래와 같다.

◎ 삽입

2004년 08월 1일 10시에 description 엘리먼트의 자식 엘리먼트로 menu 엘리먼트와 menu 엘리먼트 자식 엘리먼트로 title 엘리먼트와 내용으로 '전화번호메뉴' 텍스트가 삽입 되었음을 나타낸다.

◎ 삭제

2004년 08월 12일 18시에 mobile 엘리먼트의 하위 엘리먼트인 company 엘리먼트의 자식 엘리먼트인 name 엘리먼트의 텍스트가 삭제 되었음을 나타낸다.

◎ 업데이트

2004년 08월 15일 17시에 mobile 엘리먼트의 하위 엘리먼트인 company 엘리먼트의 텍스트 속성값으로 'SK Telecom'이 내용으로 추가 되었음을 나타낸다.

본 시스템에서 변경작업은 엘리먼트 단위로 버전이 일어난다. 예를 들어 2004년 8월 2일에 엘리먼트 company에 수정이 일어났다면 먼저 해당 엘리

먼트의 현재 버전번호를 가지는 레코드의 end_time필드의 값을 변경이 일어난 날짜 정보로 수정을 한다. 다음으로 Flag 필드의 값으로 U를 version 필드의 값은 엘리먼트에 수정이 일어난 날짜의 년월 정보와 이것이 해당 엘리먼트의 두 번째 버전이라는 것을 나타내기 위해 version 필드의 값으로 시간정보의 0408과 일련번호 2를 결합하여 0408_2를 버전 번호로 부여하고 이것을 버전 테이블에 저장한다.

본 논문에서는 버전 번호에 이처럼 시간정보를 결합함으로서 버전이 변화된 시간적인 진행 과정을 쉽게 파악할 수 있으며 또한 버전의 일어난 횟수도 쉽게 알 수 있는 장점을 제공한다.

새로운 엘리먼트의 삽입시에는 Flag 필드 값으로 I를 저장하고 현재의 시간 정보와 일련 번호 1를 결합한 값을 버전번호로 부여하고 start_time 필드 값으로는 생성시간을 end_time필드 값으로는 now를 준다. 특정 엘리먼트에 삭제가 발생할 경우 기존의 버전번호에서 시간정보와 버전번호수를 증가 시키고 end_time 필드에는 삭제가 완전히 되어진 표시로 'Expire'를 표시한다. 문서의 변화가 발생 할 때 마다 이런 과정을 수행함으로서 버전은 효율적으로 관리 될 수 있으며 이력 정보를 통한 유용한 검색 기능을 제공한다.

3.7 사용자 중심의 다양한 질의 수행

본 논문에서 제안한 방법은 사용자가 원하는 다양한 유형의 질의 수행이 가능한 효과적인 구조를 가진다.

특히, 특정 엘리먼트의 전체 버전 변경횟수에 대한 정보 검색을 질의하는 경우 또는 특정 엘리먼트의 시간적인 버전 진화 과정에 대한 정보 및 사용자가 원하는 특정 시점에 해당하는 문서의 재구성을 원할 때 효율적인 XML 문서의 재구성을 지원한다. 예를 들어 질의 유형을 세가지로 구분하면 첫째 현재시점을 기준으로 XML 문서 재구성, 둘째 과거의 특정 날짜까지 변경된 XML 문서의 재구성, 그리고 마지막으로 특정 엘리먼트 버전닝 과정을 들 수 있다.

질의 유형 1 - 현재기준

사용자가 버전 테이블의 end_time이 현시점인 now일 때 이에 대한 XML 문서의 정보를 요구할 경우 질의 처리에 따라 재구성된 XML로 문서의 형태는 (그림 12)과 같다.

```
<manual>
   <mobile>
      <title>휴대폰 메뉴얼</title>
      <company>SK Telecom</company>
   </mobile>
   <description>
      <menu1><title>기본기능</title>
         <index1><title>휴대폰 on/off</title>
                  <cont>stop버튼사용</cont>
         </index1>
         <index2><title>전화 걸기</title>
                  <cont>번호입력 후 send 버튼</cont>
         </index2>
      </menu1>
      <menu2><title>전화번호 메뉴</title>
      </menu2>
   </description>
</manual>
```

(그림 12) 현재시점(end_time = 'now')의 XML문서 재구성

질의 유형 2 - 과거의 특정 날짜 기준

사용자가 과거의 특정 날짜인 2004년 8월 13까지 변경된 문서의 내용을 검색 하고자 하는 질의 처리를 원할 경우 (그림 13)과 같이 재구성된 XML 문서를 볼 수 있다.

```
<manual>
  <mobile>
     <title>휴대폰 메뉴얼</title>
     <company></company>
  </mobile>
  <description>
     <menu1>
        <title>기본기능</title>
        <index1>
          <title>휴대폰 on/off</title>
          <cont>stop버튼사용</cont>
        </index1>
        <index2>
          <title>전화 걸기</title>
          <cont>번호입력 후 send 버튼</cont>
        </index2>
     </menu1>
     <menu2>
       <title>전화번호 메뉴</title>
     </menu2>
  </description>
</manual>
```

(그림 13) 과거 특정시점 기준의 XML문서 재구성

위 (그림 13)에서 과거의 어느 시점을 기준으로 문서의 재구성을 원할 때 예를 들면, start_time이 "2004년 8월 13일"인 것을 기준으로 적거나 같은 모든 레코드들을 버전 테이블로부터 검색하여 문서를 재구성 할 때 Xid 필

드와 Xorder 필드 기준으로 정렬 한 후 같은 Xorder 필드의 값을 가질 경우 version 번호를 비교해서 가
장 큰 값을 가지는 엘리먼트를 Xid별로 문서를 재구성 할 때 포함 시킨다.

(그림 13)에서 볼 수 있듯이 company 엘리먼트의 텍스트는 재구성을 원하는 날짜 이후에 "SK telecom" 텍스트가 생성 되었으므로 문서에 포함되지 않는다는 것을 확인할 수 있다.

질의 유형 3 - 특정 엘리먼트 버전닝 과정

사용자는 어떠한 특정 엘리먼트의 버전닝 과정을 쉽게 검색 할 수 있다. 예를 들면 날짜별 버전닝 과정과 버전번호별 버전닝 과정의 질의 처리를 이용해서 아래의 (그림 14)와 같이 추출이 가능하다.

```
날짜별 company의 버전과정
   2004-07-09 11:11 -> 2004-08-15-17:00
버전번호별 company의 버전과정
   0407_1 -> 0408_2
```

(그림 14) 특정 엘리먼트 버전과정

4. 결론

본 논문에서는 이력 정보를 활용한 새로운 XML 문서의 버전 관리 방법을 제안하였다. 기존의 방법은 문서의 동적인 변화에만 중점을 두었다면 제안한 시스템은 이력정보를 시간적 흐름에 따라 지속적으로 관리하기 위해 시간적인 정보를 추가한 버전번호 방식을 제안하였다. 또한 XML 문서의 구조의 재구성을 좀 더 빠르고 효율적으로 처리하기 위해 Dewey Order를 이용한 엘리먼트간의 포함관계 즉, 계층관계를 잘 구성 할 수 있었다.

따라서 기존의 방법과 비교해서 사용자들이 시간의 흐름에 따라 이력정보 및 질의를 다양하게 할 수 있음을 보였다.

참고문헌

[1] Shu-Yao Chien, Vassilis J. Tsotras, Carlo Zaniolo, Donghui Zhang, Storing and Querying Multiversion XML Documents using durable Node Numbers, IEEE, 2002

[2] Fabio Gran야, Federica Mandreoli, Paolo Tiberio, Marco Bergonzini, A Temporal Data Model and Management System for Normative Texts in XML Format, ACM, 2003

[3] logr Tatarinov, Stratis D.Viglas, Kevin Beyer, Jayavel Shanmugasundaram, Eugene Shekita, Chun Zhang, Storing and querying Ordered XML Using a Relational Database System, ACM, 2002

[4] Shu-Yao Chien, Vassilis J. Tsotras, Carlo Zaniolo, A Comparative Study of Version Management Schemes for XML Documents,ystem, ATimeCenter Technical Report, 2000

[5] Shu-Yao Chien, Vassilis J. Tsotras, Carlo Zaniolo, Version Management of XML Documents, WebDB 2000 Workshop, Dallas, TX, 2000

[6] 김경일, 박찬규, 조현규, 함호성, XML 제품 개발 동향, 한국 인터넷 정보학회, 2001

[7] [W3C98] W3C, "W3C Recommendation : Extensible markup Language1.0", http://www.w3c.org/TR/1998/

[8] Ramez Elmasri, Gene T.J. Wuu, Yeong-Joon Kim, THE TIME INDEX:An Access Structure for Temporal Data, VLDB, 1990

[9] 배양석, 손충범, 유재수, XML 문서 관리를 위한 효율적인 버전관리 방법,

Journal of the Research Institute for Computer and Information Communication, 2002

[10] 이홍래, 이형동, 유상원, 김형주, XML 문서상에서 키워드 검색을 위한 색인 분할 기법, 한국 정보 과학회 , 2004

[11] 김성준, 김동아, 이석균, X-treeDiff를 이용한 XML 문서의 버전관리 시스템 프로트타입 개발, 한국정보처리학회, 2003

[12] 박성호, 박우진, 이력데이터베이스에서의 시간 Domain 확장, 한국정보처리학회, 1995

[13] Patrick Cauldwell, Rajesh Chawla, Vivek Chopra, Professional XML Web Services, Wrox출판사, 2001

[14] H. M. Deitel, P. J. Deitel, J. P. Gadzik, k. Lomeli, S. E. Santry, S. Zhang, Java Web Services for Experienced Programers, Deitel Developer Series, 2003

감사의 글

본 논문을 위해 힘써주시고 많은 가르침을 주신 조우현 교수님께 깊은 감사를 드립니다. 또한 부족함이 많은 저의 논문을 심사해주신 서경룡 교수님, 송하주 교수님께 진심으로 감사드립니다. 또한 학위 과정 동안 부족한 저에게 많은 가르침을 주셨던 학과의 여러 교수님들께도 감사드립니다. 마지막으로 학부시절 저에게 많은 도움을 주신 오암석 교수님께 감사를 드립니다.

대학원 생활 처음부터 밝게 저를 맞아 주셨던 우리의 영원한 본드걸 희숙선배, 늘 조용하면서 대학원의 기둥이 되어 주셨던 창수선배, 저에게 언제나 옆에서 한결같이 도움을 준 무뚝뚝하지만 다른 사람들은 모를 따뜻함과 정열이 언제나 불타오르는 성주선배, 우리 대학원의 가장 최고의 꽃미남이자 자칭 카리스마 석주선배, 바람처럼 다니시는 주현선배, 우리 랩을 지켜나가고 있는 우리의 동생들 락기, 상호, 상훈 이하 대학원의 모든 선후배, 동기들에게 감사의 마음을 전합니다.

저를 언제나 믿어주시고 보살펴 주신 우리 부모님, 우리 형 항상 사랑합니다. 또한 제가 대학원 생활을 시작할 무렵 하늘나라로 가신 우리 할머니께 이 논문을 바칩니다.

2005년 2월 김 성 록 드림