# 압축된 문서에 대한 질의 처리를 지원하는 XML 압축 알고리즘의 설계 및 구현

## Design and Implementation of a XML Compression Algorithm Supporting Query Processing for Compressed Documents

이석재
충북대학교 정보통신공학과

강영준
충북대학교 정보통신공학과

유재수
충북대학교 정보통신공학과

조기형
충북대학교 정보통신공학과

Seok-Jae Lee (sjlee@netdb.chungbuk.ac.kr)
Department of Computer & Communication Engineering, Chungbuk National University
Young-Jun Kang (penguiney@netdb.chungbuk.ac.kr)
Department of Computer & Communication Engineering, Chungbuk National University
Jae-Soo Yoo (yjs@cbucc.chungbuk.ac.kr)
Department of Computer & Communication Engineering, Chungbuk National University
Ki-Hyoug Cho (khjoe@cbucc.chungbuk.ac.kr)
Department of Computer & Communication Engineering, Chungbuk National University

*중심어 : XML, 질의 처리, 압축*

*Keyword : XML, Query Processing, Compression*

**요 약**

인터넷의 급속한 확산에 따라 사회 전반의 디지털화와 지식정보화가 급속도로 진행되고 있다. 많은 사용자들은 웹 상에서 다양한 작업을 하고 서비스를 이용하고 있다. 이러한 작업들의 대부분은 XML을 이용한다. XML은 개발자가 필요시 문서의 논리 구조를 정의할 수 있으며, 내용과 스타일이 분리되어있어 문서의 재사용성이 뛰어나다. 하지만 XML은 문서의 내용을 단순히 텍스트 형태로 다루고 문서의 구조를 표현하기 위해 많은 태그들이 추가되기 때문에 문서의 크기가 커질 수 밖에 없다. 팜탑, PDA등 용량이 제한된 기기에서 XML 문서를 보다 잘 활용하기 위해서는 XML문서를 효율적으로 압축해서 사용할 필요가 있다. 이러한 이유로 최근 XML 문서를 효과적으로 압축하고 다루기 위한 XML 압축 기법에 대한 연구가 일부 이루어지고 있지만, 기존 연구들의 대부분은 압축된 XML 문서에 대한 질의 처리를 고려하지 않았다.

**Abstract**

With the spread of internet, the digitalization and the knowledge informatization are in progress rapidly. Specially, numerous users make the various works and use the services on the web. For the most part, these works make use of the XML. The XML shines the reusing of the documents because it is separated from contents and styles. Also, it can re-define the logic structure of the document for requirement of the developer. However, the XML document's size is much larger than common text document because it handles the document type and adds numerous tags for representing structure of the document. To utilize the limited storage devices of Palmtop, PDA and so on, it is necessary to compress and handle the documents efficiently. Recently, the compression techniques for efficiently handling and compressing the XML documents are under way to solve this problem.

* 본 연구는 정보통신연구진흥원 기초기술연구지원사업 (과제번호 2003-1-00899) 지원으로 수행 되었습니다.
접수번호 : #031229-001
접수일자 : 2003년 12월 29일, 심사완료일 : 2004년 3월 13일
*교신저자 : 이석재, e-amil : sjlee@netdb.chungbuk.ac.kr

본 연구에서는 기존의 방법들보다 XML 문서를 효과적으로 압축을 하여 저장 공간의 활용도를 높이고, 압축된 XML 문서에 대해 질의처리를 가능하게 하여 보다 빠른 질의 처리를 할 수 있는 XML 압축 알고리즘을 설계 및 구현한다.

But most of the existing researches don't support the query processing for the compressed XML documents. In this paper, we design and implement the XML compression algorithm that compresses the XML document and processes the query of compressed XML document faster and more efficiently than previous techniques.

## Ⅰ. 서론

최근 초고속 인터넷의 보편화로 인해 대부분의 기업체와 개인들이 인터넷 상에서 XML(Extensible Markup Language) 문서를 사용하고 있다[1]. XML문서는 문서표현을 위해 내용을 텍스트 형태로 다루고 있으며 문서의 구조를 표현하기 위해서는 많은 태그, 애트리뷰트 등의 메타 데이터를 적용해야 한다. 이러한 이유로 문서를 표현하려는 내용보다 구조상의 틀을 표현하는 부분에 의해 문서의 크기가 커지게 된다. 또한, 팜탑, PDA등의 장치에서 많은 문서 데이터를 이용하여 업무 등을 처리하는 모바일 장치 사용자가 급증하고 있다. 그러나 이러한 장치들은 제한된 공간 즉, 한정된 메모리를 가지고 있다. 이러한 문제점 해결을 위해 XML 압축 알고리즘 연구가 진행되고 있다.

XML 문서의 데이터 크기를 줄임으로써 저장 공간 활용도를 높이고, 전송 비용을 줄이기 위해서 다양한 방법으로 압축을 하여 저장을 한다[1],[5],[6],[7],[8]. 모바일 장치에서는 문서에 대해 질의 처리를 위해서 문서 전체를 압축 해제하여 질의를 처리해야 하므로 메모리 부족 등의 비효율성을 가져올 수 있다. 또한 문서에 대한 질의 처리시에 문서의 크기가 커질수록 질의를 처리하는 데에 따른 검색 시간이 길어지게 된다. 같은 문서에 대해서 질의 처리 시에 해당 문서에 대해서 질의를 내려서 처리를 하는 것보다 압축된 문서에 대해서 압축 해제 후에 질의 처리를 하게 되면 질의 처리 시간이 훨씬 줄어들게 된다. 하지만 XML문서를 압축할 때, 압축된 문서에 대해 질의 처리를 가능하게 하면 질의 처리에 대한 처리 성능이 훨씬 좋아지게 된다. 즉, 압축된 문서에 질의 할 때, 그 문서에 맞게 변환하여 질의를 내리고 해당 질의에 대한 결과는 압축 해제후의 질의 처리보다 좋은 성능을 나타낼 수 있다.

XML문서에 대한 압축기법들이 많이 연구되고 있다. gzip은 XML문서를 포함한 일반 문서들에 사용되는 압축방법이다[2],[3]. gzip은 실제 XML문서구조의 특성에 맞게 적용된 것이 아니다. XML구조적 특징을 이용한 것으로 xmill 압축 알고리즘이 있다[4]. xmill은 XML특성인 엘리먼트내의 태그와 실제 텍스트 데이터를 가지고 있는 특성을 이용한다. 질의 처리시에는 문서를 압축 해제 후 질의 처리 과정을 거치게 된다.

Semantic Lossy Compression of XML Data 알고리즘은 xmill의 대부분을 이용하고 문서의 구조를 다차원(multidimension)으로 표현한다[9]. 문서에서 표현되는 실제 데이터들을 모두 손실 압축 하는 것이 아니라 몇 몇 부분에 대해서만 손실 압축을 사용한다. 이 방법 역시 질의 처리를 위해서는 압축된 문서의 모든 내용을 압축 해제하는 오버헤드가 있다. 질의 처리에 대한 오버헤드를 줄이고 압축된 문서에 대해 질의 처리를 가능하게 해주는 알고리즘으로써 XGrind라는 알고리즘이 있다[10].

XGrind는 XML문서에 압축 알고리즘을 적용하여 압축을 하고 압축된 문서에 대해서 압축 해제 후에 질의 처리를 하는 오버헤드를 줄일 수 있는 구조를 가지고 있다. 즉, 압축된 문서 자체 내에서 질의 처리에 대한 결과를 얻어 낼 수 있도록 하여 성능이 좋아진다.

본 연구에서는 기존의 압축 알고리즘들에 대한 문제점을 보완하고 보다 나은 성능을 가지는 알고리즘을 구현하고자 한다. DTD문서를 파싱하여 각각의 엘리먼트와 애트리뷰트들에 대한 부호화된 DTD 테이블 문서를 생성한다. DTD 부호화 테이블은 엘리먼트와 애트리뷰트들에 대한 가변 비트가 부여된 값들이다. 이 과정을 거친 후에 XML문서 파서(parser)를 통해 XML문서를 파싱하면서 부호화된 DTD테이블을 이용하여 문서의 메타 데이터들에 대해서 부호화 테이블을 구성하고, 문서의 각 내용들에 대해서는 내용 분포 테이블을 구성하고 각각의 내용들을 토큰화한다. 이 두 과정을 거쳐 메타 데이터 부호화 및 허프만 압축과정(huffman-encoding)을 거쳐 압축된 XML 문서를 작성하게 된다. 압축된 문서에 대한 질의 처리는 질의 압축기를 통해 사용자의 질의를 압축을 하고, 각각에 따라서 구조 질의 및 내용 검색 질의 처리

를 할 수 있도록 한다. 본 논문의 구성은 다음과 같다. 먼저 II장에서 기존의 연구에 대하여 논의한 뒤 본 논문의 접근 방향을 서술하고, III장에서 제안하는 XML 문서 압축 알고리즘의 압축 및 질의 처리기의 구조에 대해서 서술한다. IV장에서는 성능 평가를 통하여 제안하는 기법의 우수성을 증명한다. 마지막으로 V장에서 결론을 맺는다.

## II. 기존의 XML 문서 압축 알고리즘의 구조 및 접근 방향

일반 데이터의 크기를 줄이기 위해 여러 압축 알고리즘들을 사용한다. 대표적인 압축 알고리즘으로는 크게 손실 압축과 무 손실 압축으로 구분되어진다. 문서에 대해 압축을 적용하는 알고리즘은 문서의 각 텍스트들이 손실 없이 압축을 해야하기 때문에 무손실 압축기법을 사용한다. 이러한 일반 문서에 대한 압축 알고리즘으로는 gzip, LZ77 알고리즘을 많이 활용하고 있다. 이러한 일반문서의 압축 알고리즘을 XML문서에 적용할 수도 있다. 하지만 XML문서의 구조적 특성을 고려하여 압축을 한다면 압축률에 대한 성능은 좋아진다. XML문서에 대한 알고리즘으로는 xmill, Semantic Lossy Compression of XML Data, XGrind 등이 있다. 각각의 XML문서에 대한 압축 알고리즘의 내용은 다음의 각 절에서 자세히 설명한다.

### 1. xmill

xmill은 XML의 구조적 특성 즉, XML 태그와 애트리뷰트의 메타 데이터와 실제 문서에 보이는 내용의 데이터 값으로 나누어진다. 이러한 특징을 이용하여 메타 데이터에 대해서는 토큰화를 시키고 각각에 대하여 사전 부호화 방식으로 부호화하고 실제 문서 표현의 데이터 값들에 대해서는 부모 태그에 대한 컨테이너들을 할당한다.

부호화 하는 방법은 각각의 엘리먼트와 애트리뷰트는 문자캐릭터와 숫자의 조합으로 이루어진다. 엘리먼트와 애트리뷰트의 끝은 기호 '/' 로 표현한다. 콘텐츠들에 대한 압축은 각각의 특성을 고려한 데이터들을 컨테이너에 그룹화 시킨다. 즉, 엘리먼트로 묶여있는 콘텐츠들을 고려하여 각 컨테이너들에 대해서 해당 값들의 특성에 맞춰 줄-길이 부호화(run-length encode), 열거형 부호화(enumeration encode), 상수 압축기(constant compressor)등의 각각에 맞는 압축 기법을 사용한다. xmill 알고리즘으로 XML문서를 압축 하면, gzip알고리즘보다 압축률이 두 배 이상 높아진다. 질의 처리를 위해서는 압축된 XML문서들을 모두 압축해제 후에 질의 처리를 해야 하므로 질의 처리시간이 길어지는 단점이 있다.

### 2. Semantic Lossy Compression of XML Data

*Semantic Lossy Compression of XML Data*알고리즘은 xmill의 압축 방법을 그대로 적용하여 사용한다. xmill과 다른 특징은 메타 데이터의 구조를 다차원으로 표현하고, 콘텐츠들의 몇몇 부분에 대해서 손실 압축을 하게 된다. 손실 압축 방식을 적용하는 예로는 다음과 같다. 순서를 가지고 있는 수들에 대한 Wavelet기반의 압축, 문자열에 대한 손실 압축, 접속사나 관사 등을 제거하는 일반 텍스트 손실 압축 등이 있다.

이 압축 알고리즘을 사용하면 xmill보다 압축률이 좋아진다. 하지만 에러 요소를 고려해야한다. 즉, 문서는 손실 압축 알고리즘을 사용하여 압축을 하고 난 후에 다시 압축 해제를 할 때 원본데이터와 같은 내용을 가지고 있어야 한다. 이 알고리즘을 적용하면 압축된 문서의 압축 해제시 원본 문서와 다른 내용이 될 수 있기 때문에 치명적일 수가 있다.

### 3. XGrind

이전의 압축 알고리즘들은 주된 연구방향이 XML 문서에 대한 압축률을 높이는 것이었다. 그러나 질의 처리에 대한 고려를 하지 않아, 압축된 문서의 압축 해제 후에 질의를 처리하는 과정을 거친다. 따라서, 질의 처리 시간이 오래 걸리는 단점이 있다. XGrind는 최근의 압축 알고리즘이다. XGrind는 문서에 대한 질의를 내렸을 때, 압축된 문서에 대하여 압축 해제 후에 질의를 처리하지 않고, 직접 압축된 문서 자체에 질의를 입력하여 처리를 한다. 해당 질의에 대한 결과 값이 있다면 압축문서 중에 그 해당하는 부분 또는, 문서 전체를 압축 해제하여 질의를 내린 사용자에게 되돌려준다. 메타 데이터의 표현은 xmill 압축 알고리즘 기법의 표현 방법과 비슷하다. 우선 엘리먼트에서 시작 태그는 'T'와, 엘리먼트들의 구별 및 레벨을 나타내는 숫자의 조합이다. 애트리뷰트에서는 'A'와 애트리뷰트들의 구별을 짓기 위한 숫자의 조합으로 구성된다. 그리고 엘리먼트와 애트리뷰트의 끝을 나타내기 위해 '/'를 사용한다. 실제 문서의 내용에 대해 적용하는 알고리즘은 비적응형 허프먼 압축 알고리즘(non-adapive huffman compression algorithm)이다. 메타 데이터의 부호화와 컨텐츠들의 문맥 자유 압축 스키마에서 얻어진 결과값으로 XML과 같은 반구조화(semi-structured)상태의 압축된 문서를 만든다. 그림 1은 XGrind의 전체 구성도이다.

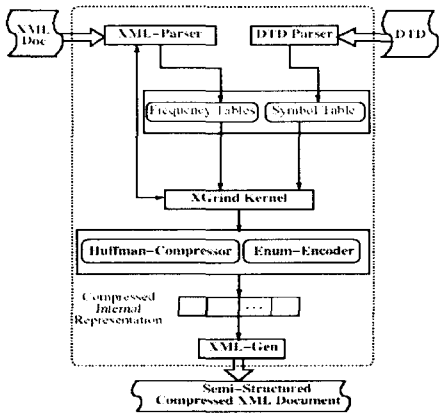


그림 1. XGrind 구성도

DTD파서와 XML파서를 실행하여 문서에 대한 유효성을 검사하고 두 번의 파싱 작업을 수행한다. 처음 파싱작업에서는 각 엘리먼트와 비 열거형 애트리뷰트들에 대해 빈도(frequency table) 테이블을 구성하고 초기화 시킨다. DTD문서에서 얻어진 메타데이터들 중에 열거형 애트리뷰트들은 심볼 테이블(symbol table)에 올려놓는다. 다시 한번 파싱 작업을 수행하여 처음 파싱 작업으로 얻어진 빈도테이블에 각 엘리먼트와 비 열거형 애트리뷰트에 대해 분석된 통계량들을 테이블에 작성한다. 재파싱하는 과정에서 각각의 태그, 애트리뷰트, 데이터 값들을 토근화된 형태를 만든다. 허프만 압축알고리즘과 열거형 애트리뷰트 압축 알고리즘을 이용하여 압축 작업을 수행한다. 이전에 심볼 테이블에 구성해놓은 열거형 애트리뷰트들에 대해서는 열거형 압축 알고리즘을 적용하여 압축을 한다. 이러한 과정을 거치고 나서 압축된 문서의 형태를 구성한다.

질의 처리과정은 문법 분석기와 질의 파서를 통해 경로와 메타 데이터를 검출한다. 검출한 경로와 메타 데이터를 가지고 질의 처리를 수행한다.

XGrind의 특징은 압축된 문서가 반구조화(semi-structure)되어 있다는 점이다. 질의 처리시에 압축된 문서를 해제하지 않고 직접 압축된 문서에 질의 처리를 하고 그 부분에 대한 압축 해제를 하여 결과를 얻어내는 방식을 취하고 있다.

### 4. 본 논문의 접근 방향

이상에서 언급된 압축 일고리즘을 정리하면 다음과 같다. xmill과 Semantic Lossy Compression of XML Data는 압축률을 높이는데 중점을 두고 있다. 그러나 이 기법들은 압축된 문서에 질의 처리를 지원하지 않는다. 질의 처리를 위해서는 압축된 문서를 압축 해제한 후에 질의에 대한 결과 값을 얻어낸다. XGrind는 이전의 두 압축 기법에 비해 압축률이나 압축시간 면에서 성능이 떨어진다. 하지만 이 기법은 압축된 문서에 대한 질의 처리시에 압축 해제를 하지 않고, 직접 압축된 문서에 질의를 처리하여 결과값을 반환해준다.

본 논문에서는 xmill과 Semantic Lossy Compression of XML Data에서의 질의 처리를 하지 못하는 문제점을 해결하고, XGrind에서의 압축률 및 압축시간에 대한 오버헤드를 줄여 효과적인 알고리즘을 제시한다. 또한 보다 향상된 질의 처리를 지원할 수 있는 질의 처리 알고리즘을 제안한다.

## III. XML 문서 압축 알고리즘

본 논문에서 제안하는 압축 알고리즘은 XML문서에 대해 문서 구조를 이용한 효율적인 압축을 한다. 또한, 압축된 문서에 대해서 압축 해제를 하지 않고 직접 압축된 문서에 효율적으로 질의 처리를 하는 알고리즘을 설계 및 구현 하고자 한다. 본 논문에서 제안하는 압축 알고리즘은 XENDER(XML Encoder)라 명명한다. 이 장에서는 다음의 큰 두 가지 알고리즘을 보인다. 우선 XML 문서를 효율적으로 압축하기 위한 압축 알고리즘에 대해서 설명한다. 다음으로 압축된 문서에 압축 해제를 하지 않고 직접 압축된 XML 문서에 질의 처리를 지원하는 알고리즘을 설명한다.

### 1. 제안하는 압축 알고리즘

#### 1.1. 문서 압축 알고리즘 구성도

제안하는 알고리즘의 전체적인 구성은 다음과 같다. 그림 2의 XENDER의 구성도에서 XML 문서에 대해서 SAX 파서로 파싱 과정을 수행한다. 파싱 과정에서 얻어지는 메타 데이터에 DTD문서의 각 엘리먼트 및 애트리뷰트 정보 값들을 가지는 DTD 부호화 테이블 문서를 이용하여 각각의 메타 데이터에 대한 가변비트 부호화 테이블을 구성한다.

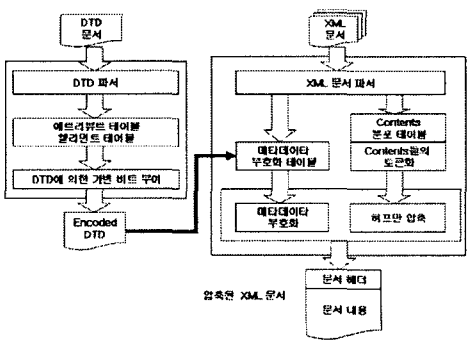


그림 2. XENDER 구성도

메타데이터 이외의 내용들 즉, 데이터들에 대해서는 파싱 과정에서 내용 분포테이블을 구성하고 각각의 데이터를 토큰화하여 테이블에 올려놓는다. XML문서의 파싱 과정을 마친 후에 XML압축 생성기를 통하여 메타 데이터에 대한 부호화 테이블을 가지고 각각의 메타 데이터에 대한 부호화 작업과 허프만 압축과정을 통해 얻어지는 압축된 문서를 생성한다.

### 1.2. DTD 부호화 테이블 문서 작성

그림 2의 XENDER 전체 구성도에서 압축과정을 수행할 때 메타데이터에 가변 비트를 부여하는 테이블을 구성해야 한다. DTD 부호화 테이블 문서에서 그 테이블의 역할을 수행한다. DTD 부호화 테이블 문서는 실제 엘리먼트 및 애트리뷰트에 대한 정보와 각각의 가변비트 값을 가진다. 이 정보를 가지고 문서에 대해 두 번의 파싱을 거치지 않고도 질의 처리가 가능한 압축된 XML 문서를 생성할 수 있다. 그림 3과 같은 과정을 통해 DTD 문서를 파싱하여 부호화된 DTD 테이블 문서를 생성한다.

각각의 엘리먼트에 대한 자식 엘리먼트에 부여된 가변 비트를 할당하고 해당 엘리먼트들의 실제 데이터의 유무, 그리고 '?', '*', '|', '+'등의 표시자에 대한 정보를 가지고 있다. 애트리뷰트에 대해서는 해당 애트리뷰트에서 부여된 실제 내용들의 가변 비트값 및 표시자등에 대한 정보를 가지고 있다.

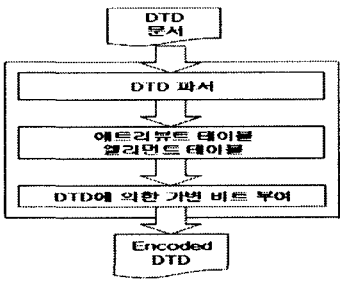


그림 3. DTD 부호화 테이블 성과정

이러한 DTD 부호화 테이블 문서를 먼저 생성하고 압축 문서 작성을 위한 과정을 수행한다. 다음은 DTD 문서를 이용하여 DTD 부호화 테이블에 대한 가변 비트 구성의 내용을 보여준다.

주어진 DTD문서를 가지고 DTD 부호화 테이블 문서를 구성하는 방법은 다음과 같다. 우선 최상위 루트 엘리먼트가 가질 수 있는 자식 엘리먼트의 수를 확인하고, 가변비트를 부여한다. 즉, 루트 엘리먼트가 가지는 자식 엘리먼트를 표현하기 위해 부여되는 가변비트 수는 ⌈log2K⌉ (K는 정수)로 표현한다. 위의 예에서 book의 자식 엘리먼트는 3개이다. 식 log2K에 대입하면 1.5849가 나오게 된다. 여기서 주어진 값을 넘는 최소 정수가 가변 비트수이기 때문에 자식 엘리먼트를 표기하기 위한 비트수는 2이다. 자식 엘리먼트는 title, author, section이므로 각각을 표현하려면 00(title), 01 (author), 10(section)으로 표기하면 각각의 엘리먼트를 구분지을 수 있다. 그리고 각각의 자식엘리먼트가 하위 엘리먼트를 가지고 있다면 각각의 엘리먼트에서 자식 엘리먼트의 부호화를 위한 가변비트를 설정한다. 그림 4와 5에서 DTD문서를 이용하여 DTD 부호화 테이블 문서를 구성한 예를 보인다.

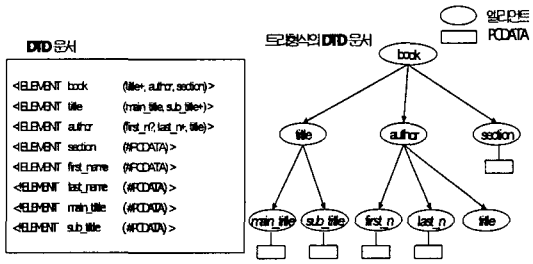


그림 4. DTD 문서

**Encoded DTD**

| | |
|---|---|
| book | [ title : 00 author : 01 section : 10 ] |
| title | [ main_title : 0 sub_title : 1 ] |
| author | [ first_n : 00 last_n : 01 title : 10 ] |

그림 5. DTD 부호화 테이블

### 1.3. 문서 압축 처리 과정

XML 문서압축 과정은 부호화된 DTD를 이용하여 XML문서를 압축한다. 그림 6은 XML문서의 예이다.

```
XML Document
<book>
    <title>
      <main_title> Hamlet </main_title>
      <sub_title> drama </sub_title>
      <sub_title> four tragic drama</sub_title>
    </title>
    <title>
      <main_title> Romeo & Juliet</main_title>
    </title>
    <author>
      <first_n> William </first_n>
      <last_n> Shakespeare</last_n>
      <title>
        <main_title>King Lear</main_title>
      </title>
    <section> Hamlet is without question the most famous play
in the English language. Probably written in 1601 or 1602....</section>
</book>
```

그림 6. XML 문서 예

DTD 부호화 테이블 문서에서의 메타 데이터 정보를 그림 6의 XML문서의 메타 데이터에 적용한다. 생성한 DTD 부호화 테이블 문서를 이용하여 해당 메타데이터에 가변 비트를 부여한다. 그림 7은 가변 비트가 부여된 형태의 XML 문서를 보여주고 있다.

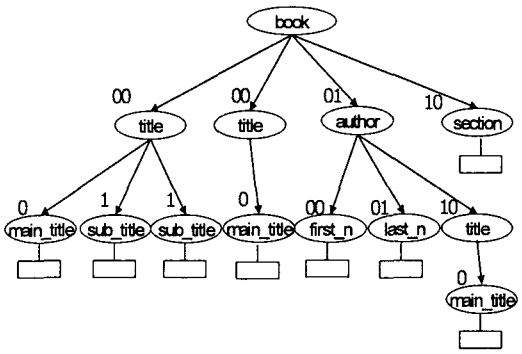


그림 7. 가변비트 포함한 XML 문서 트리

문서상에 표현되는 데이터 값들에 대해서는 내용 분포 테이블을 이용하여 각각의 데이터들을 허프만 방식의 부호화 과정을 수행한다. 우선 최초의 루트 엘리먼트가 가지는 자식 엘리먼트들의 가변 비트 부호화 값을 나열한다. 헤더부분에 가변 비트로 표현된 전체 크기와 자식 엘리먼트의 개수를 각각의 엘리먼트들에 대한 가변비트 부호화 값을 나열하기 전에 작성한다. 애트리뷰트가 있다면 엘리먼트에 나타나는 각각의 애트리뷰트의 개수와 할당된 크기를 헤더 부분에 작성하고 그후에 애트리뷰트의 부호화 값을 나열한다. 자식 엘리먼트들의 각각에 대해 PCDATA 또는 CDATA를 가지고 있으면 분포 테이블에 저장되어 있는 각각의 내용들을 허프만 압축 알고리즘을 이용하여 압축한다. 압축할 때, 해당 엘리먼트들의 자식 정보를 이용하여 압축된 데이터의 내용이 어느 엘리먼트에 대한 내용인지를 알 수 있도록 헤더 부분에 정보를 추가한다. 그림 8은 각 레벨 단위의 압축정보이다.

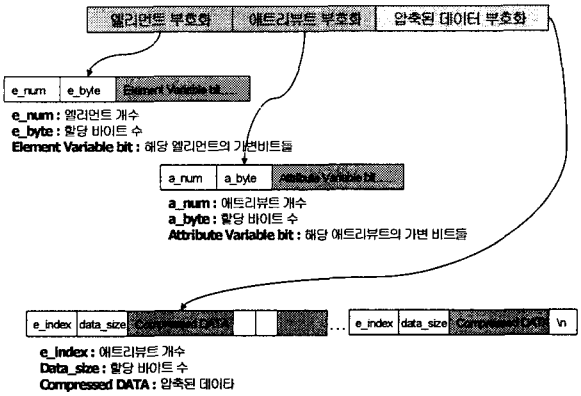


그림 8. 레벨단위의 압축 정보

그리고 마지막 엘리먼트와 엘리먼트 사이의 구분 즉, 각 레벨단위의 구분은 '\n'으로 표기한다. 각 레벨에서의 엘리먼트를 구분하여 다음 레벨의 정보를 확인할 수 있다. 그림9는 같은 레벨 단위의 압축된 정보를 보여주고 있다.

이렇게 생성된 압축된 문서의 내용은 그림 9와 같다.

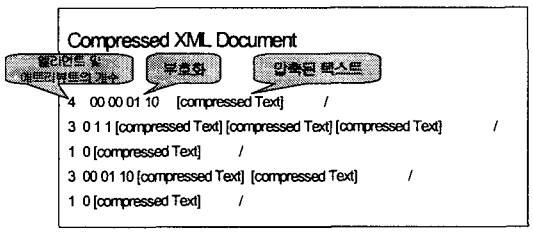


그림 9. 압축된 XML 문서

## 2. 질의 처리 알고리즘

질의 처리는 앞 절의 과정을 거쳐 압축된 문서를 압축 해제하지 않고 직접 압축된 문서에 대해서 질의를 처리한다. 일반적인 압축 방법을 이용하여 XML 문서를 압축하게 되면, 압축된 문서에 대한 질의 처리는 많은 오버헤드를 발생시킨다. 즉, 압축된 문서를 압축 해제를 하고 질의 처리를 해야 한다. 이 절에서는 앞 절의 알고리즘을 이용해 생성된 압축된 문서에 대해 압축 해제하지 않고 직접 질의를 압축하여 처리하는 알고리즘을 보인다.

### 2.1. 질의 처리 구성도

질의 처리과정은 그림 10과 같다.

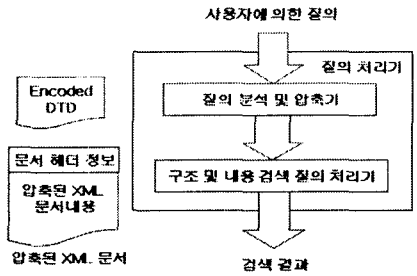


그림 10. 질의 처리 구성도

우선 질의 분석기를 통하여 사용자의 요청에 의한 질의를 분석한다. 분석한 질의 내용에서의 각 엘리먼트와 애트리뷰트들에 대한 정보를 DTD 부호화 테이블 문서를 통해 요청한 질의를 가변 비트 부호화 및 질의 압축을 하게 된다. 이러한 과정 후에 질의 처리기를 통해 압축된 질의를 생성한다. 앞 절에서 구현한 압축 알고리즘을 이용하여 생성된 압축된 XML 문서에 질의 처리기를 통해 압축된 질의를 한다. 질의 결과를 얻어내어 원하는 결과의 여부를 확인하고 질의 값을 반환한 후에 마치게 된다.

### 2.2. 질의 처리

그림 11은 질의 처리 과정이다. 사용자가 압축된 XML 문서에 질의를 내리게 되면 질의 처리기에서는 DTD 부호화 테이블 문서의 메타데이터 테이블을 가져온다. 가져온 메타데이터 테이블의 메타 데이터를 분석한 사용자의 질의에 매핑시켜 사용자의 구조 및 내용 검색질의를 얻어낸다. 엘리먼트와 애트리뷰트 그리고 실제 내용의 데이터로 토큰화 한 후에 질의 압축기를 통해 압축된 XML문서에 맞게 압축을 시킨다. 그림 11의 예에서 사용자의 질의가 title내의 main_title을 모두 검색하는 요구라 할 때, title 엘리먼트 내의 main_title이 포함된 모든 부모 자식관계의 엘리먼트들을 찾는다.

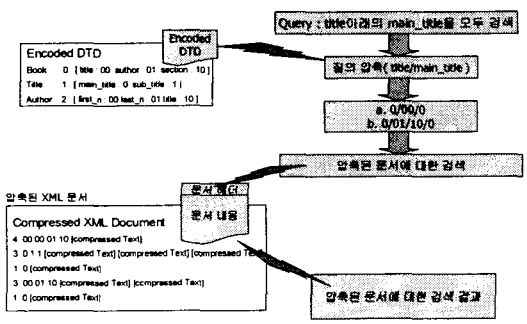


그림 11. 질의 처리 과정

찾은 결과 값을 가지고 압축된 문서에 질의 처리를 할 수 있도록 질의에 DTD 부호화 테이블의 가변비트를 부여한다. 그림 11에서 보여지는 title내의 main_title에 대한 압축된 질의는 두 가지이다. 하나의 질의는 title의 부모 엘리먼트가 Book인 경우와 또 다른 하나의 질의는 title의 부모 엘리먼트가 author인 경우이다. 이 두 가지의 질의를 질의 압축기로 질의를 내리면 [0 00 0]의 질의와 [0 01 10 0]이다. 이 질의에서 각각의 루트 레벨의 엘리먼트는 명시적으로 나타나는 부분이기 때문에, 깊이 1부터 압축된 문서를 검색한다. 즉, 루트 엘리먼트의 자식 엘리먼트들로부터 시작하여 해당 질의에 대한 값을 찾아 들어가면서 값을 확인하고 결과를 출력한다. 위 예에서 나온 결과는 그림 12이다.

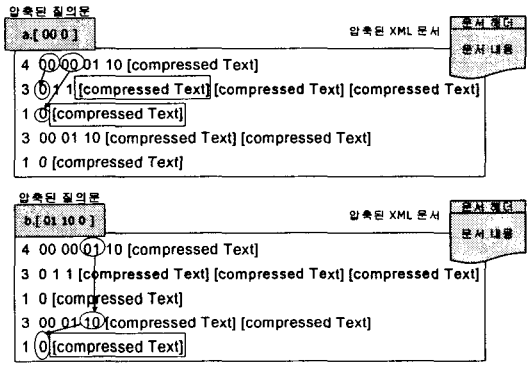


그림 12. 질의 처리 결과

## IV. 구현 및 성능평가

제안한 알고리즘에서의 성능평가는 XGrind와 xmill을 비교하여 다음과 같은 세 부분으로 나누어 측정하였다. 기존 문서를 압축했을 때의 성능을 나타내는 압축률 비교, 압축을 하는데 있어서의 압축 처리 시간, 그리고 질의 처리시간의 세 가지를 비교하였다.

### 1. 구현 환경

구현 환경은 OS로 리눅스 9.0을 채택하였다. CPU는 PENTIUM-III 700Mhz이고 RAM은 256MB의 사양에서 진행을 하였다.

### 2. 측정 데이터

표 1. XML 문서 종류

| XML Document | Size | Depth | Element | Attrs | Enum |
|---|---|---|---|---|---|
| shakespeare | 5MB | 6 | 22 | 0 | 0 |
| student | 3MB | 3 | 6 | 2 | 2 |
| university | 4MB | 4 | 16 | 10 | 10 |

shakespeare는 쉐익스피어의 희극을 XML문서로 변환한 것으로 크기는 5MB이고, 문서의 깊이(depth)는 최대 6까지 내려간다. 엘리먼트로만 구성이 되어 있으며 DTD에 정의된 엘리먼트타입의 개수는 22개이다. 그리고 student와 university는 성능 평가를 위해 임의로 구성한 XML문서이다. student 문서는 엘리먼트수가 적고, 애트리뷰트가 모두 열거형 타입의 데이터이다. university는 16개의 엘리먼트에 10개의 애트리뷰트중에 5개의 열거형을 갖는다.

## 3. 측정 결과

(1)압축률

각각의 구현된 알고리즘을 이용한 방법에서의 압축률 비교는 다음과 같다. 압축률 계산은 다음과 같은 식을 통해서 얻어진다.

$$\text{압축률} = 1 - \frac{\text{압축된 } XML \text{문서의 크기}}{\text{실제 } XML \text{문서의 크기}} \quad (1)$$

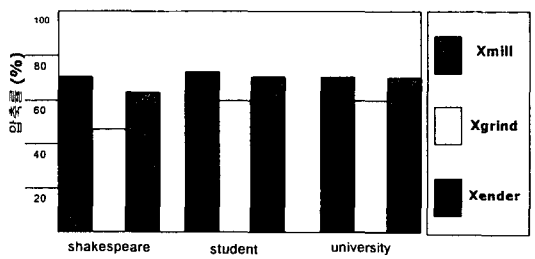


그림 13. 압축률

그림 13은 압축률 비교 평가이다. 압축률에서는 xmill이 가장 좋은 압축률을 보이고 있다. 그 이유는 xmill은 압축률만을 고려하여 태그로 묶여진 데이터들에 대해서는 산술형 압축 알고리즘을 적용하였기 때문이다. 본 논문에서 제안하고 있는 XENDER 압축 알고리즘은 질의 처리가 가능하도록 실제 데이터들에 대해서는 허프만 압축 알고리즘을 적용하였다. 그럼에도 xmill과의 비교에서 압축률에서 큰 차이를 보이고 있지 않다. 그 이유는 메타 데이터로 가변 비트를 사용하고 있기 때문에, 바이트 단위로 부호화하는 xmill과 비교하여 큰 차이가 없다. XGrind에서는 압축률이 가장 좋게 나온 경우가 애트리뷰트들이 열거형을 가진 경우이다. 하지만 그림 13에서 알 수 있듯이 student는 애트리뷰트들을 열거형으로 갖는 XML문서임에도 본 논문에서 제안한 알고리즘보다 압축률이 떨어진다. 이 부분 역시 메타 데이터를 표현하는데 있어서 가변 비트를 사용하고 있기 때문이다.

(2)압축시간

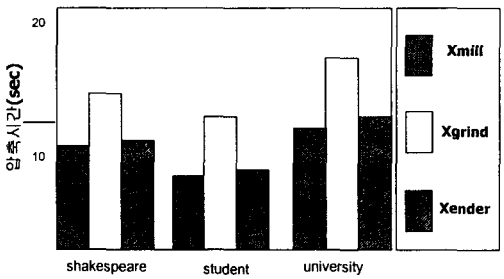


그림 14. 압축 시간

그림 14는 압축 시간을 비교 평가한 내용이다. DTD문서를 이용하여 DTD 부호화 테이블 문서로 작성하는데 걸리는 시간은 측정시간에서 제외시켰다. 그 이유는 DTD 부호화 테이블 문서를 생성하는데 걸리는 시간은 압축시간 성능 평가에 영향을 주지 않는 정도로 작은 밀리초단위(ms)의 값을 갖기 때문에 포함시키지 않았다. 압축 시간 비교에서는 shakespeare문서가 university 문서에 비해서 문서량이 많긴 하지만 메타데이터로 엘리먼트만을 가지고 있기 때문에 압축시간에서 더 적게 나오는 것을 보이고 있다. XGrind 알고리즘이 다른 알고리즘에 비해서 압축시간이 많이 나오는 이유는 다음과 같다. xmill과 본 논문에서 제시한 XENDER는 한번의 파싱으로 압축 작업을 수행하지만, XGrind에서 사용하는 Huffman 압축이 Huffman 트리를 탐색하면서 압축값을 부여하므로 일반적으로 Dictionary 압축에 비해 압축시간이 많이 소요되어 약 1.5배 이상의 차이를 보이고 있다.

(3) 질의 처리 시간

아래 그림 15은 질의 처리 시간에 대한 성능 평가이다. 질의 처리 시간의 결과는 다음과 같다. xmill은 압축된 문서에 대한 질의 처리를 지원하지 않기 때문에 평가대상에서 제외 시켰다. 본 논문에서 제시하는 알고리즘의 질의 처리에

따른 시간 측정은 XGrind보다 향상된 질의 처리시간을 보이고 있다. 질의 처리 시에 DTD 부호화 테이블 문서를 이용하여, 질의를 압축된 문서에 맞게 가변 비트를 부여하고 질의 처리를 한다. 질의 처리 시에 메타 데이터를 읽어서 확인 시에 같은 레벨의 메타데이터를 한꺼번에 읽어서 확인하기 때문에 이전에 제안한 XGrind 알고리즘보다 향상된 질의 처리 시간을 보이고 있다. 압축된 질의에 대한 오버헤드는 크지 않다.

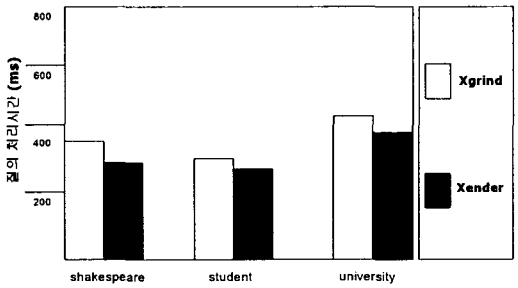


그림 15. 질의 처리 시간

## V. 결론

본 논문에서는 XML문서를 효율적으로 저장, 관리하기 위하여 질의 처리가 가능한 XML 문서 압축 알고리즘을 설계하고 구현하였다. XML문서의 구조적 특성 즉, 구조표현에서의 메타데이터의 표현을 부호화한다. XML문서 파싱 후에 DTD 부호화 테이블 문서에 있는 메타 데이터의 가변비트 및 엘리먼트 정보를 구성한다. 메타데이터 정보와 실제 텍스트들의 정보를 DTD 부호화 테이블 문서와 허프만 압축 알고리즘을 이용하여 압축된 XML문서를 구성하였다. 또한 질의 처리 시 압축을 해제 한 후에, 요청한 질의 처리의 오버헤드를 줄이기 위해 압축된 문서 자체에 대해 질의를 처리하는 알고리즘을 제안하였다. 질의 처리 시 압축된 문서의 부호화된 메타 데이터 형식에 맞게 질의를 압축한 후에 처리한다.

향후 연구방향으로는 xmill 압축 알고리즘 방법보다 떨어지는 압축률을 더 높여보고자 한다. 압축률에서 주어진 XML 문서에서의 확장을 위하여 적용되는 엘리먼트와 애트리뷰트에 대한 표시자들의 처리 문제를 보다 명확성 있게 해야할 필요가 있다. 또한 부호화된 DTD테이블 문서에 대한 유효성 검사의 범위가 필요하다. 또한 데이터 값의 형식에 대한 내용 부분에서 CDATA와 PCDATA의 구별을 하여 실제로 적용되는 문서에 대한 접근을 더 높여보고자 한다.

## 참 고 문 헌

[1] H. K. Reghbati. "An Overview of Compression Techniques," IEEE Computer, April 1981.

[2] "gzip," http://www.gzip.org

[3] "gzip algorithm," http://www.gzip.org/algorithm.txt

[4] H. Liefke and D. Suciu, "XMill: An Efficient Compressor for XML Data," Proc. of ACM SIGMOD Intl. Conf. on Management of Data, May 2000.

[5] E. Lefons, A. Silvestri, and F. Tangorra. "An Analytic Approach to Statistical Databases," Proc. of VLDB Conf., October 1983.

[6] D. Batory. "Index Coding: A Compression Technique for Large Statistical Databases," Proc. of 2nd Intl. Workshop on Statistical Database Management, September 1983.

[7] G. V. Cormack. "Data Compression in Database Systems," Comm. of ACM, December 1985.

[8] B. R. Iyer and D. Wilhite. "Data Compression Support in Databases," Proc. of VLDB Conf., September 1994.

[9] Cannataro M., Carelli G., Pugliese A., Sacc, D., "Semantic lossy compression of XML data," 8th Int. Workshop on Knowledge Representation meets Databases.

[10] P. M. Tolani and J. R. Haritsa. "XGRIND: A Query-friendly XML Compressor," In Proceedings of 18th International Conference on Database Engineering, February 2002.

이 석 재(Seok-Jae Lee) 정회원

2000년 2월 : 충북대학교 정보통신공학과(공학사)

2002년 2월 : 충북대학교 정보통신공학과(공학석사)

현재 : 충북대학교 정보통신공학과 박사과정

<관심분야> : 데이터베이스 시스템, 메모리 상주형 데이터베이스 시스템, 저장 시스템

강 영 준(Young-Jun Kang) 준회원

2002년 2월 : 충북대학교 컴퓨터공학과 (공학사)
2002 ~ 현재 : 충북대학교 정보통신공학과(석사과정 재학)

<관심분야> : DBMS, XML, Multi-DBMS

유 재 수(Jae-Soo Yoo) 종신회원

1989년 2월 : 전북대학교 컴퓨터공학과 (공학사)
1991년 2월 : 한국과학기술원 전산학과 (공학석사)
1995년 2월 : 한국과학기술원 전산학과 (공학박사)
1995 ~ 1996 : 목포대학교 전산통계학과 전임강사
1996 ~ 현재 : 충북대학교 전기전자컴퓨터공학부 부교수
<관심분야> : 데이터베이스 시스템, 정보검색, 멀티미디어 데이터베이스, 분산 객체 컴퓨팅

조 기 형(Ki-Hyoug Cho) 정회원

1966년 2월 : 인하대학교 전기공학과 (공학사)
1984년 2월 : 청주대학교 산업공학과 (공학석사)
1992년 2월 : 경희대학교 전자공학과 (공학박사)
1988 ~ 현재 : 충북대학교 전기전자컴퓨터공학부 교수
<관심분야> : 데이타베이스, 소프트웨어 시스템 설계 및 구현, 컴퓨터 네트워크 설계