TV-Anytime 메타데이터의 저장방법 비교 분석

이용희[O] 박종현 이민우 정민옥 강지훈
충남대학교 컴퓨터과학과
{lyhcool[O], jhpark, cslmw, ultra999, jhkang}@cs.cnu.ac.kr

Comparison and Analysis on the Storage Scheme of TV-Anytime Metadata

Yong-Hee Lee[O] , Jong-Hyun Park, Min-Woo Lee, Min-Ok Jung, Ji-Hoon Kang
Dept. of Computer Science, Chungnam National University

요 약

디지털 방송 환경의 발전은 방송 서비스 제공자가 방송 프로그램을 사용자에게 단방향으로 서비스를 제공하는 것뿐만 아니라 방송 서비스 제공자와 사용자 사이에 양방향의 액션을 통한 서비스 제공으로 형태가 변화하고 있다. 이러한 방송 서비스를 위한 규격을 만드는 국제표준화단체인 TV-Anytime 포럼에서는 차세대 디지털 방송용 메타데이터 국제 표준을 만드는 작업을 진행하고 있다.

TV-Anytime 메타데이터를 실제의 응용에 적용하기 위해서는 대용량의 메타데이터를 저장하고 관리하는 것이 필수적인 요구 사항이다. 이에 TV-Anytime 메타데이터를 실제 디지털 방송 환경에서 효율적으로 관리할 수 있도록 메타데이터 저장 엔진을 설계 및 구현한다. 또한 다른 시스템과의 시험을 통해 TV-Anytime 메타데이터 이외에도 XML 문서를 저장하고 관리하기 위한 다른 응용에서도 저장방법의 기준으로 삼을 수 있는 자료가 될 것으로 기대된다.

1. 서론

디지털 방송 환경의 발전으로 채널 수와 프로그램 수의 급격한 증가가 두드러지게 나타나고 있다. 이러한 방송 환경의 변화는 방송 서비스 제공자가 방송 프로그램을 사용자에게 단방향으로 서비스를 제공하는 것뿐만 아니라 방송 서비스 제공자와 사용자 사이에 양방향의 액션을 통한 서비스 제공으로 형태가 변화하고 있다. 이러한 방송 서비스를 위한 규격을 만드는 국제표준화단체인 TV-Anytime 포럼[1]에서는 차세대 디지털 방송용 메타데이터 국제 표준을 만드는 작업을 진행하고 있다.

TV-Anytime 포럼[1]에서 제안하고 있는 메타데이터 즉, TV-Anytime 메타데이터는 XML 스키마[5]를 기반으로 그 구조를 기술하고 있으며 메타데이터의 많은 부분은 멀티미디어 컨텐츠를 위한 메타데이터 표준인 MPEG-7[2]의 멀티미디어 기술 구조를 이용한다.

XML[3]로 표현되는 TV-Anytime[1] 및 MPEG-7[2] 메타데이터의 저장과 관리는 결국 XML 문서의 저장과 관리의 문제가 된다. 현재 XML 문서를 저장하고 관리하기 위한 연구는 많은 업체와 연구기관에서 활발히 진행 중이다. 특히, 데이터베이스 개발 업체에서는 XML 전용 데이터베이스와 기존의 데이터베이스에 XML을 지원할 수 있도록 하는 XML-Enabled 데이터베이스를 개발 중에 있다. 이들 각각은 서로 다른 방법으로 XML 문서를 저장하고 관리한다[9]. 본 논문에서는 실제 TV-Anytime 메타데이터를 이용하여 기존 시스템과의 성능을 분석한 후, TV-Anytime 메타데이터의 관리를 위하여 어떤 환경이 가장 적절한지 평가한다. 이렇게 나온 결과는 향후 보다 효율적이고 안정적인 디지털 방송을 구축하는데 일익을 담당할 것이다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구에 대해 살펴보고, 3장에서는 본 논문에서 설계하고 구현한 TV-Anytime 및 MPEG-7 메타데이터 저장 엔진에 대해 설명하고, 4장에서는 본 논문에서 제안한 시스템과 다른 XML 저장 관리 시스템과의 성능평가를 수행하며, 마지막으로 5장에서는 본 논문의 결론 및 향후 과제에 대해 기술한다.

2. 관련 연구

2.1 XRel

XRel[8]은 관계형 데이터베이스에 대한 확장을 요구하지 않으며 기본적인 테이블 구성은 엘리먼트 타입에 따라 테이블을 만들고 여기에 패스 정보를 가지게 한 테이블을 추가한 방법이다. 이것은 DTD나 엘리먼트 타입에 대한 정보 없이 데이터베이스에 고정적으로 만드는 테이블이다. 테이블의 구성은 Element, Attribute, Text 그리고 Path 테이블로 구성된다.

```
Element(docID, pathID, start, end, index, reindex)
Attribute(docID, pathID, start, end, value)
Text(docID, pathID, start, end, value)
Path(pathID, pathexp)
```

테이블의 칼럼 이름으로 사용된 docID, pathID, start, end, value는 저장한 문서의 고유 식별자, 각 노드들의 패스 표현에

대한 고유 식별자, 각 노드들이 가진 위치 정보의 시작 위치, 각 노드들이 가진 위치 정보의 끝 위치 그리고 각 노드들의 값을 각각 나타낸다. 여기서 노드들이 가진 위치 정보는 XRel에서 XML [3]문서를 트리 형태로 나타냈을 때 각 노드에 고유한 번호를 부여하여 그 번호를 가지고 위치 정보를 나타낸다. Element 테이블에서 index와 reindex는 문서에서 형제 엘리먼트 노드 사이의 순서 그리고 그 역 순서를 각각 나타낸다. 마지막으로 Path 테이블의 pathexp는 각 노드들의 패스 표현을 나타낸다.

XRel[8]의 특성은 노드들의 위치 정보를 테이블 구조에 사용한다는 것과 패스 정보를 별도의 테이블로 저장한다는 점이다. 특히, 패스 정보를 저장하는 방법은 기존 방법들이 패스 길이에 비례하여 조인을 수행하던 것에서 조인의 횟수를 줄일 수 있는 장점을 가지고 있다.

XRel[8]과 본 논문의 차이점은 Element, Attribute 그리고 Text 테이블의 정보들이 하나의 Element 테이블에 들어간다는 것이다. 또한 XRel에서는 각 노드의 위치 정보를 고유 식별자로 사용하고 있으나 본 논문에서는 Dewey 방법을 사용하여 각 노드의 고유 식별자를 부여하여 사용한다. 이는 각 노드들간의 위치정보를 관리하기 위하여 보다 효율적으로 사용할 수 있다.

2.2 XParent

XParent[6]는 LabelPath, DataPath, Element and Data 테이블로 구성된다.

LabelPath(ID, Len, Path)
DataPath(Pid, Cid)
Element(PathID, Did, Ordinal)
Data(PathID, Did, Ordinal, Value)

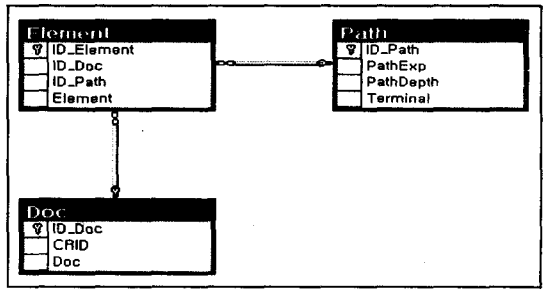
LabelPath 테이블은 라벨 패스 정보를 저장한다. 각 라벨 패스는 고유한 식별자로 구분하며 Path 칼럼에 저장한다. 라벨 패스의 간선의 수는 Len 칼럼에 저장한다. DataPath 테이블의 Pid와 Cid는 데이터 패스에서의 간선의 부모 노드 식별자와 자식 노드 식별자를 각각 나타낸다. Element와 Data 테이블에서 PathID는 LabelPath의 ID를 외래키로 가지며 Did는 각 노드의 식별자이다.

XParent[6]는 XRel[8]과 비슷한 구조를 사용하나 차이점은 Did 칼럼이 XRel에서는 start 칼럼과 end 칼럼의 쌍으로 나타나게 된다. 또한 XParent는 B-tree와 같은 인덱스 메커니즘을 사용하여 그 효율성을 높였다.

XParent[6]와 본 논문에서는 Path 테이블을 별도로 두고 패스 비교 과정의 처리 속도를 높이기 위해 패스 길이 정보를 함께 저장하는 방법을 사용하는 점에서는 유사하다. 그러나, Element, DataPath, Data 그리고 Ancestor 테이블의 정보들이 하나의 Element 테이블에 들어간다는 점에서 차이가 있다. 이는 문서의 구조정보를 관리할 경우 유용하게 사용될 수 있다.

3. 저장 엔진의 설계

본 논문에서는 TV-Anytime 메타데이터를 위한 스키마를 분석하여 Edge 단위로 동일한 엘리먼트의 이름을 가지는 모든 노드는 동일한 Table에 저장하는 Binary방법을 사용하였고 또한, 각각의 엘리먼트 마다 고유한 노드의 ID를 부여하기 위하여 Dewey 방법[7]를 사용하여 단일 문서에서 모든 엘리먼트 노드는 서로 다른 ID 즉, 고유한 키를 갖도록 하였다. Dewey ID는 크게 두 가지로 사용된다. 첫째, 데이터베이스 스키마에서 부모 자식 테이블간 연결을 맺는 포인터의 역할을 하여 분해된 문서를 쉽게 재조합 할 수 있다. 둘째, Dewey ID는 루트 노드부터 각 노드까지의 경로이므로 간단한 질의표현으로 정확한 질의결과를 얻을 수 있다.



[그림 3-1] Element, Path, Document Table의 관계

[그림 3-1]는 각 Element, Path, Document Table의 관계를 나타내고 있다. TV-Anytime 메타데이터의 기술에 사용되는 모든 엘리먼트 노드들은 동일한 이름을 가지는 노드끼리 동일한 Table에 저장되고 모든 Element Table은 [그림 3-1]와 같이 Path Table과 Document Table과 연결되는 구조로 데이터베이스 스키마는 구성되어 있다. Path Table의 PathExp필드는 각 엘리먼트 노드의 Path를 저장하여 사용자가 질의한 XQuery의 XPath[4] 표현을 한번의 비교만으로 직접 해당 Table을 선택 할 수 있도록 함으로써 검색 시간을 효과적으로 줄일 수 있다.

4. 실험 및 분석

4.1 실험 환경

XML 저장 관리 시스템 중에서 성능 평가를 위해, XML-Enabled DB 에서는 Oracle 9i를 선택하고 Native XML DB 에서는 eXcelon을 선택하였다.

사용한 질의어로는 TV-Anytime 메타데이터의 XML 스키마[5]를 분석하여 실제 사용자가 많이 사용하는 질의어의 종류 중 대표가 되는 8개를 예제 질의어로 정의하여 선택하였다. 각각의 예제 질의어는 크게 '//'과 Predicate를 사용하여 Xpath[4] 형태로 만든다. 이러한 형태의 질의어는 방송 환경에서 사용될 질의어의 대표적인 것들로 질의어의 종류는 [표 4-1]과 같다.

[표 4-1] 예제 질의어

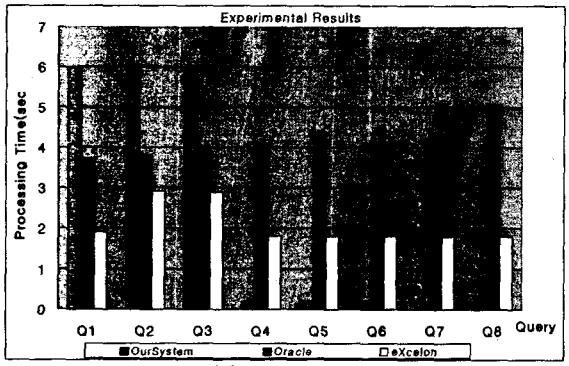
Q1 : 말단 노드
Q2 : '//' 1개 사용
Q3 : '//' 2개 사용
Q4 : Predicate 2개 사용
Q5 : Predicate 5개 사용
Q6 : '//' 1개 사용, Predicate 2개 사용
Q7 : '//' 2개 사용, Predicate 5개 사용
Q8 : Predicate 10개 사용

4.2 실험 결과 및 분석

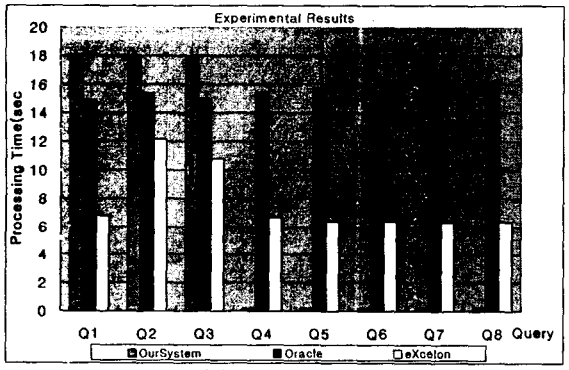
각 시스템의 시간 측정은 질의어를 데이터베이스에 보내 질의를 요청하는 시간을 시작 시간으로 하고 질의의 결과를 재조합하여 사용자가 원하는 결과를 생성하는 시간을 종료 시간으로 하여 수행시간을 측정하였다.

본 논문에서 제안하고 있는 시스템은 데이터베이스에서 값을 가져와서 문서를 재조합 하여 String 데이터 타입으로 사용자에게 결과를 가져오는데 걸리는 시간을 측정하였고

Oracle은 Oracle에서 제공하는 함수를 사용하여 CLOB 데이터 타입으로 결과를 가져오고 가져온 CLOB 타입의 데이터를 다시 String 타입으로 변환하는 데까지의 시간을 측정하였으며 eXcelon은 eXcelon에서 제공하는 함수를 사용하여 String 데이터 타입으로 결과를 가져오는데 시간을 측정한 것이다.



(a) Data Size 5M



(b) Data Size 20M

[그림 4-1] TV-Anytime 문서의 처리 시간

[그림 4-1]에서 질의어 Q1, Q2 그리고 Q3에 대해서는 eXcelon이 가장 좋은 성능을 보이고 있다. 그 이외의 나머지 질의어에 대해서는 본 논문에서 제안한 방법이 월등한 성능을 보이고 있다.

본 논문에서 제안한 시스템에서 질의어 Q1, Q2 그리고 Q3에서 처리 시간이 다른 시스템 비하여 늦게 나오는 이유는 문서를 재조합 하는데 걸리는 시간이 가장 큰 이유가 된다. 나머지 질의어에 대해 본 논문에서 제안한 방법이 월등한 성능을 보이는 이유는 검색 조건을 많이 주어도 기존의 관계형 데이터베이스에서 그것을 처리하는데 걸리는 수행시간이 월등히 빠르다는 것을 의미한다. 검색 조건을 처리하기 위해 Path 테이블을 이용하여 XPath[4]에 기술된 말단 노드의 이름을 가지는 테이블로 직접 접근하여 조건을 처리할 수 있기 때문이다. 이는 Path 테이블을 이용하여 불필요한 조인의 횟수를 줄이기 때문에 가능하다. 반면 문서의 일부분을 반환할 경우, 반환할 메타데이터의 최상위 노드에 해당하는 정보를 테이블에서 가져오는 시간은 얼마 걸리지 않으나, 각기 서로 다른 테이블에 나누어 저장된 자식 노드의 정보를 조인을 통하여 얻어오기 때문이다.

위의 결과를 종합해보면 본 논문에서 제안한 시스템은 검색 조건이 주어지는 경우 월등한 성능을 보이며 검색 조건의 수와 무관한 질의 처리 시간을 보인다. 이것은 디지털 방송 환경에서 방송용 메타데이터 검색을 위해서는 조건이 필요하므로 메타데이터 검색 시 효율적이라 할 수 있다. 또한 데이터 크기에 큰 영향을 받지 않고 검색 조건에 대한 질의를 처리하므로 대용량의 방송용 메타데이터 구조에 적절하다는 장점을 가진다. 그러나 문서 재조합을 요구하는 질의 처리 시간은 다른 시스템에 비해 많은 시간이 필요하다. 이것은 문서 재조합의 속도를 향상시키는 구조를 추가하여 효율적으로 사용할 수 있다.

5. 결론 및 향후 과제

본 논문에서는 TV-Anytime 메타데이터를 실제 디지털 방송 환경에서 효율적으로 관리할 수 있도록 메타데이터 저장 엔진을 설계하고 구현하였다. 또한, 본 논문의 메타데이터 관리시스템과 XML을 지원하는 상용 데이터베이스와의 성능 평가를 통하여 어떤 환경에서 방송용 메타데이터가 효율적으로 저장되고 관리되는지 시험하고 평가하였다. 이는 방송용 메타데이터를 관리하기 위해 보다 효율적인 환경 제공하기 위한 기반이 될 것으로 사료된다. 또한, 이를 통해 TV-Anytime 메타데이터 이외에도 XML 문서를 저장하고 관리하기 위한 다른 응용에서도 기준으로 삼을 수 있는 자료가 될 것으로 기대된다.

향후 과제로는 메타데이터 관리 시스템의 효율성을 높이기 위해 본 논문에서 제안한 메타데이터 저장 방법에서 문서 재조합의 성능 개선을 위한 연구가 이루어져야 한다.

5. 참고 문헌

[1] TV-Anytime Specification Series, August, 2003.
[2] Overview of the MPEG-7 Standard, Dec. 2001.
[3] W3C, Extensible Markup Language (XML) Version 1.0, Recommendation, Feb. 1998.
[4] W3C, XML Path Language (XPath) Version 1.0, Recommendation, Nov. 1999.
[5] W3C, XML Schema Part 0: Primer, Recommendation, May 2001.
[6] Haifeng Jiang, Hongjun Lu, Wei Wang, Jeffrey Xu Yu. "Path Materialization Revisited: An Efficient Storage Model for XML Data", Proc. 13th ADC 2002, Melbourne, Jan. 2002.
[7] 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.
[8] 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, Volume 5, pp.110-141, Aug. 2001.
[9] 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.
[10] 김병규, 박종현, 강지훈 " 데이터베이스를 이용한 TV-Anytime 메타데이터의 저장" , 한국정보과학회, pp.593-595, 제주, Apr. 2003.