韓國 컴퓨터情報學會 論文誌
第13卷 第4號, 2008. 7.

2008-13-4-2-3

디지털 방송을 위한 Set-Top Box기반 TV-Anytime 메타데이터 관리 시스템

박 종 현*, 강 지 훈**

TV-Anytime Metadata Management System based on a Set-Top Box for Digital Broadcasting

Jong-Hyun Park*, Ji-Hoon Kang**

요 약

디지털 방송은 양방향 통신을 기반으로 하여 고객의 요구를 만족시키는 다양한 부가 서비스를 제공한다. 새로운 방송 환경을 위한 중요한 요소 중 하나는 분산되어있는 환경에서 여러 소비자와 공급자 간의 상호운용성의 유지에 있다. 이를 위하여 디지털 방송을 위한 메타데이터의 표준이 제안되었고, TV-Anytime 메타데이터는 이러한 요구를 만족시키기 위한 차세대 방송 표준 메타데이터의 하나이다. 한편, 사용자 측면에서 다양한 방송 서비스 및 부가 서비스를 원활히 활용하기위해서 방송 사용자 단말(Set-Top Box: STB) 환경에서 메타데이터를 효율적으로 관리하기 위한 연구들이 진행 중이다. 본 논문에서는 TV-Anytime 메타데이터를 저 비용, 저 사양의 STB에서 효율적으로 관리하기위한 메타데이터 관리 시스템을 제안한다. STB 기반 관리시스템은 메타데이터의 저장을 위한 저장엔진과 검색을 위한 XQuery엔진으로 구성되며, 효율적인 저장과 검색을 위해서 인덱싱 방법을 제안하고 있다. 또한, XML 검색을 위한 표준 질의어인 XQuery를 방송용 메타데이터 검색을 위한 질의어로 사용하는 우리의 메타데이터 관리 시스템은 향후 다양한 방송 응용들 사이에서 상호운용성(Interoperability)을 보장할 뿐만 아니라 메타데이터의 확장에도 유연하게 대처할 수 있다.

Abstract

Digital Broadcasting serves a variety of broadcasting services for satisfying the requirement of customers. One of main factors for new broadcasting environment is interoperability between providers and consumers. For this interoperability, metadata standards are proposed for the digital broadcasting and TV-Anytime metadata is one of these standards. On the one hand, there are some researches for efficiently managing the broadcasting metadata on Set-Top Box. This paper proposes the metadata management system for efficiently managing the broadcasting metadata based on the STB which is low-cost and low-setting. Our system consists of a storage engine to store the metadata and an XQuery engine to search the stored metadata and uses special index for storing and searching. We expect that our system will keep the interoperability amongst a variety of applications for broadcasting because we adopts the XQuery for searching the metadata and the XQuery is a standard language for searching XML data.

▸ Keyword : Metadata Management, TV-Anytime Metadata, XQuery Processing, XML Indexing

• 제1저자 : 박종현 교신저자 : 강지훈
• 접수일 : 2008. 4. 17, 심사일 : 2008. 7. 7, 심사완료일 : 2008. 7. 25.
* 충남대학교 소프트웨어연구소 전임연구원 ** 충남대학교 전기정보통신공학부 교수
※ 본 연구는 21세기 프론티어 연구개발사업의 일환으로 추진되고 있는 지식경제부의 유비쿼터스컴퓨팅및네트워크원천기반기술개발사업의 08B3-O1-30S 과제로 지원된 것임

Ⅰ. 서 론

초고속 인터넷 망의 확산과 디지털화된 방송 프로그램 제작 환경의 도래 등 통신과 방송의 융합으로 인해 기존 TV의 기능이 다양하게 변화하고 있다. 디지털 방송 서비스는 메타데이터를 활용하여 다양한 부가 서비스와 사용자 맞춤형의 개인화된 방송서비스를 제공 한다 [1,2]. 특별히, 방송 사용자 측면에서 디지털 방송 서비스를 원활하게 제공 받기 위해서는 Set-Top Box(STB)가 필요하며, 방송용 메타데이터는 STB 환경에서 관리 된다 [1]. 그러므로 방송 서비스와 부가서비스를 충분히 지원하고, 방송 서비스 제공자와 사용자 모두가 최적으로 관리 할 수 있는 표준 메타데이터가 요구되었고, 이러한 요구사항을 위하여 TV-Anytime Forum에서는 디지털 방송을 위한 메타데이터 표준으로 TV-Anytime 메타데이터를 제안하였다. 이러한 메타데이터를 효과적으로 저장하고 검색하기위한 방법은 다양하고 고품질의 디지털 방송 서비스 제공하기위하여 반드시 필요하다.

TV-Anytime 메타데이터는 단일의 XML Schema를 기반으로 하는 XML 형식으로 표현 된다 [3]. 그러므로 디지털 방송 서비스를 위한 메타데이터의 저장과 검색은 결국 TV-Anytime 메타데이터를 활용한 XML 문서의 저장과 검색의 문제이다. 다만 그 응용 분야가 방송용이고, 특별히 저비용, 저 사양의 STB를 기반으로 관리해야한다는 특징을 가지고 있다. 현재, 디지털 방송 서비스를 위한 방송용 메타데이터의 표준은 아직까지 없는 상태이므로, 본 논문에서는 차세대 디지털 방송을 위한 메타데이터 표준인 TV-Anytime 메타데이터[3]를 디지털 방송 서비스를 위한 메타데이터로 사용할 것을 제안하며 이를 기반으로 하는 메타데이터 관리시스템을 제안 한다. 메타데이터 관리시스템은 크게 메타데이터의 저장을 위한 저장엔진과 검색을 위한 XQuery엔진으로 구성된다. 본 논문에서는 검색을 위해 XML 검색을 위한 표준 질의어인 XQuery[4]를 사용함으로써, 향후 메타데이터를 활용한 다양한 방송 응용들 사이에 상호운용성을 보장한다.

우리는 이미 본 연구의 앞선 연구 결과로 방송 서비스 제공자 측면에서 방송용 멀티미디어 데이터와 TV-Anytime 메타데이터 그리고 MPEG-7 메타데이터들을 어떻게 효율적으로 관리할 것인지를 연구하고 그 결과의 우수성을 입증한바 있다[5]. 그러나 방송 서비스 제공자 측면과 달리 본 논문에서의 환경은 저 비용, 저 사양의 STB 환경에서 메타데이터를 효율적으로 관리하기위한 방법이다. 그러므로 앞선 우리의 연구에서 이미 검증된 메타데이터 저장 방법과 검색 방법을 성능의 저하 없이 어떻게 STB를 위해서적용할 것인가 하는 것이 본 논문의 목표이다.

본 논문의 나머지 구성은 다음과 같다. 2절은 방송용 TV-Anytime 메타데이터의 특징과 XML 메타데이터의 저장 및 관리를 위한 관련연구를 알아보고, 3절에서는 메타데이터 저장 엔진과 검색을 위한 XQuery 엔진 그리고 Special 인덱스의 구조를 설명한다. 4절에서는 우리의 방법을 기반으로 하는 프로토타입 시스템과 기존 XML 데이터를 관리하기위한 시스템들과의 성능 평가를 통하여 우리 방법의 성능을 평가한다. 마지막으로 5절에서는 향후 연구계획과 결론을 내린다.

Ⅱ. 관련 연구

TV-Anytime 메타데이터는 저장매체를 갖는 단말 즉, PDR(Personal Digital Recorder) 또는 PVR(Personal Video Recorder) 환경에서 원하는 AV 콘텐트를 원하는 시간에 선택하고 소비할 수 있는 Anytime 서비스를 위한 방송용 메타데이터 표준이다 [3]. TV-Anytime 메타데이터는 하나의 방송 콘텐츠를 기술하기 위해 4개 그룹으로 구성되며, 각 그룹은 콘텐츠를 참조하기위한 식별자로 CRID(Content Referencing Identifier)를 사용한다. 이러한 TV-Anytime 메타데이터는 단일 XML Schema를 사용하여 정의 되며, 메타데이터의 인스턴스(instance)는 XML 데이터로 표현된다. 그러므로 STB에서 TV-Anytime 메타데이터의 관리는 XML 문서의 저장과 검색의 문제로 볼 수 있다.

한편, 관계형 데이터베이스[5, 6, 7, 8, 9, 10], XML 전용 데이터베이스[11, 12], 파일 시스템[13, 14] 등에서 효율적으로 XML을 저장하고 검색하기 위한 많은 연구와 상용 시스템들이 존재한다. 그러나 [10, 11, 12, 13]는 일반적인 XML 데이터들을 관리하기 위한 방법으로 방송용 메타데이터의 특성을 고려한 방법이라고 보기 어려울 뿐만 아니라, 이미 [8]에서 방송용 메타데이터의 특성을 고려한 시스템과의 성능 평가를 통하여 방송용 메타데이터 관리를 위해서는 최적화 되지 않았음을 볼 수 있다.

방송용 메타데이터가 대용량이라는 특성 때문에 관계형 데이터베이스와 XML 전용 데이터베이스를 이용하여 XML 저장 관리 시스템을 구현하는 연구가 활발히 진행 중이다. 그 가운데 본 연구진이 기 수행한 연구를 포함하여 일반적으로 방송용 XML 메타데이터를 관계형 데이터베이스에 저장하기 위한 [7, 8]의 연구들은 구조화된 XML 문서를 관계형 데이터베이스에 저장하기 위해 XML 스키마 정보를 분석하여 노드 수만큼 테이블을 생성하고, 노드와 값을 저장하는 방법을 사용한다. [7]은 노드들 사이의 구조 정보를 유지하기 위해서 Fragment 기반 저장방법을 사용하며, [8]은 Dewey 노드 넘버링 방법을 사용

한다. 또한 두 시스템 모두 검색을 위해 XPath 및 XQuery 언어를 지원한다. 그러므로 사용자 질의로부터 SQL 질의로 변환하기 위한 모듈을 가지고 있으며, 효율적인 검색(selection, projection, join의 신속한 처리)을 위해서 인덱싱 방법을 사용한다. 이러한 방법들은 XML로 기술된 대용량의 방송용 메타데이터를 효율적으로 저장하고 검색하기위해서 매우 바람직해 보인다. 그러나 두 시스템 모두 관계형 데이터베이스를 그 근간으로 하고 있으므로 저 비용, 저 사양의 STB를 위한 최적화된 방법으로 보기 힘들다. 그러나 [8]의 방법은 방송용 메타데이터 검색을 위하여 기존의 상용 XML 전용 데이터베이스 또는 XML-Enabled 데이터베이스들 보다 그 성능이 월등히 뛰어나므로 본 논문에서는 우리의 앞선 방법들을 STB를 위한 방법으로 변형하여 그 성능의 우수성을 유지하고자 한다.

III. 메타데이터 관리 시스템

메타데이터 관리 시스템은 콘텐츠 소비자를 위한 부가 서비스, EPG(Electronic Program Guide) 그리고 콘텐츠 검색 등을 위해서 메타데이터를 효율적으로 관리하기 위한 시스템이다. 본 논문에서 제안하는 메타데이터 관리 시스템은 STB로 입력된 메타데이터의 구문 분석을 통해 얻어진 정보를 활용하여 메타데이터의 저장, 관리, 효율적 검색을 목적으로 한다. 또한 시스템은 STB 환경을 고려한 파일 기반 저장 방법과 특별히 정의된 인덱스를 사용한다.

<그림 1>은 STB 내에서 메타데이터 관리시스템의 기능과 역할을 보여주고 있다. 시스템은 운영체제와 독립적으로 자바 플랫폼(platform)위에서 구현되어 프로그램 이식성을 높였다. 제안된 시스템에서 중요한 두 가지 모듈은 Storage 엔진과 XQuery 엔진이다.

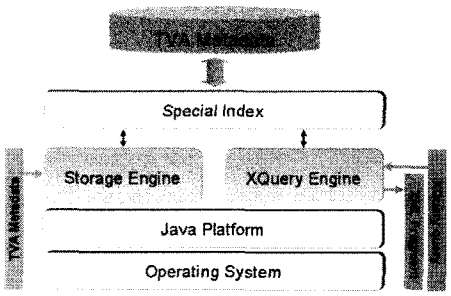

그림 1. 메타데이터 관리시스템의 구조
Fig 1. An architecture of Metadata Management System

메타데이터의 저장은 STB의 H/W와 S/W적인 제약사항을 고려하여 본 논문에서는 파일 시스템을 기반으로 메타데이터들을 저장하고 검색한다. 물론 우리는 관계형 데이터베이스를 비롯한 Native XML 데이터베이스, 메모리 데이터베이스와 같은 다양한 종류의 데이터 관리 시스템을 고려했으나 비용과 자원의 제약성 때문에 기존의 시스템을 적용하기는 어렵다고 판단하였다. 또한 논문은 사용자의 STB 특성을 고려하여 대량의 메타데이터를 관리하는데 초점을 두기보다는 메타데이터의 검색 효율성에 그 초점을 맞추었다. Storage 엔진은 메타데이터의 저장과 함께 인덱스를 생성한다. 이때, 인덱스는 새로운 메타데이터의 삽입과 저장된 메타데이터의 삭제 그리고 수정을 위해 사용된다. XQuery 엔진은 저장 엔진이 생성한 인덱스를 이용하여 사용자 질의를 처리한다. 이미 언급한 것처럼 본 시스템에서는 상호운용성을 보장하기위해 질의어로 XQuery를 사용한다. XQuery 엔진은 XQuery로 질의를 받고, 질의를 처리하고 질의의 결과를 반환 한다. 이때, Special 인덱스는 저장된 메타데이터를 빠르게 검색하고 반환될 XML 단편(Fragment)의 위치 제공을 위해 사용된다.

3.1 저장 엔진

저장 엔진은 <그림 2>와 같이 TV-Anytime 메타데이터의 삽입, 삭제, 그리고 갱신을 위하여 InsertDoc, UpdateDoc 그리고 DeleteDoc의 3개 모듈들로 구성된다.

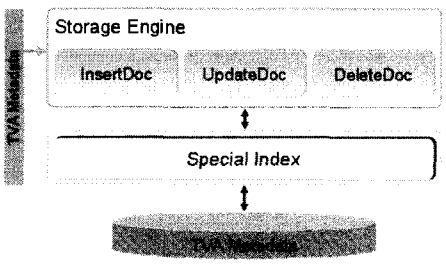

그림 2. 저장 엔진의 구조
Fig 2. An architecture of Storage Engine

InsertDoc은 STB로 전송된 메타데이터를 구문 분석하여 메타데이터에 기술되어 있는 내용정보와 구조정보를 얻어와 인덱스 파일을 생성하고, 폴더에 메타데이터를 저장한다. DeleteDoc은 메타데이터의 CRID와 매치되는 인덱스 파일 정보와 메타데이터를 삭제한다. UpdateDoc은 저장된 메타데이터와 인덱스 정보를 갱신하기 위한 모듈로 CRID와 매치되는 인덱스 정보와 메타데이터를 우선 DeleteDoc을 이용해

서 삭제하고, InsertDoc 모듈로 추가하는 방식으로 갱신 작업을 수행한다. 다음은 메타데이터 저장 시 차후 효율적인 검색과 관리를 위해서 각 데이터들을 위한 인덱스를 생성하기위한 방법을 설명한다.

방송용 메타데이터를 위한 인덱스 구조

메타데이터 관리시스템에서 인덱스는 효율적인 메타데이터 검색을 위해 반드시 필요하며, 인덱스 구조에 따라 검색의 성능이 결정된다. 앞서 언급한 것처럼, 본 논문에 앞서 우리는 관계형 데이터베이스 기반 TV-Anytime 메타데이터의 저장 및 검색을 위한 연구를 수행했고, 실험을 통해서 검색 속도의 우수성을 보였다.[8]. 그러나 본 논문에서 제안한 시스템은 관계형 데이터베이스가 아닌 파일 시스템을 기반으로 한다. 그러므로 이전 연구의 저장 방법을 그대로 사용하더라도 파일시스템 기반의 시스템에서는 관계형 데이터베이스가 제공하는 테이블 인덱스 등을 사용할 수 없으므로 동일한 성능 향상을 기대하기는 어렵다. 그러므로 본 논문에서는 이전 연구의 장점과 파일 시스템의 특징을 고려해 저장된 메타데이터에 대해 효율적으로 질의할 수 있는 인덱스 방법을 제안한다. 우리의 인덱스는 입력된 메타데이터의 구문 분석을 통해 얻어진 정보를 이용하여 텍스트 파일로 저장된다. 〈표 1〉은 실제로 인덱스 파일을 테이블 형태로 보인다.

표 1. 인덱스의 구조
Table 1. An architecture of Index

No.	Name	Value
1	Doc_Name	BBC_20070107..xml
2	Node_Position	2:4-2:30
3	Dewey_Num	1:1:1
4	Doc_id	1
5	Path_id	6
6	Attribute_Name	lang
7	Attribute_Value	korean
. .	. . .	. . .
10	Text_Value	주몽

인덱스 파일의 이름은 TV-Anytime 메타데이터에 정의된 모든 엘리먼트 노드의 이름으로 정의되어있다. 즉, 〈표 1〉의 인덱스 파일의 이름은 'Title.txt'이다. 결국, 인덱스 파일의 개수는 TV-Anytime 메타데이터 스키마에 정의되어있는 전체 엘리먼트의 개수와 동일하다. 이러한 개념은 [10]에서 제안하고 있는 관계형 테이블 기반의 Binary 방법과 유사한 개념이다. 인덱스 파일은 〈표 1〉의 No. 1과 같이 "Title" 엘리먼트 노드가 정의되어있는 문서의 이름을 저장하고 해당 문서에서 "Title" 엘리먼트 노드의 시작 위치와 종료 위치를 Node_Position 값으로 표현하고 있다. 즉, 위 예제의 경우, "Title" 엘리먼트 노드는 "BBC_20070107..xml" 문서의 2행 4번째 위치에서 시작하여 30번째 위치까지 그 내용이 기술되어있다는 것을 알 수 있다. 이는 차후 사용자가 질의한 XQuery 질의에 기술된 "Title" 엘리먼트 노드의 반환을 위하여 매우 유용하게 사용된다. 인덱스 가운데 3의 Dewey_Num은 [6]에서 제안된 방법으로 해당 메타데이터에서 "Title" 엘리먼트 노드의 고유한 구조 정보를 표현한다. 그러므로 "Title" 엘리먼트 노드가 문서 내에서 어느 노드의 형제 노드이며 어느 노드의 조상 또는 자손노드인지 쉽게 알 수 있다. 예를 들어, 〈표 1〉의 경우 "BBC_20070107..xml" 문서에서 "Title" 엘리먼트 노드의 조상 엘리먼트 노드의 Dewey_Num은 "1"과 "1:1"일 것이고, 자손 엘리먼트 노드의 Dewey_Num은 "1:1:1"fh 시작할 것이다. 4의 Doc_id는 메타데이터 문서의 고유 id를 표현한다. 결국, Doc_id와 Dewey_Num을 이용하면 현재 시스템에 저장된 메타데이터들 가운데 "Title" 엘리먼트 노드의 고유 위치 정보를 취할 수 있다.. 5의 Path_id는 Path 인덱스 파일과 연관되어있는 고유 id이다. Path 인덱스 파일은 TV-Anytime 메타데이터를 저장할 때, 문서를 구문 분석하여 메타데이터에 포함된 모든 엘리먼트 노드의 전체 경로를 XPath 표현을 이용하여 저장하고 있는 파일이다. 예를 들어 Path 인덱스 파일에는 "Title" 엘리먼트 노드의 전체 경로를 표현하는 XPath 표현인 "TVAMain/ProgramDescription/ProgramInformation Table/ProgramInformation/BasicDescription/Title"을 저장하고 있으며, 그 id는 6으로 정의되어 있을 것이다. 이는 차후 사용자 질의인 XQuery 질의에 기술된 XPath 표현을 처리하기위해서 사용될 수 있다. 물론 XPath 표현에는 특정 노드를 지정하기위하여 전체 경로를 모두 기술하지 않고 "TVAMain/ProgramDescription//Title"과 같이 간결하게 기술하는 방법이 존재한다. 이러한 경우 본 논문에서는 사용자가 질의한 경로 표현과 Path 인덱스 파일에 저장된 경로들의 문자 비교를 통하여 사용자가 표현한 경로가 어떤 전체 경로들과 매칭 되는지 검색할 수 있다. 본 논문에서 Path 인덱스 파일을 구성할 수 있는 이유는 TV-Anytime 메타데이터가 단일의 스키마를 기반으로 작성되고 자손 노드에서 조상 노드로의 재귀가 발생하지 않기 때문에 가능하다. 다시 말하면, 단일의 스키마를 따르므로 표현할 수 있는 전체 경로의

수가 한정되어 있으며, 조상 노드로의 재귀가 없으므로 무한정 경로가 발생할 수 없기 때문에 결국 유한개의 경로로 Path 인덱스 파일을 구성할 수 있다. 실제로 본 논문에서 실험을 통하여 약 300여개의 TV-Anytime 메타데이터 파일을 저장했을 경우 Path 인덱스 파일에 저장된 전체경로는 약 200여개에 이르지 못하였다. 6과 7의 Attribute_Name과 Attribute_Value는 해당 엘리먼트 노드가 속성을 갖는다면 그 속성의 이름과 값을 저장하는 부분이다. 마지막으로 Text_Value는 해당 엘리먼트 노드가 내용을 가질 경우 그 값을 저장하는 부분이다. 6과 7의 Attribute_Name과 Attribute_Value는 해당 엘리먼트 노드가 속성을 갖는다면 그 속성의 이름과 값을 저장하는 부분이다. 마지막으로 Text_Value는 해당 엘리먼트 노드가 내용을 가질 경우 그 값을 저장하는 부분이다.

〈그림 3〉은 사용자가 질의한 XQuery 질의를 우리의 XQuery 엔진이 인덱스를 이용하여 처리하는 모델을 보인다. 사용자 XQuery 질의는 "Title"이 "유리와 재회"라는 내용을 포함하는 프로그램의 세그먼트를 검색하여 해당 세그먼트의 "Description"을 모두 보여 달라는 질의이다. 이를 처리하기 위해서 수행되는 첫 번째 작업은 사용자 XQuery 질의로부터 where절에 기술된 변수의 전체 XPath 경로를 생성하여 Path 인덱스 파일로부터 해당 Path_id를 모두 얻는다. 아마도 위 예제의 경우 "Title" 엘리먼트를 위한 Path_id는 63일 것이다. 다음은 "Title" 인덱스 파일로부터 Path_id가 63이고 Text_Value가 "유리와 재회"를 포함하는 모든 "Title" 엘리먼트 노드들의 Doc_id와 Dewey_Num을 취할 것이다. 그 다음 XQuery 엔진은 조건을 만족하는 "Description" 엘리먼트 노드를 얻기 위하여 "Description" 인덱스 파일에 기술된 "Description" 엘리먼트들 가운데 Doc_id가 앞서 검색된 "Title" 엘리먼트 노드의 Doc_id와 동일하고 Dewey_Num가 "Title" 엘리먼트 노드의 Dewey_Num에 포함되는 "Description"엘리먼트 노드를 모두 취한다. 이렇게 얻어진 "Description" 엘리먼트 노드들은 사용자가 질의한 조건을 모두 만족하는 결과일 것이다. XQuery 엔진의 마지막 단계는 사용자가 질의한 XQuery 질의에 선언된 반환 형식에 맞게 검색 결과를 구성하는 것이다. 예제의 경우 Node_Position에 기술된 "Description" 엘리먼트 노드의 위치 정보를 이용하여 TV-Anytime 메타데이터에들로부터 필요한 정보를 취한 후 이들을 취합하여 사용자에게 반환한다.

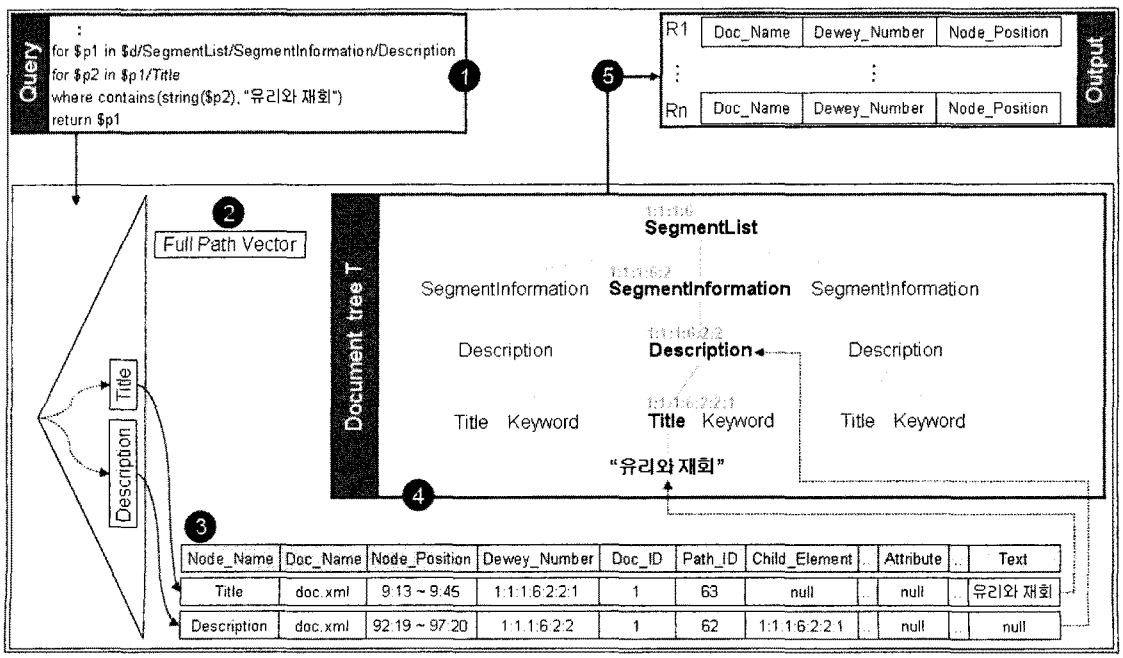

그림 3. 메타데이터 검색 모델
Fig 3. Metadata Search Model

3.2 XQuery 엔진

TV-Anytime 메타데이터는 XML로 표현된다. 그러므로 저장된 XML 데이터에서 원하는 정보를 얻기 위해서는 XML 데이터를 위한 효율적인 질의어가 필요하다. 논문에서 검색을 위한 질의어로 XML 검색을 위한 표준 질의어인 XQuery를 사용한다. XQuery는 XML 데이터의 저장 형식에 상관없이 XML 데이터를 검색할 수 있도록 하기 위하여 W3C에서 제안한 XML을 위한 질의 언어이다 [4].

〈그림 4〉는 XQuery 엔진의 구조를 보이고 있다. XQuery 엔진의 입력은 사용자가 질의한 XQuery 질의이고 그 결과는 TV-Anytime 메타데이터 문서 또는 문서 일부의 집합이다. Path Generator는 사용자 질의로부터 필요한 전체 경로를 생성하는 역할을 수행하고, Where Clause Processor는 사용자 XQuery 질의의 where절에 기술된 조건들을 처리하고 이를 만족하는 Doc_id와 Dewey_Num를 얻는다. Return Clause Processor는 Where Clause Processor로부터 얻어진 결과를 이용하여 이를 만족하는 반환 값을 얻는 역할을 수행하고, Result Constructor는 사용자 질의에 기술된 반환 구조에 맞는 반환 문서를 생성한다.

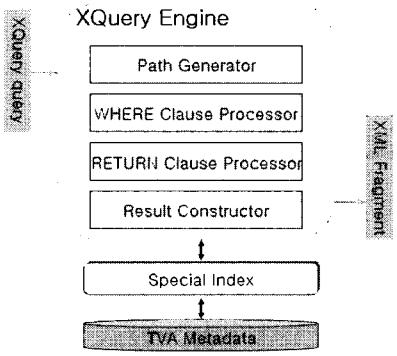

그림 4. XQuery 엔진의 구조
Fig 4. An architecture of XQuery Engine

Ⅳ. 성능 평가

본 논문에서 실험을 위하여 사용한 실험 환경은 다음과 같다. 운영체제는 Linux x86을 사용하였으며, CPU는 Intel Process 750MHz, 메모리는 128MB, 그리고 프로그래밍 언어로는 JDK 5.0을 사용하였다. 성능평가를 위한 비교대상이 되는 시스템은 XML 전용 처리 시스템인 SaxsonB[11]와 XML-Enabled 데이터베이스인 Oracle[12]을 사용하여 본 논문에서 제안한 시스템과 성능평가를 수행했다. 〈표 2〉는 성능평가를 위해서 사용된 XQuery 질의의 특성을 보인다.

표 2. 성능 평가를 위한 XQuery 질의의 특성
Table 2. Feature of XQuery queries for Performance Evaluation

XQuery	XPath Condition	Return Value
Q1	Single condition	Single terminal element
Q2	Single condition	Single root element
Q3	Single condition	Multiple root elements
Q4	Three conditions	Single root element
Q5	Five conditions	Multiple root elements
Q6	Three conditions	Multiple Terminal and Non-Terminal elements

예제질의의 특성은 Q1, Q2, Q3의 경우 단일의 조건만으로 각각 하나의 말단노드, 하나의 중간노드, 여러 개의 문서 전체를 반환하도록 하는 질의로, 반환되는 문서의 크기에 따른 성능을 측정한다. Q4, Q5, Q6의 경우 여러 조건을 기술하여 하나의 말단노드, 여러 개의 중간노드, 여러 개의 문서 전체를 반환하도록 하는 질의로 조건과 반환 값에 따른 성능을 측정한다.

〈그림 5〉은 100개의 견본 TV-Anytime 메타데이터에 예제 질의를 수행하여 얻은 시간을 측정한 그래프이다. 성능 측정 결과, 6개의 견본 질의 모든 경우에 본 논문에서 제안한 시스템이 효율적인 처리속도를 보이고 있다. 이는 우리의 방법이 XQuery 질의의 조건 개수와 반환 값의 크기에 독립적이라는 것을 입증한다.

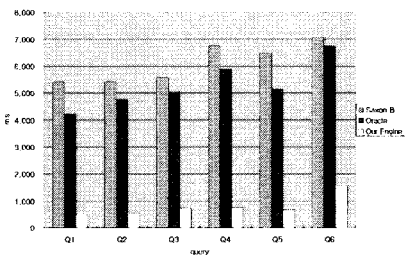

그림 5. 질의 수행시간
Fig 5. Comparison of Query Processing Times

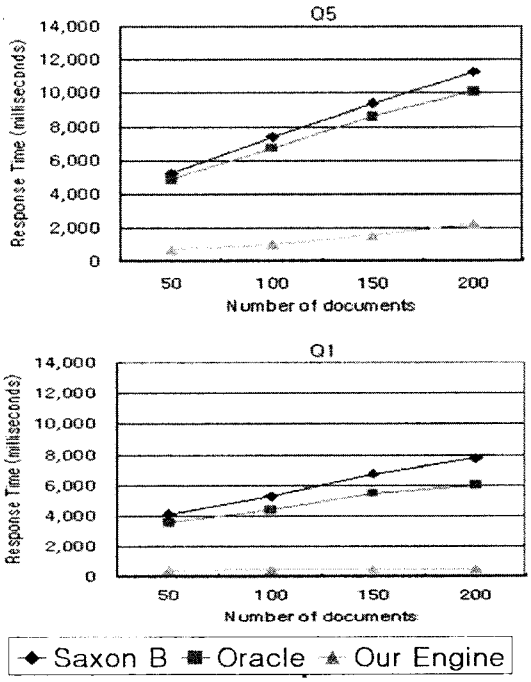

그림 6. 데이터 크기에 따른 성능 평가
Fig 6. Performance evaluation for scalability property

〈그림 6〉은 Q1과 Q5의 질의를 견본 데이터의 크기에 따라 성능을 측정한 그래프로 50개, 100, 150, 200개의 견본 메타데이터를 이용하여 성능을 측정한 결과이다. 나머지 질의들의 경우 역시 위 두 질의의 결과와 크게 다르지 않다. 성능 측정의 결과 Saxon B와 Oracle모두 데이터의 크기가 증가하면 처리속도 역시 유사한 비율로 증가하나 우리의 시스템은 증가비율이 크지 않았다. 그 이유는 메타데이터의 검색을 위해 Path 인덱스를 이용하여 직접 말단 노드에 해당하는 인덱스에 접근한 후 해당 내용만 검색하므로 구조와 내용을 비교하기 위한 시간이 커지지 않기 때문으로 사료된다.

V. 결론

본 논문에서는 디지털 방송 서비스를 위해 STB 기반에서 효율적으로 동작할 수 있는 방송용 TV-Anytime 메타데이터 관리 시스템을 제안하고 그 성능을 평가하였다. 우리의 실험 결과로 알 수 있는 것처럼, 본문에서 제안하고 있는 방송용 메타데이터 처리 방법은 저 사양, 저 비용의 STB에서 효율적으로 사용될 수 있는 방법들 가운데 하나이다. 본 논문의 결과는 현재 매우 활발하게 진행되고 있는 XML 데이터의 저장과 검색 방법에 대한 연구와 상호 보완적 성격을 가지므로 서로에게 도움을 줄 수 있을 것으로 기대된다. 또한, 우리의 XQuery 엔진은 TV-Anytime 기반 방송 시스템에서뿐만 아니라 향후, 유사한 응용에서 XML 기반 메타데이터를 관리하기위해서 사용될 수 있을 것으로 사료된다.

참고문헌

[1] 권영환, 최준균, "IPTV 기술 및 표준화 동향", 텔레콤, 22권 1호, 2006년 6월.

[2] 권수갑, "IPTV 개념 및 해외 동향", 전자부품연구원 전자정보센터, 2006년 1월.

[3] TV-Anytime Forum, "TV-Anytime Specification Series", http://www.tv-anytime.org.

[4] W3C, XQuery 1.0: An XML Query Language, W3C Recommendation, 23 January 2007, http://www.w3.org/TR/2007/REC-xquery-20070123/

[5] D.Florescu & D.Kossmann, "Storing and Querying XML Data Using an RDBMS", IEEE Data Engineering Bulletin, Vol.22 No.3, 1999.

[6] I.Tatatinov, S.D.Viglas, K.Beyer, J.Shanmugasundaram, E.Shekita, & C.Zhang, "Storing and Querying Ordered XML Using a Relational Database System", Proc. of ACM SIGMOD Conf, June 2002.

[7] 신효섭, "디지털 TV 방송 환경에서 내장형 시스템을 위한 XML 데이터의 저장 및 검색 방법", 데이터베이스연구, 제 19권 3호, 2003년 9월.

[8] Jong-Hyun Park, Ji-Hoon Kang, Byung-Kyu Kim, Young-Hee Lee, Min-Woo Lee and Min-Ok Jung, "An XQuery-based TV-Anytime Metadata Management", Proc. of DASFAA05 Conf, April 2005.

[9] J. Zhou, M. Wang, S. Zhang, & H. Sun, "Semi-Structure Data Management by Bi-Directional Integration Between XML and RD", Proc. of CSCWD 2006, Nanjing, China, May 2006.

[10] 이상태, 임종선, 주경수, "관계형 DBMS를 이용한 XML 스키마기반의 XML DBMS 설계", 한국컴퓨터정보학회, 9권 4호, 2004년.

[11] ORACLE, "Berkeley DB Introduction", http://www.oracle.com/database/berkeley-db/.

[12] T.Fiebig, S.Helmer, C.-C.Kanne, J.Mildenberger,

G.Moerkotte, R.Schiele,& T.Westmann, "Anatomy of a Native XML Base Management System," Technical Report 01, University of Mannheim, 2002.

[13] SAXONICA, "SAXON XQuery Engine", http://www.saxonica.com/

[14] ORACLE, "Oracle XML Data Synthesis or XDS", http://www.oracle.com/technology/tech/xml/xds/

저 자 소 개

박 종 현

1999년 우송대학교 컴퓨터과학과 학사 졸업.

2002년 충남대학교 컴퓨터과학과 석사 졸업.

2007년 충남대학교 컴퓨터과학과 박사 졸업.

2007년~현재 충남대학교 소프트웨어 연구소 전임연구원.

주관심분야: XML, XQuery, Ontology, 분산 데이터베이스, 유비쿼터스 컴퓨팅, 웹정보시스템, 추론, Semantic Web

강 지 훈

1979년 서울대학교 계산통계학과 학사 졸업.

1981년 한국과학기술원 전산학과 석사 졸업.

1996년 한국과학기술원 전산학과 박사 졸업.

1996년~1998년 미국 버지니아대학교 컴퓨터과학과 방문교수.

2000년~2002년 충남대학교 정보통신원장.

1985년~현재 충남대학교 전기정보통신공학부 교수.

주관심분야: Semantic Web, 추론 알고리즘, XML, XQuery, 데이터베이스 시스템, 웹 정보 시스템, 디지털도서관