1994

A Conceptual Model for Multimedia Database Systems

Shunge Li

Xiangning Liu

Bharat Bhargava
Purdue University, bb@cs.purdue.edu

Report Number:
94-046

A Conceptual Model for Multimedia Database Systems

Shunge Li, Xiangning Liu, and Bharat Bhargava
Computer Sciences Department
Purdue University
West Lafayette, IN 47907

CSD-TR-94-046
June, 1994

A Conceptual Model for Multimedia Database Systems *

Shunge Li Xiangning Liu Bharat Bhargava
Department of Computer Sciences
Purdue University
West Lafayette, IN 47907
{lis,xl,bb}@cs.purdue.edu

Abstract

The representation and manipulation of synchronization and other temporal relationships among multimedia operations are important issues in multimedia database systems. This paper presents a conceptual model for multimedia database systems. This model extends the object-oriented data model by introducing a new abstraction mechanism for specifying temporal relationships among multimedia objects. More specifically, an mechanism called *clustering* is introduced to facilitate the representation and manipulation of multimedia objects among which sophisticated relationships pertain. Furthermore, the issue of potential synchronization conflicts is also addressed and some detection solutions are given. A framework for a multimedia prototype system is then proposed. Finally, the limitations of the current model and directions for future research are summarized.

1 Introduction

The formulation of a data model lies at the heart of the construction of a multimedia database system. The present research grew out of a study of the issues encountered in such data models. Multimedia database systems are characterized by the involvement of multiple types of data and the existence of complex temporal and spatial constraints among these data. Traditional data models are simply not equipped to handle these data types and constraints, necessitating the development of new and more powerful data models. In this paper, we focus particularly on the formulation of a mechanism with which temporal

*This research is supported in part by Army Research Laboratory.

constraints among objects and database events can be naturally represented and be easily manipulated.

Unlike most research on temporal databases where temporal data is the main concerns, our work focuses on the temporal relationships among multimedia operations since the multimedia data itself is in general not time-various.

In this paper, a model for representing and manipulating synchronization and other temporal relationships among multimedia objects is proposed. This model is based on the framework of the object-oriented data model. Besides its inheritance of such object-oriented features as complex objects, classes, methods, it introduces a new type of abstraction mechanism called *clustering* so that arbitrary n-ary synchronization and other temporal relations can be represented and manipulated at a conceptual level. Furthermore, many operations, including forward/backward presentations and stoppage, are supported in the model.

In actual multimedia systems, potential synchronization conflicts may cause systems to enter infinite loops or simply abort. Since the detection of possible synchronization conflicts is therefore of prime importance, this paper treats this issue in some detail and establishes set of equations to solve this problem.

The rest of this paper is organized as follows: section 2 is dedicated to our proposed model; section 3 shows relevant database schemas; section 4 discusses the issues of potential synchronization conflicts and its solutions; section 5 shows the framework of MM-Raid multimedia prototype; section 6 briefly presents some research work related to synchronization and data models in multimedia database systems, and compares them with our model; section 7 concludes the paper by pointing out future research works.

2 The Proposed Data Model

2.1 An Example

Before providing formal definitions of our model, let us first examine a typical example of a multimedia application in order to provide a context for the concepts to appear later.

Figure 1 illustrates a situation typical to many multimedia systems. The several data streams illustrated may represent a data type, which may be an image, a video, audio, or a text. Two or more objects in any data streams are connected via a synchronization link, of which the endpoints must be synchronous in time. For example, a link connecting an object in an image data stream with an object in an audio data stream requires that the operations on these two objects be synchronous in order that the image be displayed at the same time the audio is played. There are two types of synchronization links; interstream links connect data from different streams, while intrastream links connect data within a single stream. These links are usually specified by users through a medium such as a specification language. One of the tasks of the system is to guarantee these synchronization constraints.

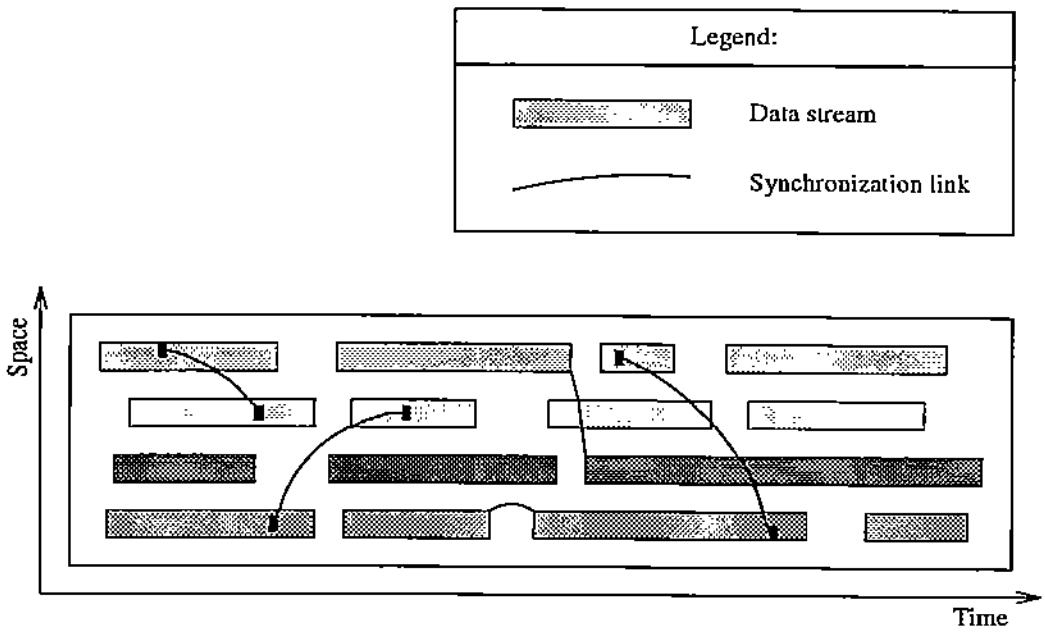

Figure 1: Synchronizing Multistream Multimedia Data

In general, users may specify synchronization links among arbitrary objects. However, it usually makes sense for users to specify synchronization constraints only among content-related objects. From the perspective of data abstraction, we view objects associated with links as tightly-coupled and objects without links as loosely-coupled. This partitions the set of objects into several classes of objects according to the nature of their synchronization links. Each class is termed a *cluster*.

Although, in this simple example, synchronization links represent only point synchronization, other binary temporal relationships defined in [All83] can be represented by conversion into point synchronizations. Continuous synchronization can be represented by specifying point synchronizations at the two endpoints of the continuous interval. As will be discussed later, synchronization links can also represent other types of synchronization (e.g., spatial synchronization) and constraints. For this reason, we will omit the word *synchronization* and simply use *link* in the formal definitions presented in next section.

2.2 Definitions

Object: An object is a composite entity made up of data and a set of methods. Data characterizes its states and attributes, while methods characterize its dynamic behavior. Structurally, an object is an Abstract Data Type (ADT), which also specifies constraints on methods. Objects in databases are persistent. An object is said to be **atomic** if it cannot be decomposed into smaller objects.

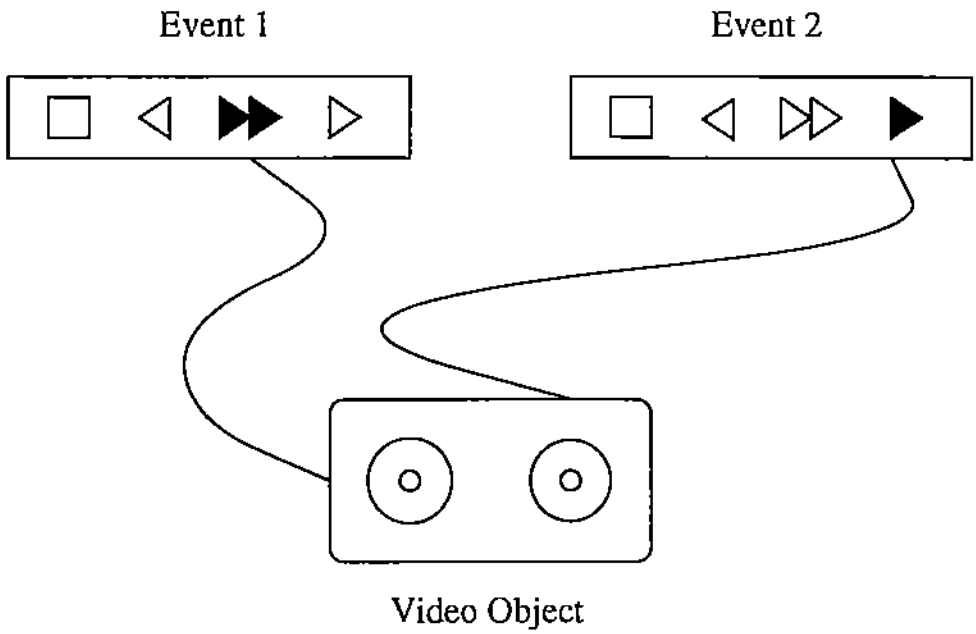

Figure 2: Two Events Performing Simultaneous Operations on an Object

Event: An event is an execution of a method on an object.

Objects are sharable, while events (which are implemented as processes) are not. The differences between objects and events parallel to those between programs and processes. Figure 2 illustrates the simultaneous performance of operations by two events on a single video object. One event is fast-forwarding the video object, while the other is playing it at normal speed. Although these two events share a single video object, they are independent of each other.

Note that although an event characterizes an operation on an object, it can itself be viewed as an object. However, unless otherwise stated, we will treat events as dynamic entities.

Active Event: An event is active if it is currently running, otherwise it is dormant. This concept is defined for use in modeling such situations as a pause in a presentation.

Working Object: An object is active when it is currently being acted upon by an event; it is dormant when it is not working.

Anchor: An anchor is an entity *within* an object to which links may be hooked. Links may be hooked to objects only through anchors. An atomic object itself can be an anchor. With this anchoring mechanism, finer-grained temporal relationships, such as within-component relationships (e.g., links within an object), can be defined. This mechanism facilitates handling of different levels of synchronization precision.

Link: A n-ary link ($n \geq 2$) is a n-ary relation that represents a constraint among n events. A binary link is denoted by link(p, q) if there exists a binary link between event

p and event q. By default, links are undirected; therefore, link(p, q) is true if and only if link(q, p) is true.

Path: A path exists between event p and event q, denoted by path(p, q), if either link(p, q) is true or there' exists an event r such that both path(p, r) and path(r, q) are true. The latter case reflects the fact that *path* relations are transitive.

Cluster: A cluster is an event which contains a (maximum) set of events in which there is a path between any two events. Events belonging to a cluster are called **members** of that cluster. For example, playing a movie consisting of videos, audios, and captions can be abstracted as a cluster in which videos, audios, and captions must be played synchronously.

An independent event, which has no links pointing to or from any other events, forms a **free cluster**.

Clustering: Clustering is the process of forming a cluster. Higher-level temporal relationships (such as relationships among clusters) transcending simple peer-component-level temporal relationships, can be specified via the clustering mechanism. For example, if information is shared among clusters, it may be specified for a cluster common to all members.

Cluster Object and Cluster Hierarchy: A cluster itself can be viewed as an ordinary object and can be a member of other clusters. For example, the events of playing a movie and playing an advertisement are both clusters, but they can also be viewed as objects in a larger context. Constraints among them can be specified, too. A user may specify that an advertisement be played after every 10 minutes or after each 100 scenes of a movie. A higher-level cluster can then be defined with the events (clusters) of playing movies and playing advertisements as its members. This process can proceed recursively, forming a cluster hierarchy.

Time Interval: Each event is associated with a time interval indicating the lifespan of the event.

Lifespan of An Object: The lifespan of an object is the time interval from the birth of the object to its death. Since objects in databases are persistent, their lifespans are generally larger than the time durations during which they are in memory.

The definitions indicates that events are volatile, while objects are persistent. Furthermore, a cluster contains the constraints among its member events. In order to forestall occasional specification of these constraints by users, a *fixation* mechanism is provided to cause clusters to be persistent.

Cluster Size: The size of a cluster can be measured in a variety of ways. When links represent temporal relationships among member objects, time can be taken as a measure, and the size of a cluster will be the union of the time intervals of all its member objects. When links represent spatial relationships among member objects, space can be taken as a measure, and the size of a cluster will be the union of the spaces occupied by all its member objects. In general, the size of a cluster is defined as the union of the sizes of all its member objects as defined by a consistent measure. A cluster is therefore the minimum upper bound of the collection of its member objects.

Figure 3 illustrates the concepts of cluster and clustering. The offset of a member

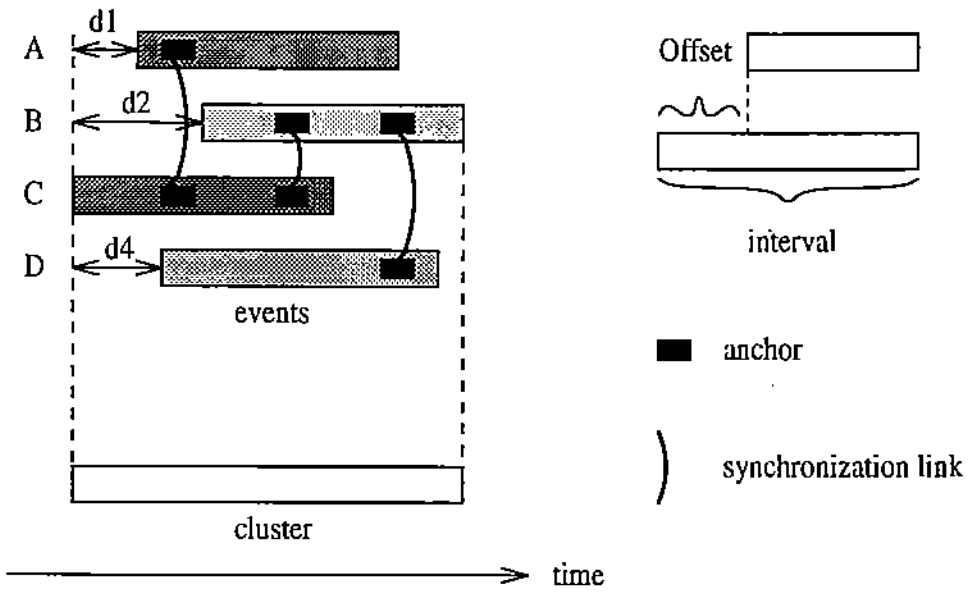

Figure 3: Cluster and Clustering

indicates its absolute position (order) within the cluster. This offset can be defined relative to the ending time of the cluster, permitting the modeling of reverse presentations. Figure 4 illustrates the cluster hierarchy which corresponds to Figure 3. The legend in Figure 4 is expanded in Figure 5.

2.3 Properties

Information Hiding and Data Abstraction: Clustering is a type of data abstraction. For example, a user may specify temporal relations among clusters but be uninterested in the temporal relations within clusters. Clustering allows within-cluster temporal relations to be hidden from outside view. Since all constraints among member objects of the cluster must by definition have been satisfied, we can move our attention to higher-level constraints.

Disjointness of Clusters: By the definition of a cluster, no links may be present between members in different clusters. The presence of such links would imply that these are actually a single cluster. Thus objects cannot belong to more than one cluster (with respect to certain constraints), and clusters therefore cannot have common members.

Tightness: Any member object in a cluster is restricted to the space available within the cluster. The distance between any two members of a cluster is therefore no greater than the diameter or the diagonal length of the cluster, and the size of any member object of a cluster is no larger than that of the cluster. If some relation holds true for a cluster, it must also hold for its members. Properties defined under interval inclusion relations [OT93] are

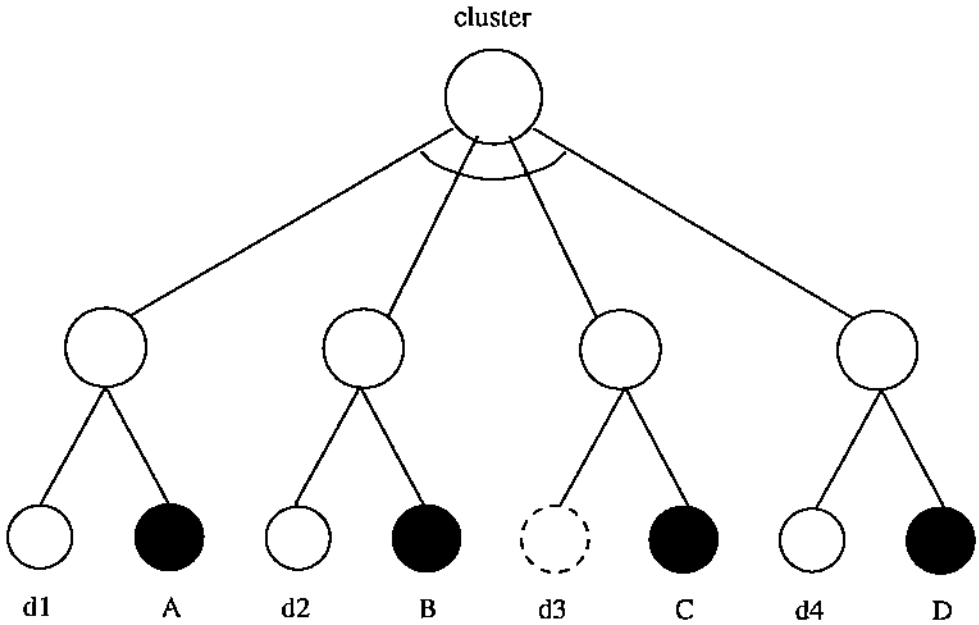

Figure 4: Cluster Hierarchy

therefore included in this tightness property.

Strongly-Connected Graph: A graph within which the members of a cluster form vertices and the links among them are edges is a strongly-connected graph. The members of a cluster therefore form an equivalence class under the relation of reachability.

Inheritance: Since some information common to cluster members is specified in the cluster, some cluster attributes can be inherited by its members.

Orthogonality of Generalization, Aggregation, and Clustering: While clustering may intuitively appear synonymous with aggregation, the latter addresses only composition-related information such as the number and nature of a composite object's components. Aggregation does not address the organization of those components within composite objects. In contrast, clustering deals with the layout and ordering of members within a cluster. As a simple analogy, a car consists of many parts, but a heap of parts with no specification of spatial relationships among them does not form a car. Moreover, the members of a cluster are not necessarily the components of the (virtual) object on which the cluster performs. Furthermore, the member objects of a composite object can be shared, whereas those of a cluster are not sharable. Finally, aggregation deals with static components, whereas clustering deals with dynamic events.

In fact, clustering may be viewed as a special type of associations with restricted forms of relationships. Clustering is restricted to spatio-temporal relationships, while associations in general can represent arbitrary relationships among objects. This forms a distinction between clustering and general association.

Table 1: Comparison Among Different Abstractions

Abstraction	Concerns	Relationships	Hierarchy
Generalization	Properties	IS_A/A_KIND_OF	Class Hierarchy
Aggregation	Components	A_PART_OF	Object Hierarchy
Clustering	Positions	Spatio/Temporal	Cluster Hierarchy

Clustering also differs from generalization in that inheritance exists in a strict form in class hierarchy but not in cluster hierarchy. Every subclass inherits some aspects of its superclass, whereas members of a cluster need have no relationship of inheritance with the cluster.

Table 2.3 summarizes the differences among these abstractions.

2.4 Applications

1 Slide Presentations

Figure 5 illustrates the cluster hierarchy for slide presentations. Each atomic event represents a presentation of either slides or annotated voices, which must be synchronized in pairs, each of which forms a cluster. All clusters must be played sequentially. From the figure we can see that, although the fourth slide presentation uses the same slide as the second, it must invoke a separate process (event).

2 Spatial Synchronization

We can extend the example of Figure 3 by allowing the constraints to be specified across two-dimensional space, rather than only in a one-dimensional system such as an interval. For example, the links among objects can indicate their spatial relationships.

For this reason, the notion of synchronization must be extended to include spatial as well as temporal aspects. Just as two objects that are synchronous in time must be executed simultaneously, two objects that are synchronous in space must be displayed at the same place. Spatial relationships such as left/right of, above/below, adjacent, contained, surrounded, or overlapping, can be defined by analogy to temporal relationships. [Li94] has more discussions on this topic.

3 From Data Model to Database Schema

It is assumed that temporal relations are specified by users. After users submit their specifications, the Specification Language Processor (SLP) is invoked to transform these specifications to a database, following the database schema given below. An SLP is a

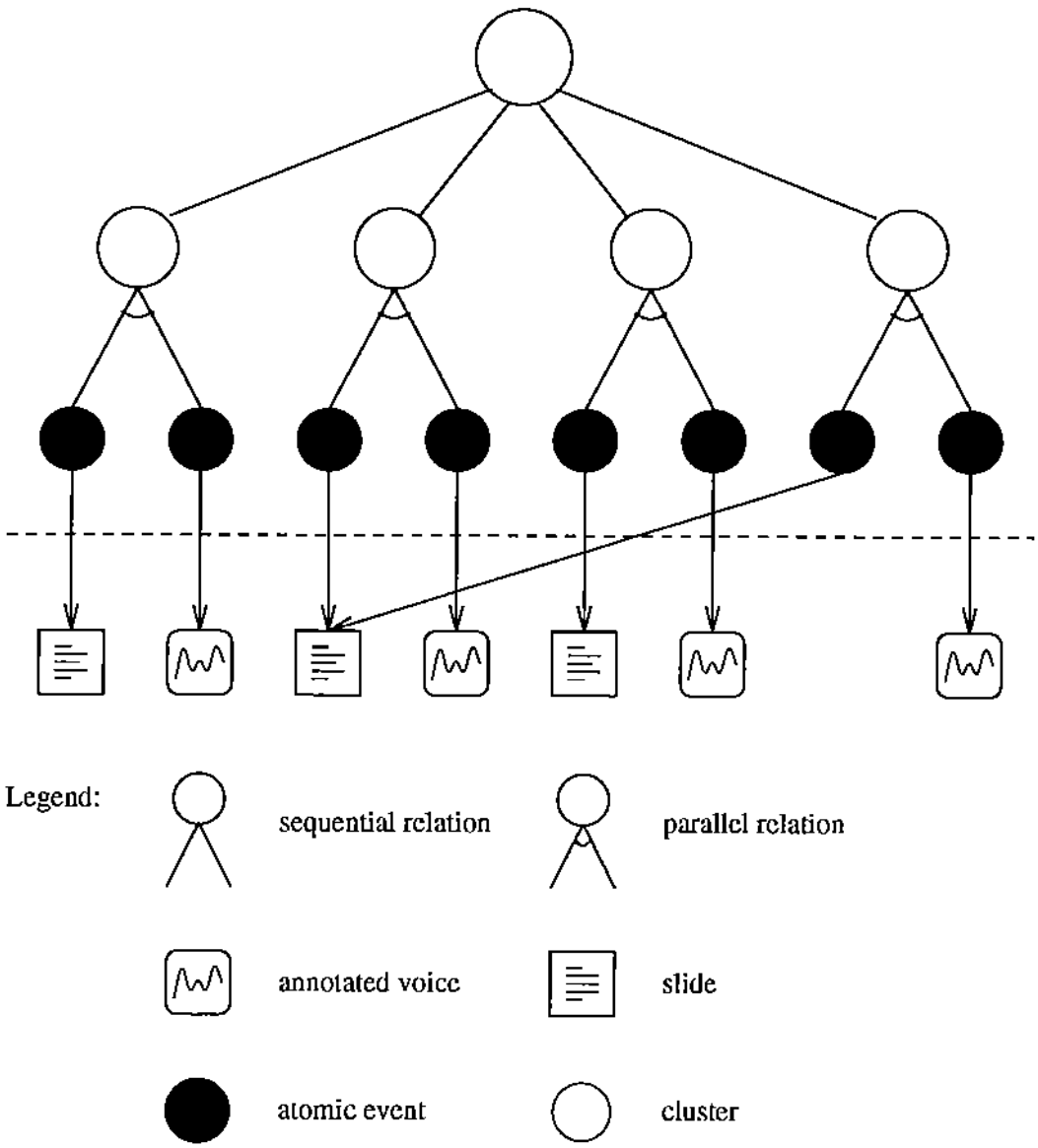

Figure 5: Cluster Hierarchy for Slide Presentation

modeling tool which consists of three parts: a scanner and parser, a conflict detector, and a cluster hierarchy builder. The scanner and parser ensures the lexical and syntactic correctness of specification languages. The conflict detector seeks out any potential user-defined synchronization conflicts, an issue to be addressed later in greater detail. Finally, the cluster hierarchy builder constructs a cluster hierarchy from which a database can be easily established. Figure 6 illustrates this process.

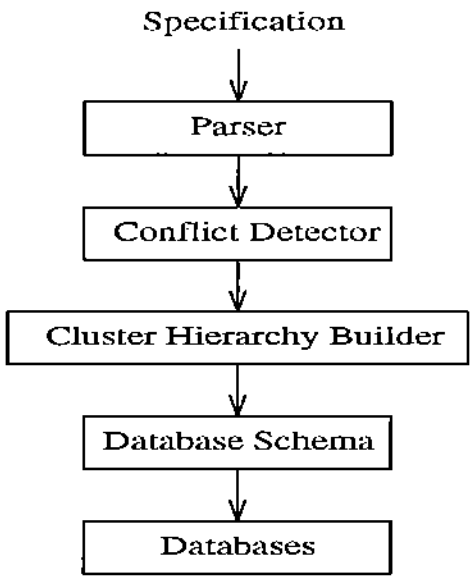

Figure 6: Flow of Clustering

The following outlines the database schemas of the basic components of this model.

3.1 Database Schema of Atomic Events

The database schema of an event includes (but is not limited to) the following fields (Table 3.1):

Table 2: Database Schema of Atomic Events

EID	Active	OID	OP	TimeInterval	PCID	Offset	AnchorList	...

- **EID:** A unique identifier assigned automatically by the system to each event upon its creation.

- **Active:** A flag indicating whether this event is active or dormant.
- **OID:** The OID of the object on which the operation will be performed.
- **OP:** Fields relating to the operation performed by the event, including name, signatures, and other constraints.
- **TimeInterval:** The time interval defined by the time instants of the two endpoints of the event, from which its length can be computed.
- **PCID:** The CID (described below) of the parent cluster.
- **Offset:** The relative time instant with respect to the parent cluster. This can be defined relative to the starting time or ending time of the parent cluster.
- **AnchorList:** A pointer to the linked list of anchors, a collection of all anchors possessed by the event. Each node in the linked list contains such information about each anchor as the offset and precisions.

3.2 Database Schema of Clusters

The database schema of a cluster includes (but is not limited to) the following fields (Table 3.2):

Table 3: Database Schema of Clusters

CID	N_1	N_2	N_3	Interval	PCID	Offset	IndexPtr	Anchor List	...

- **CID:** A unique identifier for each cluster assigned automatically by the system to each cluster upon its creation.
- $\mathbf{N_1}$: The number of members within the cluster.
- $\mathbf{N_2}$: The number of media involved in the cluster. This indicates the number of I/O channel types to be involved during the lifespan of the cluster.
- $\mathbf{N_3}$: The maximum number of parallel events involved in the cluster. This indicates the maximum number of processes required throughout the lifespan of the cluster. It is equal to the maximum number of I/O channels needed during that time period.
- **Interval:** The time interval computed from the intervals of cluster members.
- **PCID:** The CID of the parent cluster.
- **Offset:** The relative time instant with respect to the parent cluster.
- **IndexPtr:** A pointer to the index table recording the offsets of its members.
- **AnchorList:** A pointer to the cluster-level linked list of anchors.

3.3 Database Schema of Links

The database schema of a link includes (but is not limited to) the following fields (Table 3.3):

Table 4: Database Schema of Links

LID	Degree	Strict	Endpoints	...

- **LID:** A unique identifier assigned automatically by the system to each link upon its creation.
- **Degree:** The arity of the link.
- **Strict:** A flag indicating the strictness of the synchronization constraints represented by the link. If a synchronization constraint is strict, then the pausing of one event must cause the pausing of all related events.
- **Endpoints:** A linked list that contains all the anchors pointed to by the link. Each node in the linked list points to the data structure of each anchor.

3.4 Database Schema of Anchors

The database schema of an anchor includes (but is not limited to) the following fields (Table 3.4):

Table 5: Database Schema of Anchors

ANID	Offset	CID	ENID	Precision	...

- **ANID:** A unique identifier assigned automatically by the system to each anchor upon its creation.
- **Offset:** The relative time instant with respect to the cluster.
- **CID:** The ID of the cluster to which the anchor belongs.
- **ENID:** The ID of the entity at which the anchor is located.
- **Precision:** An indication of the maximum tolerable delay and minimum acceptable delay.

4 Potential Synchronization Conflicts

4.1 Classification of Synchronization Conflicts

Three general categories of synchronization conflicts may arise in multimedia manipulations [BvRW91].

- **User-defined Synchronization Conflicts:** An unreasonable synchronization constraint may have been defined (directly or indirectly) by a user.
- **Conflicts caused by Device Limitations:** Device characteristics may limit the ability of a particular environment to support a given operation. For example, audio cannot be played on machines without speakers.
- **Conflicts caused by Environments:** When navigating through a document, a reader/viewer/listener may wish to fast-forward (fast-backward) to a document section that contains a number of relative synchronization constraints for which the source or destination is not active [BvRW91].

In this paper, only the user-specified conflicts will be addressed. Conflicts of the other two categories are device-dependent and are really system restrictions. User-specified conflicts will be exacerbated by the existence of multiway links, which make conflicts easier to introduce.

The detection of synchronization conflicts is an important issue. Users typically specify temporal relationships among objects through specification languages. If no mechanism is provided for conflict detection, the system will enter into infinite loops or simply abort, greatly harming system robustness and degrading system performance. A synchronization conflict detection mechanism allows a system to automatically detect and refuse conflicts at specification time, rather than crashing the system at run time.

4.2 Solutions to Synchronization Conflicts Detection

Although user-defined synchronization conflicts involve the semantics of the constraints, a syntactical approach may be employed in their detection.

4.2.1 Establishing the Set of Equations — A Naive Solution

Assume there exist n data streams (Figure 1). The ith data stream has n_i intervals whose lengths are known a priori. Assume further that there some orderings exists among intervals within the same data stream, i.e., the ith interval is always ahead of the jth interval for $j > i$. Thus, there are only serial synchronizations among intrastream intervals. The gap between two adjacent intervals (the kth and $k+1$st interval) within the ith data stream is denoted by g_{ik}. Assume there exist m links among n data streams. Each link specifies a point synchronization between two points within two intervals. Links can represent both

serial and parallel synchronization. Other binary temporal relations defined in [All83] can be converted into point synchronization.

Suppose that some link relates a point x within the pth interval of the ith data stream to a point y within the qth interval of the jth data stream. We can then establish a basic equation as follows:

$$g_{i0} + \sum_{k=1}^{p-1}(g_{ik} + I_{ik}) + d_x = g_{j0} + \sum_{k=1}^{q-1}(g_{jk} + I_{jk}) + d_y$$

where I_{ik} are the lengths of the kth interval in the ith data stream, and d_x and d_y are offsets within their intervals, respectively. The set of equations with appropriate constraints can be established similarly:

$$\begin{cases} g_{i0} + \sum_{k=1}^{p-1}(g_{ik} + I_{ik}) + d_{xl} = g_{j0} + \sum_{k=1}^{q-1}(g_{jk} + I_{jk}) + d_{yl}; & 1 \le l \le m,\ 1 \le i, j \le n, \\ & 1 \le p \le n_i,\ 1 \le q \le n_j \\ g_{ik} \ge 0; & 1 \le i \le n,\ 1 \le k \le n_i \\ \text{minimize } z = \sum_{i=1}^{n} g_{i0} \end{cases}$$

with g_{ik} being unknowns. This is a typical linear programming problem with the goal function being:

$$z = \sum_{i=1}^{n} g_{i0}$$

and can be solved with various approaches.

4.2.2 Reducing the Number of Variables by Clustering

It may have been noted that the number of variables will be very large when there are many data streams or when there are many intervals within data streams. In such instances, direct solution of this set of equations would be expensive and cumbersome. An alternative approach is to partition the intervals among data streams into several clusters, to establish the set of equations for each cluster, and to solve them separately.

More formally, if a link exists between the pth interval of the ith data stream and the qth interval of the jth data stream, then these two intervals are in the same cluster. Clusters can be easily identified on the basis of their properties of reflexity, symmetry, and transitivity (the members form an equivalence class).

Each single interval, to and from which no links point, initially forms a separate equivalence class (i.e., free cluster). It can be recursively merged with adjacent equivalence classes whenever necessary.

We have thus reduced the original problem to several subproblems of smaller size.

4.2.3 Solutions to Rate-Variable Synchronization Conflicts Detection

In previous sections, it is implied that the rates of presentation of data streams are all equal and have been normalized to 1. However, user may specify

- A variable rate of presentation for each interval; and
- That the required presentation times of two intervals of different lengths are equal, causing a difference in their presentation rates.

In either case, each interval is associated with a presentation rate. Let R_{ik} be the presentation rate of interval I_{ik}. The basic equation will therefore be modified as follows:

$$g_{i0} + \sum_{k=1}^{p-1}(g_{ik} + \frac{I_{ik}}{R_{ik}}) + \frac{d_x}{R_{ik}} = g_{j0} + \sum_{k=1}^{q-1}(g_{jk} + \frac{I_{jk}}{R_{ik}}) + \frac{d_y}{R_{ik}}$$

4.2.4 Solutions to Tolerable Synchronization Conflicts Detection

Until this point, only precise synchronization has been considered. However, latency, network transmission delay, the overhead of context-switching, or other factors may prohibit precise point synchronization. In such instances, we must consider a lower level of synchronization precision [BvRvL91].

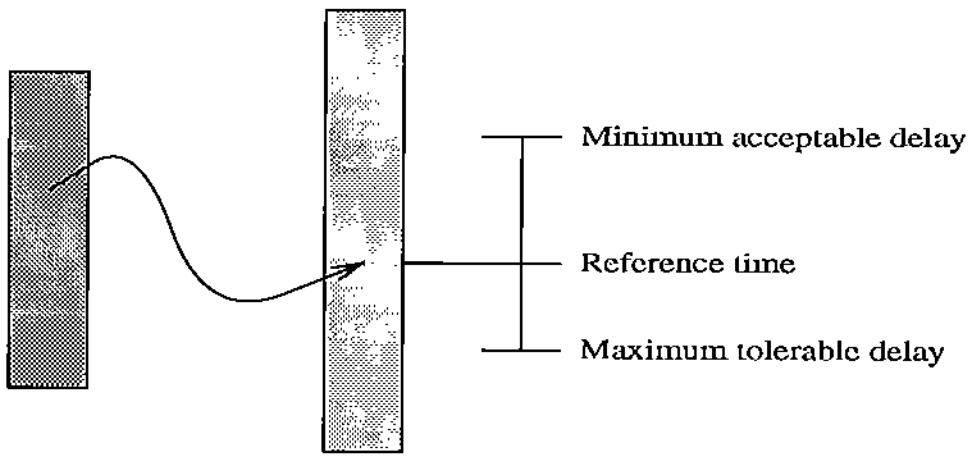

Figure 7: Synchronization Delay Parameters

For each link, let the precise point synchronization be corrected by considering total reference times (t_{ref}), minimum acceptable delays (δ), and maximum tolerable delays (ϵ) (Figure 7). The general synchronization equation then becomes

$$t_{ref} + \delta \leq t_{actual} \leq t_{ref} + \epsilon$$

where t_{actual} is actual time [BvRvL91].

The following equations must then be added to the original set (Figure 8):

$$\epsilon_{ik} + \delta_{i(k+1)} \leq g_{ik}$$

for $1 \leq i \leq n$ and $1 \leq k \leq n_i$.

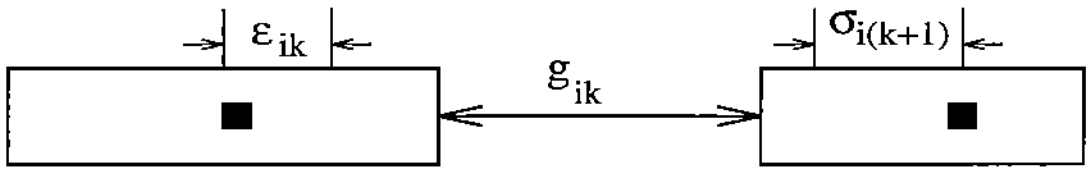

Figure 8: Tolerable Conflicts Detection Solution

If an interval contains more than one synchronization point, the corresponding delays must be accumulated.

5 MM-Raid – A Framework

MM-Raid is a multimedia database prototype system which we propose to superimpose upon the O-Raid system [BJS93], an object-oriented extension of a distributed database management system called Raid [BR89]. Figures 9 illustrate the architecture of MM-Raid.

Using an incremental approach to build MM-Raid, traditional Raid applications will still be supported without modifications. All features dedicated to multimedia applications are constructed as an extension upon the O-Raid system, which is itself left intact. Furthermore, the functions supported by O-Raid, such as object composition, class definition, indexing [JLB94], user-defined functions, and communications facilities, can be reused by MM-Raid.

The six layers of MM-Raid have the following functions:

- Layer 1 is the physical organization of data, including the multimedia data itself and metadata databases. Multimedia data consists of audio/video data, image data, documents, and formatted data. Metadata databases contain class definitions, cluster definitions, bibliographic data, descriptional data, and semantic data (content data) for multimedia data.
- Layer 2 is the Raid relational distributed database management system, combined with the communication facility. The Raid system supports transaction processing over LAN/WAN environments. To improve overall performance, Raid was designed to permit databases to be replicated in different sites. Consistency among all replicas is automatically guaranteed by the system. The communication facility will be enhanced

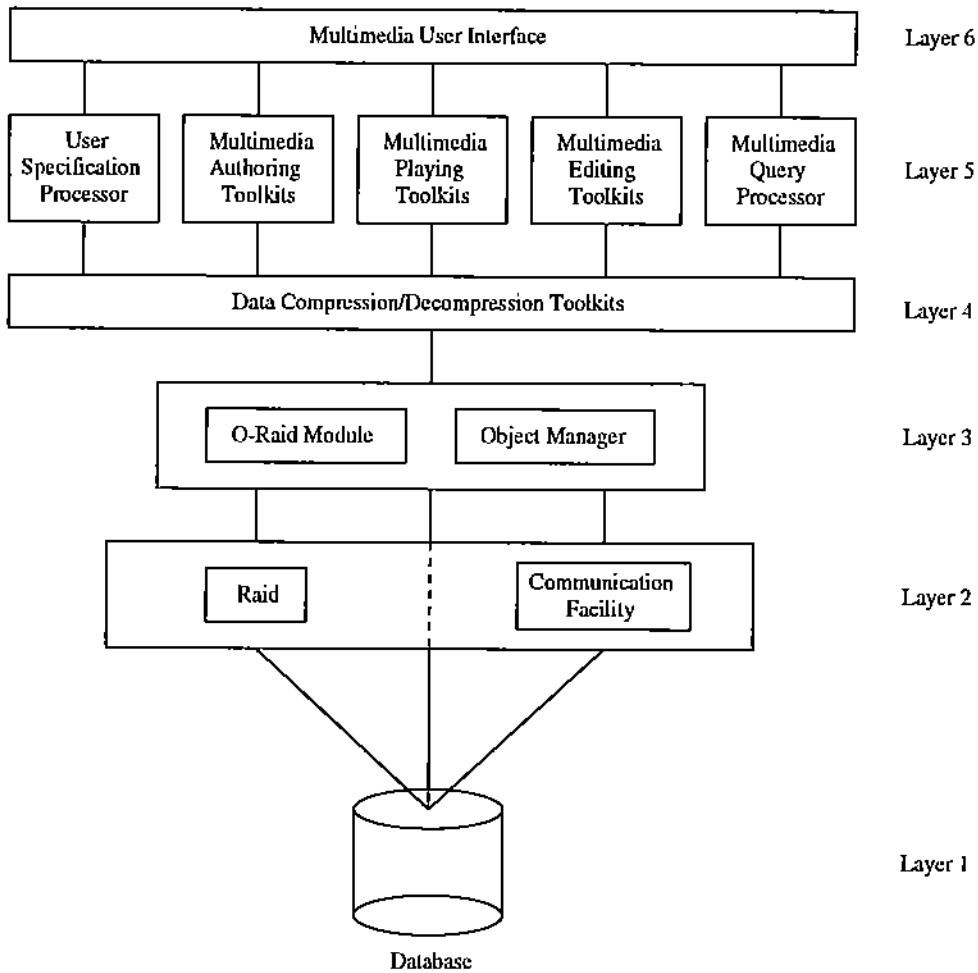

Figure 9: The architecture for a multimedia database system

to support the transmission of large volume of data. This layer is the physical interface to the Internet.

- Layer 3 is the object layer of the MM-Raid system. It consists of two major parts: the O-Raid module and the object manager. The O-Raid module allows the definition and registration of user-defined classes and the composition of complex objects. The object manager is responsible for object and cluster definition, manipulation, and access. Inheritance, object migration, and classification are also supported here. For multimedia applications, the object manager directly accesses databases for metadata, descriptive data, and compressed multimedia data without going through the Raid module (see the dash line in the figure), thus improving performance and compatibility.
- Layer 4 consists of toolkits, such as the MPEG encoder and decoder, which are necessary for the compression and decompression of multimedia data. This layer performs conversions among data formats appropriate for storage, transmission, presentation, and editing.
- Layer 5 consists of several modules: a specification language processor (SLP), multimedia authoring, playing, and editing toolkits, and a multimedia query processor. It allows users to define and manipulate multimedia data. The user specification language processor (SLP) parses and evaluates user specifications for such application constraints as synchronization constraints and the temporal relationships among objects. Any synchronization conflicts present in user specifications will be discovered by the conflict detector. If no conflicts exist, cluster hierarchy is built and the corresponding data and cluster information are stored in the database. Based on these results, the authoring toolkits then allow users to build customized applications. Query language statements can be embedded into user specification languages to facilitate specification. In addition to building customized multimedia systems, users are allowed to play, browse, and edit multimedia data selectively. The query processor transforms higher-level user queries, such as fuzzy or visual queries, into statements in the SQL++ language, which is the query language of O-Raid. Such queries may alternatively be transformed into internal object-oriented query statements to assist the object manager in finding targets. In general, multimedia data is retrieved through the object manager by descriptions, contents, or bibliographic data. Of course, users may bypass the query processor and use SQL++ directly for retrieving multimedia data.
- Layer 6 provides a uniform interactive user interface that integrates underlying modules.

6 Discussions on the Proposed Model

6.1 Comparison with Other Models

Various approaches have been proposed to the modeling of synchronization and other spatio-temporal relationships among multimedia objects. Literature on this topic includes [LG90], [LG93], [LY93], [Mas91], and [CTB92].

[LG90] presented a Petri-Net-based model (OCPN) for handling synchronization in multimedia applications. [LY93] extended this method by associating each class of places with a parameter weight that determines its relative importance compared to other classes. Synchronization represented in a Petri-net is automatically triggered when the firing conditions are satisfied. In spite of the expressive power of this approach, it does suffer from the weakness in supporting finer-grained synchronization after subnet substitution. It also does not support such operations as the reverse presentation of an object or stoppage of a dynamic presentation. This method is also unable to detect any possible conflicts among temporal relations.

OMEGA ([Mas91]) extended the object-oriented data model to introduce four types of relations: temporal precedence relations, temporal synchronization relations, spatial precedence relations, and spatial synchronization relations. However, only those objects among which explicit temporal and spatial relationships exist are defined under these four classes. Moreover, the method of specifying the temporal relationships among objects according to temporal adjacency has several drawbacks [Li94]:

- It represents a temporal object incrementally by referencing the starting time of its predecessor. In some instances, temporal relations can be more easily represented in terms of ending times, which would require additional calculation in this approach.
- Actions that are temporally adjacent need have no constraints on their temporal relationship. Conversely, temporally constrained objects may be linked through irrelevant intermediate objects or may even be unlinked simply because they are not temporally adjacent.
- The deletion or update of an object will cause other temporally adjacent objects to be updated. This process may propagate through an extensive sequence of objects.

One of the contributions of TEDM [CTB92] is to characterize the evolutionary features of some time-varying objects. Objects can dynamically fuse or be split into new objects, and their attributes can evolve with time. Temporal-relation object constructs are also provided to deal with temporally related objects. However, these constructed are defined only for objects of restricted classes, such as objects and their super-types or aggregated types.

The concepts of anchor and link used in our model are borrowed from the DEXTER hypermedia model [HS94]. Unlike our model, however, the DEXTER model does not consider temporal relations. [HBvR94] extends the DEXTER model through the addition of

the concepts of time and context. This extension permits the explicit modeling of synchronization and other timing relations. However, no mechanism comparable to our concept of clustering is supported.

6.2 The Advantages of the Proposed Model

Our model differs from those described above in the following ways:

- A new abstractive mechanism is used to characterize ordering information, including temporal and spatial relationships among objects. Temporal relations not explicitly specified by users can also be handled.
- An interval is used to represent the lifespan of an object, while an offset represents the time instant of each member within a cluster. Updating and deletion is thus simplified by this independent treatment of the representations of each member of a cluster.
- Each object is assigned a flag indicating the state of its current activity, thus modeling the stoppage of a presentation. Offsets can be defined relative to both the starting and ending time of a cluster, permiting the modeling of backward presentation.
- Both point and continuous synchronization are addressed, and different levels of synchronization precision are also considered. Arbitrary n-ary spatio-temporal relations with $n \geq 2$ can be represented by this model, and conflicts of temporal relations can be detected.

7 Conclusions and Future Research and Experimental Directions

In this paper, a model for the development of multimedia database systems has been proposed. The proposed framework extends the traditional object-oriented data model to include consideration of the temporal relationships among objects. A new abstraction mechanism, called clustering, permits the representation and manipulation of user-specified temporal and spatial relationships among objects. Through clustering, complex spatio-temporal relationships are layered in a cluster hierarchy, while the features of the object-oriented approach are preserved. We have demonstrated that this model is well equipped to handle the complex relationships which exist among multimedia objects.

Synchronization conflict is a critical issue in the development of practical systems. In this paper, the problem of user-specified synchronization conflicts was investigated, and solutions to detecting potential synchronization conflicts were explored.

Finally, a framework for the construction of a multimedia prototype as an extension of O-Raid system was proposed.

The next step in our research in this issue is the construction of a multimedia database prototype on the basis of the proposed model. Since our underlying system is distributed,

the proposed model must be extended to consider distributed environments. Such system design issues as the development of an object-oriented query language and of multimedia data indices will be addressed. Performance control will be a significant consideration in the implementation of the prototype system. In order to evaluate the overall performance of the complete system, we may consider the construction of a benchmark for it.

The prototype MM-Raid will be tested in the following contexts:

- Communication experiments will be conducted over an ATM network using UDP/TCP or other protocols. The relationship between communication costs and the Quality of Services (QOS) will be explored.
- Some multimedia applications will be implemented. These may include VOD systems, multimedia e-mail databases, digital libraries, and multimedia (hypermedia) dictionary or encyclopedia.
- Various types of queries will be compared and integrated, including content-based query, fuzzy query, and visual query.
- An MM-Benchmark will be built to evaluate system performance.

References

[All83] J. F. Allen. Maintaining knowledge about temporal intervals. *The Communication of the ACM*, 26(11):832–843, Nov. 1983.

[BJS93] B. Bhargava, Y. Jiang, and J. Srinivasan. O-Raid: Experiences and experiments. In *International Conference on Intelligent and Cooperative Information Systems, Rotterdam, Holland*, May 1993.

[BR89] Bharat Bhargava and John Riedl. The Raid distributed database system. *IEEE Transaction on Software Engineering*, 15(6), June 1989.

[BvRvL91] Dick C. A. Bulterman, , Guido van Rossum, and Robert van Liere. A structure for transportable, dynamic multimedia documents. In *Proceedings of 1991 Summer USENIX Conference, Nashville TN*. USENIX, June 1991.

[BvRW91] Dick C. A. Bulterman, Guido van Rossum, and Dik Winter. Multimedia synchronization and UNIX -or- if multimedia support is the problem, is the UNIX the solution? In *EurOpen Autumn 1991 Conference*. EurOpen, 1991.

[CTB92] W. W. Chu, Ion Tim Ieong R. K. Taira, and C. M. Breant. A temporal evolutionary object-oriented model and its query language for medical image management. In *Proceedings of the 18th International Conference on Very Large Data Base*, 1992.

[HBvR94] Lynda Hardman, Dick C. A. Bulterman, and Guido van Rossum. The Amsterdam Hypermedia Model: Adding time and context to the Dexter model. *The Communication of the ACM*, 37(2):50–62, Feb. 1994.

[HS94] Frank Halasz and Mayer Schwartz. The DEXTER Hypertext Reference Model. *The Communication of the ACM*, 37(2):30–39, Feb. 1994.

[JLB94] Yinhe Jiang, Xiangning Liu, and Bharat Bhargava. Reevaluating indexing schemes for nested objects. Submitted to CIKM '94, New Jersey, May 1994.

[LG90] Thomas D. C. Little and Arif Ghafoor. Synchronization and storage models for multimedia objects. *IEEE Journal on Selected Areas in Communications*, 8(3):413–427, Apr. 1990.

[LG93] Thomas D. C. Little and Arif Ghafoor. Interval-based conceptual models for time-dependent multimedia data. *IEEE Transactions on Knowledge and Data Engineering*, 5(4):551–563, Aug. 1993.

[Li94] Shunge Li. A model for multimedia database systems. Master's thesis, Department of Computer Sciences, Purdue University, May, 1994.

[LY93] K. Lakshman and Raj Yavatkar. Polyscheuse: An intergrated framework for distributed multimedia applications. In *Proceedings of IEEE Workshop on Advances in Parallel and Distributed Systems (PADS)*, pages 64–69, Princeton, NJ, Oct. 1993. IEEE.

[Mas91] Yoshifumi Masunaga. Design issues of OMEGA: An object-oriented multimedia database management system. *Journal of Information Processing*, 14(1):60–74, 1991.

[OT93] E. Oomoto and K. Tanaka. OVID: Design and implementation of a video-object database system. *IEEE Transactions on Knowledge and Data Engineering*, 5(4):629–643, Aug. 1993.