DESIGNING A MULTIMEDIA DATABASE FOR ELECTRONIC PUBLICATION SYSTEMS

Chien-Chih Yu

摘要

在整合性辦公室資訊系統的許多應用中，都包含有各式文件的產生與傳遞。為有效管理異質分散環境中的文件製作流程，多項系統功能如資料庫建構、資訊查詢，及設備整合控制等都很重要。同時，能用来定義、處理及控制包含數字、文字、影像、圖形等在內的多媒體資料庫管理系統，更形需要。此外，電子印刷或電子排版系統，在辦公室自動化中的基礎工具角色也引人愈來愈多的注意。然，多媒體資料庫管理與電子印刷是智慧型文件管理的主要因素，但如何達成兩者之間的整合，仍有待進一步的探討。本文的目的，即在提出一個概念模式方法來連結多媒體資料庫和文件製作系統，並提供一個實體系統建構的軟體整合方法。

Abstract

In an integrated office information system, applications involve generation and distribution of different kinds of documents in a heterogeneous distributed environment. Diverse features of system functions including data base creation, information retrieval and presentation, and control of physical devices, etc. are needed for efficiently managing the document generating process. Within this system architecture, a multimedia database management system which is capable of defining, manipulating and controlling various types of information objects such as numbers, characters, image, and graphics is in great demand. Desktop publishing or electronic publication systems, on the other hand, are drawing more and more attentions for their roles as basic tools in facilitating office automation. Although multimedia database management and electronic corporate publishing are key factors for intelligent document management, efficient linkage of these two systems is yet to be accomplished. The goal of this paper is to propose conceptual document modeling techniques and processes for integrating multimedia database with electronic publishing systems and to provide a software integration approach for supporting physical system implementation.

Keywords: multimedia database, electronic publication, office automation (OA), software integration.

*作者為本校資訊管理系教授
I. Introduction

Database has been defined as a mechanized, shared collection of interrelated data. It is formally defined and capable of satisfying varied information needs in an organization. A document can be considered as a structured presentation of related information in a prespecified format. Database management then encompasses the use and control of data from various sources for the processing of information and documents. Major functions of a database management system (DBMS) consists of database definition, creation, revision, restructuring, query, update, backup, recovery, concurrency control, data communication interfaces, programming facilities, security and performance monitoring mechanisms. Associated management tools for database applications include system administrator, form editor, report writer, application generator, data dictionary, graphical user interface, 4 GL, and data exchange programs, etc. The advantages obtained from applying database approach to the development of an information system are minimal data redundancy, consistency of data, integration of data, sharing of data, enforcement of standards, ease of application development, uniform security, privacy, and integrity control, data accessibility and responsiveness, data independence, and reduced program maintenance [14].

Multimedia database extends the database capabilities to facilitate multiple forms of information contents such as number, text, images, graphics, voice, animation, and video in the same system environment. Multimedia database management system is a DBMS for defining, manipulating, and controlling multimedia databases and for generating multimedia documents. Research prototypes and commercial packages that intend to meet the requirement specifications of a multimedia DBMS are still under continuous development. Only a few multimedia authoring tools such as Hypercard (Apple), Supercard (Silicon Beach), Toolkit (Asymetrix), Guide (Owl), and Authorware Professional (Authorware), etc. have been available in the software market for the past few years. These systems provide no comprehensive database management functions, but the operating environments for editing and organizing multimedia information of electronic documents as a non-linearly linked network of frames that can be navigated.

Desktop publishing (DTP) softwares, unlike cut-and-paste only word processors, consider publishing activities in a greater depth. Page layout design, table formulation and placing, word processing, hyphenation, typographic control, on-line dictionary, spell checking, automatic text reflowing, graphics manipulation, image handling, searching and replacing, file translation, print preview, and WYSIWYG (What
You See Is What You Get) output control are some of the baseline features. The applications of DTP aim at supporting high quality document generation in the areas of professional publications, business communications, and office administration. Commercial products available for various levels of microcomputer users include Aldus Corp’s Pagemaker, Ventura Software’s Ventura Publisher, IMSI’s Pages, Spinnaker Software Corp’s PFS: First Publisher, Timeworks Inc.’s Publish-It, and Quark Inc.’s Quark Xpress, etc. [2,16,25]. Electronic publication software systems such as DynaText [22] and MediaView [19] provide industrial strength electronic documentation environments that extend the flexibilities and power of document construction and distribution functions to process multimedia components, allow electronic delivery to remote sites, and permit reuse of information objects for other applications. In the views of document modeling and management, we can treat the printing style DTP and the navigating style electronic publication systems as softwares of the same category with their terms being interchangeable in the rest of this paper.

Many applications in office automation (OA) have stimulated the need for linking database and electronic publishing systems to increase efficiencies and productivities for document management [1,8,9,13,21,28,29]. Information collected, saved, and manipulated from internal and external sources of an office environment may include alphanumerical data, spreadsheets, forms, graphics, images, and full-text documents. Efficient multimedia database as well as multimedia DBMSs that accommodate various types of information objects are increasingly demanded by business users. There systems should be capable of allowing users not only to define, store, retrieve and modify heterogeneous types of information but also to design, edit, and generate customized multimedia documents. In this category of demand, integrating desktop publishing or electronic publication systems into the multimedia database environment can replace the traditional plain-text report writers as more effective document generating, managing and printing tools. On the other hand, electronic publishing systems evolved from composition and pagination softwares is becoming high-end document processing and distribution systems in an organization. Information objects gathered by film recorders, scanners, videotex, online databases, word processors, printers, graphics devices, and so on can be imported and integrated to produce complex multimedia documents. The produced documents can then be published and/or distributed in whole or by part in printed or electronic forms to different users on a variety of output and storage devices such as laser printers, microfilms, optical disks, and/or through electronic mail. In order to accomplish the goals of integrating and sharing multiple information objects in diverse
electronically generated documents for satisfying various publications and version control needs, a common document database with sufficient management functions is desired. In this case, the multimedia database system approach has great potentials to become an optimal solution to this problem.

Although modern information technologies have explosively expanded its powers in the past few years, such as the faster processors, higher storage capacities, more accessible expanded/extended memories, less expensive multimedia data processing softwares and hardwares, more flexible and acceptable communication protocols, more compatible networking facilities, more efficient compression and decompression techniques, more friendly graphic user interfaces, and more complete application development tools etc., it is still lacking of a functionally powerful and compact multimedia DBMS capable of manipulating various types of documents in a single data base environment. To develop a multimedia database and management system that equips with electronic publishing functions or to provide DTP or digital publication softwares with embedded database management facilities for defining, manipulating, and controlling multimedia publications are by no means easy tasks to be accomplished. An integrated office information system which can take advantages of both multimedia database management and electronic publishing capabilities and achieve desired productive performances in office works is even a further destination to reach.

Considerable papers have been devoted to this area of research in recent years. Related topics in the literature include modeling techniques for multimedia data [12,24,26,30], standards for open document processing [5,22], multimedia document presentation and management [7,17], image base management systems [6,8,20], multimedia database architecture [3,17,30], high level languages for pictorial or graphic query processing [4,18,23], system integration and applications for linking database and multiple data processing components [9,10,11,26,27,31,32], and so on. Since previous research works focussed on specific technical issues and solved the target problems only partially, besides, no systematic approach for integrating multimedia database and electronic publication systems has been presented, the goal of this paper is to propose an uniform database modeling scheme for designing conceptual multimedia document structure in which multiple types of data in varying degrees of details are described, and then, to present a system architecture as well as operating processes for integrating multimedia database and electronic publication systems. This approach starts from an object-oriented (OO) analysis and design process and then converts the OO model to relational model for guiding physical implementation. Conventional relational DBMS can then be extended to have
Designing a Multimedia Database for Electronic Publication Systems

the capabilities of manipulating multimedia documents in consistent manner. The remainder of this paper is divided into three sections. In Section II, fundamental concepts for modeling different types of information and documents in the same database are discussed. An architecture as well as a process for linking DBMS with DTP or electronic publication softwares to establish a multimedia document database management environment are presented in Section III. A conclusion is given in Section IV.

II. FUNDAMENTAL CONCEPTS FOR DOCUMENT MODELING

A multimedia document consists of structured information contents displayed in a specified format. In traditional DTP environment, the created documents are stored by pages via image files. User can only retrieve information by full pages. There is no attributes and relationships defined on information objects in a page to allow for more flexible document processing. However, there are increasing needs from users to access information by not only a full page but also some specific portions of a page and to generate an alternative version of document in which information objects are reorganized and presented in a different format. Thus, instead of defining and storing documents by pages, information objects that form a page as well as the relationships of these objects in that page should be structured into a conceptual data model. Also defined are the mapping procedures for obtaining page-related objects and for publishing the associated page views. Managing information objects for documents this way greatly enhances the sharibility and integratibility of heterogeneous information objects for satisfying varied document needs. Data redundancy is minimized and data independence can be achieved. Interrelated objects can be accessed through ad hoc queries and be assembled to form a new page. To efficiently and effectively design such a conceptual model for multimedia documents, one must take into account not only the representation of a logical content structure but also the specifications for document layout control. Also to be considered is the convertibility to a DBMS - dependent program accessible model for physical implementation and execution. Since the modeling technologies included in the ongoing development of the international standards for open multimedia document processing such as SGML (Standard Generalized Markup Language) [22] and ODA (Office Document Architecture) [5] have not yet been well established, some feasible but functionally restricted attempts for actual document model creation and database manipulation still based on usual relational DBMSs with
some degrees of modifications in the constructs and processes of language tools [4,10,18,23,28,31]. By considering the rapidly emerging object-oriented methodologies and their potentials in the multimedia database design as well as document management domains, we adopt the object-oriented data modeling concepts along with an extended relational database approach capable of defining BLOB (Binary Large OBject) typed fields and map these concepts and approaches to the conceptual and implementation design phases for constructing multimedia document models and databases. Detailed descriptions of the object-oriented model and the relational model with BLOB capabilities are beyond the scope of this paper. Related papers can be found in the reference [12,26,30,24]. In this section, only fundamental modeling concepts and corresponding design processes are presented in accordance with the resulting document models.

1. Object-oriented modeling concepts and process

As a relatively new field of researching interests, no universal standards in terminologies as well as designing processes have been commonly accepted. Basically, an object can be seen as an abstract representation of a real world material. Each object is formally defined by specifying its state and behavior through a set of encapsulated properties (attributes) and operations (methods). Objects with the same attributes and operations are grouped to form a class. There are several kinds of relationships can be defined among classes including generalization, aggregation, and association (reference). In the case of generalization/specialization relationship, several similar but mutually exclusive specific classes inherit common attributes and operations from a generic class. This type of relationship is also known as an inheritance or is-a relationship. Classes in both ends of this relationship are termed superclass and subclass. A class may consist of attributes and operations collected from several other classes. This type of relationship reveals an aggregation of a group of component classes to form a composite class and is also known as a has-a or part-of relationship. Both generalization and aggregation relationships may stretch to multiple levels. When a class refers to properties and/or activate operations of another class independent to it, this establishes an association, or a reference, relationship. Methods can be invoked by passing messages between classes. Contents of a message include specifications of the target class, the chosen method defined in the target class, and the parameters for executing this chosen method.

For effectively using the object-oriented modeling concepts to construct a
class hierarchy for representing structured contents as well as layout formats of the multimedia documents, we propose a conceptual design process as follows:

(1) identifying objects from contents of a document,
(2) specifying attributes and operations for all objects,
(3) identifying classes and relationships,
(4) building a hierarchical class structure for modeling document contents,
(5) identifying layout control objects,
(6) as in (2)-(4), building a class structure for document presentation,
(7) integrating class structures created in steps (4) and (6) to form an uniform conceptual data model for multimedia documents.

Following this process, an object-oriented multimedia document model for representing example views of a DTP page and an electronic publication page illustrated in Figure 1 (a) and (b) respectively is created and shown in Figure 2. For simplicity, only the class hierarchy related to the Main Body class is extended and displayed in this model. Levels of hierarchy in this model can go deeper if more detailed decompositions of page contents are desired. Attributes and operations for major related classes are described below.

Class Document
Attributes: document ID, doc. title, subject, author, publishing dept., publication date, version, number of pages, keywords, security level, languages, user list,
Methods: create, view, modify, save, delete, copy, print preview, print, send,

Class Author
Attributes: author ID, name, photo, signature, biography, publications,
Methods: show author information, show publications, print info.,

Class Main Body
Attributes: number of chapters, number of illustrations, chapter IDs, illustration IDs,
Methods: insert illustrations, merge files, show chapter list, show illustration list,
Figure 1. Example page views for printed and electronic documents
Figure 2. An Object-Oriented Multimedia Document Model
Class Chapters
Attributes: chapter ID, chapter title, number of pages, chapter file name, number of sections, section IDs.
Methods: edit, view, save, delete, print, show section list,

Class Illustrations
Attributes: illustration ID, caption, type, description, characteristics, illustration file name, index for retrieval.
Methods: edit, view, save, delete, print, import, export,

Class Sections
Attributes: section ID, section title, number of paragraphs, paragraph IDs.
Methods: edit, view, delete, print,

Class Paragraphs
Attributes: paragraph ID, number of lines,
Methods: edit, view, delete, print,

Class Images
Attributes: size, color status, file format, number of segments, segment IDs,
Methods: scan, decompose, assemble,

Class Voices
Attributes: time, volume, tone, file format,
Methods: record, tune, play,

Class Video
Attributes: number of frames, speed, file format,
Methods: Capture, grasp, rewind, play,

Class Blocks
Attributes: block ID, block size, block type,
Methods: generate, duplicate, place, move, clear, change, hide, show,

Class Text Blocks
Attributes: number of paragraphs, paragraph IDs, font type ID,
Designing a Multimedia Database for Electronic Publication Systems

Methods: fill, copy, cut, paste.

Class Byte Blocks
Attributes: illustration ID,
Methods: insert, copy, cut, paste,

Class Buttons
Attributes: button symbol, button description, target object ID,
Methods: link, activate,

Class Fonts
Attributes: font type ID, font type, points, effects,
Methods: choose, show font list, show font scale units, show font effects,

Class Frames
Attributes: frame ID, frame type, number of blocks, block IDs, block positions, number of embedded frames, frame IDs, frame positions,
Methods: create, group, place, hide, show,

Class Printed Pages
Attributes: page number, styles ID, number of blocks, block IDs, block positions,
Methods: chose printing styles, print preview, print,

Class Electronic Pages
Attributes: number of blocks, number of frames, block IDs, block positions, frame positions,
Methods: setup page buttons,

Class Printing Styles
Attributes: styles ID, margins, spacing, orientation, indent, justification,
Methods: setup printing styles, apply, modify styles,

Class Pages
Attributes: page ID, page type,
Methods: setup page, create page, view page.
2. Extended relational modeling with BLOB

BLOB is a data type created for the relational model to accommodate multimedia data within the same database structure. Data items in a relation can be defined to be either text BLOBs or byte BLOBs to represent diverse binary data stream in which any complex object such as a photo, a graph, a spreadsheet, a table, or a digitized voice can be contained. In addition to defining a BLOB typed information object as an entire unit, a binary large object can by itself be divided into detailed binary objects. These segmented objects can then be organized and defined into separate but related files. The degree of decomposition depends on the needs of how precise the objects are to be referred individually. Several types of queries that may interest users include:

— to retrieve and view document with criteria setting on alphanumeric attributes,
— to retrieve and view document with criteria setting on binary objects or on characteristics of binary objects,
— to specify some criteria and format to obtain multiple types of related information objects for ad hoc document generation.

Multimedia objects and associated text data can often be stored separately on different portions of a disk or on different mass storage devices for efficient data processing. When this happens, the actual value of the BLOB typed data item can be assigned to an index key pointing to the storage location of the object file. This extension to the relational model aims at supporting definition and manipulation of multimedia information using relational DBMS with BLOB capabilities. Conditions and rules for converting an object-oriented document model to the extended relational model are described in the following four sets. The resulting relational file structure is shown in Figure 3 (redundant relationships are included to indicate more possibilities of references).

Set 1: Directly map a class into a relation with an uniquely identifiable primary key item. In this relation, attributes having complex binary types are defined to be either Text BLOBs or Byte BLOBs. Examples can be seen as bold fields in Author, Illustrations, and Buttons relations shown in figure 3.

Set 2: For classes binded by a generalization relationship, add a field representing types of subclasses to the mapped superclass relation for distinguishing
Designing a Multimedia Database for Electronic Publication Systems

Figure 3. An extended relational model for multimedia documents
among subclasses, and then assign a common primary key to all the mapped superclass as well as subclass relations for defining an one-to-one relationship. Examples are shown among Illustrations and Images, Voices, Video relations.

Set 3: As for aggregation, there may be one-to-one, one-to-many, and many-to-many relationships between composite and component objects. Leave the first two types of relationships as the same by setting the primary key of the mapped composite relation as a common partial key to every mapped component relations. Upon the many-to-many relationship, insert a new connector typed relation with a composite key in which primary keys from both the mapped composite and component relations are grouped to breakup the many-to-many relationship into 2 one-to-many relationships. Examples include Printed Pages, Printed Page Blocks, and Blocks relations as shown in figure 3.

Set 4: The reference relationship can be easily translated to one-to-one or one-to-many relationships according to its nature. A common key can be assigned to the mapped associated relations in a similar way as stated in Set 2 and Set 3 to be a foreign key or part of a composite key. Examples include Document, Doc Author, Author relations and Buttons, Blocks relations as shown in figure 3.

III. A SOFTWARE INTEGRATION APPROACH FOR DOCUMENT MANAGEMENT

There are a few commonly used approaches for integrating different software functions. Basing on the levels of integration, we can classify these approaches into the following four categories:

(1) Compatible integration — In this style of integration, a group of compatible stand-alone softwares are linked by providing among them some commom data interchange formats and predefined translation procedures. Such exchange facilities resided in different softwares are responsible for performing transformation between specific software-dependent file formats. For instance, Aldus’ PageMaker provides Import/Export filters for converting text files in different formats such as ASCII, DOC, WK1, DBF, etc. to and from popular softwares such as MS Word (Microsoft), Lotus 1-2-3 (Lotus), and dBASE IV (Ashton-Tate). Also provided is a conversion routine to translate graphics files for file formats like PCX,
PICT, EPS, BMP, and TIFF.

(2) Multi-tasking (surface) integration — In this category, a graphical-oriented windowing system software is used as a multi-tasking operating environment to support linking of different software applications through a common data structure for exchanging intermediate data files. Various applications from diverse softwares can be selected and assigned to different windows on the screen. Users can switch back and forth among these windows. But, at any moment, only one application is actually in action. IBM’s Presentation Manager, Apple’s Mac System, and Microsoft’s Windows are belonging to this category. Object Linking & Embedding (OLE) and Dynamic Data Exchange (DDE) are key features provided in the Windows system for the purpose of facilitating software integration.

(3) Packaged (inclusive) integration — In this case, different software functions are combined into an all-in-one package. Separate programs can be coordinated directly, or indirectly at the same level via interfaces. Lotus’s Symphony, Ashton-Tate’s Framework are among those that fall into this category. Spreadsheet, data management, word processing, and graphics editing are major functions often packed into these integrated softwares.

(4) Synergistic integration — Multiple software functions are accommodated into separate but coordinated components. In any one of these components, functions from other components can be referred in a single operation without actually going into these components. Micro Data Base Systems’ Guru which, as an example, integrates word processor, graphic generator, report writer, data base management system, inference engine, ... etc. into a synergistic software environment, is one of the kind.

As can be seen, the capabilities of the first integration approach are limited. It does not establish the environment for smooth software integration, but merely provide the file compatibilities. Only translation interfaces among a limited set of commercial softwares are provided. No control mechanism is available for the purpose of integrating various functions from different softwares. Users have to manually step in and out all the needed software environments and to run all needed functions independently. The second approach gives an open multi-tasking and windowing environment for software integration. But the real integrating power is restricted to the cut and paste function for moving information to the requesting applications. Users are themselves responsible for driving the integrating procedure by switching among separate windows to activate the desired functions. Protocols and interfaces for dynamically exchanging data among applications are receiving more and more attentions and are adopted by many vendors for upgrading their softwares. But
the environment as well as software dependencies still limit the availabilities and linkabilities of desired softwares. The packaged integration approach operated as a closed system, usually arranges software functions into a hierarchical structure. Users traverse the hierarchy to execute the desired functions. In most cases, various capabilities provided in different components of resultant all-in-one package are sacrificed comparing to that of stand-along softwares. Although the synergistic integration is an ideal approach for integrating multilpe software functions, there is still no available synergistic packages that are suitable for our needs to link multimedia database with DTP or electronic publication systems.

In our approach of software integration, DBMS and DTP softwares are linked by a string of low-level commands such as an extended batch file, a keyboard macro, or a dynamic link library script depending on the operating system environment. Open architecture, dynamic data exchange, automatic process execution, and user transparency are significant features to be taken into account. The system architecture for linkage is presented in Figure 4. In this set up, users interact with the system through a menu provided in the application interface. Picking an option item from the menu directs the Control module to load and execute the chosen application using the corresponding softwares. Database management system, desktop publishing or electronic publication packages, word processor (WP), graphics processor (GP), image processor (IP), and other data and document processing tools are component softwares integrated within this operating system environment. Data defined and created by DBMS are stored in the usual database. Textual, graphical, and image information generated by using WP, GP, and IP respectively or via the DTP process as well as audio and video information objects captured by audio and video recorders can be stored separately in magnetic and/or optical storage devices. All softwares have the right to access information objects using proper dynamic linking facilities with some compatible file formats.

Two directions of the linkage are possible in the proposed system environment. The first direction treats the DTP document as the desired multimedia report generated by executing a database query and information retrieval process. A DBMS procedure designed according to the related logical access mapping from the conceptual document data model is executed for processing the query and presenting the resulting document. The second direction reverses the process by mapping from page oriented DTP objects to DBMS processible relations and fields for defining a conceptual document database schema to represent both the logical structure
Figure 4. A system architecture for integrating DBMS softwares
and presentation layout and then to guide the creation of the physical document database. Processes with varying degrees of transparencies for operating these two directions are described below.

1. Document retrieval

In general, the process is directed by macro command operations. A relational DBMS, in this situation, acts as the front end system to run the associating database query and retrieval processes for supporting the document processing applications which are linked to or embedded in the system user interface environment. And then the retrieved information objects are stored in a working database for the back end DTP like systems to actually generate the desired document. Major steps to be carried out in this direction are as follows.

1. Start the application and invoke the system main menu for document management.
2. Pick the desired menu item for browsing and generating specific documents.
3. Choose a presentation format for the document generation.
4. The control module loads the DBMS and runs the related procedures.
5. Process the database query and retrieve data by choosing or entering proper parameters and commands.
6. Execute the corresponding program functions to open the desired document file for browsing or printing if the complete document has been saved and stored as a single file, and then, go to step 14. Otherwise, go to the next step.
7. Access values of the selected fields including the BLOB keys to retrieve all the information objects from relative data and object bases and then transfer these objects to the working database.
8. Generate the desired document for browsing and printing if the operating process contains some dynamic linking facilities that are capable of handling and assigning blocks for binary objects according to the chosen presentation format as well as pasting the retrieved objects to their assigned blocks in a page. Then go to step 14, otherwise, go to the next step.
9. Create and save only the document layout forms with text data and blank areas for binary objects, and then transfer files including the
Designing a Multimedia Database for Electronic Publication Systems

unfinished forms, all related binary objects, and a text describing the relationships between these objects and the forms to the working database.

10. Quit from the DBMS environment and turn the control back to the operating system commands.

11. Return to the system main menu or directly invoke a macros to activate the desired DTP functions.

12. Load and run the DTP software or other electronic publication softwares.

13. Retrieve all the related files from the working database, modify and compose pages of the desired document by using proper pointing devices or keyboard commands.

14. Browse or print the desired document and then return to the system main menu for processing other applications.

2 Document database definition

In this direction, DTP or electronic publication softwares are the front end process systems. The control steps are listed in the following.

1. Start the application and invoke the system main menu for creating and describing the document data.

2. The control module loads and runs the DTP or similar software.

3. Retrieve all the related text and object files and create the page.

4. Cut and save all decomposed objects as desired.

5. Record the file names, index keys with block ranges and positions of all objects as well as other related information of the page to text files, and then save them using some database interchangable file formats.

6. Transfer all saved files to the working database.

7. Exit the DTP environment.

8. Return to the system main menu or activate DBMS functions automatically.

9. Load and run the DBMS software.

10. Create the document database if it is not available.

11. Otherwise, import the text files from the working database and perform a proper conversion routine to automatically append the data to appropriate relations and fields of the existing document database.

12. Quit the DBMS environment and return to the system main menu for
processing other applications.

Technical problems of software integration and data linkage have been experimented with varying degrees of transparencies on different platforms including Microsoft Windows, and Macintosh system environments to validate the proposed architecture and processes. In the Windows environment, PageMaker (Aldus) and Toolbook (Asymetrix) are used as DTP and electronic publication systems while Object Manager and Data Manager (Raima) and Superbase 4 (Precision Software) are used as DBMSs. In the Mac system environment, linked softwares include Hypercard (Apple) and Supercard (Silicon Beach) as the publication softwares and 4th Dimension (Acius) as the DBMS.

IV. CONCLUSION

In this paper, we discussed the backgrounds and needs, the database modeling concepts and processes, and a software integration approach for integrating multimedia database management to electronic publishing systems.

Linking DBMS and DTP typed softwares to form an uniform document management environment facilitates the creating, modifying, retrieving, and reusing of heterogeneous features of information within the same database and allows for the designing and editing of documents containing multiple types of information objects. The integrated DTP plus DBMS environment is capable of defining and manipulating documents by page contents and layouts and can be used for printing and navigating multimedia documents.

Object-oriented and field-driven extended relational modeling approaches have been adopted and combined in a systematic way to design the conceptual data models for efficiently and flexibly representing multimedia documents and to design the implementing processes for effectively supporting the physical document database creation and operation. A software integration approach consisting of an architecture as well as two direction binding processes have also been proposed to establish the multimedia database operating environment for actually managing the printed and electronic documents.

Future extensions to this research will focus on the construction of a more functionally-complete prototyped open document processing and management environment in which sufficient modeling and operating tools as well as version control facilities can be provided.
REFERENCES


