USING XML AS A MEDIUM FOR DESCRIBING, MODIFYING AND QUERYING AUDIOVISUAL CONTENT STORED IN RELATIONAL DATABASE SYSTEMS

Iraklis Varlamis[1], *Michalis Vazirgiannis*[2], *Panagiotis Poulos*[3]

1,2) Dept of Informatics,
Athens University of Economics & Business,
Patision 76, 10434,
Athens, HELLAS
{varlamis,mvazirg}@aueb.gr

3) Dept of Electrical and Computer Engineering
National Technical University of Athens
9, Iroon Polytechniou Str., 157 73,
Athens, HELLAS
ppoulos@otenet.gr

ABSTRACT

The digitization and annotation of audiovisual programs results in a huge amount of information that becomes useful only if it is properly organized and if the appropriate query mechanisms exist. Usually audiovisual information and meta-information is stored in relational database schemata that consist of decades of tables, relationships and constraints that complicate the information querying tasks. The PANORAMA* platform has been developed to cover needs for manipulating video information, by attaching meta-information for both audio and visual content of video sources. This paper presents this meta-information model and the database interface developed in terms of the PANORAMA platform. The model works as keystone in the creation of the database and the database interface implements a mechanism for converting XML documents to relational data. This mechanism allows information users to use a more understandable way to communicate with the database.

1. INTRODUCTION

Video companies are interested in digitization, annotation, and re-use of their audiovisual information. This process requires the co-operation of different applications that perform the different tasks under a common information model. To give an example, video annotating applications analyze and describe the contents of digitized videos. These descriptions are associated to the audiovisual content and are stored in a database so that they can be searched and re-used in other applications. It is desirable that the information exchanged among the different co-operative applications has a common structure. In the case of PANORAMA, XML is used to model the exchanged information and XML-Schema [1] defines the structure of the information. This information model follows the MPEG-7 definition for video metadata and is described in [2]. Furthermore all the audiovisual information transferred among the users, annotators and administrators is stored in a relational database management system in order to be available to other applications that do not use XML.

A digitized audiovisual program is associated to a set of meta-information such as contributors, technical specifications, existing copies of a program and maintenance state, descriptions of content etc. This meta-information is equally important to the digitized audiovisual program itself.

In order to cover this need for information storage and retrieval based on an easily understandable model, we developed a system where the interchanged information is expressed in XML and is centrally stored in an RDBMS. The system produces XML documents that contain meta-information regarding the audiovisual content and can be easily viewed in a web browser. The system is developed, to allow the video source owner to apply syntactic and semantic information to digitized programs and store it in a database, and the end-user to search this information.

In this paper focus is given on the database part of Panorama. A short description of the whole system architecture is performed and a more detailed analysis of the database component follows.

Apart from the general requirements, a set of specific requirements for the database system have been addressed concerning support for distant (web) users running

* The PANORAMA project was funded by the EPET framework for Research & Technology

different operating systems and having different levels of access to the database.

2. THE PLATFORM ARCHITECTURE

2.1. Database

The aim of the system was the development of a flexible interface for describing, modifying and querying video information that is stored in a relational database. The underlying DBMS in this case was Oracle 8i that supports multimedia data and handles efficiently the huge amount of data created by video decomposition and description process. Oracle Call Interface is used to communicate with the database, but an effort was made to create a database independent model, which will work with any underlying database management system using standard SQL commands.

2.2. Web & Local Interface

The whole annotation and query system, is supported by three applications:

- a web based application that uses Active Server Pages technology to create a user interface for querying and presenting the database contents,
- a local application developed in C++ that allows the media administrator to organize media and video content information into the database,
- a local application developed in C++ to allow video annotators to work separately and save their work into files that are forwarded to the media administrator.

The database is accessed directly only by web users and administrators, while annotators produces data to be stored in the database but have no direct access to it. The web application performs only select statements to the database while the media administrator's application can perform insert, update, delete, select and in special cases create statements to the database. In addition to this, web application users need to have different access levels to the information stored in the database therefore several types of connection privileges are created.

2.3. Database Interface

The use of XML for modeling the information transferred among applications, offers application developers a common communication protocol, but also induces the need for an extra application that will interfere between the XML documents and the relational database. This application is implemented as a COM object developed in C++ and become available for both web based and local applications. A novelty for the system is that the information transferred among the applications follows a common model, an XML-Schema [2], which defines the structure of the information. This model is automatically mapped into a relational schema that is used to create, populate and query the database.

2.4. XML

The implemented data model is based on the directives of MPEG-7 [3] format for video metadata. The work of Jain and Hampapur [4] is also proved very useful in the development of our system. The emerging XML-Schema notation is used to express the structure of the information model.

The use of XML conveys many advantages to the system:

- It offers a common model for information representation
- Information produced by annotators can be saved in XML files, which can be reviewed and corrected at any time before being sent to the media administrator. The content of an XML document can be easily read or modified using a simple text-editor and validated using the XML-Schema.
- This information can be saved in XML files, which are presented as informative documents to users.
- The analysis of the physical model of information can easily produce the XML-Schema. The XML language - which is inherently object oriented - becomes a standard, and as a result object oriented analysis tools will soon provide the ability to export the model of information in an XML-Schema file, which can be used as input to our system.

2.5. Information flow

The general architecture of PANORAMA is presented in Figure 1. The flow of information among the various system components is depicted in the following steps:

1) The administrator of the Video-Information Model, analyses the information structure by describing the different entities and relationships using XML-Schema.

2) The X-Database interface receives the XML document created into step 1 (XML-Schema is also an XML document) and generates SQL commands to create the database.

3) The annotator provides descriptions of the programs' contents and generates an XML document containing content information for each video.

4, 5) The media-administrator matches the digitized program descriptions with real media or media copies and adds all the media information related to each audiovisual object. The administrator inserts, updates, deletes and selects information from the database (step 4) and stores the digitized video files into the database (step 5)

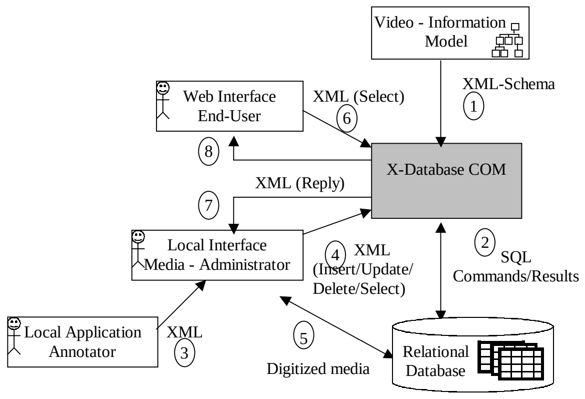

Figure 1. Short description of the system

6) The end-user can query the database through a web-based interface. The appropriate XML documents are sent to the database interface.
7,8) The X-Database interface receives the XML document created during steps 4 and 6 and generates a set of SQL commands. It retrieves the query results and constructs an XML-Reply document, which is sent either to the media administrator (step 7) or to the end-user (step 8).

3. THE X-DATABASE (XML TO RELATIONAL) COMPONENT

The X-Database component consists of two parts. The first part is responsible for parsing XML-Schema documents and collecting information on the structure of XML documents that follow this schema definition. The second part parses XML documents and constructs the appropriate SQL commands that are processed by the database. It also takes database results and formulates valid XML documents as reply to user queries.

3.1. The Database structure

XML's object oriented nature offers great advantages in describing information structure in a more comprehensible way. However, object oriented features like element nesting and inheritance are not supported in relational databases. Therefore, the database interface must be adapted to support such features, thus allowing users to access the database contents having in mind the structure of video information entities and not the structure of the database tables.

The following example describes a typical case of inheritance in XML. Entities "Video_Tape" and "DVD" are declared as extensions of entity "Medium" so they inherit all the attributes and elements of Medium in addition to their own attributes and elements. A "Program" entity makes a reference to the "Medium" it is stored in, but the referred id may be the id of a Video_Tape or a DVD.

In order to support this in the relational database schema a table named OBJECTIDS is created that keeps record of the id and type of each element that may inherit or be inherited. A group of triggers certifies that a reference to children elements through their parent's name is stored correctly as a reference to the children element in the database.

To give an example of XML nesting, an audiovisual object stores its information into 34 different tables. As a result, an attempt to insert an audiovisual object into the relational database evokes 34 insert SQL commands, which are executed recursively into the database.

3.2. Converting XML commands to SQL commands

During the creation of the XML-Schema all the physical relations among the various entities were expressed as relations of containment or reference among the respective complex types. During the database creation process all the elements of the XML-Schema are mapped into database tables and constraints.

Insert, update, delete or select XML documents result in a set of insert, update, delete or select queries to the database. A correct set of constraints, both database and programming ones, guarantee the integrity of the database into the aforementioned actions. Some of the constraints are:

a) Certain entities exist only as ***part of*** other entities. They appear only as sub-elements of other elements and cannot appear inside a ***DBInsert*** command. To give an example: the syntactical information is related to an audiovisual medium, so SyntacticDS appears only inside an AudioVisualDS, thus avoiding to store syntactic information that does not belong to any audiovisual medium.

b) Certain entities can be ***re-used by more than one*** entity. The complex types that correspond to such entities must appear as sub-elements of ***DBInsert***. When a complex type refers to another complex type, the latter must already have been inserted into the database.

c) Certain entities contain ***ordered instances*** of other entities. The order in which sub-elements appear in an element is important hence extra information must be stored in the database during the XML document parsing.

d) One or more commands can be send into the X-database module at the same time. These commands usually evoke a reply from the database, so the DBReply module must be able to group the database replies for each command.

These constraints are incorporated into the XML-Schema in the definition of ***extension elements***. A pre-parsing of the XML-Schema document gives the X-Database module all the information needed for the parsing of the XML documents.

The four database commands supported by the module are:
1) Insert: One or more ***top-level*** complex types can be found inside a DBInsert element. The parsing starts from the top-level element and continues recursively to all sub-elements, thus generating and executing a series of SQL INSERT statements.
2) Update: A ***DBUpdate*** element contains one or more ***top-level*** elements. All elements having a ***negative id*** are inserted. The rest of the elements, are updated.
3) Delete: Only ***top-level*** complex types with attribute ***"id"*** can be found inside a DELETE element. The database cascade deletes the information of all sub-elements.
4) Select:
A DBSelect element has three parts:

- A ***where*** part that contains the filter of the Select query.
- A ***from*** part where the elements that contain the attributes to be filtered appear.
- A ***return*** part that contains the element(s) to be returned.

The whole element is returned after a select statement.

The value of where attribute is the ***"where_clause"*** of our query. From the nested structure of elements inside the "from" part of the command, the parser is able to create all the ***"join conditions"*** among the tables that take part in the query. From the elements that appear in the "from" and "return" elements the ***"list_of_associated_tables"*** is generated.

The return element has a reference to the complex type(s) that must be selected. These references give the name of the table that corresponds to the ***"return_entity"***. So the first query that is send to the database has the following structure:

```
SELECT return_entity.id   FROM
 "return_entity", "list_of_associated_tables"
WHERE ("join_conditions")
AND ("where clause")
```

This query returns a list of ids of the entity to be returned. Using these ids a set of recursive select commands is addressed to the database to obtain all the information of the entities to be returned. The DBReply element contains the entities in their complete structure.

4. EXPERIMENTAL EVALUATION

In order to test the reliability but also the scalability of the X-Database module a series of test transactions is performed to the database. The system has been experimentally tested using a simple interface, where XML documents are inserted as text and the resulting XML file is displayed in a web browser. The simulation ran on a Pentium II computer with 128MBytes of RAM and an IDE HD.

In order to test the schema complexity we measured the number of tables created, the number of foreign key and triggers that guarantee the schema integrity, as well as the database creation time for XML-Schema files of different complexity. For the complete XML-Schema (68 complexTypes, 18 inherited elements) 140 tables, 77 triggers and 260 foreign key constraints are generated in 52 seconds.

Attempting to measure the insertion and selection time, we found that both values increase linearly to the number of elements inserted given that all the elements are of the same type. When elements of different type are inserted, the number of consequent insertions differs and as a consequence the relation to time is not linear.

5. RELATED WORK

Several research and commercial systems provide automatic indexing and querying of visual contents ([5], [6]). Such platforms provide solutions on the audiovisual content annotating and indexing, but unfortunately are not available for web applications. Nonetheless, they can be used as a basis for extracting information from the audiovisual sources that can be wrapped in our XML model.

The X-Database interface offers a fairly simple query mechanism combining the default structure of XML documents with the simplicity of SQL "Select" commands. Most of the query categories proposed in [7] could be performed using our DBSelect element, such as Simple Visual Feature Query, Feature Combination Query, Query by Example etc.

Compared to the interfaces used by commercial relational database management systems ([8], [9], [10],

[11]), X-Database provides a complete solution in XML documents manipulation. All the above systems do not provide the ability to create the database schema based on the XML-Schema and moreover to use the same XML-schema to validate all the XML documents and commands that are forwarded to the database.

6. CONCLUSION AND FURTHER WORK

PANORAMA is a platform that handles digitised video data and meta-information. In this platform different applications access a common data repository and exchange data in XML format. The structure of XML information is defined in an XML-Schema document that guarantees the validity of transferred documents. The same XML-Schema has been used to build the database where information is stored.

The X-Database module works as an interface between the database and the co-operating applications. The applications interact with the relational database system only using XML documents without taking into account the underlying database schema. This provides a simple and database independent mechanism for storing and retrieving XML documents' information into relational DBMSs. Although the mechanism is customised for video meta-information, any information model can be supported. Fault tolerance is achieved by adding appropriate control procedures to the database, such as triggers that check the validity of inserted values, reference constraints that guarantee cascade removal of an object (and all the objects it contains) from the database. The last is important since information for an object may be stored in more than one table in our database.

Further research will focus into retrieval and advanced query tasks. The retrieval process can be easily accelerated if the appropriate indexes are created. For this task the current XML-Schema can be enriched to precisely define the information that must be indexed, or even more the information entities on which to perform similarity search etc. Finding an efficient notation to describe such requirements in XML-Schema is crucial in creating a database schema that will serve advanced retrieval needs.

7. REFERENCES

[1] "XML Schema Part 0: Primer," W3C Working Draft, Sept. 2000
(http://www.w3.org/TR/xmlschema-0)

[2] G. Akrivas, S. Ioannou, E. Karakoulakis, K. Karpouzis, Y. Avrithis, A. Delopoulos, S. Kollias, I. Varlamis, M. Vazirgiannis, An Intelligent System for Archiving and Retrieval of Audiovisual Material Based on the MPEG-7 Description Schemes, WSES/IEEE Multiconference (2001).

[3] ISO/IEC JTC1/SC29/WG11, "MPEG-7 Overview (v. 1.0)," Doc. N3158, Dec. 1999.

[4] R. Jain, A. Hampapur: Metadata in Video Databases. SIGMOD Record 23(4): 27-33 (1994)

[5] M. Flickner et al, Query by Image and Video Content: The QBIC System, IEEE Computer Vol. 28, No. 9, September 1995.

[6] A. Gupta, Visual Information Retrieval Technology, A VIRAGE Perspective white paper, Virage, 1995.

[7] Y. Alp Aslandogan and Clement T. Yu, "Techniques and Systems for Image and Video Retrieval", IEEE Transactions on Knowledge and Data Engineering, vol. 11, no. 1, Jan.-Feb. 1999.

[8] IBM's DB2 extender for XML
(http://www-4.ibm.com/software/data/db2/
extenders/xmlext.html)

[9] Microsoft SQL Server XML support,
(http://msdn.microsoft.com/msdnmag/issues/
0300/sql/sql.asp)

[10] Informix and XML,
(http://www.informix.com/xml/)

[11] Steve Muench, Using XML and Relational Databases for Internet Applications, Oracle Corporation
(http://technet.oracle.com/tech/xml/info/htdocs/
relational/index.htm#ID795)