The Design and Implementation of a Distributed Web Document Database *

Timothy K. Shih
Dept. of Computer Science and
Information Engineering
Tamkang University
Tamsui, Taiwan 251, R.O.C.
email: TSHIH@CS.TKU.EDU.TW

Jianhua Ma and Runhe Huang
Dept. of Computer Software
The University of Aizu
Aizu-Wakamatsu City
965-8580 Japan
jianhua@u-aizu.ac.jp

Abstract

Distance learning has become a very important mechanism for virtual university operation. In order to realize such an operation smoothly, it is necessary to consider distance learning from three perspectives: **administration**, **awareness**, *and* **assessment**. *We are currently implementing a virtual university environment according to these guidelines. In this paper, we propose a new Web documentation database as a supporting environment of the Multimedia Micro-University project* [1]. *The design of this database facilitates a Web documentation development paradigm that we have proposed earlier. From a script description, to its implementation as well as testing records, the database and its interface allow the user to design Web documents as virtual courses to be used in a Web-savvy virtual library. The database supports object reuse and sharing, as well as referential integrity and concurrence. In order to allow real-time course demonstration, we also propose a simple course distribution mechanism, which allows the pre-broadcast of course materials. The system is implemented as a three-tier architecture which runs under MS Windows and other platforms.*

Key words: Multimedia Database, WWW, Virtual University, Information Retrieval, Virtual Library, Distance Learning

1 Introduction

Distance learning is a possible revolution of future education. With the growing amount of users of Internet, Web documentation based course delivery becomes widely available. Distance learning is thus a trend toward future university. In order to facilitate the teaching and learning environment, Web browsers, audio/vedio communication tools, and data conferencing tools are widely developed. We have surveyed a number of such systems and come out with a suggestion guideline of the integration of virtual university tools. We believe that, a well-considered virtual university supporting system needs to meet the following three criteria:

- **Administration Criterion:** Unlike Web document based course delivery, a virtual university environment needs to have administration facilities to keep admission records, transcripts, and so on. These administration tools should be available to administrators, instructors, and students (e.g., checking transcript information).
- **Awareness Criterion:** Distance learning is different from traditional education. Since instructors and students are separated spatially, they are sometimes hard to "feel" the existence of each other. A virtual university supporting environment needs to provide reasonable communication tools such that awareness is realized.
- **Assessment Criterion:** Assessment is the most important and difficult part of distance education. Tools to support the evaluation of student learning should be sophisticated enough to avoid unbiassed assessment.

These criteria are difficult to achieve. However, it is possible to design such a virtual university environment upto a certain degree of satisfaction. Multimedia Micro-University (MMU) is a joint research project with the participation of researchers from around the world. The primary goal of the MMU consortium is to develop technologies and systems for the use of virtual university. In this paper, we propose a software architecture for a multimedia-based virtual course system. The architecture supports multi-platform due to the availability of Web browsers and the Java virtual machine. We aim to provide a system on the Internet for instructors to design and demonstrate lectures. This system serves as a step toward our research goal - virtual university over the Internet. The primary directive of our MMU systems has four goals:

[1] Multimedia Micro-University is a joint research project with the participation of researchers from the University of Pittsburgh, USA, Tamkang University, Taiwan and the University of Aizu, Japan.

- **Adaptive to changing network conditions:** The system adapts to QoS requirements and network conditions to deliver different levels of service.
- **Adaptive to changing user needs:** Users are using the system from different perspectives. Types of users include students, instructors, and administrators. The system supports the demand of various kinds of information delivery from time to time.
- **Adaptive to Web-based environment:** The system is Web-savvy. That is, a standard Web browser is the only software required to students and administrators. The instructors will use our system running on a Web browser in conjunction with some commercial available software.
- **Adaptive to open architecture:** A minimal compatibility is defined as the requirement for the open architecture. Compatibility requirements include presentation standard, network standard, and database standard.

On the instruction design side, we encourage instructors to use the Microsoft FrontPage editor, or an equivalent on a Sun workstation, to design virtual courses. Virtual courses may also be provided via some Java application programs, which are embedded into HTML documents. Since HTML and Java are portable languages, multi-platform courses are thus feasible. An instruction annotation editor, written as a Java-based daemon, is also running under the Java virtual machine (which is supported by Web browsers). This annotation daemon allows an individual instructor to draw lines, text, and simple graphic objects on the top of a Web page. Different instructors can use the same virtual course but different annotations. These annotations, as well as virtual courses, are stored as software configuration items (SCIs) in the virtual course database management system. A SCI can be a page shows a piece of lecture, an annotation to the piece of lecture, or a compound object containing the above. A class administrator performs book keeping of course registration and network information, which serves as the front end of the virtual course DBMS. The implementation of the virtual course DBMS uses JDBC (or ODBC) as the open database connection to some commercial available database systems, such as the MS SQL server, Sybase, Informix, or Oracle servers. Currently, our system uses the MS SQL server.

On the other side of the system architecture, a student can use an ordinary Web browser to traverse virtual lectures. However, some underlying sub-systems are transmitted to a student workstation to allow group discussions, annotation playback, and virtual course assessment. We also provide an interface to the virtual course library. The interface allows object searching and browsing. These sub-systems, again, are written as Java-based daemons running under the Internet Explorer or the Netscape Navigator. Help facilities are provided.

This research project, besides providing a prototype system for virtual university realization, also focuses on some research issues in multimedia computing and networking. From the perspective of software engineering, several paradigms were used in software development (e.g., water-fall model, spiral model, object-oriented model, etc.). Can these models be applied to multimedia course development? Or, can we refine these models to be used? On the other hand, how do we estimate the complexity of a course and how do we perform a white box or black box testing of a multimedia presentation are research issues that we have solved partially. From the perspective of CSCW, the virtual course system maintains the smooth collaboration and consistency of distributed course designs. A software configuration management system allows checking in/out of course components and maintain versions of a course. All instruction systems require assessment. The virtual course system has methodologies which support the evaluation of student progress and achievement.

In this paper, we focus our discussion on a virtual course database management system. We have proposed a Web document development paradigm earlier. The objective of this new course database is to support such a paradigm and its application tools. The paper is organized as the following. In section 2, we present some issues of multimedia databases which were developed by others. We then propose our database design in section 3. The implementation techniques, which include mechanisms for object reuse, distribution and sharing are given in section 4. We propose a Web-based virtual library in section 5. And a short conclusion is given in section 6.

2 Related Works

Multimedia database management system design is an important and interesting research topic in the community of multimedia computing and networking. In order to support the production of multimedia applications, the management of multimedia resources (e.g., video clips, pictures, sound files) is essential. For instance, multimedia presentations can be designed as building blocks which can be reused. To facilitate multimedia application design, many articles indicate the need of a multimedia database [1, 5, 7]. A multimedia database is different from a traditional relational database in that the former is object-oriented while the latter relies on entity relations. Moreover, a multimedia database needs to support continuous resource types of large and variable sizes. Due to the amount of binary information that need to be processed, the performance requirement of a multimedia database is high. Clustering and indexing mechanisms support multimedia databases are thus important. The discussion of research issues in multimedia database management systems can be found in [5]. A distributed database supporting the development of multimedia applications is introduced in [1]. A mechanism for formal specification and modeling of multimedia object composition is found in [4]. The work discussed in [4] also considers the temporal properties of multimedia resources. A database system for video objects is discussed in [3]. A content-based querying mechanism for retrieving images is given in [7]. Layered multimedia data mod-

eling [6] suggests a mechanism to manage multimedia data. In addition to the general discussion on multimedia database management systems (MDBMSs), there are other articles take a similar approach to ours. The work discussed in [2] proposes a multimedia data model and a database to support hypermedia presentations and the management of video objects. Its specialized video server with an incremental retrieval method supports VCR like functions for heterogeneous video clips. The design of multimedia DBMS is from the scratch, which is similar to our approach. The system also supports object composition/decomposition. However, no specific reuse mechanism is emphasized in the discussion. Only an object-oriented data model was proposed. The system also provides a global data sharing mechanism, including a video and an image collaboration tools, which are integrated with a distributed environment.

3 Specification of the Web Document Database

To support the storage requirement of our Web document development paradigm, we have designed a Web document database. We use an off-the-rack relational database system as the underlying supporting system. In this section, we discuss the design considerations of our database system.

We design the database based on three objectives: object reuse, object distribution, and resource sharing. The mechanisms to achieve these goals are discussed in section 4. In this section, a three-layered database hierarchy is proposed. In the Web document DBMS, multiple Web document databases are allowed. Each database can have a number of documents. Each document is identified by a unique *script name*. A script, similar to a software system specification, can describe a course material, or a quiz. With respect to a script, the instructor can have different tries of implementation. Each implementation contains at lease one HTML file, and some optional program files, which may use some multimedia resources. A course implementation can be used by different instructors. An instructor can use our annotation tool to draw lines and text to add notes to a course implementation. Thus, an implementation may have different annotations created by different instructors. To test the implementation, test records are generated for each implementation. And, bug reports are created for each test record. The database has three layers. Objects in the three layers contain the following attributes:

- **Database Layer**
  - Database name: a unique name of the database.
  - Keywords: one or more keywords are used to describe the database.
  - Author: author name and copyright information of the creator.
  - Version: the version of this database.
  - Date/time: the date and time this database was created.
  - Script names: pointers to script tables belong to the database.
- **Document layer**
  - Script Table: content of a script object.
  - Implementation Table: content of an implementation object.
  - TestRecord Table: content of a test record.
  - BugReport Table: content of a bug report.
  - Annotation Table: content of an annotation.
  - HTML files: standard HTML files used in the implementation
  - Program files: add-on control program files used in the implementation
  - Annotation files: Annotation files which store document annotations
- **BLOB layer**
  - Multmedia sources: multimedia files in standard formats (i.e., video, audio, still image, animation, and MIDI files). Objects in this layer are shared by instances and classes.

In the database hierarchy, each object may be linked to other relative objects. A link in the hierarchy is associated with a label, which has a reference multiplicity indicated in its superscript. A "+" sign means the use of one or more objects. And a "*" sign represents the use of zero or more references. Database objects can be reused. A number of database objects are grouped into a reusable component. The component can be duplicated to another compound object with modifications. However, the duplication process invloves objects of relatively smaller sizes, such as HTML files. BLOBs in large sizes are shared by different compound objects, including different scripts and implementations. However, BLOB resource sharing is limited to a workstation. Upon demand, BLOB objects may be duplicated in other workstations in order to realize real-time course demonstration.

The document layer contains the most important items of a Web document. Since we use a relational database management system to implement our object hierarchy, we summarize some of the major tables here.

- **Script Table**
  - Script name: a unique name of the document script.
  - Keywords: kewords of the script.
  - Author: the author of the document.
  - Version: the version of the document.
  - Date/time: the creation date and time.

  - Description: the content of the script, which is described in text. However, the author may have a verbal description which is stored in a multimedia resource file.
  - Expected date/time of completion: a tentative date of completion.
  - Percentage of completion: the status of current work.
  - Multimedia resources: file descriptors point to multimedia files.
  - Starting URLs: foreign key to the implementation table.
  - Test record names: foreign key to the test record table.
  - Bug report names: foreign key to the bug report table.
  - Annotation names: foreign key to the annotation table.
- **Implementation Table**
  - Starting URL: a unique starting URL of the Web document implementation.
  - HTML files: implementation objects such as HTML or XML files.
  - Program files: implementation objects such as Java applets or ASP programs.
  - Multimedia resources: implementation objects such as audio files.
  - Script name: foreign key to the script table.
  - Test record names: foreign key to the test record table.
  - Bug report names: foreign key to the bug report table.
  - Annotation names: foreign key to the annotation table.
- **TestRecord Table**
  - Test record name: a unique name of the test record.
  - Testing scope: local or global.
  - Web traversal messages: windowing messages which control a Web document traversal.
  - Script name: foreign key to the script table.
  - Starting URL: foreign key to the implementation table.
  - Bug report names: foreign key to the bug report table.
- **BugReport Table**
  - Bug report name: a unique name of the bug report.
  - Quality assurance engineer: name of the QA person.
  - Test procedure: a simple description of the test procedure.
  - Bug description: the test result.
  - Bad URLs: a number of URLs which can not be reached.
  - Missing objects: multimedia or HTML files missing from the implementation.
  - Inconsistency: a text description of inconsistency.
  - Redundant objects: a list of redundant files.
  - Test record name: foreign key to the test record table.
- **Annotation Table**
  - Annotation name: a unique name of the annotation.
  - Author: the author of the annotation.
  - Version: the version of the annotation.
  - Date/time: the creation date and time.
  - Annotation file: a file descriptor to an annotation file.
  - Script name: foreign key to the script table.
  - Starting URL: foreign key to the implementation table.

The Web document database, when updated, should be proceeded in a consistent way. Each Web document SCI has a number of references. We use these references to maintain the referential integrity of the database. We maintain a referential integrity diagram. Each link in the diagram connects two objects. If the source object is updated, the system will trigger a message which alerts the user to update the destination object. Each link in the diagram is associated with a label, with various number of possible alert messages. For instance, if a script SCI is updated, its corresponding implementations should be updated, which further triggers the changes of one or more HTML programs, zero or more multimedia resources, and some control programs.

Due to the locking mechanism used in object-oriented database systems, we have defined an object locking compatibility table. In general, if a container has a read lock by a user, its components (and itself) can have the read access by another user, but not the write access. However, the parent objects of the container can have both read and write access by another user. Of course, the accesses are prohibited in the current container object. Locking tables are implemented in the instructor workstation. With the table, the system can control which instructor is changing a Web document. Therefore, collaborative work is feasible.

4 Implementation of the Web Document Database

Web documents are reusable. Among many object reuse paradigms, classification and prototyping are the most common ones. Object classification allows object properties or methods at a higher position of the hierarchy to be inheritanced by another object at a lower

position. The properties and methods are reused. Object prototyping allows reusable objects to be declared as templates (or classes), which can be instantiated to new instances. A Web document in our system contains SCIs for script, implementation and testing. As a collection of these three phrases of objects, a Web document is a prototype-based reusable object. Object reuse is essentially important to the design of Web documents. However, the demonstration of Web documents may take a different consideration due to the size and the continuous property of BLOB.

Web documents may contain BLOB objects which is infeasible to be demonstrated in real-time when the BLOB objects are located in a remote station due to the current Internet bandwidth. However, if some of the BLOB objects are preloaded before their presentation, even the process involves the use of some extra disk space, the Web document can be demonstrated in real-time. However, BLOB objects in the same station should be shared as much as possible among different documents. We aim to provide a system to make distributed Web documents to be reused in a reasonable efficient manner.

The design goal is to provide a transparent access mechanism for the database users. From different perspectives, all database users look at the same database, which is stored across many networked stations. Some Web documents can be stored with duplicated copies in different machines for the ease of real-time information retrieval. A Web document may exist in the database at different physical locations in one of the following three forms:

- Web Document class
- Web Document instance
- Web Document reference to instance

A document class is a reusable object which is declared from a document instance. A document instance may contain the physical multimedia data, if the instance is newly created. After the declaration of the document instance, the instance creates a new document class. The newly created class contains the structure of the document instance and all multimedia data, such as BLOBs. The original document instance maintains its structure. But, pointers to multimedia data in the class is used instead of storing the original BLOBs. When a new document instance is instentiated from a document class, structure of the document class is copied to the the new document instance and pointers to multimedia data are created. This design allows the BLOBs to be stored in a class. The BLOBs are shared by different instances instantiated from the class.

A document instance is a physical element of a Web document. When a database user looks at the Web document from different network locations, the user can access the Web document in two ways. The first is to access the document directly. The second mechanism looks at the document via document reference. A document reference to instance is a mirror of the instance. When a document instance is created, it exists as a physical data element of a Web document in the creation station. References to the instance are broadcasted and stored in many remote stations.

When a document instance is retrieved from a remote station more than a certain amount of iterations (or more than a watermark frequency), physical multimedia data are copied to the remote station. The duplication process may include the duplication of document classes, which contain the physical BLOBs.

The duplication process is proceeded according to a hierarchy distribution strategy. Assuming that, N networked stations join the database system in a linear order. We can arrange the N stations in a full m-ary tree according to a breadth first order. A full m-ary tree is a tree with each node contains exactly m children, except the trailing nodes. The n-th station, where $1 \leq n \leq N$, in the linear joining sequerence has its i-th child, where $1 \leq i \leq m$ at the following position in the linear order:

$$m * (n-1) + i + 1$$

In a Web document system which utilizes a listance learning system, an instructor can broadcast lectures to student work stations. Essentially, the broadcast process is a multi-casting activity. With the appropriate selection of m, the propogation of physical data can be proceeded in an efficient manner, starting from the instructor station as the root of the m-ary tree. The implementation of this multi-casting system has a broadcast vector contains a linear sequence of woakstation IP addresses. The system maintains the sizes of m's, based on the number of workstations and the physical network bandwidth for different types of multimedia data. This design achieve one of our project goals: adaptive to changing network conditions.

On the other hand, a student can look at an off-line lecture presentation prepared by the instructor. In this case, the instructor station serves as a lecture server. Lecture presentations are transmitted to student work stations upon demands. The broadcast route can use an inverse function of equation 4. The k-th station, where $1 \leq k \leq N$, in the linear joining sequerence has its unique parent at the following position in the linear order:

$$(k-i-1)/m + 1,$$
$$where\ i = (k-1)\ \texttt{mod}\ m,\ if\ i \mid m;$$
$$i = m,\ otherwise$$

The duplication of lecture presentations are upon demand. A child node in the m-ary tree copies information from its parent node. However, if a workstation (and its child workstations) does not review a lecture, it is not necessary to duplicate the lecture. The station only keeps a document reference in this case. Since the duplication process may involve extra disk space, one may argue that disk spaces are wasted. However, the duplicated document instances live only within a duration of time. After a lecture is presented, dupli-

cated document instances migrate to document references. Essentially, buffer spaces are used only. However, the instructor workstation has document instances and classes as persistence objects. The above equations are proved by mathematical induction and double induction techniques. They are also implemented in our system.

Another issue of object propagation is that objects in the BLOB layer of the database are shared by objects at a higher level in the hierarchy. That is, in both the instructor station and the student station, BLOBs are shared among different lecture presentations. Since an individual multimedia resource is used only by a presentation in a workstation with respect to a time duration, concurrent access is not a consideration. This strategy avoid the abuse of disk storage.

5 A Web Document Virtual Library

As discussed in section 4, the database has three sorts of objects: classes, instances, and references. Document classes support object reuse. Instances are physical objects of a Web document, which are referred by document references. In the proposed virtual university architecture, in order to support off-line learning, we encourage students to "check out" lecture notes from a virtual library. Web Document instance are stored in the virtual library. An instructor has a privilege to add or delete document instances, which contain lecture notes as Web pages. Students can check out and check in these Web pages. However, in general, there is no limitation of the number of Web pages to be checked out. The check in/out procedure serves as an assessment criteria to the study performance of a student. We provide a browsing interface which allows students to retrieve course materials accoring to matching keywords, instructor names, and course numbers/titles. This virtual library is Web-savvy. That is, the searching and retrieve processes are running under a standard Web browser. The library is updated as needed. The mechanism follows another guidance of our project goals: adaptive to changing user needs. We are developing three Web courses based on the virtual library system: introduction to computer engineering, introduction to multimedia computing, and introduction to engineering drawing.

6 Conclusions

Distance learning, virtual university, or remote class room projects change the manner of education. With the tremendous growing amount of Internet users, virtual university is a step toward the trend of future university. However, most development of distance learning systems rely on the high bandwidth of a network infrastructure. As it is not happening everywhere on the Internet to meet such a high requirement, it is worthwhile to investigate mechanisms to cope with the real situation. Even in the recent future, with the next generation of Internet, the increasing amount of users consume an even higher network bandwidth. The primary directive of Multimedia Micro-University Consortium is looking for solutions to realize virtual university. Some of our research results, as pointed out in this paper, adapt to changing network conditions. Using an off-line multi-casting mechanism, we implemented a distributed virtual course database with a number of on-line communication facilities to fit the limitation of current Internet environment. The proposed database architecture and database system serve as an important role in our virtual university environment. We are currently using the Web document development environment to design undergraduate courses including introduction to computer science and others.

References

[1] Te-Chih Chen, Wei-Po Lin Chin-An Wu, and Chih-Shen Shen, "A Client-Server Database Enviornment for Supporting Multimedia Applications," in proceedings of the 18th IEEE annual international computer software and application conference (COMPSAC'94), Taipei, Taiwan, 1994, pp 215–220.

[2] C.Y. Roger Chen and Dikran S. Meliksetian and Martin Cheng-Sheng Chang and Larry J. Liu, "Design of a multimedia object-oriented DBMS," Multimedia Systems, Vol. 3, 1995, Springer-Verlag, pp 217–227.

[3] Keh-Feng Lin, Chueh-Wei Chang, and Suh-Yin Lee, "Design of an Interactive Video Database," in proceedings of the 1994 HD-Media Tech. and Application workshop, Taipei, Taiwan, pp PO2-17–PO2-22.

[4] Thomas D. C. Little and Arif Ghafoor, "Synchronization and Storage Models for Multimedia Objects," IEEE Journal on Selected Areas in Communications, Vol. 8, No. 3, April, 1990, pp 413–427.

[5] Raymond Paul, M. Farrukh Khan, Ashfaq Khokhar, and Arif Ghafoor, "Issues in Database Management of Multimedia Information," in proceedings of the 18th IEEE annual international computer software and application conference (COMPSAC'94), Taipei, Taiwan, 1994, pp 209 – 214.

[6] Gerhard A. Schloss and Michael J. Wynblatt, "Presentation Layer Primitives for the Layered Multimedia Data Model," in Proceedings of the IEEE 1995 International Conference on Multimedia Computing and Systems, May 15-18, Washington DC, 1995, pp 231 – 238.

[7] Atsuo Yoshitaka, Setsuko Kishida, Masahito Hirakawa, and Tadao Ichikawa, "Knowledge-Assisted Content-Based Retrieval for Multimedia Database," IEEE Multimedia Magazine, Winter 1994, pp 12 – 21.