

XML 기반 디지털 방송용 메타데이터 관리 시스템

(Metadata Management System for XML-based Digital Broadcasting)

박 종 현 [†] 김 병 규 ^{††} 이 용 희 ^{†††}

(Jong-Hyun Park) (Byung-Kyu Kim) (Young-Hee Lee)

이 민 우 ^{****} 정 민 옥 ^{*****} 강 지 훈 ^{*****}

(Min-Woo Lee) (Min-Ok Jung) (Ji-Hoon Kang)

요약 차세대 디지털 방송은 다양한 서비스와 함께 방송 사용자와 제공자 사이의 양방향 통신을 가능하게 한다. 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. TV-Anytime 메타데이터는 향후 다양한 서비스로의 확장을 위하여 그 말단이 MPEG-7으로 정의되어 있다. MPEG-7은 멀티미디어 컨텐츠를 기술하기 위한 메타데이터 표준으로, 방송용 컨텐츠를 기술하기 위해서 사용된다면 내용기반 검색과 같은 다양한 서비스를 제공한다. 이러한 방송용 메타데이터를 효율적으로 관리하기 위한 시스템은 실제 방송 환경에서 사용자에게 보다 질 좋은 서비스를 제공하기 위해서 필수적이다. 방송용 메타데이터의 가장 큰 특징 중 하나는 단일의 XML 스키마를 기반으로 XML로 기술된다는 것이다. 이러한 점은 기존의 XML 관리 시스템을 사용하여 방송용 메타데이터를 관리할 수 있다는 가능성을 보인다. 그러나 이들 대부분은 범용적인 방법을 사용하여 XML데이터를 관리하고 있으므로 방송용 메타데이터를 관리하기 위한 특화된 방법으로 보기는 어렵다. 본 논문에서는 방송용 메타데이터의 특성을 파악하여 방송 환경에 적절한 방송용 메타데이터 관리 시스템을 설계하고 구현한다. 우리의 메타데이터 관리 시스템은 실제 방송 환경에서 사용되는 표준 메타데이터를 기반으로 구현하므로, 방송 환경에 최적의 기능을 수행할 수 있을 것으로 기대된다. 또한, 우리는 방송용 메타데이터의 검색을 위한 질의어로 XML 표준 질의어인 XQuery를 사용하여 시스템 간의 상호 운용성을 확보할 수 있도록 한다.

키워드 : TV-Anytime, MPEG-7, XQuery, XML 관리 시스템, 디지털 방송 시스템.

Abstract The goal of next generation digital broadcasting is offering the interaction among consumers and providers as well as variety services. One of the important factors for this new broadcasting environment keeps the interoperability among providers and consumers since the environment is distributed. Therefore a standard metadata for digital broadcasting is required and TV-Anytime metadata is one of the metadata standards for digital broadcasting. The terminal nodes of TV-Anytime metadata are defined by using MPEG-7 metadata. MPEG-7 metadata is standard

· 본 연구는 충남대학교 BK21과 소프트웨어 연구센터의 지원을 받았음

[†] 학생회원 : 충남대학교 컴퓨터과학과

jhpark@cs.cnu.ac.kr

^{††} 정 회 원 : 한국과학기술정보연구원 지식정보센터 연구원

yourovin@kisti.re.kr

^{†††} 정 회 원 : 한국전자통신연구원 디지털홈연구단 연구원

lyhcool@etri.re.kr

^{****} 정 회 원 : 한전 KDN 발전사업팀 연구원

cslmw@kdn.com

***** 정 회 원 : (주)한글과 컴퓨터 XML 기술팀 연구원

mojung@haansoft.com

***** 종신회원 : 충남대학교 전기정보통신공학부 교수

(corresponding author)임

jhkang@cs.cnu.ac.kr

논문접수 : 2004년 11월 16일

심사완료 : 2005년 4월 13일

metadata for describing multimedia content. Therefore, if we use the MPEG-7 metadata for describing broadcasting content can offer multimedia search services like content-based search by the extension of metadata. The efficient management system for these metadata is important for offering the services with high quality on real broadcasting environment TV-Anytime metadata and MPEG-7 metadata are technically defined using a single XML schema, so its instances are XML data. Currently, a lot of systems for managing XML data are proposed in many researchers and we can expect to adapt these systems for managing broadcasting metadata. But the methods used in these systems are not specific methods for managing broadcasting metadata because of methods for general-purpose. In this paper, we find the properties of broadcasting metadata and develop an efficient metadata management system that is based on the found properties. Since our system is implemented on real broadcasting environment, we expect that the system is most efficient and suitable. Also our system is interoperable since we use XQuery as query language for querying broadcasting metadata.

Key words : TV-Anytime, MPEG-7, XQuery, XML Management System, Digital broadcasting system

1. 서 론

디지털 방송은 차세대 방송의 새로운 패러다임으로서, 화질의 획기적인 개선은 물론이고 사용자 서비스 측면에서 기존의 공중파 방송에서는 불가능하였던 다양한 서비스를 가능하도록 한다. 컨텐트 검색, 사용자 기록/선택도 관리, On-Demand 서비스 등의 새로운 부가 서비스를 원활하게 지원하기 위해서 현재 TV-Anytime Forum[1]에서는 TV-Anytime 이라는 차세대 디지털 방송용 메타데이터 표준을 제안하고 있다. TV-Anytime은 기존의 방송 시스템에서 방송 서비스 제공자가 사용자에게 방송 프로그램을 일방적으로 전송하는 것과는 달리 사용자가 직접 방송 프로그램을 검색하고 선택할 수 있으며, 사용자가 소유하고 있는 저장 장치에 프로그램을 저장하여 언제든지 필요할 때 방송 프로그램을 이용하는 것을 목표로 하고 있다. 이러한 것을 가능하게 하기 위한 수단으로서 방송 서비스용 메타데이터를 표준화한 것이 TV-Anytime 메타데이터이다[2,3].

MPEG-7[4]은 멀티미디어 컨텐츠를 기술하기 위한 메타데이터 표준으로, 멀티미디어 데이터들을 내용 기반으로 검색하고 처리하는 것을 가능하게 한다. 또한 MPEG-7은 표현자(Descriptor)와 표현구조(Description Scheme)를 이용하여 사용자가 원하는 멀티미디어 데이터의 일부분만을 사용할 수 있다는 특성을 갖는다. 그러므로, MPEG-7을 방송용 메타데이터로 이용 한다면, TV-Anytime 메타데이터에 기술할 수 없는 멀티미디어 컨텐츠에 대한 기술이 가능하므로 사용자에게 내용기반 검색과 같은 질의를 가능하게 한다.

이러한 방송용 메타데이터는 방송이라는 환경의 특성상 실시간의 검색을 필요로 하고, 대용량이다. 그러므로 본 논문에서는 방송 환경에서 최적의 서비스를 제공하기 위해서 방송용 메타데이터를 관리하기 위한 메타

데이터 관리 시스템을 설계하고 구현한다. 메타데이터 관리 시스템은 양방향의 서비스를 제공하는 디지털 방송 환경에서 대용량의 데이터를 관리하고 보다 양질의 서비스를 제공하기 위해서 필수적인 시스템으로 앞선 연구[5]에서 우리는 이미 TV-Anytime 메타데이터를 효율적으로 관리하기 위한 방법을 제안한바 있다. 본 논문에서는 이러한 방법을 기반으로 하여 TV-Anytime 메타데이터를 관리하기 위한 시스템을 설계하고 구현한다. 또한 추가적으로 사용자에게 보다 풍부한 서비스를 제공하기 위하여 MPEG-7 메타데이터를 TV-Anytime 메타데이터와 연계하여 내용기반 검색을 가능하게 한다. 이는 향후 디지털 방송 환경에서 다양한 서비스를 확장 할 수 있는 가능성을 보인다. 현재 방송용 메타데이터를 검색하기 위한 표준 검색어는 존재하지 않는다. 그러므로 우리는 XML 데이터의 검색을 위한 표준 질의어인 XQuery를 방송용 메타데이터의 검색을 위한 질의어로 사용함으로써 방송 메타데이터의 공급자들과 소비자들 사이의 상호 운용성을 유지한다. 우리의 메타데이터 관리 시스템은 실제 디지털 방송 환경을 기반으로 설계하고 구현하였으며, 여기서 사용되는 방송용 메타데이터를 기반으로 구현하였으므로 향후, 보다 효율적이고 안정적인 방송 환경을 구축하기 위한 기반이 될 것으로 사료된다.

본 논문의 나머지 구성은 다음과 같다. 2절에는 관련 연구로 XML[6]문서의 관리를 위한 기준의 연구들과 방송용 메타데이터를 관리하기 위해 이미 제안된 시스템에 대해서 설명하고 있으며, 3절에서는 방송용 메타데이터 관리 시스템의 구조와 시스템을 위한 우리의 접근 방법을 기술하고 있다. 4절에서는 방송용 메타데이터 관리 시스템의 구현을 기술하고 있으며, 5절에서는 결론과 함께 기대효과를 기술한다. 부록에는 우리의 메타데이터 관리 시스템과 기존 XML 관리 시스템간의 성능평가를 기술하고 있다.

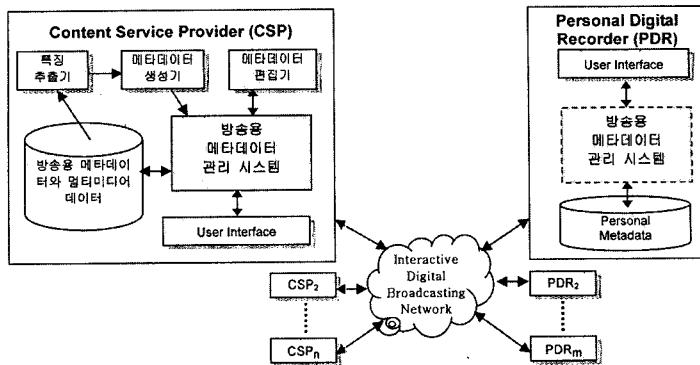


그림 1 메타데이터 기반 방송서비스 시스템

2. 관련연구

XML로 기술되는 방송용 메타데이터의 관리는 결국 XML 데이터의 관리와 직결 된다. XML 데이터를 관리하기 위한 시스템으로는 파일 시스템, 전용 시스템 그리고 기존의 데이터베이스 관리 시스템을 사용하여 관리하는 시스템들이 있다. 가장 손 쉽게 사용되고 있는 파일 시스템 기반의 관리시스템은 대용량의 데이터를 처리하기 위해서는 비효율적이라는 단점이 있다. XML 데이터 모델에 Semistructured 기법을 사용하여 만들어진 XML 전용 데이터베이스 시스템은 XML을 위한 전용 시스템이기 때문에 XML 데이터 관리를 위해서는 효율적일 수 있으나, XML 데이터 이외의 데이터를 함께 사용 할 수 없다는 단점을 가지고 있다. 반면, XML 데이터의 저장을 위하여 관계형 데이터베이스를 사용하는 시스템은 오랜 시간 동안 집중적으로 연구되었던 관계형 데이터베이스의 강력한 장점을 그대로 이용하므로 다른 시스템들보다는 여러 관점에서 효과적일 수 있다. 그러므로 관계형 데이터베이스를 이용한 XML 데이터의 저장, 검색에 관한 연구는 그 동안 많이 이루어 졌고, 현재도 많은 연구 중에 있다[7-19]. 그러나 이러한 방법들은 모두 일반적인 XML 데이터의 관리에는 효율적일 수 있으나 방송용 메타데이터의 특성을 고려하지 않았으므로 본 논문에서 필요로 하는 방송용 메타데이터를 관리하기 위한 특화된 방법으로 보기는 어렵다.

XML 데이터를 관리하기 위한 대표적인 시스템 중 하나는 Inlining의 방법[13]을 이용하여 XML 데이터를 관리하는 시스템으로 [18]에서 연구된 바 있다. 이는 문서의 검색과 재조합 시, Join의 수를 감소시켜 저장과 검색의 효율을 높이려는 방법이다. 그러나 TV-Anytime 메타데이터의 경우, 메타데이터를 구성하는 약 80%의 노드가 Inline될 수 없으므로, 본 논문에서 목표로 하는 방송용 메타데이터를 관리하기 위해서 큰 효과

를 기대하기 힘들다. 또한 TV-Anytime 메타데이터는 단일의 스키마를 따르지만 그 말단이 MPEG-7으로 정의되어 있으므로 데이터베이스 스키마의 확장에 유연하게 대처할 수 있어야 한다. 그러나 [18]에서 제안하고 있는 방법은 XML DTD를 기반으로 데이터베이스 스키마를 생성하므로 DTD가 확장 될 경우 데이터베이스 스키마를 다시 설계해야 한다는 어려움이 존재한다.

디지털 방송을 위한 메타데이터의 특성을 고려하여 이를 효과적으로 저장하고 검색하기 위한 연구는 [20]에서 이미 연구된 바 있다. [20]에서 제안하고 있는 방법은 TV-Anytime 메타데이터 문서의 크기가 크고 문서의 일부분이 주로 재사용된다는 특성을 고려하여, 메타데이터를 단편으로 쪼개어 저장하고 검색하는 방법을 제안하고 있다. 그러나 [20]에서 제안하고 있는 방법은 메타데이터의 검색을 위한 질의어에 관한 언급은 없다. 그러므로 실제 응용에서 메타데이터 검색을 위하여 XPath[21] 또는 XQuery[22]와 같은 XML 질의어를 이용했을 경우 많은 문제들이 발생할 수 있으며 이 문제들을 해결하기 위한 해결책은 실시간의 처리를 요구하는 방송환경에서 반드시 필요하다. 본 논문에서는 XML 데이터 검색을 위한 표준 질의어인 XQuery를 방송용 메타데이터 관리 시스템의 검색을 위한 질의어로 사용한다. 그러므로 응용들 간의 상호 운용성을 확보할 수 있을 뿐만 아니라, XQuery를 이용하여 메타데이터를 검색할 경우 발생할 수 있는 문제점들을 해결하기 위한 방법을 모색한다.

3. 방송용 메타데이터 관리 시스템의 설계

3.1 메타데이터 기반 방송 시스템

TV-Anytime 메타데이터 기반의 방송 시스템은 현재 한국 전자통신 연구원에서 개발 중에 있으며 그림 1은 시스템의 기능 구성도이다[23].

메타데이터 기반 방송 서비스 시스템은 크게 방송 서비

스 제공자(Content Service Provider)와 서비스를 사용하는 방송 사용자 단말(Personal Digital Recorder)로 구성되고, 방송 사용자 단말은 다시 모바일 폰과 같은 하위의 방송 사용자 단말(sub-PDR)로의 데이터 전송이 가능하다. 방송서비스 제공자는 방송용 멀티미디어 데이터에서 특징 및 내용을 추출하여 메타데이터를 생성하고, 추출된 메타데이터에 편집이 필요한 부분은 메타데이터 편집기를 이용하여 방송용 멀티미디어 데이터에 대한 메타데이터를 완성한다. 이때 방송용 메타데이터로는 TV-Anytime 스키마에 정의되어 있는 정보와 추가적으로 MPEG-7의 Visual Descriptor의 일부를 추출한다. 이렇게 생성된 메타데이터는 저장되고, 방송망을 통하여 방송 사용자 단말로 전송된다. 이러한 과정은 메타데이터 기반 방송 시스템은 기존 공중파 방송 서비스 시스템과는 달리 방송망(유·무선에 관계없이)으로 전송되는 데이터가 방송용 멀티미디어 데이터뿐 아니라 메타데이터 역시 함께 전송한다는 특징을 보인다. 방송 사용자 단말은 수신된 방송용 멀티미디어 데이터와 메타데이터를 저장 장치에 저장하여 두었다가 언제 어디에서든지 사용자의 요청이 있을 경우 제공한다. 사용자가 원하는 멀티미디어 데이터를 찾고자 할 경우 사용자는 메타데이터 검색을 통하여 멀티미디어 데이터를 검색하고 검색한 메타데이터에 해당하는 멀티미디어 데이터를 방송 서비스 제공자에게 요청할 수 있다.

본 논문에서 제안하고 있는 메타데이터 관리 시스템은 현재 방송 서비스 제공자 측면에서 기능을 수행하고 있으나 향후, 방송 사용자 단말에서도 사용이 가능할 것으로 기대된다. 특히 우리의 메타데이터 관리 시스템은 메타데이터 검색으로 XQuery를 사용하므로 다른 CSP 또는 PDR들과의 상호운용을 보장받을 수 있다는 장점이 존재한다.

3.2 메타데이터 관리 시스템의 구조

메타데이터 관리 시스템은 방송 서비스 제공자의 주요모듈로 방송용 메타데이터 스키마의 구조에 적절하게 정의된 데이터베이스 스키마에 메타데이터를 효율적으로 저장, 검색하고, 관리하는 역할을 담당하며, 그 구조는 그림 2와 같다.

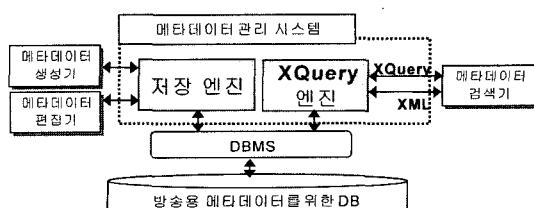


그림 2 메타데이터 관리 시스템의 구조

메타데이터 관리 시스템은 방송용 메타데이터의 양이 대용량이라는 문제점을 효과적으로 처리하기 위하여 데이터베이스 기반 저장 관리 시스템을 사용하며, 시스템의 상용화에 대비하여 보다 고급의 검색 서비스를 할 수 있도록 MPEG-7 메타데이터를 추가하여 내용 기반 검색을 가능하도록 한다. 메타데이터 관리 시스템은 크게 저장 엔진과 XQuery 엔진의 두 가지 엔진으로 구성된다. 저장 엔진은 메타데이터 생성기와 편집기에서 요구하는 메타데이터의 삽입, 삭제, 생성의 기능을 수행하며, XQuery 엔진은 메타데이터 검색기로부터 XQuery를 입력으로 받고 질의를 처리하여 그 결과를 XML 문서로 재조합하여 반환하는 역할을 수행한다.

3.3 메타데이터 저장 엔진

메타데이터 기반 방송 시스템에서 방송용 멀티미디어 데이터의 메타데이터를 효율적으로 저장하고 관리하기 위해서 사용되는 메타데이터 저장 엔진은 그림 3과 같다.

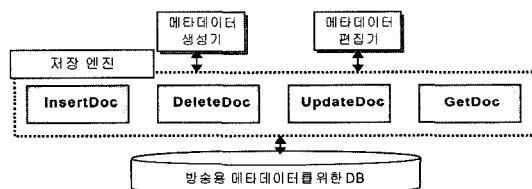


그림 3 메타데이터 저장 엔진의 구조

메타데이터 저장 엔진은 메타데이터의 삽입, 삭제와 생성을 위하여 InsertDoc, DeleteDoc, UpdateDoc, GetDoc의 네 개의 모듈로 구성된다. InsertDoc은 메타데이터 생성기 또는 편집기로부터 입력 받은 메타데이터를 구문 분석하여 DOM[24] Tree를 순회하면서 메타데이터에 기술되어 있는 내용 정보와 구조 정보를 얻어온 후, 데이터베이스 스키마에 적절한 SQL 문을 구성하여 메타데이터를 데이터베이스에 삽입하는 역할을 한다. 이때, MPEG-7 메타데이터는 TV-Anytime 메타데이터와의 연계를 위하여 TV-Anytime 메타데이터의 CRID(Content Reference Identifier)와 SegmentID의 쌍을 고유한 키로 정의한 후 저장한다. DeleteDoc은 TV-Anytime 메타데이터의 고유한 CRID를 입력으로 받아서 CRID에 해당하는 데이터를 모두 제거하는 역할을 담당한다. UpdateDoc은 새로운 메타데이터를 입력으로 받아 이미 저장되어 있는 메타데이터를 삭제하고 새로운 메타데이터를 삽입하는 과정을 통하여 갱신을 수행한다. 현재, XQuery는 갱신에 관해서는 정의하지 않고 있기 때문에 향후, XQuery가 갱신을 지원하면 XQuery를 이용하여 갱신 해야 할 것이다. GetDoc은 CRID를 입력으로 해당 메타데이터를 얻기 위한 모듈로써 메타

데이터 편집기가 메타데이터 검색기를 이용하지 않고 CRID로 직접 TV-Anytime 메타데이터를 불러 올 수 있도록 한다.

3.4 방송용 메타데이터 저장을 위한 데이터베이스 스키마 설계

3.4.1 방송용 메타데이터를 데이터베이스에 저장하기 위한 방법

비정형 데이터인 XML 데이터를 데이터베이스에 저장하기 위한 방법은 현재, 많은 업체와 연구기관에서 활발히 진행 중이다. 특히, 관계형 데이터베이스에 XML 데이터를 저장하는 방법[7-15]은 XML 데이터의 구조 정보를 어떻게 유지하는가에 많은 연구가 집중되고 있다. 그러나, 이러한 방법들은 일반적인 XML 데이터의 저장 방법을 제안하기 때문에 장점과 함께 단점을 함께 포함한다. 그러므로 본 논문에서는 TV-Anytime 메타데이터와 추가적 정보인 MPEG-7 메타데이터의 저장과 검색을 위하여 기존 방법들의 단점을 제거하고, 장점들을 조합하여 방송용 메타데이터에 특화된 저장방법을 사용하여 사용자의 질의를 보다 효율적으로 수행할 수 있도록 한다.

TV-Anytime 메타데이터는 그 밀단이 대부분 MPEG-7으로 정의되어 있다. 이러한 이유는 TV-Anytime 메타데이터의 밀단에 MPEG-7 메타데이터를 연결하여 멀티미디어 검색과 같은 다양한 서비스를 제공하기 위함이다. 그러나 이러한 특성은 메타데이터가 확장될 때 데이터베이스 스키마 역시 유연하게 확장되어야 한다는 요구사항을 야기시킨다. 또한, TV-Anytime 메타데이터의 하위 노드, 또는 하위 노드의 집합은 재사용이 빈번하게 발생한다[20]. 예를 들어 ProgramInformation Element의 자식 노드들은 EPG(Electronic Program Guide)를 생성하기 위하여 재사용되며, 한 프로그램의 출연자에 관한 메타데이터는 다른 프로그램의 메타데이터를 생성하기 위해 재사용 되기도 한다. 이러한 TV-Anytime 메타데이터의 특성은 메타데이터를 구성하는 Element 노드들을 개별적으로 관리해야 한다는 요구사항을 야기시킨다. 본 논문에서는 이러한 방송용 메타데이터의 요구사항을 만족시키기 위하여 Binary 방법[11]에서 제안한 아이디어를 사용한다. Binary 방법은 XML 문서에서 동일한 이름을 가지는 모든 Element 노드들을 하나의 테이블에 저장하는 방법으로, XML 문서의 구조정보를 가장 손쉽게 표현할 수 있는 방법 중 하나이다. Binary 방법의 장점은 데이터베이스 스키마의 확장에 유연하게 대처할 수 있으며, 메타데이터를 Element 노드 단위로 관리하기 쉽다는 것이다. 이러한 장점은 앞에서 언급한 TV-Anytime 메타데이터의 특성을 고려하여 데이터베이스 스키마를 생성할 수 있도록

한다. 그러나 본 논문에서 사용하고 있는 방법은 Binary 방법의 기본 아이디어인 데이터베이스 테이블을 Element 노드 단위로 나누어 저장하는 방법만을 취한다. Binary 방법의 단점은 노드의 구조를 검색하기 위해서는 반드시 join연산을 필요로 한다는 것이다. 그러므로 본 논문에서는 Element 노드 간의 연관관계를 나타내기 위하여 Binary방법과 함께 Dewey numbering 방법[12]을 사용한다.

데이터베이스에 저장된 XML 데이터는 이를 검색하고 재조합 하기 위하여 각 노드들의 구조정보를 유지하면서 저장해야 한다. 이를 위해 현재, 노드들의 구조 정보를 표현하기 위한 여러 가지 방법[12,15,25,26]들이 제안되었으며, 본 논문에서는 그 제안된 방법들 중 Dewey Numbering 방법[12]을 사용한다. Dewey number는 최상위 노드부터 해당 노드까지의 경로를 Dewey Decimal Classification을 이용하여 나타내는 방법으로, 모든 Element 노드의 ID가 될 수 있을 뿐만 아니라 부모-자식의 연결, 형제 노드들의 순서 등과 같은 각 노드들의 연관관계를 알 수 있다. 이러한 특성은 기존의 Binary 방법에서 노드의 구조를 표현하기 위해서 사용된 부가적인 정보(부모-자식간의 정보, 형제들 간의 정보)들을 제거할 수 있도록 하므로 저장공간을 절약할 수 있다는 장점을 갖는다. Dewey 방법의 단점은 문서의 생성 시, 해당 노드의 우측 자손 노드들의 Dewey number를 재구성 해야 한다는 것이다. 그러나 TV-Anytime 메타데이터의 경우 문서의 생성은 극히 드문 일이고, 만약 문서의 생성이 발생하면 [27,28]에서 제안하는 ORDPATH 방법을 사용하여 Dewey number의 재구성 없이 문서의 생성을 수행할 수 있다.

MPEG-7 메타데이터는 TV-Anytime 메타데이터 이외의 부가적인 정보를 가진 메타데이터로 단지 검색을 위해서만 사용된다. 그러므로 저장 역시 검색을 위한 형태로 저장한다. 본 논문에서는 MPEG-7 메타데이터의 실제 이미지 특징 같은 CLOB을 이용하여 저장하고, 이미지와 TV-Anytime 메타데이터와의 연동을 위하여 CRID와 SegmentID를 두어 어떤 TV-Anytime 메타데이터의 어느 부분과 연결되는지를 관리한다.

3.4.2 데이터베이스에 저장된 메타데이터의 검색 방법

앞에서 이미 언급한 것처럼 Binary 방법은 서로 다른 이름을 갖는 Element 노드들마다 서로 다른 테이블에 나누어 저장되므로 구조 검색을 위하여 테이블간의 많은 Join이 필요하다는 단점이 존재한다. 이러한 문제를 해결하기 위하여 우리는 Dewey numbering 방법을 취하였다. 그러나 문자로 표현된 XPath 표현을 효율적으로 처리하기 위해서는 Dewey number와 함께 문자로 표현된 Path의 정보가 별도로 필요하다. 그러므로 우리

는 Path 테이블 방법[15]의 아이디어를 사용한다. Path 테이블 방법은 각 노드에 해당하는 모든 Path들을 하나의 테이블에 저장하고 구조검색 시 저장된 정보를 이용하는 방법으로, Binary 방법의 단점인 구조검색을 위한 불필요한 Join을 수행하지 않고도 문서의 모든 노드를 검색할 수 있다는 장점이 존재한다. 본 논문에서 TV-Anytime 메타데이터를 검색하기 위해서 Path 테이블을 사용하는 방법이 가능한 것은 TV-Anytime 메타데이터가 단일의 스키마를 기반으로 기술된다는 특성이 존재하기 때문이다. 만약 이러한 특성이 존재하지 않는다면, 메타데이터에 따라 수없이 많은 Path가 생성될 수 있으므로 Path 테이블의 사용은 불가능하다. 기존에 제안된 Path 테이블 방법[15]은 Path 테이블 이외에 각 노드들의 정보를 저장하기 위하여 Element 테이블, Attribute 테이블, Text 테이블의 세 종류의 테이블을 사용한다. 그러나 본 논문에서는 Binary 방법을 이용하여 각 Element 노드마다 하나의 테이블을 생성하므로, 방송용 메타데이터와 같은 대용량의 메타데이터를 검색하고 관리하기에 효율적인 방법을 제안하고 있다. 또한, Path 테이블 방법은 XQuery에 표현되는 중첩된 경로들을 파악하고 처리하기 위해서는 String 비교를 해야 한다는 단점이 존재한다. 그러나 본 논문에서는 노드의 구조정보를 표현하기 위하여 Dewey number를 사용하도록 이를 이용하여 String 비교 없이 중첩된 경로를 알 수 있다는 장점이 존재한다.

3.4.3 메타데이터의 재조합을 위한 방법

본 논문에서 방송용 메타데이터의 검색을 위해서 사용하는 XQuery의 질의 결과는 TV-Anytime 메타데이

타 문서의 전체 또는 일부분이다. 그러므로 데이터베이스에 나누어 저장된 메타데이터를 사용자에게 반환하기 위해서는 문서의 재조합이 필요하다. 현재 문서의 재조합을 효율적으로 수행하기 위한 연구들[10, 16, 18]이 진행 중에 있으나 대부분의 방법들은 문서의 검색 문제와 연관되어 있다. 즉, 효율적인 검색을 위해서는 되도록 XML 문서를 많이 나누어 저장해야 하지만, 문서의 재조합을 위해서는 문서를 나누지 않고 저장하는 것이 효율적이다. 그러나 기존의 대부분 연구들에서는 각각 한 가지 문제만을 다루고 있기 때문에 실제 응용에 적용하기 위해서는 두 가지 특성을 모두 고려하여야 한다. 앞서 본 논문에서 사용한 방법들은 모두 효율적인 검색을 위한 방법들로 문서의 재조합을 위한 방법과는 거리가 멀다. 그러므로, 우리는 문서의 재조합을 위해서 TV-Anytime 메타데이터 문서 전체를 저장하고 각 노드에 해당하는 위치정보를 이용하여 문서의 재조합 없이 원하는 결과를 얻어올 수 있도록 한다. 그러나 말단 노드의 경우에는 재조합을 위해서 Join이 발생하지 않으므로 위치 정보를 저장하지 않는다. 이러한 방법은 데이터의 중복 저장이라는 문제를 야기하지만, 방송용 메타데이터를 관리하는 응용의 경우, 그 특성상 실시간 검색을 요구하고 있으므로 저장의 측면보다는 효율적인 검색을 위한 최소한의 중복저장을 허용한다.

3.4.4 방송용 메타데이터 저장을 위한 데이터베이스 스키마

그림 4는 ‘네이비’라는 드라마를 위한 TV-Anytime 메타데이터의 ‘본문문서이고, 그림 5는 ‘네이비’와 ‘러브 스토리’라는 드라마의 메타데이터가 저장된 데이터베이

```
<TVAMain version="1">
<ProgramDescription>
  <ProgramInformationTable>
    <ProgramInformation programId="crid://www.imbc.com/navy103052300001">
      <BasicDescription>
        <Title type="main">네이비</Title>
        <Synopsis> 이지스함 선발요원이 되기 위해 경쟁중인 민석과 현수는 소연을 둘러싼 둘 사이의 대립에 의해 항상 충돌하는데...</Synopsis>
        <Keyword>           </Keyword>
      </BasicDescription>
    </ProgramInformation>
  </ProgramInformationTable>
  <ProgramLocationTable>
    <OnDemandProgram>
      <Program crid="crid://www.imbc.com/navy103052300001"></Program>
      <ProgramURL>D:\media\drama\imbc_navy.mpg</ProgramURL>
    </OnDemandProgram>
  </ProgramLocationTable>
  <SegmentInformationTable timeUnit="PT1001N30000F">
    <SegmentList>
      <SegmentInformation segmentId="SID_0_0_148">...
      </SegmentInformation>...
    </SegmentList>
  </SegmentInformationTable>
</ProgramDescription>
</TVAMain>
```

그림 4 Sample TV-Anytime 메타데이터 문서

ID_Path	PathExp
1	TVAMain
2	TVAMain/ProgramDescription
...	...
6	TVAMain/ProgramDescription/ProgramInformationTable/ProgramInformation
7	TVAMain/ProgramDescription/ProgramInformationTable/ProgramInformation/BasicDescription
8	TVAMain/ProgramDescription/ProgramInformationTable/ProgramInformation/BasicDescription/Title
9	TVAMain/ProgramDescription/ProgramInformationTable/ProgramInformation/BasicDescription/Synopsis

'Path' Table		
ID_DOC	CRID	CONTENT
1	crid://www.imbc.com/navy103052300001	...
2	crid://www.imbc.com/LoveStory17042002	...

'Doc' Table							
ID_DOC	ID_TITLE	ID_Path	TITLE	ID_DOC	ID_PROGRAMINFOTMATION	ID_Path	POSITION
1	1:1:1:1:1:1:1:1	8	네이비	1	1:1:1:1:1	6	650, 789
2	2:1:1:1:1:1:1	8	러브스토리	2	2:1:1:1:1	6	690, 870
2	2:1:1:1:1:2:1	8	Love Story				

'Title' Tables		'Program Information' Tables	
ID_DescriptionImage	Att_CRID	Att_SegmentId	DescriptionImage
1	crid://www.imbc.com/navy103052300001	SID_0_0_148
2	crid://www.imbc.com/LoveStory17042002	SID_0_0_148

'MPEG7 For TVA Time' Table	

그림 5 TV-Anytime과 MPEG-7메타데이터의 저장을 위한 데이터베이스 스키마

스 스키마의 기본 구조 예이다.

TV-Anytime 메타데이터를 저장하기 위한 데이터베이스 스키마의 기본 구조는 Path 테이블, Doc 테이블, Element 테이블들로 구성된다. 그림 5는 TV-Anytime 메타데이터의 저장을 위한 세 가지 종류의 테이블들과 MPEG-7 메타데이터의 저장을 위한 테이블을 보인다. Path 테이블은 데이터베이스에 저장된 TV-Anytime 메타데이터의 각 노드들에 대한 모든 경로를 저장하고 있는 테이블로 각 행들은 Element 테이블과 사상된다. 즉, 하나의 Element 테이블은 자신을 표현하는 하나 이상의 Path 테이블의 행과 사상된다. Path 테이블에 저장된 값들은 사용자가 질의한 XQuery 질의의 구조정보를 테이블 간의 Join 없이 직접 찾을 수 있도록 함으로써 검색의 시간을 효과적으로 줄일 수 있다. Doc 테이블은 TV-Anytime 메타데이터 문서를 CLOB형태로 저장하고 있는 테이블로 사용자가 전체문서의 반환을 요청할 경우, 또는 문서의 중간 노드의 반환을 요청할 경우에 문서의 재구성을 위해서 테이블의 Join 없이 문서를 생성할 수 있도록 한다. Element 테이블은 TV-Anytime 메타데이터의 Element 노드들 중 동일한 이름을 갖는 노드들을 저장하는 테이블로 크게 노드가 말단일 경우와 그렇지 않은 경우로 나뉜다. 노드가 말단 노드인 경우 그림 5의 Element 테이블 중 'Title' 테이블의 구조와 동일한 구조를 갖으며, 말단이 아닐 경우 'ProgramInformation' 테이블과 같은 구조를 갖는다. 말단 노드를 저장하는 테이블과 말단이 아닌 노드를 저장하는 테이블의 가장 큰 차이는 Position 필드의 유무

이다. Position 필드는 Doc 테이블에 저장된 메타데이터에서 해당 노드의 위치를 저장하고 있는 필드로, 말단 노드의 경우 Position필드를 이용하는 시간과 이용하지 않고 노드를 재 조합하는 시간이 거의 유사하므로 Position필드를 이용하지 않는다. 아래 그림의 경우 'ProgramInformation' 테이블의 첫 번째 노드의 Position 값은 'Doc' 테이블에 CLOB으로 저장된 문서들 중 ID가 1인 문서에서 시작 위치가 650이고 끝 위치가 789까지의 문서 일 부분을 나타내고 있다.

MPEG-7 메타데이터는 TV-Anytime 메타데이터 이외의 부가적인 정보를 가진 메타데이터로 단지 검색을 위해서만 사용된다. 그러므로 저장 역시 검색을 위한 형태로 저장한다. 그림 5의 마지막 테이블은 MPEG-7 메타데이터의 저장을 위한 테이블의 구조이다. CRID와 SegmentID 필드에는 해당 MPEG-7 메타데이터가 어떤 TV-Anytime 메타데이터의 어느 부분과 연결되는지를 위한 정보가 저장된다. 이는 TV-Anytime 문서의 삽입, 삭제와 갱신이 발생할 경우 MPEG-7 메타데이터 역시 수정이 일어나야 하고, 키워드 검색과 내용기반 검색을 혼합하여 사용하기 위함이다. DescriptionImage 필드에는 실제 이미지의 특징 값을 저장하고 검색엔진에서 SR-Tree[29]를 이용하여 검색하기 위해서 사용된다.

3.5 메타데이터 검색용XQuery 엔진

메타데이터 검색을 위한 XQuery 엔진의 구조는 그림 6과 같다. 본 논문에서는 메타데이터의 검색을 위해서 현재, 인터넷 표준화 단체인 IETF와 W3C에서 XML

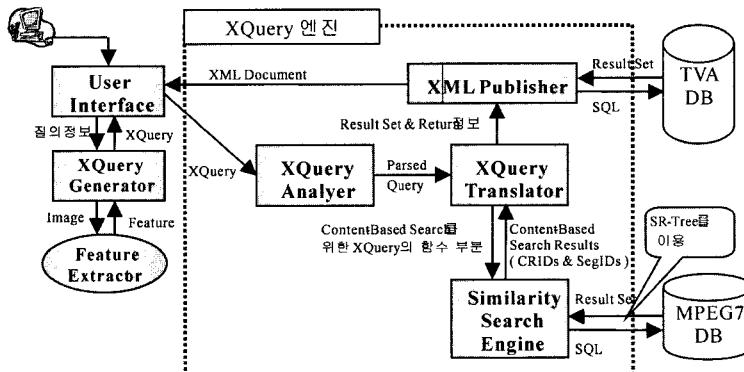


그림 6 XQuery 엔진의 구조

데이터 검색을 위한 국제 표준 질의어로 제정 중(현재 Working Draft)에 있는 XQuery를 사용한다[22]. XQuery는 기존의 XML 데이터 검색을 위한 질의어들의 장점을 포함하는 질의어로, XQuery를 방송용 메타데이터의 검색을 위해서 사용한다면 방송용 메타데이터의 저장 형식에 무관하게 질의가 가능하므로 다른 응용과의 상호운용성을 유지할 수 있다. 또한, XQuery의 사용자 정의 함수를 이용하여 내용기반 검색과 같은 질의의 확장을 가능하도록 한다. 그러나 XQuery는 일반사용자가 사용하기에는 어렵다는 단점이 존재한다. 그러므로, 우리는 일반사용자가 사용하기 편리한 형태의 사용자 인터페이스를 제공하여 검색조건을 얻고, 이를 취합하여 XQuery를 생성한 후, XQuery 엔진의 검색어로 사용한다.

사용자가 인터페이스를 통해 찾고자 하는 질의 정보를 입력하면, 질의 정보는 XQuery Generator에 의해 XQuery로 구성된다. 만약 사용자가 질의 입력 정보에 이미지를 이용한 내용 기반 검색을 선택하였다면, XQuery Generator는 Feature Extractor를 사용하여 이미지에 관한 특징을 추출하여 내용 기반 검색 정보를 혼합한 XQuery를 생성한다. 여기까지가 클라이언트에서 수행되는 모듈로써, 생성된 XQuery는 검색을 위하여 서버 쪽으로 전송된다. 서버 쪽으로 전송된 XQuery는 XQuery Analyzer에 의해 분석된다. XQuery Analyzer는 본 논문에서 구현한 XQuery Parser를 이용하여 XQuery가 문법에 맞는지 확인하면서 Syntax Tree를 생성한다. 구문 분석을 마치면, XQuery Analyzer는 Syntax Tree를 순회하면서 질의의 특성에 맞게 정보를 추출한다. XQuery Translator는 분석된 XQuery 정보를 이용하여 실제 메타데이터의 내용이 저장되어 있는 데이터베이스에 질의하기 위한 SQL을 생성하는 역할을 담당한다. 이때, XQuery 정보에 내용 기

반 검색 정보가 존재할 경우 이 정보를 Similarity Search Engine으로 제공한다. Similarity Search Engine은 입력 받은 이미지의 특징 정보와 기존에 데이터베이스에 저장되어 있던 이미지 정보를 SR-Tree[29]를 이용하여 유사한 이미지들의 CRID와 SegmentID의 목록을 XQuery Translator에 반환한다. XQuery Translator는 내용기반 검색이 존재할 경우는 검색된 이미지의 CRID, SegmentID와 키워드 기반 검색정보를 가지고 SQL을 생성하고, 내용기반 검색이 존재하지 않을 경우는 키워드 기반 검색 정보만을 가지고 SQL을 구성한다. XQuery Translator는 생성된 SQL문을 수행하고 수행된 결과를 사용자에게 반환할 형태인 XML 문서로의 재조합을 위하여 XML Publisher의 입력으로 제공한다. 이때, XQuery Translator는 SQL과 함께 XQuery의 Return절에 기술된 반환해야 할 XML 문서의 구조정보를 함께 제공한다. XML Publisher는 이 두 가지 정보를 이용하여 데이터베이스에 질의하고, 검색결과를 사용자가 원하는 형태의 XML 문서로 재구성하여 사용자에게 전송한다.

4. TV-Anytime 메타데이터 관리 시스템 구현

4.1 메타데이터 저장 엔진

메타데이터 저장 엔진은 현재 문서단위로 삽입, 삭제와 갱신을 수행하며, 그림 7은 메타데이터 저장을 위한 사용자 인터페이스이다.

저장엔진은 TV-Anytime 메타데이터의 고유한 ID인 CRID(Content Reference IDentifier)를 이용하여 데이터베이스에 저장되어 있는 메타데이터들을 관리한다. 사용자는 메타데이터를 불러오기 위해서 CRID를 검색 조건으로 하는 XQuery를 생성하여 문서를 불러올 수 있다. 또한, 메타데이터의 삭제 역시 CRID를 이용하여 삭제한다. 이때, CRID를 입력으로 TV-Anytime 메타

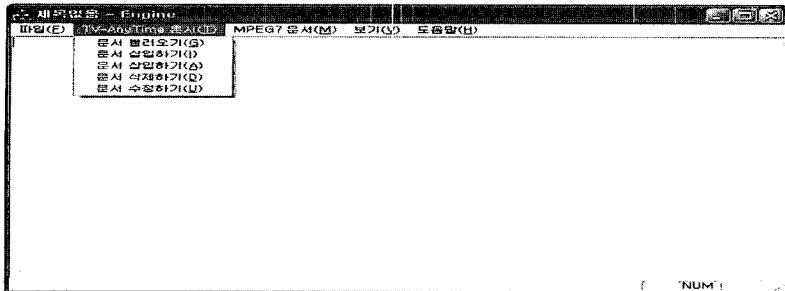


그림 7 저장엔진 인터페이스

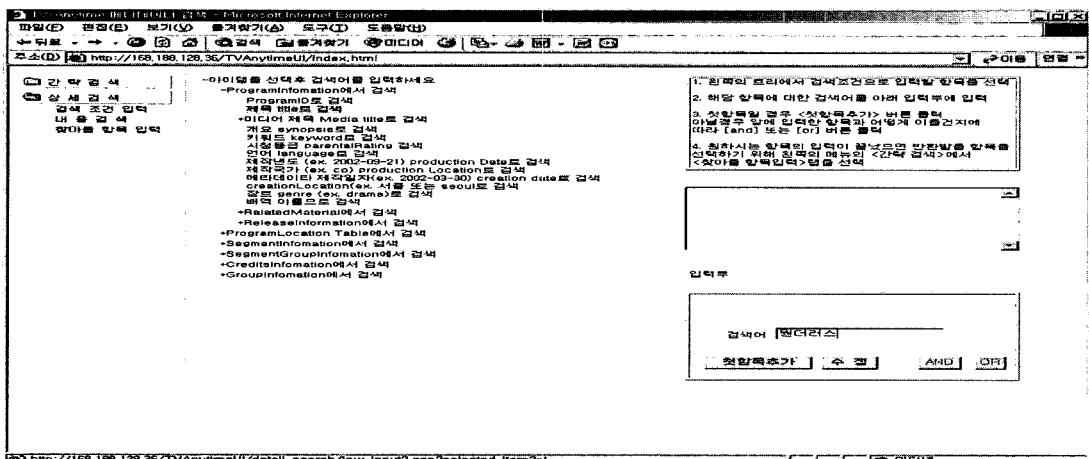


그림 8 메타데이터 검색을 위한 사용자 인터페이스

이터를 삭제하면, 이와 연관된 MPEG-7메타데이터 역시 동시에 삭제된 후, SR-Tree가 생성된다. 메타데이터의 삽입은 문서 혹은 풀더 단위로 저장이 가능하고, MPEG-7 메타데이터의 삽입은 삽입과 동시에 SR-Tree를 생성하므로 사용자는 실시간으로 생성된 메타데이터를 검색할 수 있다.

본 논문에서 설계하고 구현한 메타데이터 저장 엔진은 Oracle 9i와 SQL 서버의 두 가지 데이터베이스 기반에서 구현하였다. 그러나, 데이터베이스 관리를 위한 SQL은 표준 SQL을 사용했으므로 다른 종류의 관계형 데이터베이스에서 변경 없이 사용할 수 있도록 개발하였다.

4.2 메타데이터 검색용 XQuery 엔진

방송용 메타데이터를 검색하기 위한 인터페이스는 그림 8과 같이 TV-Anytime 메타데이터를 분석하여 질의가 가능한 모든 노드를 대상으로 질의 할 수 있도록 설계하였으며, 사용자가 메타데이터의 구조를 쉽게 파악하기 위하여 Tree구조를 이용하였다.

사용자 인터페이스의 좌측은 간략검색과 상세검색으로 나뉘어 있다. 그림 8은 좌측의 상세검색에 “검색 조

건 입력”을 선택했을 경우 나타나는 화면으로, TV-Anytime 메타데이터를 분석하여 사용자가 질의할 수 있는 Element와 Attribute들을 Tree형태로 나타내고 있다. 그림에서는 최상위 노드들과 중간 노드 일부만을 표현하고 있으나 자식 노드가 존재하는 목록은 펼쳐서 말단 노드까지 질의가 가능하다. 사용자는 Tree를 이용하여 원하는 목록을 선택하여 키워드로 질의할 수 있으며, “AND”와 “OR”를 이용하여 논리 연산 검색을 수행할 수 있다. 사용자는 키워드 검색과 함께 또는 키워드 검색과는 별도로 그림 9와 같은 내용기반 검색을 할 수 있다. 현재, 내용기반 검색은 Color와 Texture 검색을 지원한다. 이를 각각은 개별적으로 SR-Tree index를 구성하여 검색할 수 있도록 하였으므로 필요에 따라 다른 특징 값의 추가가 용의하도록 구현하였다.

사용자의 검색조건 입력이 완료되면 사용자는 반환 받을 값을 선택하기 위하여 인터페이스의 좌측에 있는 “찾아올 항목 입력”을 선택한다. 찾아올 항목을 위한 인터페이스는 그림 8과 동일한 형태의 인터페이스를 제공하여 사용자가 전체문서, 문서의 일부 혹은 문서 일부분의 조합을 선택하여 반환 구조를 정의할 수 있도록 한

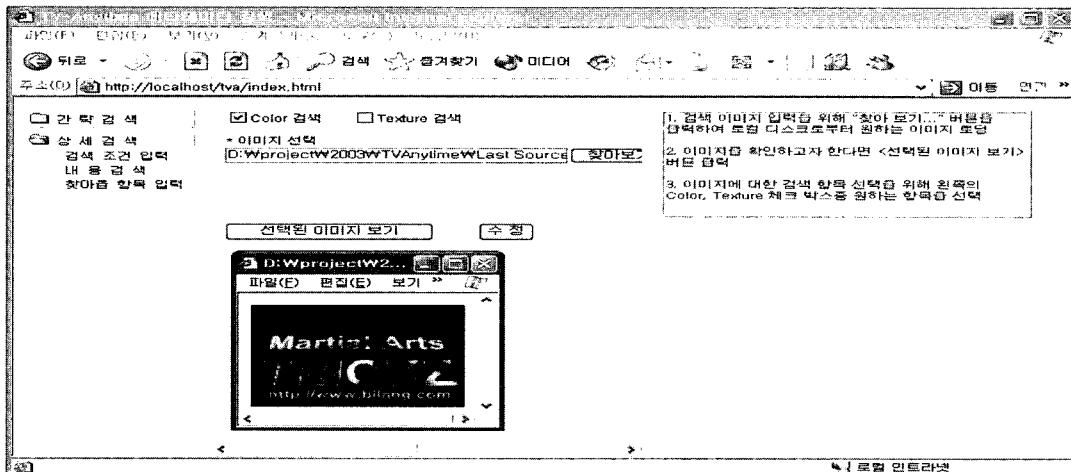


그림 9 내용기반 검색을 위한 사용자 인터페이스

```

V[] : XQuery의 WHERE와 RETURN절에 존재하는 변수 집합
FV[] : XQuery의 FOR절에 기술된 변수들의 집합
XE[] : XQuery의 FOR절에 기술된 XPath표현들의 집합
FXE : Full XPath Expressions

If(ImageSearch){
    ImageCRID=getInformation(); (0)
}
for (each V[])
{
    find XE[] assigned to V[];
    find the last Element node name described into XE[]];
    generate the FROM clause of SQL query by using last Element node name; (1)
    fnXE = the first Element node described into XE[]];
    while(Document Node != fnXE) {
        for(each FV[])
        {
            if(FV[k] == fnXE){
                FXE += XE[j];
                XE[j] = XE[k];
                fnXE = the first node of XE[j];
                newTableName = find the terminal Element node name of XE[]];
                if(isn't a newTableName into FROM clause){
                    add new table(including path table) into the FROM clause of SQL query; (2)
                }
                generate the WHERE clause of SQL query for comparing ancestor-descendant relationship; (3);
            }
        }
    }
    generate WHERE clause of SQL query for retrieving the structural information of V[] by using FXE; (4)
    if(V[] == WHERE variable) {
        generate WHERE clause of SQL query for retrieving the content of V[]]; (5)
    }
    Else{
        generate the SELECT clause of SQL query; (6)
    }
}

```

그림 10 XQueryToSQL 알고리즘

다. 메타데이터의 간략검색은 키워드 검색 시 미리 검색 항목을 지정해 놓은 후, 사용자는 원하는 아이템을 선택하여 검색할 수 있도록 하는 검색으로, 사용자 인터페이스를 통하여 입력 받은 검색조건은 특정한 알고리즘을 거쳐 검색엔진에서 사용할 수 있는 질의어인 XQuery로 변환된다.

그림 10은 XQuery를 SQL로 변환하기 위해서 본 논문에서 사용한 알고리즘이고, 그림 11은 견본 XQuery

와 변환된 SQL이다.

그림 11의 XQuery는 멀티미디어 컨텐츠의 제목 (Title)이 “네이비”이고, 특정 이미지의 Color를 추출하여 유사한 특징을 지닌 메타데이터의 Program-Information을 반환을 요구하는 질의이다. 변환 알고리즘의 (0)부분은 XQuery에 내용기반 검색이 존재할 경우 처리하는 부분으로 예제의 경우 line 9와 10이 처리된다. 내용기반 검색의 결과는 해당 이미지와 연결되는

```

1: <Results>
2: for $d in input("TVAnyTime")
3: return <Result>
4: distinct-values(
5:   for $p1 in $d/TVAMain/ProgramDescription
6:     for $p2 in $p1/ProgramInformationTable/ProgramInformation
7:       for $p3 in $p2/BasicDescription/Title
8:         for $p4 in $p1/ProgramLocationTable/OnDemandProgram/Program/@crid
9:           for $CRS in ImageSearchForColor()
10:             for $ImageCRID in $CRS/ColorResult[position()<=1] /@CRID
11:               where contains(string($p3), "네이버") and $p4=$ImageCRID
12:             return $p2
13: )</Results>
14: </Results>
15: SELECT DISTINCT ProgramInformation.id_Doc, ProgramInformation.Position
16: FROM ProgramInformation, Title, Program, ProgramDescription, Path Path0, Path Path1, Path Path2
17: WHERE Title.Title like '%네이버%' AND
18:   Title.id_Path = Path0.id_Path AND
19:   Path0.Pathexp = '/TVAMain/ProgramDescription/.../BasicDescription/Title' AND
20:   Program.att_crid = 'crid://...103040700624' AND
21:   Program.id_Path = Path1.id_Path AND
22:   Path1.Pathexp = '/TVAMain/ProgramDescription/.../OnDemandProgram/Program' AND
23:   ProgramInformation.id_Path = Path2.id_Path AND
24:   Path2.Pathexp = '/TVAMain/ProgramDescription/.../ProgramInformation' AND
25:   Title.id_Title like ProgramInformation.id_ProgramInformation || '%' AND
26:   Program.id_Program like ProgramDescription.id_ProgramDescription || '%' AND
27:   ProgramInformation.id_ProgramInformation like ProgramDescription.id_ProgramDescription || '%'

```

그림 11 견본 XQuery와 변환된 SQL

메타데이터의 CRID 또는 CRID와 segmentID의 집합으로 예제의 경우 가장 유사한 이미지를 포함하고 있는 메타데이터의 CRID 하나만을 반환한다. XQuery의 처리는 XQuery의 Where 또는 Return절에 기술된 변수 (\$p3, \$p4, \$p2)를 기반으로 SQL로 변환된다. 알고리즘의 (1)부분은 먼저 예제 XQuery의 Where절에 기술된 \$p3에 해당하는 For절(line 7)의 XPath 표현 중 가장 말단의 Element 노드의 이름을 이용하여 SQL의 FROM절을 생성하는 단계로, Line 6의 Title 테이블을 생성한다. 예제의 \$p4와 \$p2역시 각각 Program과 ProgramInformation 테이블을 생성한다. 알고리즘의 while loop 부분은 XQuery의 \$p2, \$p3, \$p4에 해당하는 XPath 표현을 이용하여 말단 노드부터 최상위 노드까지 전체 경로를 생성해가면서 FOR절에 의해서 나누어진 부모-자손의 관계를 확인하는 부분이다. 이때 만약 각 변수들이 동일한 조상으로부터 나뉘었다면, 이를 구분하기 위한 SQL을 생성한다. 알고리즘의 (2)에 의해서 생성된 SQL은 예제의 Line 16 중 ProgramDescription, Path0, Path1, Path2 테이블이고, (3)에 의해서 생성된 SQL은 Line 18, 21, 23, 25~27이다. 알고리즘의 (4) 부분은 변수에 해당하는 노드의 경로를 검색하기 위한 부분으로 예제의 Line 19, 22, 24를 생성한다. 알고리즘의 (5) 부분은 SQL의 Where절 중 조건을 생성하는 부분으로 예제의 Line 17, 20을 생성하며, 알고리즘의 (6)은 SQL의 Select절(Line 15)을 생성한다.

현재, 본 논문에서는 사용자 인터페이스를 이용하여 방송용 메타데이터 검색을 위한 XQuery의 subset을 사

용하고 있다. XQuery subset의 범위는 XQuery의 14가지 표현 중 XPath 표현과 FLOWR 표현을 처리하며 논리 연산자인 'and'와 'or' 연산자, 비교 연산자와 Constructor의 일부를 사용하고 있다. 그러나 FLOWR 중 중첩된 질의는 처리하지 않고 있다. 위 알고리즘에서 볼 수 있듯이 우리의 XQuery 처리 방법은 XQuery 내에 선언된 FLOWR절에서 XPath 표현을 추출하여 처리하고 있다 그러므로 향후, 보다 풍부한 질의를 제공하기 위해서는 XQuery의 표준화 작업에 맞추어 모든 XQuery와 함수들을 지원해야 할 것이다.

5. 결 론

본 논문에서 제안한 방송용 메타데이터 시스템의 기여는 다음과 같다. 첫째, 기존의 XML 데이터를 저장, 검색 또는 재조합 하기 위한 제안들을 분석하고 이들의 장점을 취합하여 방송용 메타데이터의 특성에 맞는 시스템을 제안하였다. 둘째, 방송용 메타데이터 검색을 위해 XML 데이터 검색을 위한 표준 질의어인 XQuery를 사용하여 상호 운용성을 높였다. 셋째, 방송용 메타데이터 표준인 TV-Anytime과 멀티미디어 컨텐츠를 기술하기 위한 표준인 MPEG-7을 연계하여 사용자가 방송용 컨텐츠를 검색하기 위해 보다 풍부한 질의를 가능하게 하였다. 넷째, 프로토 타입을 설계하고 구현한 후, 기존의 상용 시스템과 평가하였다. 이는 향후, 보다 안정적이고 효율적인 디지털 방송 환경을 확립하기 위한 자료로 사용될 것이다. 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 시스템의 서버에서뿐만 아니라 향후, 사용자 단말에서 검색을 위한 모듈로도 사용될 수 있을 것으로 사료된다. 현재, XQuery는 아직 표준화 과정이 진행 중인 단계이므로, 본 논문에서는 전체 XQuery를 지원하기보다는 방송용 메타데이터의 검색을 위한 XQuery의 일부분만을 지원하고 있다. 그러나 향후, XQuery의 표준이 완성되면 전체 XQuery를 지원하여 사용자에게 보다 풍부한 질의를 가능하도록 할 것이다.

참 고 문 헌

- [1] TV Anytime Specification Series, August 2001. (<http://www.tv-anytime.org/>)
- [2] S. Pfeiffer & U. Srinivasan, "TV Anytime as an application scenario for MPEG-7," Proc. ACM Multimedia 2000, Los Angeles, October 2000.
- [3] 한국정보통신기술협회, "TV-Anytime Forum 최근동향," Proc. IT Forum korea 2002, May 2002.
- [4] J. M. Martinez, Overview of the MPEG-7 Standard. ISO/IEC JTC1/SC29/WG11 N4509, December 2001. (<http://mpeg.telecomitalialab.com/standards/mpeg-7/mpeg-7.html>)
- [5] J. H. Park, B. K. Kim, Y. H. Lee, M. W. Lee, M. O. Jung & J. H. Kang, "XQuery-based TV-Anytime Metadata Management," In Proc. Of the DASFAA Conf., April 2005.
- [6] W3C, Extensible Markup Language (XML) Version 1.1, Recommendation, February 2004. (<http://www.w3.org/TR/2004/REC-xml11-20040204/>)
- [7] J. McHugh, S. Abiteboul, R. Goldman, D. Quass, & J. Widon, "Lore: A Database Management System for Semi-structured Data," ACM SIGMOD Record Vol. 26, No. 3, September 1997.
- [8] A. Deutsch, M. Fernandez, & D. Suciu, "Storing Semistructured Data with STORED," In Proc. of the ACM SIGMOD Conf. Management of Data, June 1999.
- [9] A. Schmidt, M. Kersten, M. Windhouwer, & F. Waas, "Efficient Relational Storage and Retrieval of XML Documents," Proc. WEBDB 2000, Dallas, May 2000.
- [10] D. Scheffner & J.-C. Freytag, "The XML Query Execution Engine (XEE)," Proc. BalticDB&IS 2002, Tallinn, Estonia, June 2002.
- [11] D. Florescu & D. Kossmann, "Storing and Querying XML Data Using an RDBMS," IEEE Data Engineering Bulletin, Vol. 22, No. 3, September 1999.
- [12] I. Tatarinov, S.D.Viglas, K.Beyer, J.Shanmugasundaram, E.Shekita, & C.Zhang, "Storing and Querying Ordered XML Using a Relational Database System," Proc. ACM SIGMOD Conf., June 2002.
- [13] J. Shanmugasundaram, K. Tufte, G. He, C. Zhang, D. DeWitt, & J. Naughton, "Relational Databases for Querying XML Documents: Limitations and Opportunities," Proc. 25th VLDB, Edinburg, Scotland, September 1999.
- [14] J. Shanmugasundaram, "XPERANTO: Bridging Relational Technology and XML", IBM Research Report, June 2001.
- [15] M. Yoshikawa, T. Amagasa, T. Shimura, & S. Uemura: "XRel: a path-based approach to storage and retrieval of XML documents using relational databases," Proc. ACM Transactions on Internet Technology, Vol. 5, Augus, 2001.
- [16] I. Manolescu, D. Florescu, & D. Kossmann, "Pushing XML Queries inside Relational Databases," INRIA Technical Report, INRIA, No. 4112, January 2001.
- [17] J. Shanmugasundaram, J. Kiernan, E. Shekita, C. Fan, & J. Funderburk, "Querying XML Views of Relational Data," Proc. 27th VLDB, Roma, Italy, September 2001.
- [18] M. Carey, J. Kiernan, J. Shanmugasundaram, E. Shekita, & S. Subramanian, "XPERANTO: Middleware for Publishing Object-Relational Data as XML Documents," Proc. VLDB 2000, September 2000.
- [19] S. Banerjee, V. Krishnamurthy, M. Krishnaprasad, R. Murthy: Oracle8i-The XML Enabled Data Management System. Proc. ICDE 2000, San Diego, California, USA, March. 2000.
- [20] 신효섭, "디지털 TV 방송 환경에서 내장형 시스템을 위한 XML 데이터의 저장 및 검색 방법", 데이터베이스연구회지, Vol. 19, No. 03, September 2003.
- [21] W3C, XML Path Language (XPath) 2.0, Working Draft, November 2002. (<http://www.w3.org/TR/xpath20/>)
- [22] W3C, XQuery 1.0: An XML Query Language, Working Draft, 29 October 2004. (<http://www.w3.org/TR/2004/WDxquery-20041029/>)
- [23] K. Kang, J. G. Kim, H. K. Lee, H. S. Chang, S. J. Yang, Y. T. Kim, H. K. Lee & J. W. Kim, "Metadata Broadcasting for Personalized Service: a Practical Solution", ETRI Journal, Vol. 26, No.5, October 2004.
- [24] W3C, Document Object Model (DOM) Level 1, Recommendation, October 1998. (<http://www.w3.org/TR/REC-DOM-Level-1/>)
- [25] Q. Li & B. Moon, "Indexing and Querying XML data for Regular Path Expressions," Proc. VLDB 2001, Roma, Italy, September 2001.
- [26] X. Wu, M. L. Lee & W. Hsu, "A Prime Number Labeling Scheme for Dynamic Ordered XML Trees," Proc. ICDE 2004, Boston, USA, March

2004.

- [27] P. O'Neil, E. O'Neil, S. Pal, I. Cseri, G. Schaller & N. Westbury, "ORDPATHs: Insert-Friendly XML Node Labels," Proc. SIGMOD 2004, Paris, France, June 2004.
- [28] E. Bertino, B. Catania & W. Q. Wang, "XJoin Index: Indexing XML Data for Efficient Handling of Branching Path Expressions," Proc. ICDE Conf, Boston, MA, USA, March 2004.
- [29] N.Katayama & S.Satoh, "The SR-tree: An Index Structure for High-Dimensional Nearest Neighbor Queries," Proc. ACM SIGMOD Conf, Tucson, Arizona, May 1997.

부 록

본 논문에서 실험을 위하여 사용한 운영체제는 Windows XP를 사용하였으며, 데이터베이스는 Oracle(R) Enterprise Manager Version 9.2.0.1.0 Production를 사용하였다. 실험에 사용된 시스템 사양은 Intel(R) Pentium(R) 4 CPU 2.8GHz와 1.0GB RAM을 사용하였다. 그리고, 프로그래밍 언어로는 J2SDK 1.4와 JDBC 드라이버를 이용하여 데이터베이스에 접근하도록 하였다.

성능평가를 위한 비교대상이 되는 시스템은 XML 전용 데이터베이스로 eXcelon DXE Manager Version 3.1 SP2를, XML-Enabled 데이터베이스로 Oracle9i를 사용하여 본 논문에서 제안한 시스템과 성능평가를 수

행했다. Oracle 데이터베이스에 XML 데이터를 저장하는 방법에는 XML 스키마를 등록하여 저장하는 방법과 스키마 없이 XMLType만을 이용하여 저장하는 방법이 있으나, 전자의 성능은 eXcelon과 거의 유사한 결과를 보였으므로 본 논문에서는 후자를 이용하여 시험하였다. 성능 평가를 위한 표본 메타데이터들은 실제 TV-Anytime 메타데이터 저작도구를 이용해 생성한 데이터를 사용하고, 검색을 위한 질의어는 각 시스템마다 입력으로 하는 질의어가 다르므로 세가지 종류의 질의어를 사용하였다. 우리의 메타데이터 관리 시스템은 XQuery를 질의어로 사용하였으나, eXcelon과 Oracle은 아직 XQuery를 지원하지 못하고 있는 단계이므로 eXcelon의 경우에는 XPath를 사용하고, Oracle의 경우에는 순수한 XPath를 직접 사용할 수 없으므로 Oracle에서 지원하는 질의형식으로 변환하여 사용하였다. 각 입력으로 사용하는 질의어는 다르지만 XPath는 XQuery의 Subset 이므로 XPath에 기술된 조건은 XQuery를 이용하여 기술할 수 있으므로 의미적인 차이는 없다. 다만, XQuery를 질의어로 사용할 경우는 반환구조를 처리하는 시간이 소요되므로 최소한의 반환구조만을 정의하여 사용하였다. 성능 평가를 위한 질의어의 종류는 표 1과 같다.

예제질의의 특성은 Q1, Q2, Q3의 경우 XPath 표현식에 전체경로를 기술하여 각각 하나의 말단노드, 하나의 중간노드, 여러 개의 문서전체를 반환하도록 하는 질의로, 반환되는 문서의 특성에 따른 성능을 측정한다.

표 1 성능평가를 위한 예제 질의

Query	질의 특징/XPath 표현
Q1	전체경로로 표현된 노드의 값 하나를 비교 조건으로 말단 노드 하나를 반환 /TVAMain/ProgramDescription/ProgramInformationTable/ProgramInformation[@programId='crid://www.arirang.co.kr/BBCWorldNews103040600117']/BasicDescription>Title
Q2	전체경로로 표현된 노드의 값 하나를 비교 조건으로 문서의 중간 노드 하나를 반환 /TVAMain/ProgramDescription/ProgramInformationTable/ProgramInformation[@programId='crid://www.arirang.co.kr/BBCWorldNews103040600117']
Q3	전체경로로 표현된 노드의 값 하나를 비교 조건으로 전체 문서 여러개를 반환 /TVAMain[ProgramDescription/ProgramInformationTable/ProgramInformation/BasicDescription>Title[contains(text(), 'BBC')]]
Q4	단축경로(6개)로 표현된 노드의 값 3개를 비교조건으로 말단 노드를 하나를 반환 /TVAMain/ProgramInformation//BasicDescription/Name[contains(text(), '뉴스')] and //Synopsis[contains(text(), 'bbc')] and //Language[contains(text(), 'ko')]/Title
Q5	단축경로(6개)로 표현된 노드의 값 3개를 비교조건으로 문서의 중간 노드 여러 개를 반환 //ProgramDescription//ProgramInformation//BasicDescription/Name[contains(text(), '뉴스')] and //Synopsis[contains(text(), 'bbc')] and //Language[contains(text(), 'ko')]]
Q6	단축경로(7개)로 표현된 노드의 값 5개를 비교조건으로 전체 문서 여러 개를 반환 /TVAMain//ProgramInformation//DayAndYear[contains(text(), '2003')] and //ProgramInformationTable//BasicDescription/Synopsis[contains(text(), 'bbc')] and //Language[contains(text(), 'ko')] and //Name[contains(text(), '뉴스')] and //ProductionLocation[contains(text(), 'ko')]]]
Q7	전체경로로 표현된 노드의 값 하나를 비교 조건으로 대량의 문서 반환 /TVAMain[ProgramDescription/ProgramInformationTable/ProgramInformation/BasicDescription/Genre/Name[contains(text(), '드라마')]]

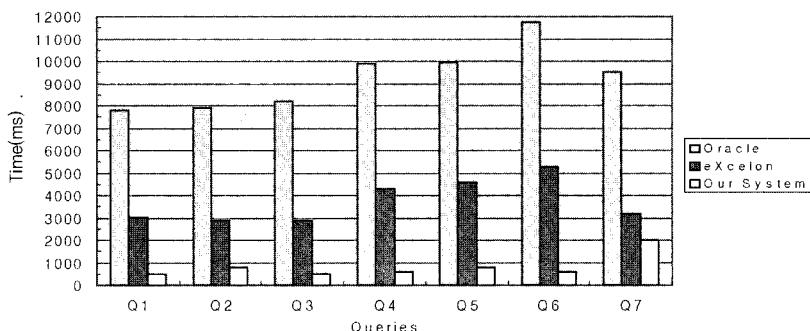


그림 1 방송용 메타데이터 검색 시간

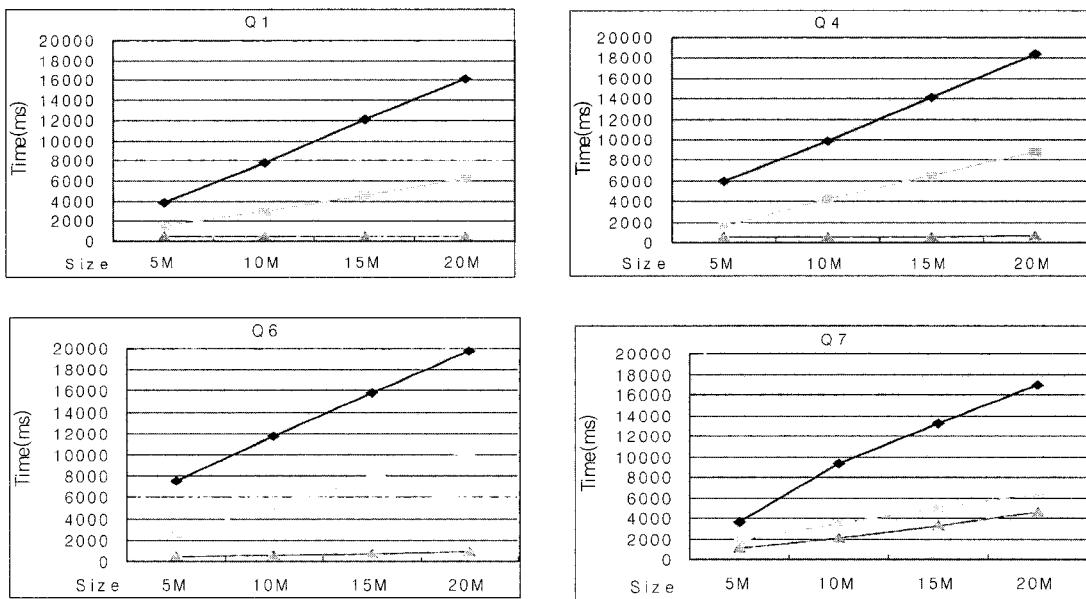


그림 2 데이터 크기에 따른 성능 측정(Q1, Q4, Q6, Q7)

Q4, Q5, Q6의 경우 XPath 표현식에 각각 서로 다른 단축경로와 여러 조건을 기술하여 하나의 말단노드, 여러 개의 중간노드, 여러 개의 문서전체를 반환하도록 하는 질의로 조건의 형태에 따른 성능을 측정한다. Q7은 대량의 문서를 반환할 경우 성능을 측정하기 위한 질의어이다. 각 시스템의 시간 측정은 질의어를 검색엔진에 보낸 후 검색엔진에서 질의를 분석하는 시간을 시작 시간으로 하고, 검색된 결과를 재조합 하여 사용자에게 반환하기 위한 최종 문서를 생성하는 시간 까지를 종료 시간으로 하여 수행시간을 측정하였다.

그림 1은 약 10MB의 견본 메타데이터에 표 1의 예제 질의를 수행하여 얻은 시간을 측정한 그래프이다. 성능측정 결과, Q7을 제외한 모든 경우에 본 논문에서 제안한 시스템이 효율적인 처리속도를 보이고 있다. 이는

질의에 표현된 구조정보를 처리하기 위해 XPath 테이블을 사용하므로 검색시간이 단축될 뿐만 아니라, 사용자에게 문서를 반환할 때 문서를 제 조합하지 않고 CLOB으로 저장되어 있는 전체문서에서 문서의 일부분을 직접 얻어올 수 있기 때문이다. eXcelon과 Oracle의 경우는 Q1, Q2, Q3보다 Q4, Q5, Q6과 같이 복잡한 질의를 처리할 경우 성능이 저하되는 것을 볼 수 있다. 그러나 본 논문에서 제안하고 있는 시스템은 질의의 복잡도와는 무관한 처리속도를 보인다. Q7과 같이 반환해야 하는 문서(위 예제의 경우 267개의 문서로 크기는 약 1.2M)가 많은 경우에, 우리의 시스템은 데이터베이스에 CLOB으로 저장된 문서를 얻어와서 반복구조를 적용하는 시간이 더 소모되므로 다른 질의와 비교하여 수행시간이 더 걸린다.

그림 2는 각각 Q1, Q4, Q6, Q7의 질의를 견본 테이타의 크기에 따라 성능을 측정한 그래프로 5M(문서 약 400개), 10M(약 800개), 15M(약 1500개), 20M(약 2000개)의 견본데이터를 이용하여 성능을 측정한 결과이다. 성능 측정의 결과 eXcelon과 Oracle 모두 데이터의 크기가 증가하면 처리속도 역시 유사한 비율로 증가하나 우리의 시스템은 Q7의 경우를 제외하면 증가비율이 크지 않았다. 그 이유는 메타데이터의 검색을 위해 Path 테이블을 이용하여 직접 말단 노드에 해당하는 테이블에 접근한 후 해당 내용만 검색하므로 구조와 내용을 비교하기 위한 시간이 커지지 않기 때문으로 사료된다. 그러나, Q7의 경우 질의의 조건이 단순하고 반환해야 할 문서의 양이 많으므로 성능을 좌우하는 요인은 문서의 재조합 시간으로 결정된다. 성능측정의 결과, 20M의 견본데이터에 Q7질의를 수행했을 경우 반환되는 문서의 크기는 약 3.2M(문서 712개)로 데이터베이스로부터 문서를 읽어오는 시간이 질의 수행시간의 대부분을 차지하였다.

위 실험의 결과 대부분의 경우에 본 논문에서 제안하고 있는 시스템이 월등한 수행시간을 보였다. 그러나, eXcelon과 Oracle 모두 특별한 색인구조를 사용하지 않았으므로 이를 감안하여야 할 것이다.



박종현

2002년 충남대학교 컴퓨터과학과 석사
2002년~현재 충남대학교 컴퓨터과학과 박사과정. 관심분야는 XML, XQuery 웹 정보 시스템, XML 데이터베이스



김병규

2003년 충남대학교 컴퓨터과학과 석사
2003년~2005년 현재 KISTI 지식정보센터 연구원. 관심분야는 메타데이터 처리 웹 정보 시스템, 정보자원개발



이용희

2004년 충남대학교 컴퓨터과학과, 석사
2004년~현재 한국전자통신연구원 디지털 홈 연구단 연구원. 관심분야는 XML 웹 정보 시스템, 디지털 홈 시스템



이민우

2004년 충남대학교 컴퓨터과학과, 석사
2004년~현재 한전 KDN 발전사업팀 연구원. 관심분야는 XML, 정보검색, 자연어 처리, 웹 정보 시스템



정민욱

2005년 충남대학교 컴퓨터과학과, 석사
2005년~현재 (주)한글과컴퓨터 XML 기술팀 연구원. 관심분야는 XML DB 메타데이터 처리, 웹 정보 시스템



강지훈

1981년 한국과학기술원 전산학과 석사
1996년 한국과학기술원 전산학과 박사
1996년~1998년 미국 버지니아대학교 컴퓨터과학과 방문교수. 2001~현재 인터넷식별자포럼 운영위원. 1985년~현재 충남대학교 정보통신공학부 교수. 관심분야는 XML, 디지털 도서관, Semantic Web, 데이터베이스 시스템, 웹 정보 시스템