압축된 문서에 대해 질의 처리를 지원하는 XML 압축 알고리즘

The XML Compression Algorithm Supporting Query Processing For Compressed Documents

강영준, 이석재, 유재수
충북대학교 정보통신공학과

Kang Young-Jun, Lee Seok-Jae, Yoo Jae-Soo
Dept. of Computer and Communication Eng.,
Chungbuk National University

요약

인터넷의 급속한 확산에 따라 사회 전반의 디지털화와 지식정보화가 급속도로 진행되고 있다. 특히 많은 사용자들은 웹상에서 다양한 작업을 하고 서비스를 이용하고 있다. 이러한 작업들의 대부분은 XML을 이용한다. XML은 개발자가 필요시 문서의 논리구조를 정의할 수 있으며, 내용과 스타일이 분리가 되어 있어서 문서의 재사용성이 뛰어나다. 하지만 XML은 기본적으로 문서의 내용을 단순히 텍스트 형태로 다루고 문서의 구조를 표현하기 위해 많은 태그들이 추가되기 때문에 문서의 크기가 커질 수 밖에 없다. 팜탑, PDA 등의 제한된 용량을 보다 잘 활용하기 위해서는 문서를 효율적으로 압축해서 사용할 필요가 있다. 이를 해결하기 위해 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. 이에 본 연구에서는 기존의 방법들보다 효과적으로 압축을 하여 저장 공간의 효율성을 높이고, 압축된 XML 문서에 대해 질의 처리를 가능하게 하여 보다 보다 빠른 질의 처리를 할 수 있는 XML 압축 알고리즘을 설계 및 구현하고자한다.

Abstract

With the spread of internet, the digitalization and knowledge-based information are in progress. 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 basically handles the document type and adds numerous tags for representing structure of the document. To utilize the limited storage 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 in progress to solve this problem. But the existing research doesn't support the query processing for that. 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 effciently than the previous techniques.

1. 서론

최근 초고속 인터넷의 보편화로 인해 대부분의 기업체와 개인들이 인터넷 상에서 XML문서를 사용하고 있다[1]. XML 문서는 문서표현을 위해 내용을 텍스트 형태로 다루고 있으며 문서의 구조를 표현하기 위해서는 많은 태그, 애트리뷰트 등의 메타데이타를 적용해야 한

다. 이러한 이유로 문서를 표현하려는 내용보다 구조상의 틀을 표현하는 부분에 의해 문서의 크기가 커지게 된다. 또한, 팜탑, PDA 등의 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 기법에 대한 연구가 진행되고 있다. XML 문서의 데이터 크기를 줄임으로써 저장공간 활용도를 높이고, 전송 비용을 줄이기 위해서 다양한 방법으로 압축을 하여 저장을 한다[2, 11, 12, 14, 16]. 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 처리하는데에 따른 검색 시간이 길어지게 된다. 같은 문서에 대해서 질의 처리시에 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 XML 문서를 압축할 때, 압축된 문서에 대해 질의 처리를 가능하게 하면 질의 처리에 대한 처리 성능이 훨씬 좋아지게 된다. 즉, 압축된 문서에 대해서 질의를 그 문서에 맞게 변환하여 질의를 내리고 해당 질의에 대한 결과는 이전의 방법보다 훨씬 좋은 성능을 나타낼 수 있다. 이러한 XML 문서에 대한 압축기법들이 근래 들어서 많이 연구되고 있다. gzip은 XML 문서를 포함한 일반 문서들에 사용되는 압축방법이다[3,4]. gzip은 실제 XML 문서구조의 특성에 맞게 적용된 것이 아니다. XML 구조적 특징을 이용한 것으로 xmill 압축 알고리즘이 있다[6]. xmill은 XML특성인 엘리먼트 내의 태그와 실제 텍스트 데이타를 가지고 있는 특성을 이용한다. 이 알고리즘은 XML문서 압축에 대한 압축 효율을 강조한 알고리즘이다. 실제 질의 처리시에는 문서를 압축 해제 후 질의 처리 과정을 거치게 된다. Semantic Lossy Compression of XML Data 알고리즘은 xmill의 대부분을 이용하고 문서의 구조를 를 다차원으로 표현한다[21]. 문서에서 표현되는 실제 데이터들을 모두 손실 압축을 하는 것이 아니라 몇몇 부분에 대해서만 손실 압축을 사용한다. 이러한 방법을 이용하여 기존의 xmill의 압축 방법보다 조금 더 좋은 압축률을 보장한다. 하지만 이 방법 역시 질의 처리를 위해서는 압축된 문서의 모든 내용을 압축 해제해야 하는 오버헤드가 있다. 질의 처리에 대한 오버헤드를 줄이고 압축된 문서에 대해 질의 처리를 가능하게 해주는 알고리즘으로써 XGrind라는 알고리즘이 나오게 되었다[22]. XGrind는 XML문서에 대한 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. DTD 문서를 파싱하여 각각의 엘리먼트와 애트리뷰트들에 대한 부호화된 DTD 테이블을 생성한다. 여기서 생성된 부호화된 테이블은 엘리먼트와 애트리뷰트들에 대한 가변 비트가 부여된 값들이다. 이후에 XML 문서 파서를 통해 XML 문서를 파싱하면서 부호화된 DTD 테이블을 이용하여 문서의 메타데이타들에 대해서 부호화 테이블을 구성하고, 문서의 각 내용들에 대해서는 내용 분포 테이블을 구성하고 각각의 내용들을 토큰화 한다. 이 두 과정을 거쳐 메타데이타 부호화 및 허프만 압축과정을 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 처리를 할 수 있도록한다. 1장에서는 XML압축 알고리즘에 대한 전반적인 내용을 설명한다. 2장에서는 XML압축 알고리즘에 관련된 여러 내용을 다룬다. 3장에서는 본 논문에서 제시하는 XML 문서 압축 알고리즘인 XENDER에 대해서 설명을 하고 4장에서 성능평가에 대한 내용을 다룬다. 그리고 마지막으로 5장에서 결론으로 본 내용을 마친다.

Ⅱ. 관련 연구

데이터의 크기를 줄이기 위해 사용되는 다양한 압축 알고리즘들이 있다. 압축 기술로는 크게 손실압축과 무

손실 압축으로 구분되어진다. 문서에 대해 압축을 적용하는 알고리즘은 문서의 각 텍스트들이 손실 없이 압축을 해야 하기 때문에 무손실 압축기법을 사용한다. 이러한 일반 문서에 대한 압축 알고리즘으로는 gzip, LZ77 알고리즘을 이용하여 많이 활용하고 있다. 이러한 일반 문서의 압축 알고리즘을 XML 문서에 적용할 수도 있다. 하지만 XML 문서의 구조적 특성을 고려하여 압축을 한다면 압축률에 대한 성능은 보다 더 좋아지게 될 것이다. 이러한 XML문서에 대한 알고리즘으로는 xmill, Semantic Lossy Compression of XML Data, XGrind 등이 있다. 각각의 XML 문서에 대한 압축 알고리즘에 대한 전반적인 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 부호화 하고 실제 보여지는 문서 표현의 데이터 값들에 대해서는 부모 태그에 대한 컨테이너들을 할당한다.

부호화 하는 방법은 각각의 엘리먼트와 애트리뷰트들은 처음 문자캐릭터와 숫자의 조합으로 이루어진다. 엘리먼트와 애트리뷰트의 끝은 '/'로 표현한다.

그리고 나서 각각의 데이터들을 컨테이너에 그룹화 시킨다. 각 컨테이너들에 대해서 해당 값들의 특성에 맞춰 run-length encoder, enumeration encoder, constant compressor 등의 각각에 맞는 압축 기법을 사용한다. 이러한 방법으로 XML 문서를 압축을 하면 일반문서를 압축하는 gzip 알고리즘보다 압축하는데 필요한 같은 시간에 두 배 이상의 압축률이 더 좋아진다. 질의 처리를 위해서는 압축된 XML 문서들을 모두 압축해제 후에 질의 처리를 해야 하는 오버헤드가 있다.

2. Semantic Lossy Compression of XML Data

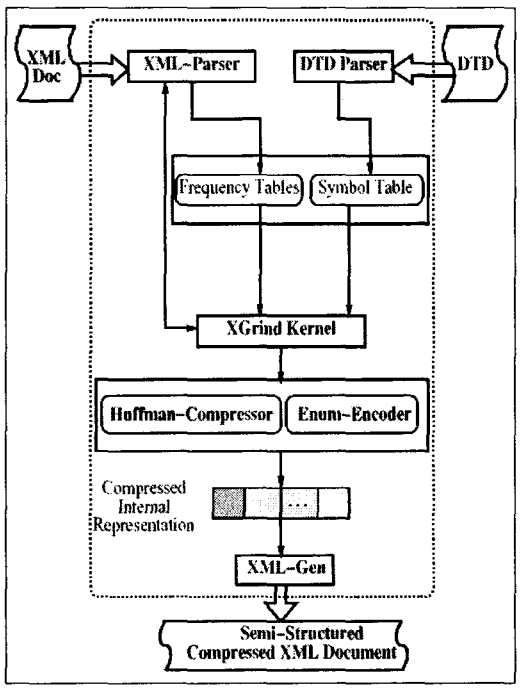
Semantic Lossy Compression of XML Data 알고리즘은 xmill의 압축 방법을 대부분 적용하여 사용한다. xmill과의 다른 큰 특징은 각각의 메타 데이터의 구조를 다차원(Multidimension)으로 표현하고, 데이터들을 모두 손실 압축을 적용하는 것이 아니라 몇몇 부분들에 대해서 손실 압축을 하게 된다. XML 문서에서 XML 문서의 구조에 대해서는 작성자에 따라 각각의 다차원 데이터큐브를 만들고 세부 레벨들에 대해서는 손실 정책을 모듈을 이용해 설정한다. 압축 방법은 메타데이터들에 대해서는 xmill에서 사용하는 태그와 애트리뷰트 표현 방식과 같다. 데이터 값들에 대해서는 그룹화 되어 있는 각 컨테이너들에 대해서 특정 컨테이너들에 대한 손실 압축 방식을 적용한다. 손실 압축방식을 적용하는 예로는 순서를 가지고 있는 수들에 대한 Wavelet기반의 압축, 문자열들에 대한 손실 압축, 접속사나 관사 등을 제거하는 일반 텍스트들에대한 손실 압축 등이 있다.

이러한 압축 방법을 사용하게 되면 압축률을 xmill보다 더 높일 수가 있다. 하지만 에러 요소를 고려해야한다. 문서는 손실 압축기법을 사용하여 압축을 한 후에 압축 해제시에 원본 데이터와 같은 내용이 있어야 하는데, 그렇지 않다면 원본 문서와 다른 내용이 될 수 있기 때문에 치명적일 수가 있다.

3. XGrind

XGrind는 지금까지 나온 기법들 중에 가장 최근의 압축 알고리즘으로써 이전 기법들이 질의 처리에 대한 고려를 하지 않았다. XGrind는 문서에 대한 질의를 내렸을 때, 압축된 문서에 대하여 압축해제 후에 질의를 처리하지 않고 직접 압축된 문서 자체에 질의를 입력하여 된 처리를 한다. 해당 질의에 대한 결과 값이 있다면 압축문서중에 그 해당하는 부분 또는, 문서 전체를 압축해제 하여 질의를 내린 사용자에게 되돌려준다. 메타 데이터의 표현은 xmill에서의 표현방법과 비슷하다. 엘리먼트에 대해서는 시작은 'T'와 엘리먼트들의 구별과 레벨을 나타내기위한 숫자의 조합으로 이루어지며, 애트리뷰트들에 대해서는 'A'와 애트리뷰트들의 구별을 짓기위한 숫자의 조합으로 이루어진다. 엘리먼트와 애트리뷰트의 끝을 나타내기 위해 '/'를 사용 한다. 일반 엘리먼트와 애트리뷰트들의 데이터값들에 대해서는 문맥 자유 압축 스키마(Context free compression

scheme)를 적용하여 사용되는 알고리즘은 비적응형 허프먼 압축 알고리즘을 사용한다. 메타 데이터와 문맥 자유 압축 스키마에서 얻어진 결과 값으로 다음의 수행 과정을 거쳐 XML구조가 가지는 semi-structured화되어 압축된 문서를 만든다.



▶▶ 그림 1. XGrind 구성도

우선 DTD 파서와 XML 파서를 실행하여 문서를 파싱한다. 각 엘리먼트 또는 비 열거형 애트리뷰트들에 대해 빈도(frequency table) 테이블을 구성하고 초기화시킨다. 열거형 애트리뷰트들에 대해서는 심볼 테이블(symbol table)에 올려놓는다. 그후에 각 엘리먼트와 비 열거형 애트리뷰트에 대해 분석된 통계량을 포함하는 빈도테이블을 올려놓는다. XML 파서를 이용하여 재파싱을 하고 각각의 태그, 애트리뷰트, 데이터 값들을 토근화된 형태를 만든다. 허프만 압축알고리즘과 열거형 애트리뷰트 압축 알고리즘을 이용하여 실제 데이터 내용들에 대해서는 허프만 압축알고리즘을 적용하여 데이터값들을 압축한다. 이전에 심볼테이블에 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 메타데이터를 검출한다. 검출한 경로와 메타데이타를 가지고 질의처리를 수행한다.

XGrind의 특징은 압축된 문서가 반구조화(semi-structure)되어 있어서 질의 처리시에 압축된 문서를 해제하지 않고 직접 압축된 문서에 질의 처리를 하는 방식을 취하고 있다. 압축시간이나 압축률 등에서는 xmill에 많이 뒤지지만 실제 질의 처리까지 고려하면 효율성이 더 높게 나타난다.

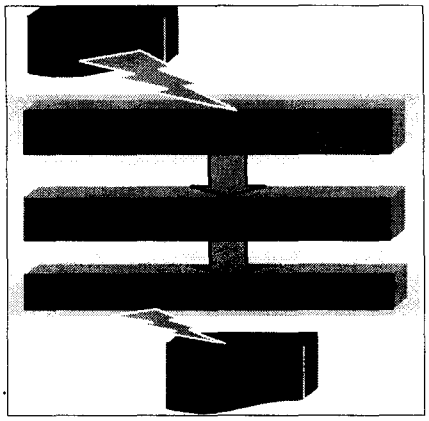
Ⅲ. XML 문서 압축 알고리즘(XENDER)

1. XENDER 구성도

본 논문에서 제안하는 XENDER 압축 알고리즘은 XML 문서에 대해 문서 구조를 이용한 효율적인 압축을 한다. 또한, 압축된 문서에 대해서 압축해제를 하지 않고 직접 압축된 문서에 대해 효율적으로 질의 처리를 하려는 알고리즘을 설계 및 구현 하고자 한다. 이 장에서는 제안하는 알고리즘에 대한 전체적인 내용을 설명한다

1.1 부호화된 DTD 테이블 문서 작성

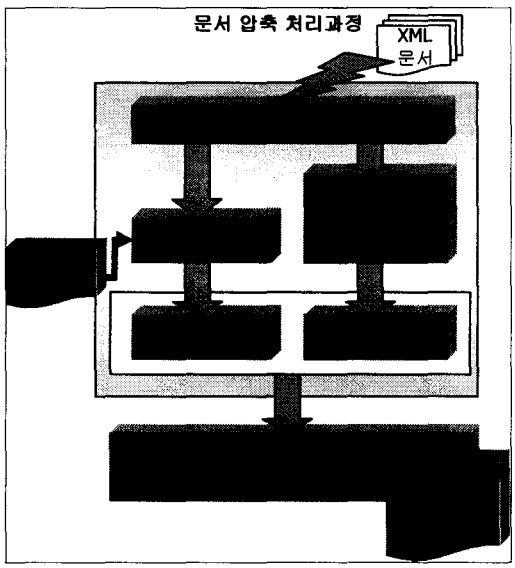
우선 XML문서를 그림2에서와 같이 DTD문서를 파싱하여 부호화된 DTD테이블 문서를 생성한다. 부호화된 DTD테이블 문서 내에 있는 항목들은 각각의 엘리먼트에서 자식 엘리먼트에 부여된 가변비트를 이용하여 할당하고 해당 엘리먼트들의 실제 데이터의 유무, 그리고 ?*|+등의 표시자에 대한 정보를 가지고 있다. 애트리뷰트에 대해서는 해당 애트리뷰트에서 부여된 실제 내용들의 가변비트값 및 표시자등에 대한 정보를 가지고 있다. 이러한 부호화된 DTD테이블 문서를 먼저 생성해 놓은 후에 다음의 과정을 수행한다.



▶▶ 그림 2 부호화된 DTD 테이블

1.2 문서 압축 전체 구성도

우선 XML 문서 파서를 가지고 파싱과정을 수행한다. 파싱과정에서 얻어지는 메타데이타들에 대해서는 이전에 작성해놓은 부호화된 DTD 테이블 문서를 이용하여 각각의 메타데이타에 대한 가변비트 부호화 테이블을 구성한다. 메타데이터 이외의 내용들 즉, 데이터들에 대해서는 파싱과정에서 분포테이블을 구성하고 각각의 데이터들을 토큰화 시킨다. 이러한 XML 문서의 파싱과정을 마치고나면 다음의 과정을 수행한다.



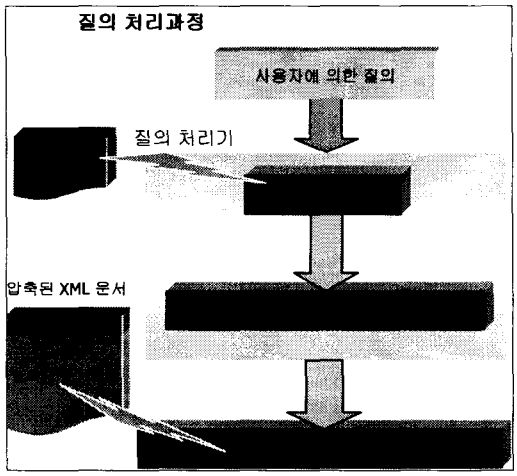
▶▶ 그림 3. XENDER 압축 처리 과정

XML 압축 생성기를 통하여 메타데이타에 대한 부호화 테이블을 가지고 각각의 메타데이타들에 대한 부호화작업과 허프만 압축과정을 통해 얻어지는 압축된 문서를 생성하게 된다.

압축된 XML 문서는 허프만 압축에 의해 생성된 분포테이블에 대한정보를 가지고 있는 문서 헤더 부분과 실제 압축된 내용의 문서를 포함한다.

1.3 질의 처리기

질의 처리과정은 다음과 같다. 우선 질의 분석기를 통하여 사용자의 요청에 의한 질의를 각각의 엘리먼트와 애트리뷰트를 이전에 생성한 부호화된 DTD 테이블 문서를 이용해 사용자의 질의를 분석 및 압축을 하게 된다. 그후에 질의 처리기를 통해 압축된 질의를 생성한다. 압축된 XML 문서에 구성된 압축된 질의를 한다. 원하는 결과가 있다면 질의에 의한 값을 반환함으로 질의 처리를 마치게 된다.



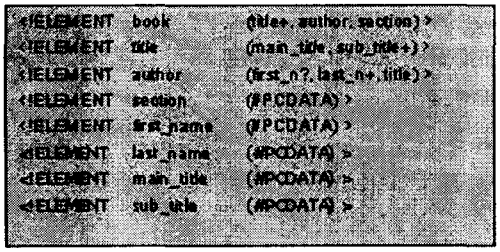
▶▶ 그림 4. 질의 처리 과정

2. 압축 처리과정 예

2.1 DTD Encoder

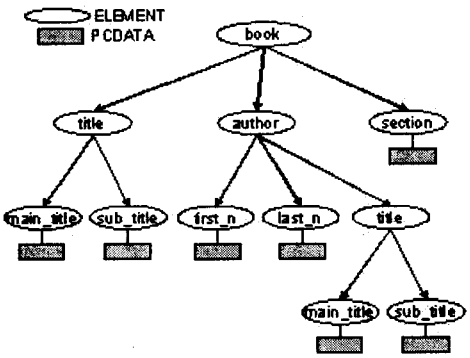
부호화된 DTD 테이블 문서는 DTD 문서를 부호화하여 테이블 형태의 문서를 생성한다. DTD 문서를 압축을 하면 다음 단계 즉, XML 문서의 파싱 및 부호화 시에 좀 더 빠른 처리를 하게 된다. 또한 문서를 압축하

고 난 후의 압축된 XML 문서에 질의 처리시에 해당 질의를 압축된 문서에 맞게 변환하여 질의를 처리하도록 한다. 다음과 같은 DTD 문서가 있다고 하자.



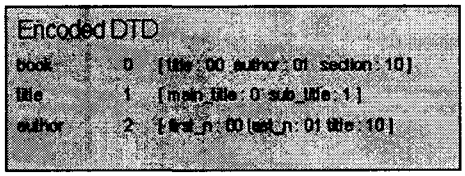
▶▶ 그림 5. DTD 문서

위 문서를 트리로 구성하면 다음과 같다.



▶▶ 그림 6. DTD문서 트리화

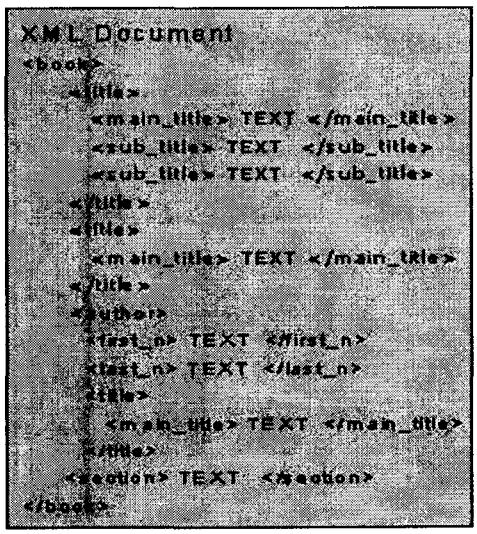
주어진 DTD문서를 가지고 부호화된 DTD 테이블 문서를 구성하는 방법은 다음과 같다. 우선 최상위 루트 엘리먼트가 가질 수 있는 자식 엘리먼트의 수를 확인한다. 부호화를 위한 방법은 엘리먼트들에 대하여 가변비트를 부여한다. 루트 엘리먼트가 가지는 자식 엘리먼트를 표현하기 위해 부여되는 가변비트 수는 ⌈log2K⌉ (K는 정수)로 표현한다. 위의 예에서 book의 자식엘리먼트는 3개이다. 식 log2K에 대입하면 1.5849가 나오게 된다. 여기서 주어진 값을 넘는 최소 정수가 가변비트수이기 때문에 자식 엘리먼트를 표기하기 위한 비트수는 2이다. 자식 엘리먼트는 title, author, section이므로 각각을 표현하려면 00(titlle) 01(author) 10(section)으로 표기하면 각각의 엘리먼트를 구분지을 수 있다. 그리고 각각의 자식엘리먼트이 PCDATA가 아니라면, 즉 하위 엘리먼트를 가지고 있다면 각각의 엘리먼트들에 대한 부호화를 위해 가변비트를 부여한다. 그러한 결과로 다음과 같은 부호화된 DTD 테이블 문서를 구성할 수 있다.



▶▶ 그림 7. 부호화된 DTD 테이블 문서

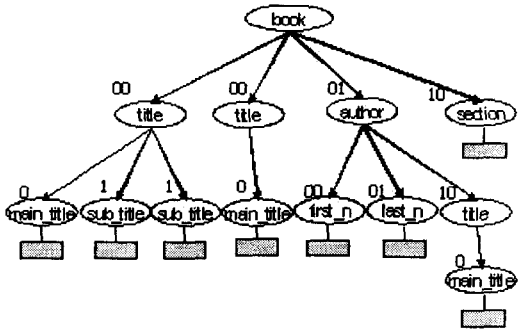
2.2 XML문서 압축

XML 문서압축 과정은 2.1절에서 생성된 부호화된 DTD를 이용하여 XML 문서를 압축된 XML 문서형태로 생성한다. 그림과 같은 XML 문서가 있다고 가정한다.



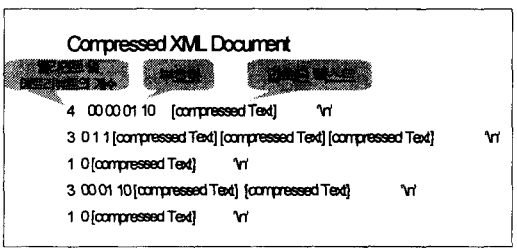
▶▶ 그림 8. XML 문서

위 문서를 트리구조로 표현하면 다음과 같다.



▶▶ 그림 9

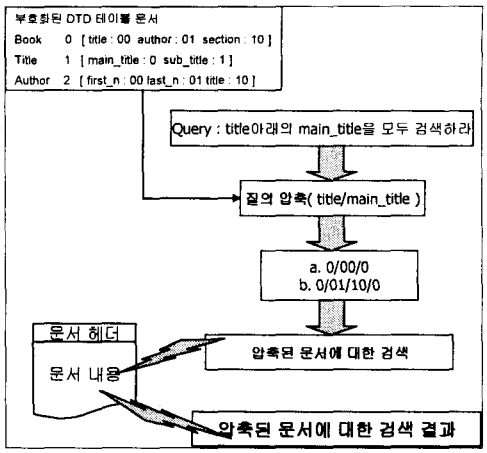
부호화된 DTD문서에서의 메타데이터들을 XML문서에 적용하여 부호화를 한다. 데이터 값들에 대해서는 내용 분포 테이블을 이용하여 허프만 방식의 부호화과정을 수행한다. 최초의 루트엘리먼트가 가지는 자식 엘리먼트들의 부호화된 값을 나열한다. 각각의 자식 엘리먼트가 하위 자식 엘리먼트를 가지고 있다면 해당 엘리먼트들에 대한 부호화된 값을 나열한다. 자식 엘리먼트들이 PCDATA를 가지고 있으며 분포테이블에서 구성한 값을 이용하여 부호화하여 표현한다. 마지막 엘리먼트와 엘리먼트 사이의 구분은 '\n'을 이용하여 각 레벨에서의 엘리먼트를 구분한다. 이렇게 생성된 압축된 문서의 내용은 다음과 같이 된다.



▶▶ 그림 10. 압축된 XML 문서

2.3 질의 처리

사용자가 압축된 XML문서에 대하여 질의를 내리게 되면 질의 처리기에서는 부호화된 DTD 테이블 문서의 문서 메타데이터 테이블을 가져온다.



▶▶그림 11. 압축된 문서에 대한 질의처리

가져온 메타데이터를 매핑시켜 사용자의 구조 의 및 내용 검색질의를 각각의 엘리먼트와 애트리뷰트 그리고 실제 내용의 데이터로 토큰화 한 후에 질의 압축기를 통해 압축된 XML문서에 맞게 압축을 시킨다. 사용자의 질의가 title내의 main_title을 모두 검색하는 요구라 할 때, title 엘리먼트 내의 main_title가 포함된 모든 부모 자식관계의 엘리먼트들을 찾는다.

Ⅳ. 성능평가

제안한 알고리즘에서의 성능평가는 XGrind와 비교를 하여 다음과 같은 세부분으로 나누어 측정하였다. 어느 정도의 압축율이 되는 지에 대한 압축율과, 압축을 하는데 있어서의 압축 처리 시간, 그리고 질의 처리시간의 세 가지를 비교하였다.

1. 측정 데이타

▶▶ 그림 12. XML 문서

shapeare는 세익스피어의 희극을 XML문서로 변환한 것으로 크기는 5MB이고, 문서의 깊이(depth)는 최

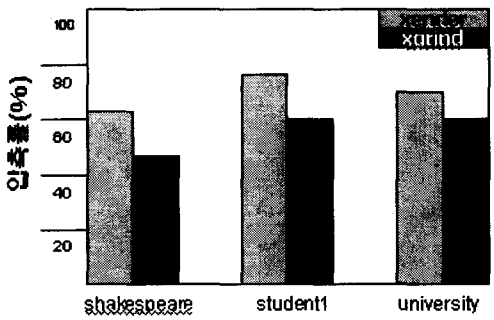
대 6까지 내려간다. 엘리먼트만 22개로 구성되어있다. 그리고 student1과 university는 성능 평가를 위해 임의로 구서한 XML문서로써 student1에서는 적은 엘리먼트 수와 애트리뷰트가 모두 열거형 타입에 대한 데이타이고, university는 16개의 엘리먼트에 애트리뷰트는 10개이며 그중에 5개의 열거형을 갖는다.

2. 측정 결과

2.1 압축률 비교

XGrind알고리즘으로 구현한 압축된 문서와의 압축률 비교는 다음과 같다.

$$\text{압축률} = 1 - \frac{\text{압축된 } XML \text{문서의크기}}{\text{실제 } XML \text{문서의크기}}$$

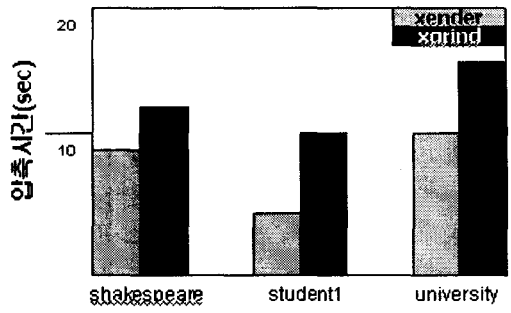


▶▶ 그림 13. 압축 시간 비교

XGrind에서는 압축률을 고려했을 때 가장 좋게 나온 경우가 애트리뷰트들이 열거형을 가진 경우이다. 하지만 그림 13에서 알 수 있듯이 student1은 애트리뷰트들을 열거형으로 갖는 XML 문서임에도 본 논문에서 제안한 알고리즘보다 압축률이 떨어진다. 그 이유는 본 연구에서 구현한 알고리즘에서는 메타데이터를 표현하기 위해 해당 메타데이터들 각각에 대하여 가변비트를 사용하였기 때문에 같은 량의 XML 문서를 처리하더라도 성능면에서 더 좋은 결과를 보이고 있다.

2.2 압축 시간 비교

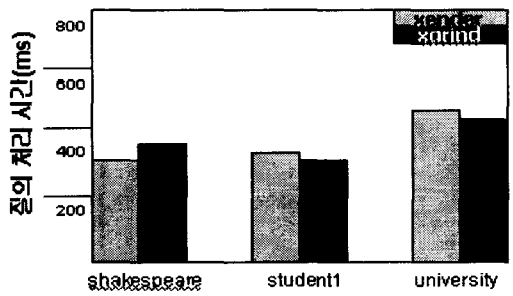
압축 시간 비교에서는 shakespeare 문서가 university 문서에 비해서 문서량이 많긴 하지만 메타데이터로 엘리먼트만을 가지고 있기 때문에 압축시간에서 더 적게 나오는 것을 보이고 있다.



▶▶ 그림 14. 압축시간 비교

3. 질의 처리시간 비교

질의 처리 시간은 다음과 같다. 질의 처리에 따른 시간 측정은 XGrind와의 비교에서 그다지 큰 차이를 보이지 못하고 있다. 질의 처리를 할 때 부호화된 DTD 테이블 문서를 이용하여 질의를 압축된 문서에 맞게 가변비트를 부여하고 질의를 처리를 하게 된다. 압축된 질의문에 대한 오버헤드는 크지 않지만, 직접 문서에 적용하여 비교를 할 때, 바이트단위의 비교가 아닌 비트단위의 메타데이터 비교를 하기 때문에 실제로 비교하는 동안 XGrind보다 더 큰 오버헤드를 보이고 있다.



▶▶ 그림 15. 질의 처리시간 비교

V. 결론

본 논문에서는 XML 문서를 효율적으로 저장, 관리하기 위하여 질의 처리가 가능한 XML 문서 압축기인 XENDER를 설계 및 구현하였다. XML 문서의 구조적 특성 즉, 구조표현에서의 메타데이터의 표현을 부호화

하고 각 메타데이터로 표현하는 데이터 값들에 대해서는 XML문서 파싱을 하고난 후에 이전의 메타데이타에 가변비트 및 엘리먼트 정보를 구성한 DTD문서와 상호 동작을 한다. 파싱을 하며 얻어지는 메타데이터 정보와 실제 텍스트들의 정보를 부호화된 DTD테이블 문서와 허프만 압축 알고리즘을 이용하여 압축된 XML문서를 구성하였다. 또한 질의 처리시에 압축 해제를 하고 난후에 요청한 질의 처리를 하는 오버헤드를 줄이기 위하여 압축된 문서 자체에 질의를 내려 처리하였다. 질의를 내릴 때는 압축된 문서의 부호화된 메타데이타 형식에 맞게 질의를 구성한다.

향후 연구방향으로는 주어진 XML 문서에서의 확장을 위하여 적용되는 엘리먼트와 애트리뷰트에 대한 표식자들의 처리 문제를 보다더 명확성있게 해야 할 필요가 있다. 또한 부호화된 DTD테이블 문서에 대한 유효성 검사의 범위가 필요하다. 또한 데이터값의 형식에 대한 내용 부분에서 CDATA와 PCDATA의 구별을 하여 조금 더 실제로 적용되는 문서에 대한 접근을 더 높여보고자 한다.

▌참고문헌▐

[1] T. Bray, J. Paoli and C. Sperberg-McQueen, "Extensible Markup Language (XML) 1.0", http://www.w3.org/TR/1998/REC-xml-19980219, February 1998.

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

[3] http://www.gzip.org

[4] http://www.gzip.org/algorithm.txt

[5] http://www.ictcompress.com/xml.html.

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

[7] http://www.research.att.com/sw/tools/xmill

[8] G. Girardot and N. Sundaresam. "Millau: an encoding format for efficient representation and exchange of XML over the Web", http://www9.org/w9cdrom/154/154.html,

[9] S. Eggers, F. Olken, and A. Shoshani. "A Compression Technique for Large Statistical databases", Proc. of VLDB Conf., September 1981.

[10] M. Bassiouni and K. Hazboun. "Utilization of Character Reference Locality for Efficient Storage of Databases", Proc. of 2nd Intl. Workshop on Statistical Database Management, September 1983.

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

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

[13] M. A. Bassiouni. "Data Compression in Scientific and Statistical Databases", IEEE Trans. on Software Eng., October 1985.

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

[15] D. W. Jones. "Application of Splay Trees to Data Compression", Comm. of ACM, August 1988.

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

[17] A. Zandi, B. Iyer, and G. Langdon. "Sort Order Preserving Data Compression for Extended Alphabets", Data Compression Conf., 1993.

[18] G. Ray. "Data Compression in Databases", Master's Thesis, Dept. of Computer Science and Automation, Indian Institute of Science, June 1995.

[19] G. Ray, J. Haritsa and S. Seshadri, "Database Compression: A Performance Enhancement Tool", Proc. of 7th Intl. Conf. on Management of Data (COMAD), December 1995.

[20] "Simple API for XML", http://www.megginson.com/SAX/

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

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