IDEA GROUP PUBLISHING

701 E. Chocolate Avenue, Suite 200, Hershey PA 17033, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group

ITP4852

Fast Search for Multimedia Metadata in an XML Data Repository

Shibin Qiu
Dept of Electrical and Computer Engineering-Univ. of New Mexico
Albuquerque, 87131, NM
sqiu@unm.edu

Shu Li
National Center for Genome Resources
2935 Rodeo Park Drive East, Santa Fe, NM, 87505
sl@ncgr.org

ABSTRACT

Performance of multimedia metadata management depends on the storage, indexing, and proper schema design of the system. In this paper, we propose a design strategy for improvement of search performance to build a prototype multimedia metadata system (MMS). The proposed design strategy includes the following aspects. First, we model and design the MMS using native XML database. The native XML database stores XML documents directly without going through normalization processes required in an RDBMS. We do not store XML document on top of an RDBMS, either. Second, we introduce an assistant document, the rotated document for each searchable descriptor. Optimal inter-document schema is designed to search and locate documents in the data repository. Third, on the rotated documents, we perform lexical processing including word separating, stemming and stop word removal. Finally, different index structures are implemented and tested to build fast indices for the system. Experiments show that the proposed design achieves a speedup factor of 9.3 compared with the direct DOM method. It can speed up even more than a system based on RDBMS. It also has a better scalability.

I INTRODUCTION

It is necessary to efficiently manage multimedia information as multimedia documents proliferate rapidly. There are basically two ways to managing multimedia information. One is content based, the other is text based. Multimedia metadata includes information that is generated in the production processes of multimedia works and descriptions extracted by content analysis. Therefore multimedia metadata involves both kinds of data.

A. Modeling Metadata with XML

Multimedia metadata is complex and semi-structured. It can be modeled with object-oriented model, relational model, and XML documents. However, it is increasingly popular to use XML and its associated RDF to model multimedia metadata. XML also provides high-level interoperability between different systems and business partners.

MPEG-7 uses XML as the language of choice for content description and structure definition [15]. The Digital Item Declaration (DID) specification and its associated Digital Item Declaration Language (DIDL) in MPEG-21 also use XML to define a standard-form metadata for multimedia works [15]. The Digital Imaging Group (DIG) has released the DIG-35 metadata specification using XML as description language [8]. The Synchronized Multimedia Integration Language, SMIL, has used XML to model multimedia content [24]. The metadata model of the MusicBrainz Metadata Initiative uses RDF and Dublin Core metadata recommendation to facilitate the standardization of audio/video related information [14]. The Open eBook Publication Structure (OEBPS) is an XML-based specification for the content, structure and presentation of electronic books [17]. IMS Learning Resource Metadata Specification also uses XML to model metadata of learning materials including multimedia materials [9].

B. Related Works

Different methods have been used to store the metadata for further processing such as insertion, update, deletion, indexing and search. Some works use relational database management systems (RDBMS) directly [5, 12, 23]. Some works use XML databases on top of RDBMS [2]. There are a few problems with managing multimedia information using RDBMS. One problem is the way relational database models data relationships. In RDBMS related data are stored in different tables after going through normalization processes. Hierarchical multimedia metadata requires many tables to handle. Movie genres, for example, are hierarchical and require a lot of related tables to model in a relational database [1, 5, 11]. When data is queried, a number of related tables are first queried and the resultant data sets are joined together for output. The process of join is time consuming and slows down the search performance [20]. Therefore the best way to store XML documents is to store them in native XML databases, which is the design strategy we adopt in this study.

Most information retrieval systems perform lexical processing before indices are built. Lexical processing includes word separating, stemming, stop word removal, etc. Necessary lexical processing can improve search precision, speedup performance and save space [21]. However data is stored as is in RDBMS without going through efficient lexical processing. Previous works on XML database systems also lacked lexical processing.

A multimedia document warehousing system was studied in [10]. It had XML active query facility. A tag-element-based query method, instead of key word search on XML data was studied in [25]. A synchronized and retrievable video/HTML lecture system for employee training has been studied in [4]. It used native XML database and provided key word search. Its data source was from HTML web pages rather than multimedia metadata. Its XML document structure was relatively simple. A multimedia data recorder with key word retrieval ability was built for wearable computers in [16]. It used XML database system to store data. The recorder was able to record information about simple multimedia works such as digital pictures and speeches. The metadata was input by human speakers. It did not study issues of search performance, schema design and lexical processing, either. In [3], an XML indexing by path encoding was proposed. When XML tags are deeply nested, the encoding is long and the index uses large spaces. Inter- document schema was not studied in this work. The XML documents were mapped into relational tables and stored in RDBMS.

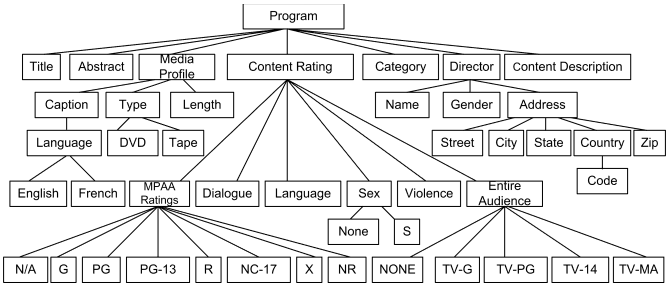
In summary, related works on XML database systems for multimedia metadata management mentioned above dealt with relatively simple XML documents. They stored

XML documents on top of RDBMS. They directly queried the documents in their data repositories. When the data collection is large, searching directly on them (using DOM or SAX) is slow. Previous works did not study optimal inter-document schema, optimal index structure, or lexical processing in XML database systems.

C. Overview of the Paper

To achieve high search performance and tackle the problems mentioned above, we improve search performance in the following ways. 1) We model multimedia metadata with XML and use native XML database to store XML documents. 2) To speed up search, we introduce assistant

Figure 1. A Simplified Model of TV Program Metadata



documents (the rotated documents) and define its relationship to the data repository through inter-document schema. When searching, query is performed against the rotated documents instead of querying the data repository directly. Indices are built both on the data repository and the assistant documents. 3) We also integrate lexical processing into the indexed rotated documents. 4) Different index structures are studied and tested. We chose Patricia trie as index structure for fast search based on test results. To show the effectiveness of the design, search performances are tested and compared among different ways of search. We use TV program information as our sample metadata for analysis and tests.

In Section II of this paper we model the metadata and design the schema. In Section III, we study index structures and lexical processing. We also analyze, test and compare performances of different ways of searching. Section IV concludes the paper.

II SCHEMA DESIGN

In this section, we explain our design strategy by using TV program information as data template. We will model the metadata by defining XML document schema of the data repository, introduce the rotated document and design inter-document schema.

A. Definition of XML Documents

We conceptually model the TV program metadata as a tree in node-centric form, as shown in Figure 1. Each node on the tree provides some description for the program, i.e., each node serves as a descriptor of the TV program. Under the content description node, children nodes can be modeled by MPEG-7 specifications. They are too large and are omitted. The simplification allows us concentrate on the design strategy, rather than being exhaustive. The content rating description follows the digital TV specification of ATSC. Figure 1 shows an instance defined for the region of USA [18].

If we normalize and store this simplified model is in an RDBMS, more than a dozen tables are required. If it is not simplified more than 100 tables are needed. Using XML document, the whole document as a tree can be stored in one record. And we do not need to handle a large number of related tables. The XML schema is shown in Figure 2. DTD can also be used to define XML document structure. But we prefer schema due to its flexibility.

B. Inter-document Schema

We can store all the documents defined by the schema in Figure 2 in one file. To model the whole TV program metadata we need more than 200 nodes. Each program instance occupies more than 2KB file space. A repository with millions of programs has gigabyte size. DOM can load all documents into memory and perform navigation on all the nodes. However when the collection of documents is large, physical memory is not enough and virtual memory must be used. We can also save each document in one file and let the file system or operating system directly mange the document files. In this case when the number of document is large, more disk access is also required. Either way, when the document collection is large more disk access is needed and performance is slowed down.

For fast response, we need a way to efficiently locate a document in a collection. We add a Program ID, progID, on each program document as shown in Figure 2. We build an index keyed off the progID node. This document index will find the document and locate the document with a pointer pointing to its physical location, usually a block number or a disk page.

To search for a program, we need to find out which documents contain the information we need by maintaining a set of Program IDs. But going through all the nodes across all the documents (like using XPath) is like table scan in a relational DB. We build additional files and indices for the most frequently used queries. We explain the design idea with the Title nodes. It is applicable to other descriptors that need be searched. On the Title node, we build the schema shown in Figure 3.

On the schema every Title points to a document. It is similar to the inverted file in information retrieval systems. We call it a rotated document. When searching, we first search in the rotated document and get program IDs and then go to the data repository to find the documents. In Figure 3, we use an IDREF type as a pointer to the data repository. Using IDREF here is to show the relationship. Automatic referencing to the document may not be the best solution, since DOM and SAX are slow working with ID and IDREF. Therefore they can be defined as integer types instead. The inter-document schema is illustrated in Figure 4.

Since the rotated document on Title has only two elements, searching on it for a title is much faster. For all searchable descriptors such as Abstract and child nodes under content description, we build the rotated documents in the same way.

III INDEX STRUCTURE AND LEXICAL PROCESSING

To design efficient indices, we study different data structures and search algorithms in this section. We perform lexical processing on the rotated documents. We also study system performance by analysis and experiments.

Figure 2. *A Schema of the Simplified Model*

```
<schema
 xmlns="http://www.w3.org/2000/10/XMLSchema" >
 <!-- Program type  -->
 <element name="Program"  type="ProgramType" />
  <complexType      name="ProgramType"/>
   <element name="progID"  type="ID" use="required"/>
   <element name="Title"  type="string" />
   <element name="Abstract"  type="string" />
   <element name="Profile" type="profileType" />
   <element name="Director"  type="DirectorType"/>
  <element name="contRating" type="ContRatingType"/>
  <element name="Category"  type="CategoryType"/>
   <element name="Description"    type="DescType"/>
   <!-- Add other nodes here -->
 </complexType>
 <complexType name="Text"  content="textOnly">
  <restriction base="string" />
 </complexType>
 <!--Director type  -->
 <complexType  name="DirectorType"/>
  <element   name="Name"  type="Text" />
  <element name="Gender"  type="string" >
    <simpleType >
     <restriction  base="string">
       <enumeration  value="male">
       <enumeration  value="female">
     </restriction>
    </simpleType>
  </element >
  <element   name="Address"  type="AddressType"/>
 </complexType>
 <!--Address type  -->
 <complexType    name="AddressType"/>
  <attribute   name="StreetNumber"    type="string" />
  <attribute   name="City"            type="string" />
  <attribute   name="State"           type="string" />
  <attribute   name="ZipCode"         type="string" />
  <attribute   name="Country"         type="string" />
 </complexType>
</schema>
```

Figure 3. *Schema of Rotated Documents*

Schema of Rotated Document on Title

```
<schema
   xmlns="http://www.w3.org/2000/10/XMLSchema" >
  <element   name="rotTitle"  type="rotTitleType" />
    <complexType  name="rotTitleType"/>
      <attribute        name="Title"  type="string" />
      <attribute        name="progIdref" type="IDREF"/>
    </complexType>
</schema>
```

Schema of Rotated Document on Abstract

```
<schema
   xmlns="http://www.w3.org/2000/10/XMLSchema" >
 <element name="rotAbstract"   type="rotAbsType" />
   <complexType name="rotAbsType"/>
      <attribute       name="Abstract"  type="string" />
      <attribute       name="progIdref" type="IDREF"/>
    </complexType>
</schema>
```

Schema of Rotated Document on Keywords

```
<schema
   xmlns="http://www.w3.org/2000/10/XMLSchema" >
 <element  name="rotKWord" type="rotKWordType"/>
     <complexType name="rotKWordType"/>
        <attribute       name="KWord"  type="string" />
        <attribute       name="progIdref" type="IDREF"/>
    </complexType>
</schema>
```

A. Lexical Processing on Text Nodes

Descriptors such as Title and Abstract in the rotated documents contain trivial words including stop words. We need further processing on them. We use these two files to explain our design strategy. The same processing can be done on all searchable text nodes.

We perform word separating, stemming and stop word removal on the texts in the rotated documents. Word separating extracts words from the text and discard white space and useless punctuations. Stemming get the stem forms of the words and remove structures such as suffix "ing", "es", "ed", etc. One kind of stop words serves for pure grammatical purposes rather than being objects of the document (e.g., of, the, and). The other kind of stop words appears so frequently that they do not provide enough descriptive meaning for searches. A stop word list and a fast stop word remover algorithm for TV programs have been proposed in [19].

Figure 4. *Inter-document Schema*

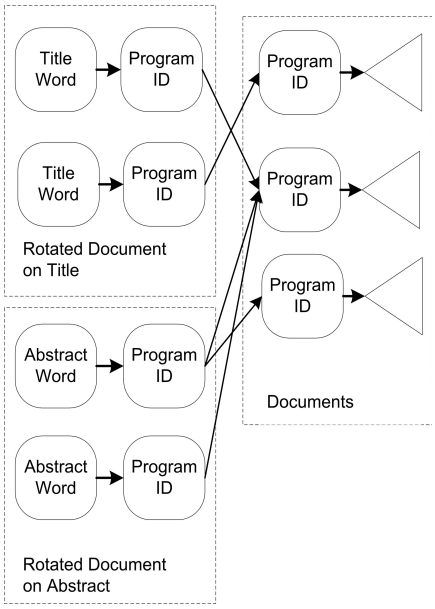
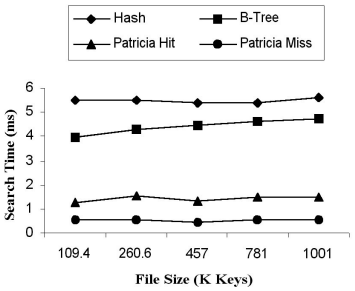


Figure 5. *Performance of Different Index Structures*



After going through lexical analysis, we split the nodes on the rotated documents so that each sub-node contains only one word. For example, suppose the title of a TV program is "Who want to be a billionaire?" After the processing, the key words "who", "want" and "billionaire" become nodes in the new document. The schema is similar to that defined in Figure 3. Each key word also has a program ID as a pointer. Rotated documents before the node splitting are temporary and are removed. Only the final indexed rotated documents persist. Boolean operations can be accomplished by going through rotated documents for different descriptors to support complex searches.

B. Index Structure

We implement hash table, B-Tree and Patricia trie as index structures to find the fastest indices for the rotated documents. For the hash index, each slot of the hash table has 10 keys on average. A hash function first calculates the ASCII value of the keys (words) then mod it by the number of slots (a prime number). For the B-Tree structure, each word is first converted into an integer value and then inserted into a B-Tree with a fan out of 32. The Patricia index is a 40-way trie. Each node contains 26 lower case letters, 10 digits and 4 other characters.

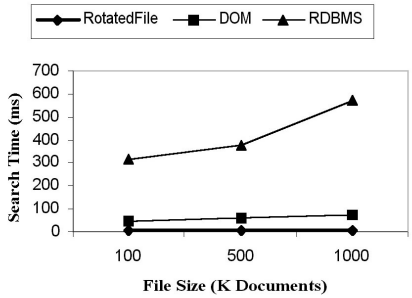
Search performances of the three data structures are shown in Figure 5, where the vertical axis indicates times consumed for searching 1000 keys in files of different sizes. On average, hash index uses 5.49 ms, B-Tree index uses 4.42 ms, Patricia trie index uses 1.43 ms on search hits and 0.53 ms on search misses. We chose Patricia trie as index structure for files that use text keys. We use B-Tree as index for integer keys. Search on a Patricia index returns faster if a search miss is encountered. This is quite contrary to searching on other index structures such as hash table, where a search miss is slower than a search hit. Patricia trie is compact than other tries, but it still uses more space than other indices. Since more memory space is available today, we can achieve high performance without worrying too much about space usage [13].

C. Analysis and Experiments

In our design, multimedia metadata is stored in native XML documents. Search is performed on rotated documents that have no stop words and use fast index structure. If we directly search on the data repository, it is slow with either the DOM or the SAX method due to the size of the collection. If XML data is stored on top of RDBMS, insertion of one data instance requires us to divide the data and insert the subsets into different tables. When searching, intermediate queries are joined together to get the resultant query. Thus searching XML documents on top of RDBMS is slow.

The proposed design can also be compared with searches in a relational model. Suppose a record in a RDBMS table stores movie titles as a field, and a client wants to find the program with a title of "Who want to be a billionaire". The whole title "Who want to be a billionaire" is the key and its value is computed for index. If the client inputs "billionaire" for the query, since no title has a value of "billionaire", the search is returned as a miss and the index on title field does not help to speed up the query. However resources must be used to maintain the index on the

Figure 6. *Performances of Different Searches*



field. For the same search in the proposed system, since every word is a key in the index, the movie is found and speedup is achieved through fast index and inter-document schema.

For experiments, we collected TV program information from TV guide magazines to make three data sets with sizes of 100K, 500K and 1000K documents. We modified the data sets so that each of them has 1000 documents containing the word "friend" in their titles. We use three different methods to query the data to return the 1000 documents. 1) Use DOM (libxml library on Linux) directly on the data repository. 2) Use the proposed method. 3) Model the data on an RDBMS (MySQL) and search with SQL. The system used is an Intel PIII PC running Red Hat Linux 7.0. The software is coded in C++. We built our XML database system using flat files. The results are shown in Figure 6. From Figure 6 we can see that to retrieve the desired documents, the RDBMS uses 364 ms on average; DOM uses 53 ms; and the proposed method uses 5.72 ms. The proposed method achieved a speedup factor of 9.3 compared with direct DOM method. It is much faster than the queries on RDBMS. Since search times do not increase very much as the data size increases, the proposed method is more scalable.

IV CONCLUSION

To support efficient search of multimedia metadata, we proposed a design strategy to build a prototype multimedia metadata system (MMS). We design the system by taking the following initiatives to enhance search performance. 1) We store XML documents in a native XML database directly. We do not map XML documents into relational tables in RDBMS. 2) We introduce the rotated document for each searchable descriptor and design optimal inter-document schema for fast search. 3) To improve search efficiency and precision, we integrate lexical processing into the rotated documents. 4) We use Patricia trie as index structure, since it is the fastest based on tests and comparison. Experiments show that the proposed method achieved a speedup factor of 9.3 compared with the direct DOM method. It is much faster than a system based on RDBMS. It is also more scalable.

The prototype MMS system can be used to manage multimedia metadata and provide search functions for VOD, digital library, DTV set top box, online program guide and other interactive services. The design strategy is applicable to managing other kinds of complex data.

REFERENCE

[1] Barton, J.M., Television Meta-data Standardization Overview and Opportunities, *IEEE International Conference on Consumer Electronics*, Los Angeles, USA, 2000, pp: 28-29.

[2] Berkley, C; Jones, M.; Bojilova, J.; Higgins, D., Metacat: a Schema-Independent XML Database System, *Proceedings of the 13th International Conference on Scientific and Statistical Database Management*, Fairfax, Virginia, 2001, pp: 171-179.

[3] Copper, B.; Sample, N.; Franklin, M. J.; G. R. Hjaltason, and M. Shadmon, A Fast Index for Semistructured Data, *Proc. of 27th International Conference on Very Large Data Bases*, 2001, Roma, Italy, pp: 341-350.

[4] Herng-Yow Chen; Jen-Shbin Hong; Yu-Te Wu, A Synchronized and Retrievable Video/HTML Lecture System for Industry Employee Training, *Proceedings of the 25th Annual Conference of the IEEE*, San Jose, CA, 1999, pp: 750-755.

[5] Michael Ehrmantraut, Theo Härder, Hartmut Wittig, Ralf Steinmetz, The Personal Electronic Program Guide—Towards the Preselection of Individual TV Programs , *Proceedings of the fifth International Conference on Information and Knowledge Management*, Rockville, Maryland, 1996.

[6] Folk, M., *File Structures* (Addison-Wesley, 1998).

[7] W. B. Frakes, R. Baeza-Yates, *Information Retrieval: Data structures and Algorithms* (Prentice Hall, 1992).

[8] International Imaging Industry Association (I3A), http://www.i3a.org/

[9] IMS, http://www.imsproject.org/metadata/imsmdv1p2

[10] Ishikawa, H.; Ohta, M.; Kato, K., Document Warehousing: a Document-intensive Application of a Multimedia Database, *Proceedings of Eleventh International Workshop on Research Issues in Data Engineering*, Heidelberg, Germany, 2001, pp: 25-31.

[11] Jasinschi, R.S., Louie, J., Automatic TV Program Genre Classification Based on Audio Patterns, *Proceedings 27th Euromicro Conference*, Warsaw, Poland, 2001, pp: 370-375.

[12] Jasinschi, R.S.; Dimitrova, N.; McGee, T.; Agnihotri, L.; Zimmerman, J.; Li, D., Integrated Multimedia Processing for Topic Segmentation and Classification, *Proceedings IEEE 2001 International Conference on Image Processing*, Thessaloniki, Greece, 2001, pp: 366 –369.

[13] Tobin J. Lehman, Michael J. Carey: A Study of Index Structures for Main Memory Database Management Systems, *Proc. 12th Int'l Conference on Very Large Data Bases*, Kyoto, Japan, pp: 294-303.

[14] MusicBrainz, http://www.musicbrainz.org/

[15] MPEG, http://mpeg.telecomitalialab.com/standards/

[16] Ohmori, Y.; Ouchi, K.; Hattori, M.; Doi, M., An XML Based Multimedia Data Acquisition and Retrieval with Wearable Computers, *Proceedings of 2001 International Conference on Distributed Computing Systems Workshop*, Mesa, Arizona, 2001, pp: 272-277.

[17] Open eBook, http://www.openebook.org/.

[18] PSIP, *Program and System Information Protocol, Document A/65* (Advanced Television Committee, Rev. A), May 2000.

[19] Shibin Qiu, Master's Degree Thesis, A Program Information Management System Design for Digital TV Broadcasting Using Information Retrieval Approach, Purdue University, Aug 2001.

[20] Ramakrishnan, R., *Database Management Systems* (McGraw Hill, 2000).

[21] Salton,G., *Automatic Text Processing, Readings* (MA: Addison-Wesley, 1989).

[22] Sedgwick, R., *Algorithms in C* (Addison-Wesley, 2000).

[23] Van Thong, J.-M.; Moreno, P.J.; Logan, B.; Fidler, B.; Maffey, K.; Moores, M. , Speechbot: an Experimental Speech-based Search Engine for Multimedia Content on the Web, *IEEE Transactions on Multimedia, Volume 4*, Issue 1, March 2002, pp: 88 -96

[24] XML, http://www.xml.com/

[25] J. Yoon, S. Kim, Schema Extraction for Multimedia XML Document Retrieval, *Proc. of International Database Symposium on Mobile, XML and Post-Relational Databases*, Hong Kong, 2000.