Data Modeling of Time-Based Media

Simon Gibbs, Université de Genève

Christian Breiteneder, Universitaet Wien

Dennis Tsichritzis, Université de Genève and GMD Bonn

ABSTRACT Many aspects of time-based media – complex data encoding, compression, "quality factors," timing – appear problematic from a data modeling standpoint. This paper proposes *timed streams* as the basic abstraction for modeling time-based media. Several media-independent structuring mechanisms are introduced and a data model is presented which, rather than leaving the interpretation of multimedia data to applications, addresses the complex organization and relationships present in multimedia.

1 INTRODUCTION

There is a qualitative difference between time-based media and the forms of data traditionally stored in database systems. Time-based media, including digital audio and digital video, music and animation, involve notions of data flow, timing, temporal composition and synchronization. These notions are foreign to conventional data models and, as a result, conventional data models are not well suited to multimedia database systems in general.

Multimedia requires a broad perspective, one accounting for both time-based media and other forms of data. This paper takes a step in this direction by proposing a data model for time-based media. Its main contributions are the development of a model, which encompasses many forms of time-based media found in practice, and the identification of three general structuring mechanisms for time-based media.

1.1 Prior Work

There has been considerable prior work on modeling multimedia data (e.g., [2][4][8][13][21][22]) and a number of these proposals have been implemented in the context of multimedia document systems [2][4] or as multimedia extensions to existing database systems [21][22].

Much of this earlier work has focussed on text and images, while time-based media have received less attention – perhaps because of their tremendous processing and storage demands (for example, one *second* of high quality digital video can occupy tens of Mbytes). Now, however, advances in compression technology and the continually decreasing costs of memory and processing cycles are making the use of time-based media viable. As a result there is a need for multimedia databases coming from two new directions. First, new multimedia applications such as video on-demand services and virtual environments stand to benefit from access to large databases of time-based material. Second, the vast "clip media" repositories now being assembled are often loosely organized collections of files and lack the power and flexibility of databases.

1.2 From Blobs to Streams

Recent proposals for multimedia database systems have introduced a *BLOB* (binary large object) data type intended for images and other very large values (e.g., [3][7][16]). While the storage of very large values is necessary for multimedia databases, it is not sufficient. The database system should also have some understanding about the internal structure of BLOBs – it must be able to "interpret" the data. There are many reasons for this. First, if the database does not maintain this structural information then the task is left to applications. In other words, information about data structure is separated from the data itself – a situation database systems were explicitly designed to avoid. Second, the structural information needed to interpret time-based media is complex, if it is lost it may be extremely difficult or infeasible to reconstruct and one is left with meaningless data. Preserving this information is crucial and the task should not be left to applications. Third, knowing the structure of time-based media permits sophisticated querying. For example, consider a digital movie with audio tracks in different languages. If the movie is represented structurally, rather than as a long uninterpreted byte sequence, it is possible to issue queries which select a specific sound track, or select a specific duration, or perhaps retrieve frames at a specific visual fidelity. Fourth, presenting time-based media requires timing information. Using a BLOB data type it is possible to read and write time-based media but, since no timing information is available to the database system, the more relevant operations of "play" and "record" have no meaning. Finally, structural information is needed when updating time-based media. For example, editing systems for digital audio and digital video take great care to perform *non-destructive* modifications: rather than reading and writing vast amounts of data in order to accomplish a modification, references to structures within the data are manipulated. The

SIGMOD 94- 5/94 Minneapolis, Minnesota, USA
© 1994 ACM 0-89791-639-5/94/0005..$3.50

benefits of this approach include preserving the original material and supporting efficient modification operations.

To summarize, a key problem confronting multimedia database systems is the description of the structure of time-based media in a form appropriate for querying, updating and presentation. In other words, a data model for time-based media is needed. This paper argues that in designing such a data model it is necessary to go beyond the mere need to store large data values and consider the *stream-like* structure of time-based media.

1.3 Outline

The remainder of this paper is organized as follows: In the following section we provide some background material on time-based media. We briefly summarize relevant developments in the areas of standards and multimedia platforms, and then informally describe some of the novel data modeling issues pertaining to time-based media. Section 3 introduces the notion of timed streams and related concepts, this section also provides definitions for several commonly used terms including "continuous media" and "event-based media." Section 4 identifies three media-independent structuring mechanisms that apply to time-based media and multimedia in general. We conclude with discussions of related work and further directions.

2 OVERVIEW OF TIME-BASED MEDIA

2.1 Examples of Time-based Media

The capture and presentation of time-based media often requires special hardware and often utilizes compression techniques as a means to reduce storage and communication costs. Consequently most data representations for time-based media originate with two groups: manufacturers of platforms for multimedia applications and the developers of data compression standards. The following lists some of the more widely used representations that have recently emerged from these groups. Space does not permit detailed descriptions, for more information the reader can consult the indicated references.

Digital Video Interactive or *DVI* [17] refers to a hardware/software environment now licensed by Intel and intended mainly for PCs. DVI is based on two digital video formats: *Production-Level Video* (PLV) and *Real-Time Video* (RTV). PLV uses a proprietary compression algorithm allowing VHS quality video to be produced from a data rate of about 1 Mbit/sec. The RTV format results in data rates similar to those of PLV, however the video quality is poorer and the frame rate may be reduced. Applications can playback both the RTV and PLV formats, and record in the RTV format.

The *Moving Pictures Experts Group* (MPEG), an ISO working group, has developed two video compression standards. The first, MPEG I, like PLV, achieves data rates of about 1 Mbit/sec for VHS quality video [9]. The more recent MPEG II yields data rates of up to 10 Mbit/sec and is intended for near-broadcast quality video. The MPEG standards have broad industrial participation and products based on MPEG I have already been announced.

The *Joint Photographic Experts Group* (JPEG), another ISO working group, has developed an image compression standard intended for continuous-tone color and grayscale images [20]. JPEG is also used with digital video by treating each frame as an image to be compressed independently. Hardware implementations of JPEG are the basis of several computer-based digital video editing systems. For a given frame rate and resolution, JPEG-compressed video has a higher data rate than techniques such as MPEG and DVI's PLV which exploit similarities between frames; but, since frames are compressed independently, it is easier to rearrange the order of the frames and to playback in reverse or at variable rates.

Compact-Disc Interactive (CD-I) is a self-contained multimedia system developed by Philips and Sony [15]. CD-I specifies a data format allowing digital audio of various qualities, digital images of various resolutions, and arbitrary application data, to be stored on a CD. Support for MPEG I video has recently been introduced.

QuickTime [6] is an addition to the Macintosh operating system allowing real-time synchronized playback of digital audio and digital video from secondary storage. QuickTime is extensible, it allows new data representations and compression schemes to be supported through the addition of software "components." The initial release includes several components intended for both synthetic (computer-generated) images and natural (continuous-tone) images. Multiple tracks of digital audio and video data are combined in a *movie* file, which can then be played by QuickTime on the various Macintosh platforms (QuickTime players are also available for PCs).

Video for Windows (VfW) is similar in functionality to QuickTime but runs on PCs under Microsoft's Windows 3.1. The file format used by VfW is called *AVI*, or Audio Video Interleaved. Like QuickTime, VfW supports several compression methods, among these is *Indeo*, a software version of DVI.

MacroMind *Director* [12] is a multimedia authoring application for the Macintosh. Director is based on a multi-track data structure (also called a movie) where individual tracks may contain graphic objects, audio and MIDI[1] events, timing information, and interactive scripts. The main tool provided by Director is the "score editor." It is used to specify track contents, how multiple graphics tracks are to be combined and overlaid, and the timing constraints between tracks. A second tool, a "movie player" is used to playback and interact with Director movies. Versions of the player tool run on the Macintosh and on the PC under Windows 3.1 [14].

2.2 Modeling Issues

The basic abstractions we propose for modeling time-based media are *timed streams* of media elements. The term "media element" includes such things as video frames, audio samples and musical notes (more precise definitions are given in the next section). During playback and recording, elements are read and written not in isolation but as synchronized sequences–these are what we call timed streams. Recording and playback involve continuous access to timed streams and it is of-

1. MIDI, or *Musical Instrument Digital Interface*, is a standard for communicating with musical devices.

ten necessary to sustain specific data rates (e.g., so many frames per second). Satisfying this requirement has many implications for database architecture and storage management but these topics are beyond the scope of this paper. Here we focus on logical aspects of time-based media. Our goal is to identify the characteristics of time-based media differentiating it from other forms of data. Among the more unique and unusual characteristics of time-based media are:

Interpretation If a database system knows that a unit of storage represents an integer, then interpretation of the data, i.e., determining which integer it represents, is considered a trivial problem. This is the case with traditional data types in general, their interpretation is not difficult and is largely taken for granted by data modeling. The same does not hold true with time-based media. In this case the storage units requiring interpretation represent timed streams. Factors that complicate interpretation include:

- heterogeneity – media elements within a stream can differ in many ways. For example, consider a stream containing a sequence of compressed digital images. Among the factors which may differ from image to image are image size, image width and height, image depth[1], color model (e.g., RGB[2] or YUV[3]), and the type of compression used.
- interleaving – in order to simplify synchronization of streams during playback, their elements may be interleaved in a single storage unit. Many of the formats discussed in section 2.1 allow interleaving.
- padding – storage units may be padded with unused data to match storage transfer rates to media data rates. This is commonly used in CD-I for example.
- out-of-order elements – some compression techniques, such as MPEG, exploit similarities between consecutive elements. "Key" elements are identified from which intermediate elements can be constructed by interpolation. Because key elements are needed at an early stage during decoding, they may be placed in storage units prior to the intermediate elements. For example, with a sequence of four elements where the first and last are "keys," the placement order could be 1, 4, 2, 3.
- scalability – certain representations for time-based media, in particular proposals for digital video [10], allow presentation at different levels of detail. For instance, a digital video sequence recorded at very high resolution may be presented in an environment requiring, or only capable of, much lower resolution. In such cases, bandwidth can be saved and processing reduced if the video sequence is "scaled" to a lower resolution by ignoring parts of the storage unit.

A data model for time-based media should provide constructs for timed streams. These constructs should abstract away information concerning the physical organization of media elements, for instance their physical ordering and placement, while permitting heterogeneity, interleaving, and padding etc.

Quality Factors Video compression algorithms such as JPEG, MPEG and those used in DVI and QuickTime are *lossy*: encoding followed by decoding is not an identity transformation. Loss of information can be thought of as a reduction in image quality. The amount of information lost can be controlled by selection of various numeric parameters used during encoding. However, these parameters should not be visible at the data modeling level. For instance, when specifying a video-valued attribute it may be useful to specify a video quality for the attribute, but it should not be necessary to do so in terms of low-level compression parameters. Instead video quality (and the same applies for audio quality) should be specified via descriptive *quality factors*. For example a particular video-valued attribute might be of "broadcast quality" or "VHS quality."

Timing We make two general observations concerning the role of time in time-based media: First, the handling (retrieval, storage, and processing) of media elements is subject to *real-time constraints*. For instance, video frames are presented at specific frame rates, audio samples at specific sample rates (the rates are determined when the elements are captured or generated). Second, *temporal correlations* can occur between media elements. It is often the case, for instance, that audio elements must be synchronized with visual elements. Satisfaction of real-time constraints, and assuring that temporal correlations are observed, is a performance and implementation issue rather than a data modeling issue. What is important in modeling time-based media is the ability to specify the real-time constraints and temporal correlations. In other words, the data model must address the *timing* of media elements.

Derivation Media objects, such as audio and video sequences, are often defined in terms of other objects, and then generated when needed rather than stored explicitly. A simple example is a video transition such as a "wipe," used when one scene ends and its image is gradually wiped away to reveal the following scene. In this case there are three sequences: the first scene, the second scene, and the wipe transition. Note that the wipe is *derived* from the scene sequences, all that is needed to recreate the wipe is the two original sequences plus various parameters describing when and how the wipe takes place. A data model for time-based media should represent the relationships between derived and non-derived media objects. It should be possible to a) store derived media objects in an implicit form, and b) to "expand" derived objects to produce actual (i.e., non-derived) objects. The decision of whether to store a derived object or to expand and instead store a non-derived object often hinges upon resource availability: if expansion can be done in real time then the derived object is all that needs be stored.

Composition An essential feature of multimedia is the mixing of sound and imagery – television and films are two obvious examples, each containing both audible and visual components. Looking at multimedia in general one finds that complex multimedia structures are built up from simpler, perhaps "single-media," components. The components are combined using various *composition* mechanisms which establish

1. The number of bits-per-pixel.
2. Colors are represented by red/green/blue intensities.
3. Colors are represented by a luminance (Y) and two chrominance (UV) values.

relationships between components; these often are spatial relationships (e.g., relative positioning during presentation) or temporal relationships (e.g., relative timing during presentation). A data model for time-based media, or multimedia in general, must be capable of representing the variety of relationships that occur when combining media.

3 BASIC CONCEPTS

The previous section introduced and motivated what we believe to be the important issues in modeling time-based media. We now present a data model that incorporates many of the notions we have just introduced. First we give a series of definitions that make more precise the notion of timed streams.

3.1 Media, Artifacts and Media Objects

The term *media* has a rich set of connotations. Often, it is related to how information is conveyed and distributed; for instance we have print and broadcast media. The term is also used to describe the materials and forms of artistic expression. This last meaning occurs when we speak of *digital media*, not in the sense of digital storage media, but in the sense of digital counterparts to *natural media*. The distinction between natural and digital media may not always be clear-cut but the idea is that natural media rely on physical elements–paper, stone, inks and paints–while digital media rely on the computer. Suppose we call *artifacts* the objects produced in a particular medium. Prints, paintings, musical performances and recordings, films and video clips are all artifacts. We use the term *media object* to refer to machine readable representations of artifacts. For instance, media objects corresponding to the artifacts just mentioned consist of digital images, digital audio recordings and digital video recordings.

Media objects represent artifacts from both natural and digital media. As an example, digital images produced by scanning photographic prints or hand drawings are representations of natural artifacts, while digital images produced by a computer paint program are representations of digital artifacts.

3.2 The Structure of Media Objects

Media objects are generally highly structured aggregates of simpler objects, often called *media elements*. Graphics objects are good examples. Complex 3D scenes contain hierarchies of graphics elements representing geometric information, shading, and lighting.

The important question from a data modeling perspective is not so much what is the structure of media objects, since this is determined by applications and standards, but what part of this structure should be captured by a database schema. The minimum a database system should know about media objects includes their type (e.g., image, audio) and encoding attributes that vary from type to type. We call such information a *media descriptor*. For example, an image descriptor includes the image width and height while an audio descriptor includes the sample size and rate.

Looking at the data representations for media objects of various types, one very general observation can be made: audio, video and (many, but not all, representations of) animation and music use a similar manner of combining media elements. In each case elements are arranged in temporal sequences. Video is a temporal sequence of frames, audio of samples, music of notes, and animation of scenes or events. Since temporal sequences of media elements are a media-independent construct, we propose such temporal sequences, what we call timed streams, as the central abstraction for modeling time-based media.

3.3 Media Types, Discrete Time Systems and Timed Streams

To more precisely define timed streams, we first introduce two other notions: media types and discrete time systems.

Definition 1 A *media type* is a specification of the attributes found in media descriptors and their possible values. For time-based media, a media type also specifies the form of *element descriptors* (these refer to individual elements rather than media objects as a whole).

As an example, the media type for CD audio would specify a sampling rate of 44.1 kHz, a sample size of 16 bits and the number of channels as 2. These attributes, together with a duration, would be included in the media descriptor. In this case element descriptors are not necessary since all elements have the same form (16 bit PCM[1] samples). Now consider ADPCM[2]-encoded audio. Some versions of this compression technique involve a set of encoding parameters that vary over an audio sequence. These parameters would be part of element descriptors.

Definition 2 A *discrete time system*, D_f, is a mapping from integers to real numbers. Members of the domain of D_f are called *discrete time values*, members of the range are called *continuous time values* and measure time in seconds (or some other unit). The mapping is of the form D_f: $i \rightarrow (1/f)i$, where f is called the *frequency* of the time system.

Some examples are $D_{29.97}$, suitable for North American video (i.e., where video frames occur at a rate of 29.97 times per second), D_{25} for European video, D_{24} for film, and D_{44100} for CD audio.

Definition 3 A *timed stream* is a finite sequence of tuples of the form: $<e_i, s_i, d_i>$, $i=1, \ldots n$. Each timed stream is based on a media type T and a discrete time system D. In particular, the e_i are media elements of T, and the s_i and d_i are discrete time values measured in system D. The value s_i is called the start time of e_i and d_i is its duration. Start times and durations satisfy: $s_{i+1} \geq s_i$ and $d_i \geq 0$.

Generally a media type imposes restrictions on the form of timed streams based on that type. For example the CD audio type specifies a sampling rate of 44.1 kHz. Given a timed stream based on this type and D_{44100} then we must have $s_{i+1} = s_i + d_i$ and $d_i = 1$ for all i. Constraints of this form are common for time-based media and give rise to several categories of timed streams (such as depicted in Figure 1):

1. Pulse Code Modulation (PCM), a simple encoding scheme for sample data.
2. Adaptive Differential Pulse Code Modulation (ADPCM), a form of audio compression used in CD-I and other multimedia environments.

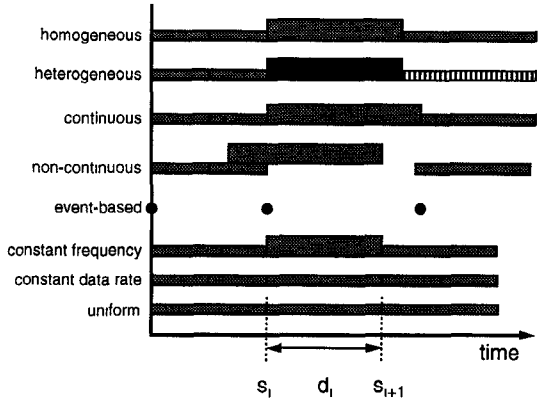

Figure 1 Examples of timed streams for different forms of time-based media. Media elements are represented by rectangles (or dots in the case of event-based streams). Four pieces of information are shown for each element: its start time and duration, size (area of rectangle), and descriptor (shading of rectangle).

- *homogeneous* streams: element descriptors are constant
  An example is CD audio, where all elements have the same form.
- *heterogeneous* streams: element descriptors vary
  Examples include compressed formats where encoding parameters may vary over the course of a recording.
- *continuous* streams: $s_{i+1} = s_i + d_i$, for $i = 1, \ldots n\text{-}1$
  In this case there is a unique element for every time value within the span of the stream (i.e., for all times t: $s_1 \le t \le s_n + d_n$). Digital audio and digital video are common examples of continuous media.
- *non-continuous* streams: $s_{i+1} \neq s_i + d_i$, for some i
  In this case there are "gaps" and/or "overlaps" among elements. Music and animation are examples. For instance, consider animation represented by sequences of elements specifying movement. At times when the animated object is at rest there are no associated media elements. Another example is a representation for music where media elements correspond to notes being produced. A chord would then require overlapping elements.
- *event-based* streams: $d_i = 0$, for $i = 1, \ldots n$
  A special case of non-continuous streams occurs when media elements are duration-less "events." An example is MIDI where elements are musical events of the form "Start Note X" and "Stop Note Y."
- *constant frequency* streams: continuous and element duration is constant
  In this common case the durations of media elements are constant, examples include fixed-frame-rate digital video and many digital audio representations.
- *constant data rate* streams: continuous and the ratio of element size to duration is constant
  Some timed streams have a constant data rate (the data rate is the ratio of size to duration of media elements).
- *uniform* streams: continuous and both element size and duration are constant
  This is a common subcase of constant data rate streams. Examples include many "raw" (uncompressed) digital audio and digital video representations.

The above categories of timed streams both encompass, and differentiate between, a wide range of representations for audio, video, music and animation. In this sense they also provide useful categories of time-based media. For example, one can say that audio and video are continuous media since representations for audio and video are generally continuous streams. Other categories than those shown in Figure 1 are certainly possible, but appear to be less common and so less useful.

4 STRUCTURING MECHANISMS

Suppose we can construct multimedia objects using attributes that take media objects as their values. For instance, a VideoClip object could possess, in addition to character-valued attributes such as the title and name of the director, a video-valued attribute containing the actual content of a video clip. Clearly VideoClip objects can be related in many ways. Two may have the same director, or concern the same topic, or one may be the sequel of, or a shorter version of, the other. These relationships are domain-specific. Other relationships are generic, they represent basic ways for structuring media objects. These relationships include *interpretation*, *derivation* and *composition*.

4.1 Interpretation

Depending upon the operation being performed, time-based media, and multimedia data in general, can be viewed at two levels. First, with low-level operations, for instance copying and deleting, there is no need to be aware of internal organization and the data can be viewed as an unstructured BLOB. However for high-level media-specific operations, such as presentation and editing, the structure of the data is important and must be visible. In this case the data is viewed as an intricately structured aggregate of media elements. Interpretation is the link between these two views. First suppose we define BLOBs as follows:

Definition 4 A *BLOB* is an attribute value that appears to applications as a sequence of bytes. The database system provides an interface by which applications can read and append data to BLOBs. (The database system may also support insertion and deletion of byte spans, however for time-based media these operations are not essential since non-destructive editing techniques are often used.)

Generally BLOBs can be very large in size (i.e., many Mbytes) and require special support by database systems. A BLOB may correspond to a region of contiguous storage or it may be fragmented, the layout of BLOBs is a performance issue and not directly relevant to data modeling. What is relevant to data modeling is the relationship between media streams and BLOBs:

Definition 5 An *interpretation*, I, of a BLOB B, is a mapping from B to a set of media objects. For each object, I specifies the object's descriptor and its placement in B. If the ob-

ject is a media sequence then for each media element *I* specifies the elements's order within the sequence, its start time, duration and element descriptor.

Example of Interpretation

As a specific example of interpretation, suppose a 10 minute PAL[1] video signal and an accompanying stereo audio signal are digitized and stored in a BLOB. PAL video frames occur 25 times per second. Assume frames are sampled at a resolution of 640 x 480 pixels, each sample providing 24 bits of RGB data. The RGB values are then converted to YUV, Y is given 8 bits per pixel, U and V are subsampled (averaged over neighboring pixels) and each is given 2 bits per pixel. There are now 12 bits per pixel. The YUV frames are then JPEG compressed using a quality factor resulting in about 0.5 bits per pixel (this will give VHS quality). Thus the original video data rate has been reduced from about 22 Mbyte/sec to roughly 0.5 Mbyte/sec. The audio data is sampled at a 44100 Hz with a sample size of 16 bits. Since there are two channels the audio data rate is 172 kbyte/sec. The audio and video data are interleaved in a single BLOB with audio samples following the associated video frame. Suppose we call the two media objects $video_1$ and $audio_1$, their interpretation is depicted in Figure 2 .

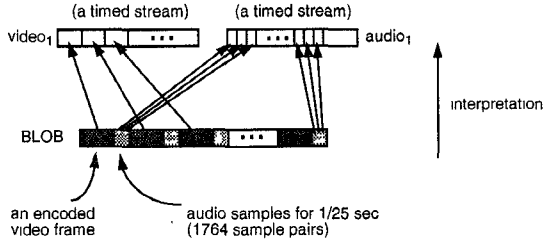

Figure 2 An interpretation of a BLOB.

The interpretation specifies the media descriptors for $video_1$ and $audio_1$, these would resemble:

$video_1$ descriptor = {
  category = homogeneous, constant frequency
  quality factor= "VHS quality"
  duration = 10 minutes
  frame rate = 25
  frame width = 640
  frame height = 480
  frame depth = 24
  color model = RGB
  encoding = YUV 8:2:2, JPEG }

$audio_1$ descriptor = {
  category = homogeneous, uniform
  quality factor = "CD quality"
  duration = 10 minutes
  sample rate = 44100
  sample size =16
  number of channels = 2
  encoding = PCM }

The descriptors should also contain information that helps allocate resources for playback, this could include the average data rate for each stream, a measure of data rate variation (for non-uniform streams), and any parameters needed for decompression.

In addition to the media descriptors, the interpretation must identify element placement, timing, and element descriptors. Both $video_1$ and $audio_1$ are homogeneous streams–elements are of the same form and element descriptor attributes are subsumed by the media descriptors. However the encoded video frames are variable sized. This means that for $video_1$ the mapping from element number to BLOB placement is not a simple multiplication; the same holds for $audio_1$ because of interleaving. As a result an explicit mapping is needed, as represented by the following tables:

$video_1$(elementNumber, elementSize, blobPlacement)
$audio_1$(elementNumber, blobPlacement)

These tables have one entry for each element, i.e., for each frame of $video_1$ and each sample of $audio_1$. The tables are a logical view of the interpretation mapping–existing storage systems for time-based media use multiple index structures, allowing rapid lookup of the element occurring at a specific time and the clustering of elements for performance reasons. (For example, QuickTime uses up to seven indexes for a single timed stream.)

The indexes used to implement interpretation should not be visible to applications, what needs be visible are the results of interpretation–the media elements and their descriptors. In other words, interpretation supports the timed stream abstraction by encapsulating information about the low-level encoding and BLOB placement of media elements.

The information needed for interpretation depends on the category of the stream, whether it is heterogeneous, homogeneous, continuous, etc. For instance, if $video_1$ were a heterogeneous and non-continuous video object, it would require a table of the form:

$video_1$(elementNumber, startTime, duration, elementDescriptor, elementSize, blobPlacement)

Definition 5 does not preclude a BLOB from having more than one interpretation. For instance, from the $video_1$ table, a second interpretation can be formed simply by removing table entries or changing their element number. The effect resembles video editing which involves cutting and reordering video sequences. Similarly, if an interpretation identifies many media objects within a BLOB, an alternative interpretation can be constructed by removing references to one of the objects from the original interpretation. The result is much like an alternative view of the BLOB (e.g., only the audio sequence is visible).

Generally modification of an interpretation is questionable (unless the underlying BLOB is also modified). The risk is that the original interpretation is destroyed in which case media elements within the BLOB may be effectively lost. It is probably a better practice if a BLOB has a single, complete, interpretation which is built up as the BLOB is captured or created and then permanently associated with the BLOB. Editing operations, and alternative views of the BLOB, can be achieved using derivation and composition.

4.2 Derivation

In data modeling, derivation refers to how the value of a specific item (an attribute, a set of attributes, an entity type etc.) can be computed from the values of other items. In general, the structure of the derived item is not the same as that of the items participating in the derivation. So, derivation in these cases can be considered as a simple viewing mechanism.

1. Phase Alternation Line (PAL) – a European video signal standard.

In the context of multimedia modeling, derivations play a prominent role and represent a very general and frequently used technique. Derivations reflect the various steps and decisions made by media designers on the way from data capture to a form of the media object ready for presentation. Information about the various production steps and their ordering are especially useful if earlier steps need to be repeated or undone. In general, rather than storing the results of derivations it is possible to store the specification of each derivation step. This notion of derivation has several advantages beyond data modeling: First, derivation reduces storage utilization by allowing alternative views to be constructed without the need for replication. Second, modification operations can be performed more efficiently. For example, to delete a video subsequence one could copy and reassemble the frame data, but it would be much more efficient to simply create a derivation representing the edit. Third, derivation provides physical data independence by separating derived objects from the underlying stored data.

We define derivation as a mapping from a set of media objects, and a set of parameters that govern the derivation, to a new media object.

Definition 6 The *derivation* (D) of a media object o_1 from a set of media objects O is a mapping of the form $D(O, P_D) \rightarrow o_1$, where P_D is the set of parameters specific to D. o_1 is called the *derived object*. The information needed to compute a derived object, references to the media objects and parameter values used, is called a *derivation object*.

The essential idea of this notion of derivation is the representation of media objects whose underlying media elements are calculated when needed. Typically, the media elements need only be stored if the calculation cannot be performed in real time (as when the time to calculate elements in a constant frequency stream is greater than their period).

The types of media objects participating in derivations are usually constrained. For example, an audio sequence cannot be concatenated to a video sequence. But there are also exceptions to this observation. As there exist many different forms of derivation, we group them into categories according to such general observations. Specific examples are given at the end of the subsection.

Derivations can be differentiated with respect to whether they change the content of a media object, its placement in time, whether they are elementary or compound, change the media type or not, allow elements of different types to participate in the derivation, are generic and apply to all or most media types or to one or a few types only. These groups of derivations are not exclusive and a derivation can appear in more than one group.

Derivations changing the content In general derivations in this group are specific to certain media types. Examples are digital filters for images, transposition of a music object to a different key, or changing the font or the font size of text. An example, in which two objects participate in the derivation is chroma keying of one video sequence over another. In this case, the content of the first video sequence is partially replaced with that of the second. Derivations that change content often require special hardware if they are to be performed in real time. Consider video fades and wipes. In video post-production, these derivations (known as "digital video effects") are calculated in real time by hardware that applies a function to "input" sequences and derives a new "output" sequence.

Derivations changing timing Derivations involving changes in timing are generic in the sense that they apply to all time-based media. For instance, temporally translating a sequence (i.e., uniformly incrementing element start times) can be performed on video sequences, audio sequences or any other time-based value. Another example is scaling (i.e., uniformly scaling element durations and start times).

Derivations changing the media type Synthesis is an important example of a derivation that changes the media type. Consider, for example, the synthesis of an audio object from a MIDI object, or the synthesis of a video object via rendering an animation sequence. Here the type changes from music to audio or animation to video. Other derivations result in a less radical change of type. Examples here include changing from one image encoding to another, or changing compression parameters.

Examples of Derivation

Up to now we have been discussing derivation very generally. This is due to the fact that derivations can be applied to various media types and have a rich variety of forms–a realistic enumeration of which would cover considerable space. In order to be more specific, Table 1 and Figure 3 illustrate some commonly used derivations. Argument and result types in the table refer to media types only (i.e., the table does not include information about the parameters of the derivations). Each derivation is also assigned to one of the categories identified above.

Color separation Printing a color image often requires a change in the color model as when images are converted from an RGB format to a CMYK[1] format. Since the mapping from RGB into the CMYK color model is not unique, additional information must be provided as parameters. In general this information is defined in *separation tables* which account for physical characteristics of inks and papers.

Audio normalization The enhancement of sound files with too little amplitude or uneven volume is done by a scaling operation called "normalization." The parameters needed are the start and end points of the audio sequence to be normalized. If no parameters are specified, normalization is performed for the whole audio object.

Video editing Editing video involves the selection and ordering of sequences that are combined to produce a new video object. The list of start and stop times of these selections is called an edit list. Edit lists are derivation objects, while edited video sequences are derived objects.

Video transition In video editing, instead of directly concatenating two video objects often an intermediate video effect is used, as for example, a fade or wipe. These transitions pro-

1. Colors are separated by cyan (C), magenta (M), yellow (Y) and black (K) intensities.

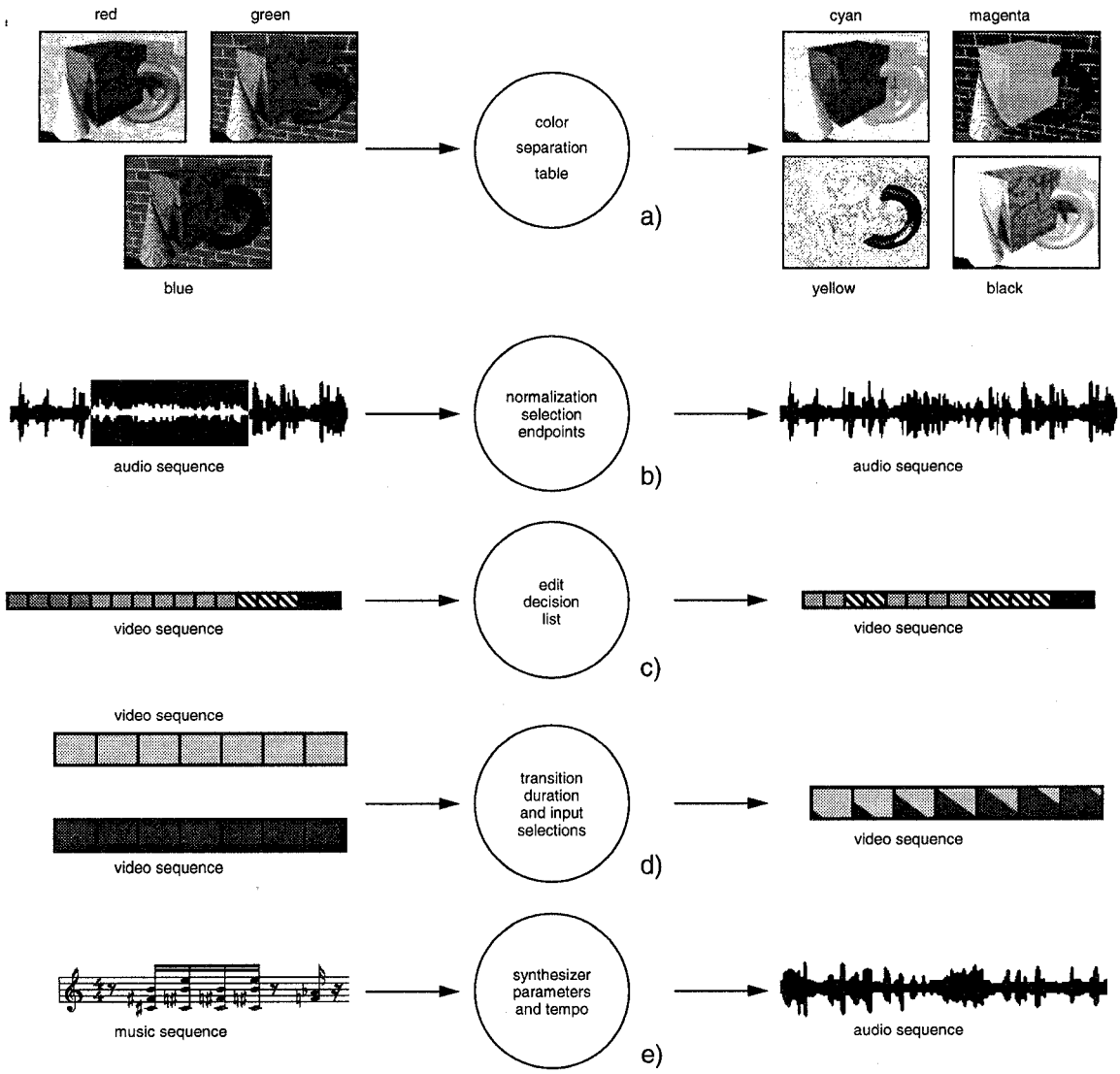

Figure 3 Examples of derived media objects: a) a CMYK separation, b) a normalized audio sequence, c) an edited video sequence, d) a video transition, e) a synthesized audio sequence. In each case the derived media object is on the right and the antecedent media objects on the left. The large circles indicate the information needed to perform the derivations.

Table 1 Examples of Derivation

Derivation	Argument Type(s)	Result Type	Category
color separation	image	image	change of content
audio normalization	audio	audio	change of content
video edit	video	video	change of timing
video transition	video	video	change of content
MIDI synthesis	music (MIDI)	audio	change of type

duce video frames that consist of data stemming from both video objects (one is faded out the other is faded in). The parameters for this kind of derivation specify the type of transition, its duration and the start time in both video objects.

MIDI synthesis The synthesis of an audio object from a MIDI object is an example of derivation for which argument and result types differ. Parameters are tempo, MIDI channel mappings and instrument parameters. (These essentially identify, for example, whether a given note is played on a piano, a violin or some other instrument.)

The use of derived media objects has many advantages. First, storage is saved since derived media objects and their associated derivation objects are relatively small (for example, a video edit list is likely many orders of magnitude smaller than a video object). Furthermore, by storing derivation objects it is possible to keep track of, and query, manipulations to media objects. Finally, sequences of derivations can be changed and reused, this is useful in multimedia authoring environments.

4.3 Composition

The *partOf* relationship is often used to model complex assemblies of simpler components. Since the essence of "multimedia" lies in assembling different media, it appears that some form of aggregation is fundamental to multimedia modeling. Looking to music and the arts, the term "composition" describes the combining and assembling of various elements. This suggests that composition is the overall organizing principle for multimedia data – it should then be modeled explicitly. In the context of data modeling we define composition as follows:

Definition 7 *Composition* is the specification of temporal and/or spatial relationships between a group of media objects. The result of composition is called a *multimedia object*, the spatiotemporally related objects are called its *components*.

Depending upon the nature of the media objects, a variety of forms of composition can be identified. They can be grouped into two general categories: spatial composition and temporal composition (e.g., [11]). The first deals with positioning objects in a 2D or 3D space. An example would be placing an image within a page of text or placing graphical objects in a scene. Temporal composition determines the relative timing and synchronization of time-based media. For instance, narrating a video sequence by combining it with an audio sequence is an example of temporal composition.

Example of Composition

As a more extensive example we will show how a multimedia object can be composed from several simpler objects, this example will also involve interpretation and derivation. Suppose we start with two audio sequences, audio_1 and audio_2, and two video sequences, video_1 and video_2. The two audio sequences contain music and narration and are intended to be presented simultaneously. For this reason they are interleaved in a single BLOB. Suppose the two video sequences result from a single capture operation (e.g., digitization of a single video tape) and so also reside in a single BLOB. These four sequences are the raw material from which a more polished multimedia sequence is to be constructed.

The first step is to construct a derived video sequence which performs a slow (10 second) fade from video_1 to video_2 (which we can assume represent different shots from the original tape). After deriving the fade sequence, video_F, we concatenate it with "cut" versions of the original sequences to produce video_3. Finally, a multimedia object is created and the three sequences audio_1, audio_2 and video_3 are added to it using temporal composition.

Figure 4 shows the relationships between the various sequences and the underlying BLOBs. Also shown are the intermediate objects, such as the four derivation objects (cut_1, cut_2, fade and concat), used to represent these relationships. The timeline diagram indicates the relative timing of the three components of the final multimedia object.

Producing multimedia objects is likely to involve many more derivations and many more components. But the procedure will be similar to that followed above: raw material is created and added to the database, and then successively refined (derived) and composed.

To summarize, interpretation, derivation and composition give us a way of moving from simple, uninterpreted data, to complex multimedia aggregates (see Figure 5). It is the higher-level abstractions–media objects and elements, multimedia objects–that are seen by applications.

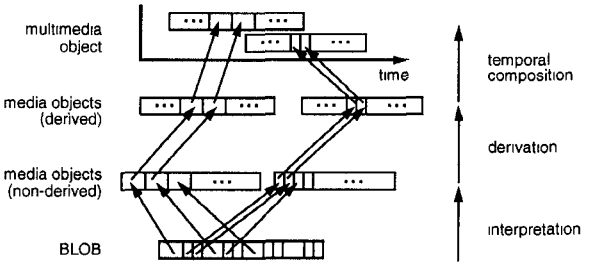

Figure 5 Successive interpretation, derivation and composition.

5 RELATED WORK

The areas that have influenced the data model include digital video and digital audio processing systems (e.g., editors, composition tools) and developments in multimedia platforms. QuickTime in particular has served as a concrete example of the mechanics of media interpretation and timing. QuickTime defines three structures, called media, tracks and movies; these correspond to what we have called non-derived media objects, derived media objects and multimedia objects. The main difference is that derivation is more general than QuickTime's media-to-track mapping. In QuickTime a track is created by rearranging the timing of media elements. Derivations, however, encompass changes in content and type in addition to changes in timing.

The notion of timed streams bears some resemblance to work in temporal databases. For example, Segev and Shoshani make use of *time sequences* for modeling temporal informa-

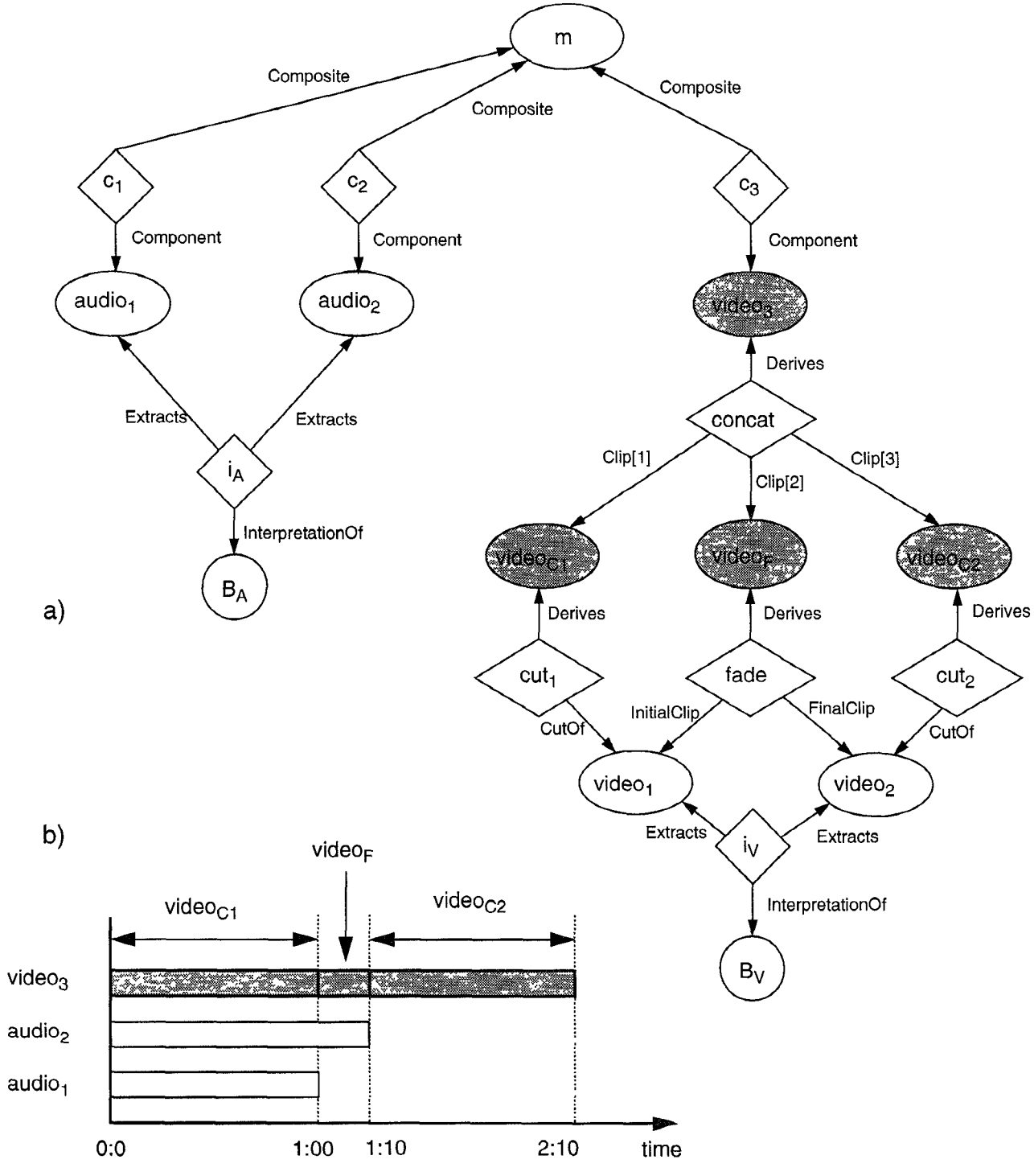

Figure 4 An instance diagram for a multimedia object is shown in a). The object, m, has three components – two audio objects and one video object. These are related to m via three temporal composition relationships, c_1, c_2 and c_3. (Instances of relationships are indicated by diamond shapes, multimedia and media objects by ellipses and BLOBs by circles. Derived objects are shaded.) The second diagram, b), shows the relative timing of the three components of m.

tion [19]. Some of the categories of time sequences, such as continuous and discrete, are similar to the categories of timed streams. However the semantics of the two constructs differ. Time sequences identify how attributes of an object, e.g., a bank account balance, change over the object's life span. Timed streams identify what is essentially scheduling information. For instance, the start time of a video frame is not the time when the frame was captured or created, but when it should be displayed relative to other frames in the stream.

Another related area is that of real-time databases. Time-based media impose real-time constraints–the database must consume and produce (record and playback) media elements at specific rates. Note, however, the deadlines are not *hard*. Divergences from element production and consumption deadlines are certainly undesirable, but can be tolerated; for example playback "jitter" can be removed by the application just prior to presentation. Support for the real-time playback of time-based media is now found in audio/video servers (e.g., [1][18]). These systems are likely precursors of more general audio/video databases.

6 CONCLUSION

We have presented a data model that addresses the representation of time-based media. The model includes the notions of media objects, media elements and timed streams. Three general structuring mechanisms are used: interpretation, derivation and composition. Many of these ideas have evolved from existing systems. Our goal has been to unify these ideas, provide a more formal foundation, and express them in a way suitable for data modeling.

Most of the examples in this paper deal with audio and video, we have given less attention to other media types such as graphics, music, and animation. However, the data model does encompass these types. The key is derivation: animation and music deal with symbolic representations from which audio or video sequences are derived. For example, music representations allow the specification of note sequences. A synthesizer then takes these sequences and derives audio sequences. Similarly video sequences are derived (via rendering) from representations of animation.

Implementation of database systems capable of supporting time-based media presents many new problems in such areas as scheduling, synchronization, resource allocation, and storage management. Authorization and electronic copyright need to be addressed. Support for time-based media requires basic changes in database architecture and the form of the database/application interface. The notion of timed streams introduced in this paper leads to a perspective where database operations are viewed as extended activities that produce, consume and transform flows of data. A database architecture based on activities and their possible interconnection is explored in [5].

REFERENCES

1. Anderson, D.P., and Homsy, G. A Continuous Media I/O Server and Its Synchronization Mechanism. *IEEE Computer*, Vol. 24, No. 10, Oct. 1991, pp. 51-57.
2. Bertino, E., Rabitti, F., and Gibbs, S. Query Processing in a Multimedia Document System. *ACM TOIS*, Vol. 6, No. 1 (Jan. 1988), pp. 1-41.
3. Carey, M.J., DeWitt, D.J., Richardson, J.E., and Shekita, E.J. Storage Managements for Objects in EXODUS. In *Object-Oriented Concepts, Databases, and Applications* (W. Kim and F.H. Lochovsky Eds.), Addison Wesley, 1989, pp. 341-369.
4. Christodoulakis, S. et al. Multimedia Document Presentation, Information Extraction, and Document Formation in MINOS: A Model and a System. *ACM TOIS*, Vol. 4, No. 4 (Oct. 1986), pp. 345-383.
5. Gibbs, S., Breiteneder, C., and Tsichritzis, D. Audio/Video Databases: An Object-Oriented Approach. *Proc. 9th Intl. Conf. on Data Engineering*, 1993, pp. 381-390.
6. Hoffert, E., et al. QuickTime: An Extensible Standard for Digital Multimedia, *Proc. of the IEEE Computer Conf.* (CompCon'92), Feb. 1992.
7. Lehman, T.J., and Lindsay, B.G. The Starburst Long Field Manager, in *Proc. Intl. Conf. on Very Large Databases (VLDB)*, Aug. 1989.
8. Klas, W., Neuhold, E.J., and Schrefl, M. Using an Object- Oriented Approach to Model Multimedia Data. *Computer Communications*, Vol. 13, No. 4, 1990, pp. 204-216.
9. Le Gall, D. MPEG: A Video Compression Standard for Multimedia Applications. *Commun. of the ACM*, Vol. 34, No. 4 (April 1991), pp. 46-58.
10. Lippman, A. Feature Sets for Interactive Images. *Commun. of the ACM*, Vol. 34, No. 4 (April 1991), pp. 92-102.
11. Little, T.D.C., and Ghafoor, A. Spatio-Temporal Composition of Distributed Multimedia Objects for Value-Added Networks. *IEEE Computer*, Vol. 24, No. 10, Oct. 1991, pp. 42-50.
12. MacroMind Inc. *Director Studio Manual*, Version 3.0, 1991.
13. Masunaga, Y. Multimedia Databases: A Formal Framework, *Proc. IEEE CS Office Automation Symposium* (Gaithersburg, MD, April 1987), IEEE CS Press, Washington, pp. 36-45.
14. Microsoft Corp. *Microsoft Windows Multimedia Programmer's Workbook*, 1991.
15. Preston, J.M. (Ed.) *Compact-Disc Interactive: A Designer's Overview*, Kluwer, Deventer NL, 1987.
16. Rengarajan, T.K. Rdb/VMS Support for Multi-media Databases. *Proc. of the ACM SIGMOD 1992 Conf. on Management of Data*, pg. 287.
17. Ripley, G.D. DVI – A Digital Multimedia Technology. *Commun. of the ACM*, Vol. 32, No. 7 (July 1989), pp. 811-822.
18. Rowe, L.A., and Smith, B.C. A Continuous Media Player. *Proc. 3rd Intl. Workshop on Network and Operating System Support for Digital Audio and Video* (San Diego 1992), pp. 334-344.
19. Segev, A., and Shoshani, A. Logical Modeling of Temporal Data. *Proc. of the ACM SIGMOD 1987 Conf. on Management of Data*, pp. 454-466.

20. Wallace, G.K. The JPEG Still Picture Compression Standard. *Commun. of the ACM*, Vol. 34, No. 4 (April 1991), pp. 30-44.
21. Woelk, D., Kim, W., and Luther, W. An Object-Oriented Approach to Multimedia Databases. *Proc. of the ACM-SIGMOD 1986 Conf. on Management of Data*, pp. 311-325.
22. Woelk, D. and Kim, W. Multimedia Information Management in an Object-Oriented Database System. In *Proc. Intl. Conf. on Very Large Databases (VLDB)*, 1987, pp. 319-329.