БЪЛГАРСКА АКАДЕМИЯ НА НАУКИТЕ . BULGARIAN ACADEMY OF SCIENCES

ПРОБЛЕМИ НА ТЕХНИЧЕСКАТА КИБЕРНЕТИКА И РОБОТИКАТА, 49
PROBLEMS OF ENGINEERING CYBERNETICS AND ROBOTICS, 49

София . 2000 . Sofia

Multimedia Data Management – Characteristics and Requirements*

Danail Dochev, Irena Koprinska, Radoslav Pavlov

Institute of Information Technologies, 1113 Sofia

1. Introduction

Data management lies at the heart of a multimedia information system. The spatial, temporal, storage, retrieval, integration and presentation requirements of multimedia data differ significantly from those for the traditional data. Hence, the goal of the multimedia data management system is to allow efficient storage, manipulation using of multimedia data in all its varied forms.

2. Multimedia Data Types

2.1. Text

Text is the basic element of most multimedia titles. Text is of concern to the developer from two main standpoints. The first is the way in which text is presented to the user: It should be easy to read and well designed. This involves the consideration font, colour, and text size. Unless the application includes a great deal of reference material (such as electronic encyclopaedia or book adaptation), text should be kept to a minimum – the user will not read long displays of text. The text intended to instruct the user how to use the program should be brief.

The second main concern of the developers is what lies behind the text; that is, the interactive "links" that the user does not see but can activate to get to additional information. Hypertext is an important aspect in reference titles. Such words may be set off from other text so it is obvious to the reader that additional information is associated with those words. This requires marking those words. Some software supports "auto" hypertext, in which the user may click on any word in the display, and any associated information is automatically displayed.

* This investigation was funded by INCO-COPRNICUS project PL961060 ARCHIMED "Advanced Multimedia-Systems Architectures and Applications for Educational Telematics".

2.2. Still images

Perhaps the most important components of multimedia applications are still images. Visual representations are generally much more effective at conveying information than text. Still images are generated by the computer in two ways: bitmaps (or painting graphics), and as vector-drawn graphics. Bitmaps are used for photo-realistic images. Bitmapped graphics are commonly used in multimedia titles and provide exceptional detail, and the software for creating them is readily available to most users. Windows has a built-in program (called Paintbrush), though there are many more versatile packages for creating or modifying bitmapped graphics. Software such as Adobe PhotoShop and PhotoFinish provide very sophisticated tools for to create and edit bitmapped graphics. Using such software, one is able to add special effects, use various filtering techniques, modify hue and saturation, and convert images to many different formats. Most also provide scanning capabilities, the usual way in which photographs and other graphics are captured for the computer. Another way to create graphics is to capture and digitise images from videos using special video capture equipment.

Vector graphic store images as a set of instructions for re-creating the image as an object consisting of geometric elements such as lines, circles, arcs, and angles. These instructions require relatively little storage and result in smaller graphic file sizes. However, complex images may take a long time to display (and require a faster processor) because the image is actually reconstructed as it is being displayed, based on the instruction set. The real advantage with vector graphics, however, is that the same image may be resized, moved, or rotated while maintaining its original quality and proportions. This is particularly important for three-dimensional (3-D) graphics and is commonly used for maps (which may be readily enlarged to view more detail) and for CAD/CAM and design work, in which an object or structure can be rotated to display different views.

2.3. Video

The two main types of video used in multimedia are analogue and digital. Analogue video is recorded and stored on videocassettes and laser disks. Of all the multimedia elements, video places the highest performance demands on the computer and its memory. Standards and formats for digital text, imagery, and sound are well established and familiar. But video is the most recent addition to the elements of multimedia. Three broadcast and video standards and recording formats are in use around the world: NTSC, PAL, and SECAM. These standards and formats are not easily interchangeable. Multi-format VCRs can play back all three standards, but typically cannot dub from one standard to another. Dubbing between standards requires specialised equipment.

Video overlay boards can capture or digitise video frames, as well as play them back from analogue video sources. Many video boards also incorporate audio input and sound management. Digitised video playback is step toward fully integrating motion video and digital computer graphics. This playback is accomplished using special software to convert the video source material from its common analogue form to a digital form manageable by the end user's computer system.

Digital video is by far the most commonly used. However it requires various data compression and decompression techniques accomplished using either software or hardware. Software codes include Intel's Indeo, IBM's PhotoMotion, Cinepak, Quick Time and Video for Windows, and they result in different file formats. The major advantage of software-based video: it requires no special boards to play video. A disadvantage is that it does not provide the same quality as television or VCR video.

Animations and digital video are sequences of bitmapped graphic scenes (frames), rapidly played back. The most commonly used animation technique is to create a

series of images that are displayed in rapid succession. Such "frame-based" animation involves a different image, or frame, for each view and works like a filmstrip. Photographs that are displayed this way can give the appearance of movies, although full-motion video is of higher quality and is becoming increasingly common in multimedia applications. Another common type of animation is "cast-based" animation in which the background image remains the same, but individual objects appearing on that background are given "instructions" to move across the background. Both techniques are also called 2-D animation, because they involve the appearance of flat images moving on the screen. Software such as Animation Works Interactive and Autodesk Animator provide the developer with sophisticated tools to create impressive animations. Another type of animation frequently used for virtual realty is 3-D animation in which three-dimensional objects are created using a mathematical model. Thus, each object may be shown in various views, giving the user a realistic sense of a third dimension. In creating 3-D animation, first must to be created a "model". This involved drawing the object in several views according to specified coordinates along x, y, and z-axes. The model is then made to look more realistic by adding shading and "rendering" the image, which involves blending the background, model, light sources, and textures to make cohesive frame transitions.

Like 2-D cast-based animation, objects are given paths of movement, but 3-D animation differs greatly because objects may turn and tilt, retaining a three-dimensional look. In addition, each frame in between a major change in orientation does not need to be drawn. This difference is due to the mathematical model, which interpolates how the object should appear in various positions and orientations. The complexity of 3-D animation requires computers with fast CPU speed, especially for full-screen animation. Special software (such as Autodesk 3-D Studio) is required to create 3-D animation that provides modelling and rendering capabilities.

Digitising and storing a 10-second clip of full-motion video requires transfer of an enormous amount of data in a very short amount of time. Reproducing just one frame of digital component video at 24 bits requires almost 1 MB of computer data; 10 seconds of video fills 300 MB hard disk. A typical hard disk drives transfer data at only about 1 MB per second. This overwhelming technological bottleneck is currently being overcome by digital image compression techniques. Real-time video compression algorithms such as JPEG, MPEG, P*64, DVI, and C-Cube are now available to compress digital video information. Most hardware-based video compression is currently based on two industry standards, JPEG and MPEG. One of the main differences between the two standards is that JPEG compresses every frame in the video, whereas MPEG compresses only the changes that occur between those frames. MPEG provides much higher compression ratio (up to 200:1 versus 100:1 for JPEG), and the quality of video is far superior to that resulting from JPEG compression.

2.4. Audio

One of the most important components of multimedia is sound, whether it is in form of music, narration to accompany text or to explain content, or special sound effects to enhance the action being displayed on the screen. The most common type of sound files incorporated into a multimedia application is "digital audio", which is created by converting analogue sound (sound from microphone, a synthesiser, existing tape recordings, live radio and television broadcasts, popular CDs, etc.) using an analogue-to-digital converter (ADC). To play back this signal the computer's sound card translates the digital information back into analogue sound using a digital-to-analogue converter (DAC).

Digitised sound is sampled sound. Every nth fraction of a second, a sample of sound is taken and stored as digital information. How often the samples are taken is

the sampling rate, and the amount of information stored about each sample is the sample size. The three sampling frequencies most often used in multimedia are CD-Audio quality 44.1kHz, 22.05kHz, and 11.025kHz. Sample sizes are either 8 bits or 16 bits. The quality of the digital sound is related to the number of channels recorded, sampling size and sampling rate. The digital data represents the instantaneous amplitude of a sound at discrete slices of time. Because it is not device-dependent, digital audio sounds the same every time it is played.

As with graphics and animation, it is possible to purchase CD-ROMs with sound libraries containing music and special effects, or the user can record his own sound using a microphone, sound board, and special sound software such as Wave for Windows and SoundTrack. This software allows the user to record sound using different sampling size and rates, add special effects such as echo and fade, and mix multiple sound files such as voice and music. Most Macintosh sound-editing software will save files in .SND and .AIF formats, and most authoring systems will read these formats. In Windows, most editing software writes .WAV files. The Convert and WaveEdit utilities in Windows allow the user to read the Macintosh .AIF format.

MIDI (Musical Instrument Digital Interface) is a communications standard for electronic musical instruments and computers. It allows music and sound synthesisers from different manufacturers to communicate with each other by sending messages along cables connected to the devices. MIDI provides a protocol for passing detailed descriptions of a musical score, such as the notes, sequences of notes, and what instrument will play these notes. But MIDI data is not digitised sound – it is a shorthand representation of music stored in numeric form. A MIDI file is a list of time-stamped commands that are recordings of musical actions that, when sent to a MIDI playback device, results in sound. MIDI files use a different format (.MID) than wave (.WAV) files and require special editing software such as Studio 3.1.

MIDI has several advantages over digital audio:

– MIDI files are much more compact than digital audio files, and the size of a MIDI file is completely independent of playback quality. In general, MIDI files will be 200 to 1000 times smaller than CD-quality digital audio files. Because MIDI files are small, they don't take up as much RAM, disk space, and CPU resources. For example, 1 minute of high-quality music requires about 10MB of storage when saved in .WAV format and only about 15 KB when saved as a MIDI file.

– In some cases, MIDI files may sound better than digital audio files if the MIDI sound source is of the high quality.

– The length of a MIDI file can be changed (by varying its tempo) without changing the pitch of the music or degrading the audio quality. MIDI data is completely editable - right down to the level of an individual note. The smallest detail of a MIDI composition can be manipulated in ways that are impossible with digital audio.

MIDI has several disadvantages:

– Because MIDI data isn't sound, its playback will be accurate only if the MIDI playback devise is identical to the devise used for production.

– MIDI cannot easily be used to play spoken dialog, although expensive and technically tricky digital samples are available.

– The preparation and programming required for creating digital audio do not demand knowledge of music theory; working with MIDI data usually does require a modicum of familiarity with musical scores as well as audio production.

2.5. Composite Objects

Composite multimedia data are created by combining basic multimedia data types and other composite multimedia data. Types can be physically mixed together to form a new type or logically mixed. A physical mix defines in a new storage format, where

data such as audio and video intermix i.e. compound objects. A logical mix defines a new data type while retaining individual data types and storage formats i.e. complex objects. However, when played the executing methods would have to deliver the data in a synchronised fashion, making it appear as though the data is a composition. Composite data may also contain additional control information describing how the information should be rendered at the client.

The software used to tie together all multimedia components includes presentation and authoring/production software. Although each provides an environment best suited for certain types of multimedia development, the boundaries separating them are becoming increasingly vague as software packages become more comprehensive and powerful.

Presentation software (such as Compel and Astound) is useful in building slide shows for presentations, allowing being incorporated effective animation or video and sound. Many presentation packages allow to access data from a database or spreadsheet to produce effective charts and other diagrams used in the slide show and may provide the user with some amount of interactivity (such as going to an adjacent screen or accessing additional information). Software used to produce presentations is easy to learn and does not require the developer to program "code". However, it does not provide as much power and flexibility as authoring software and therefore is not as well suited to create a more complex multimedia data.

Authoring software is best suited for developing titles that allow the user to intricately control the navigation and information that is presented at any given time. This is much more complicated than a sequential slide show used in presentations. Most authoring software provides an easy-to-use screen design interface that allows the user to create objects such as buttons and graphics using a set of tools.

3. Characteristics of Multimedia Data

3.1. Temporality

Some multimedia data types such as video, audio, and animation sequences also have temporal requirements, which have implications on their storage, manipulation, and presentation. The problems become more acute when various data types from possibly disparate sources must be presented within or at a given time. Temporal structures dictate the temporal layout, orchestrating the data's presentation. Basic temporal structures produce serial and parallel (hierarchical model) presentations of data. The user can also define presentations by associating a presentation time and duration (timeline model) with each multimedia object, eliminating the requirement for temporal structures. However, the model is severely limited, since it can only define static presentations. Other approaches for temporal specification include scripting languages, specification languages, and extended programming languages.

While specification languages may be mapped to temporal relationships within a multimedia DBMS, it is not so easy to do the same with scripts and programs. Scripts and programs control the presentation of multimedia data, but without a representable temporal structure they cannot easily be queried, reused, or supported by the underlying components of the multimedia DBMS (such as the storage manager).

3.2. Spatiality

Similarly, images, graphics, and video data have spatial constraints in terms of their content. Usually, individual objects in an image or a video frame have some spatial relationships between them (on the left of, on the right of, next to etc.). Such relationships usually produce some constraints when searching for objects in a database. These

relationships organise the data's visual layout on a virtual page or medium. The virtual medium may exist across multiple machines. Within a spatial presentation, users can move around inside the boundaries defined by the virtual medium and move and restructure the data.

Spatial structures may include 3D or virtual environments. Spatial constraints control 3D-object movement and inter object spatial relationships. Special tools can define spatial relationships using graphical user interfaces. Instead of providing semantically rich spatial relationships, some systems support spatial grammars, which are closer to scripting languages. However, since these spatial relationships are not directly represented within database (but only within a script), they cannot be really queried.

3.3. Need for Storage Space and Fast Transmission

Huge volumes of data also characterise multimedia information. For instance, to store an uncompressed image of 1024:728 pixels at 24 bits per pixel requires a storage capacity of about 2 Mbytes. With a 20:1 compression ratio, the storage requirement could be reduced to about 0.1 Mbytes. A 10-minute sequence of the same image at 30 frames per second requires about 38 000 Mbytes of storage, reducible to about 380 Mbytes with a compression ratio of 100:1. The potential for huge volumes of data involved in multimedia information systems become apparent when you consider that a movie could run as long as two hours and a typical video repository would house thousands of movies.

3.4. Need for Content-Based Access

The representing multimedia information as pictures or image sequences poses some problems for information retrieval due to the limitations of textual descriptions of a multimedia experience and the massive information available from it. The potential information overload means that users may find it difficult to make precise requests during information retrieval. The limitations of textual descriptions also imply the need for content-based access to multimedia information. Users need multiple cues (such as shape, colour, and texture) that are relevant to the multimedia content.

3.5. Collaborative Support Environment

Another characteristic of multimedia information is that interaction involves long-duration operations (such as with video data), and sometimes, with more than a single user (as is typical in collaborative support environments). However, in collaborative environments, it is expected that most multimedia data are likely to be accessed in a read-only mode. This assumption can be used to facilitate the provision of concurrency control algorithms.

4. Requirements for Multimedia Data Management

The goal of a multimedia database management system (MMDBMS) is to provide a suitable environment for using and managing multimedia database information. Hence, it must include the traditional DBMS functions (e.g., database definition and creation, data retrieval, data access and organization, data independence, privacy, integration, integrity control, version control and concurrency support but applied to various multimedia data types.

The functional requirements imposed on a MMDBMS can be grouped into two categories [DATAPRO]: data representation requirements and data manipulation requirements.

4.1. Data Representation Requirements

– Support for Generalization/Specialization Hierarchy

A major requirement that is imposed on multimedia applications is the support for generalization/specialization hierarchy. This hierarchy is used to define the type, subtype, and instance relationships between the various entities. For example, all the documents could be grouped into a type called "document type". If a new document is created, then it is made an instance of this type. Certain documents could have some special properties. For example, a book document could have properties, which are different from a newspaper document. Therefore, the collection of books could be grouped into a type called "book document type". Since a book is also a document, the type "book document type" is made a subtype of the type "document type". Support for generalization/specification hierarchy also facilitates schema evolution. For example, one could create a new version of document schema where the text always precedes the illustrations in the body. Any document which is constructed according to the new schema is also an instance of the old schema.

– Attribute Specification

Support for specifying the properties of a document (such as its title, author, font size, etc.) should be provided. These properties are also called the attributes of a document.

– Specifications of Operations

Another requirement is the ability to specify the operations that can be performed on a multimedia document. For example, it should be possible to change the font size of the document, change the contents of a document, retrieve the contents of the document, etc.

– Support for Composite Objects

A major requirement for modeling multimedia applications is the support for composite objects. For example, a document may be composed of front matter, body, and back matter. The front matter may be in turn composed of cover page, abstract, preface, acknowledgments, and table of contents. The cover page may consist of title, authors, organization, date of publications, and sponsors, etc.

– Object Sharing

Object sharing is the capability for different documents to share parts of their contents. Such a capability is especially necessary for multimedia documents as the amounts of storage space required to store a document might be quite large. It should be possible to represent the fact that different documents share portions of their contents.

– Ordering of Documents

The presentation of the paragraphs, images, and drawings of a multimedia document could depend on the users accessing the document. Usually constraints are imposed on the presentation of the document.

– Support for Multimedia Data

This major requirement includes extensibility, where new multimedia devices as well as new functions on multimedia information can be incorporated with ease.

– Data Independence

Database and the management functions must be separated from the application programs.

4.2. Data Manipulation Requirements

– Integration and Integrity Control

Integration means ensuring that data items need not be duplicated during different program invocations requiring the data. Unlike in traditional DBMS, data dupli-

cation is not encouraged, especially in distributed MMDBMS, due to the huge data volumes. The client-server computing model, in which a server application services multiple client applications (with the clients and server residing in possibly different machines), has proven suitable. By the integrity control consistency of the data state from one transaction to another through constraints imposed on transactions are ensured.

– Concurrency Control

This requirement ensures multimedia database consistency through rules, which impose some form of execution order on concurrent transactions. In concurrency control, a transaction is defined as a sequence of instructions executed either completely or not at all. In the latter case, the database is restored to its previous state. Defining the appropriate granularity for concurrency is a problem in multimedia databases. While traditional databases use a single record (table) as the unit of concurrency, multimedia databases typically use a single object (or composite object) as the logical unit of access. Thus, the single multimedia object could form the unit of concurrency.

– Persistence

The requirement for persistence denotes the ability of data objects to survive through different transactions and program invocations. In order to achieve persistence, the simplest method is to store the multimedia files in some operating system files. However, the huge data volumes make this approach costly to implement. Moreover, the system also needs to store the multimedia meta-data and possibly composite multimedia objects. Thus, most MMDBMSs classify the data as either persistent or transient and store only persistent data after transaction update. Transient data are used only during program or transaction execution and are removed afterwards.

– Recovery

MMDBMS must provide facility for transactions to recover from failures. Methods ensuring that results of failed transactions do not affect the persistent data storage are necessary.

– Privacy

Unauthorized access and modifications of stored data should be restricted.

– Query Support

An important requirement is that the query mechanism is suited for multimedia data. A query selects a subset of the data objects based on the user's description using some form of query language of what data to access. A query usually involves various attributes, possibly keyword-based or content-oriented, and is usually interactive. Thus, functions for relevance feedback and query formulation, similarity (rather than exact) matches, and mechanisms for displaying ranked results are important in a MMDBMS.

– Version Control

Organization and management of different versions of persistent objects are required. Version control becomes important when a persistent multimedia object is updated or modified, as some applications might need to access previous sates of the object. A DBMS provides such access through versions of the persistent objects. For a MMDBMS the huge volumes of data reinforces the importance of efficiently organizing such versions. Moreover, the available storage might limit the provision of versions. In addition, version management may involve not only versions of single objects, but also versions of the complex objects that make up the multimedia database.

– Efficient Capture, Access and Presentation of Multimedia Data

Support for the allocation and de-allocation of pages on disk, movement of pages to and from disk, and management of indexes should be provided. Also, support for the capture and presentation of various types of multimedia data is essential.

– Data Availability

Data availability refers to the fact that a data object can be retrieved from alternative storage devices as well as from different portions of the same device. This technique is also called data replication and has received major focus by researchers and developers. Replicated data can ensure that the user will be able to retrieve the requested object despite of some storage medium failure since the same object can be found also from another repository (i.e., disk, tape, etc). So, the more the replicas of an object the less the probability of not being able to deliver the object to the user. However, there are two drawbacks:

First, replication in a high degree exhausts the storage capacity. Provided that the multimedia objects are, in general, large ones, it is concluded that a database designer has a trade-off on the number of the replicas vs. the storage space willing to waste in replicas. In addition, the designer must also decide where he is going to put the replicas i.e., in which tapes, disks, etc.

Second, there is the need to keep the replicas consistent. This means that any update of the object has to be mirrored on each of the replicas. This requires tracing of each the replicas and, in case of an update, the system should traverse all the media (i.e., tapes, and disks) in order to replace the old version of the object. Notice, that in the case in which some of the replicas reside in tapes, the task of updating is very time consuming due to the very high access times in the tertiary memory (as opposed to the secondary memory, i.e., magnetic disks, RAIDs, etc).

5. Data Models for Satisfying the Data Representation Requirements

5.1. Overview

Two basic types of data models were developed: traditional models (Date [6]) and semantic models (Hull and King [7]). The former includes the network, the hierarchical, and the relational models. The entity-relationship (ER), the functional, and the object oriented (OO) data models are included in the latter data model. While traditional data models have primarily been used to represent the database, semantic data models have been used to model operational and semantic data.

Recently a third type, called hyper-semantic, has been included in the classification of data models. These models not only include the constructs provided by semantic data models (e.g. inheritance, generalization, aggregation and composition) but also provide inference capabilities which are necessary to model knowledge-based applications (Potter and Trueblood [8]).

5.2. Two Basic Data Models

5.2.1. Relational Data Model

Relational data model was proposed originally by Godd (Godd [4]). At present, it is the most popular data model mainly due to its simple representation and well-defined theory. Such a model views the world as a set of relations. For example, Fig.1 shows information about employees and departments represented by two relations, EMP and DEPT. EMP has attributes SS#, NAME, SAL, and D# (for the social security numbers, names, salaries, and department numbers of employees) and DEPT has attributes D#, DNAME, and MGR (for the department numbers, department names, and managers of the departments). Each relation has a primary key which is used to uniquely identify the rows of the tables, e.g. SS# for EMP and D# for DEPT. Several relation operators have been defined for manipulating the relations, e.g. the operator SELECT selects tuples from a relation which satisfy a certain condition,

PROJECT projects a relation onto some of its attributes, JOIN joins two relations on some common attribute (i.e. EMP and DEPT can be joined on D# in order to get association between an employee name and his/her department name.

Relation EMP

SS	#NAME	SAL	D#
1	John	20 000	10
2	Mary	30 000	20
3	James	40 000	30

Relation DEPT

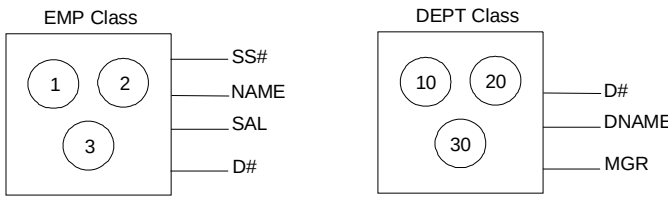
D#	DNAME	MGR
10	Personnel	Jane
20	Security	David
30	Finance	Tom

Fig. 1. Relational Representation

A lot of work has been done on developing a theory that guides the design of relational databases (Maier [7]). For example, various dependencies between the data attributes, such as functional and multi-valued dependencies have been defined. They are taken into consideration when the schema of a relational database is generated.

5.2.2. Object-Oriented Data Model

Semantic data models have been developed in order to overcome the major drawback of the relational models, namely their lack of support for complex objects. One such semantic data model that is gaining increasingly popularity is the OO data model. Unlike the relational data model, there is no standard OO data model and numerous such models have been proposed. In an OO data model, the world is viewed as a set of objects that communicate with each other by exchanging messages. Objects with similar properties are grouped together into class. Such objects are called instances of the class and the properties are called instance variables. Classes have methods associated with them. Methods are procedures that are executed when they are invoked. Methods are invoked when appropriate messages are received. An instance variable could be either a non-composite or composite instance variable. The former is further divided into Fig. 2.




```
UPDATE_SALARY (OLD, Amount);
SAL' = SAL + Amount
Write(OLD, SAL)
Return Status
Read(OLD, SAL)
E N D
```

Fig. 2. Object-Oriented Representation

simple and complex. Simple instance variables take individual objects as their values. An individual object could be a basic object such as an integer, a string, or a Boolean, or it could be an object described by a tuple value such as an employee, a department, or an automobile. For example, the simple instance variables of a book class include the "author" and the "Title", complex ones take a set or a list of individual objects as their names (e.g. set of names, set of employees). Composite instance variables describe the components of the instances, e.g. a document could be composed of a cover, table of contents, set of sections, and references.

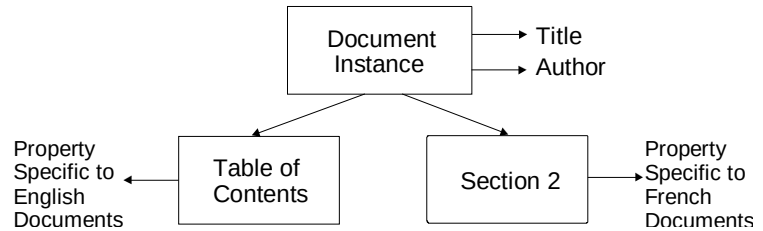


Fig. 3. IS-PART-OF Hierarchy

Any class, which has a composite instance variable, is a composite class. The instances belonging to such a class are composite objects. A composite object together with its components forms the IS-PART-OF hierarchy (Fig. 2). The link from a composite object to its component is called a composite link. A second basic hierarchy is the IS-A hierarchy (Fig. 3), where subclasses are associated with a class. The subclasses inherit all the methods and instance variables of their superclass. A subclass could also have additional instance variables and methods.

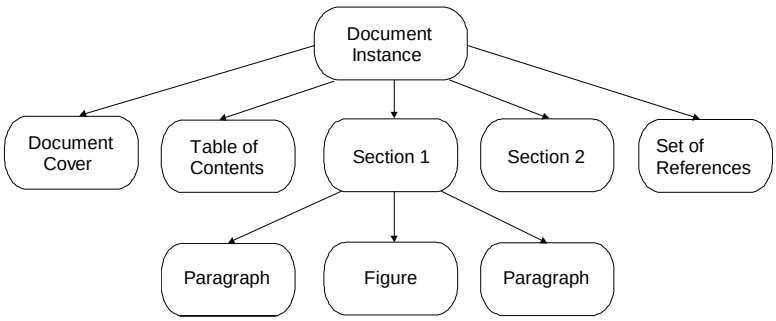


Fig. 4. IS-A Hierarchy

5.2.3. Comparison of the two models

The comparison between relational and OO data models is made with respect to both the representation and manipulation of multimedia documents.

The view of the world as a set of relations in relational model has shown to be useful for not only representation but also for manipulation. The theoretical foundations of the relational data model enable various well-defined operations to be performed on relations. As a result, the relational data model is ideal for many simple business applications. On the other hand, multimedia applications deal with complex structures. For example, the book cover, preface, introduction, various chapters, and references form the components of a book and cannot be treated as simple attributes of an entity. The book, consisting of these components, has to be collectively treated as a composite entity. It is not possible to represent composite entities using a relational model without placing a tremendous burden on the application program in order to maintain the complex structures.

Another disadvantage of the relational data model is the difficulty in specifying operations and constraints. For example, the operation of updating the contents of a document has to be embedded into an application program. Also, constraints are usu-

ally represented as rules and are not represented as relations.

Finally, the support of capturing, storing, and manipulating multimedia data has been found to be difficult using the relational data model. Multimedia data, such as video and voice, consume a large storage space. Storing them as flat files makes it difficult to efficiently access them. Also, the relational model cannot directly support specialization/generalization requirements. This makes schema evolution as well as incorporating a new device or a media difficult to support.

While the relational data model has obvious disadvantages for representing and manipulating complex structures for multimedia applications, the OO data model has constructs which support all of the requirements of multimedia applications. Such essential constructs are discussed below and it is shown how the requirements of multimedia applications can be supported.

- Class hierarchy/inheritance

The class hierarchy (or IS-A) provides the support for specialization/generalization hierarchies. For example, a class DOCUMENT that has all documents as its instances could be defined. A subclass of the DOCUMENT class could be the class JOURNAL. All journals are instances of this subclass. Inheritance mechanism enables a journal to inherit the properties of a document such as ID and Name.

- Instance variables

This feature allows specifying properties (or attributes). For example, a set of instance variables is associated with each class. They specify the properties of the instances of the class.

- Composite object hierarchy

The composite object hierarchy (also called Aggregate or IS-PART-OF hierarchy) enables composite objects to be represented.

- Methods

Methods are procedures and they are the only means of accessing objects. Executing appropriate methods can carry out operations on objects.

-Object sharing

Object sharing is supported by either the composite links or the instance variable links of two different objects pointing to another object. For example, the objects which represent documents 1 and 2 could have their instance variable links point to an object which represents the author. This means the author of the two documents is the same. However, as the two links point to the same object, the author's name does not have to be duplicated.

- Versioning

Version derivation hierarchy provides the support for representing versions and specifying operations on versions.

- Ordering of documents

Constraints on the structure and ordering of documents can be specified in the form of methods. Before a document is presented, appropriate methods are executed in order to present it in the desired form.

- Multimedia data

The class hierarchy and methods allow new kinds of data to be incorporated into

the system with ease. Furthermore, new operations on the data can also be specified. These features enable the efficient capture, storage, and presentation of multimedia documents. In addition, with the OO approach, large objects to store the large multimedia data could be created. Techniques are being developed for efficiently accessing these objects.

The discussion above shows that the OO data model has obvious advantages over the relational model for representing multimedia data.

5.3. Three Types of Multimedia DBMS Based on OO Style

5.3.1. Approach 1: RDBMS + OO Extensions (+ Multimedia Extensions)

This approach (Fig. 4a) uses a pre-existing relational DBMS (RDBMS) as basis and extends in into an OODBMS, and then further extends it into a MMDBMS. The advantage is that the core DBMS (which is relational) has been pre-developed and the design efforts can be reduced. The disadvantage is due to the mismatch between relational and OO styles. Clearly, the semantics of OODBMS is much more extensive than that of a RDBMS. This makes supporting an OODBMS with a RDBMS difficult and, from time to time, inefficient process. In other words, the system will have to live with the constraints and limitations introduced by RDBMS.

5.3.2. Approach 2: OODBMS (+ Multimedia Extensions)

This approach (Figure 4b) recognizes the problem mentioned above and redesigns an OODBMS from the beginning without relaying on pre-existing RDBMSs. However, the DBMS is still not designed for supporting multimedia information, and a multimedia extension is further introduced. Again, this system will suffer from the mismatch between multimedia data and conventional OODBMSs (which are mainly designed for non-multimedia applications).

5.3.3. Approach 3: OO Multimedia DBMS

The disadvantages mentioned above are overcome in this approach by the design of the entire multimedia OODBMS (Fig. 5c). Such a system significantly integrates the design of an OODBMS and the support of multimedia information in a single step such that all mismatches that arise in the first two approaches are eliminated. Most of the current approaches use either Approach 1 or Approach 2.

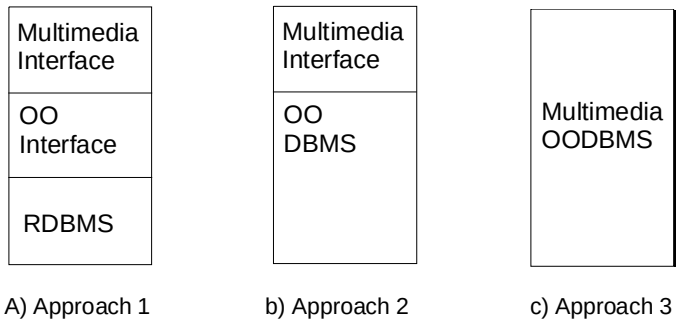


Fig. 5. Design Approaches of Multimedia DBMS

References

1. Ananger, G., T. D. C. Little. A Survey of Technologies for Parsing and Indexing Digital Video. — Journal of Visual Communication and Image Representation, **7**, March, 1995, No 1, 28-43.
2. Anjeroh, D. A., K. C. Nwosu. Multimedia database Management - Requiremets and Issues. — IEEE Multimedia, July-September, 1997, 24-33.
3. Ang, Y-H., A. D. Narasimhalu, S. Al-Hawamdeh. Image Information retrieval Systems. — In: C. H. Chen, L. F. Pau and P. S. P. Wang (eds.). Handbook of Pattern Recognition and Computer Vision. Singapore, World Scientific, 1993, 719-739.
4. Godd, E. A Relational Model for Large Shared Data Banks. — Communications of ACM, **13**, 1970, No 6.
5. DATAPRO Information Services Group. Developing Multimedia Database Management Systems, Workgroup Computing Series: Multimedia Solutions, 1031: Multimedia Concepts, November, 1992, 1-7.
6. Date, C. Introduction to Database Systems. Reading, MA, Addison Wesley, 1986.
7. Maier, D. The Theory of Relational Databases. Rockville, MD, Computer Science Press, 1983.
8. Potter, W., R. Trueblood. Traditional, Semantic and Hyper-Semantic Approaches to data Modeling. — IEEE Computer, **21**, 1988, No 6.
9. Rangan, P.V. H. M. Vin. Efficient Storage Techniques for Digital Continuous Multimedia. — IEEE Transactions on Knowledge and Data Engineering, **5**, August 1993, No 4, 564-573.

Управление мультимедийных данных — характеристики и требования

Данаил Дочев, Ирена Копринска, Радослав Павлов

Институт информационных технологий, 1113 София

(Резюме)

Анализируются мультимедийные данны разнообразного типа — текст, видео-, аудио-, и их специфические характеристики. Поставлены требования к системам их управления в двух основных направлений — к подходам их представления и их манипуляции. Анализируются преимущества и недостатки двух основных моделей — реляционного и объектно-ориентированного, а также и подходы их интегрирования.