OVID: Design and Implementation of a Video-Object Database System

Eitetsu Oomoto and Katsumi Tanaka, *Member, IEEE*

***Abstract*—This paper describes a video-object data model and the design and implementation of our prototype video-object database system named OVID based on the model. Notable features of our video-object data model are to provide (1) a mechanism to share common descriptional data among video-objects, called "interval-inclusion based inheritance," and (2) operations to composite video-objects. OVID system offers a browsing/inspection tool VideoChart, an adhoc query facility VideoSQL, and a video-object definition tool.**

***Index Terms*—Generalization hierarchy, interval inclusion inheritance, video object, video object composition, video object disaggregation, VideoSQL.**

I. Introduction

RECENTLY, much attention has been focused on multimedia information systems, and the Object-Oriented Database (OODB) Systems [4] are considered as a possible candidate for constructing multimedia information systems because of their modeling power, encapsulation capability, and so on. Many object-oriented data models, prototype systems based on their own object models, and a variety of commercial OODB systems have been developed. Also, recent workstations and high performance personal computers evolved to possess hardwares for handling multimedia data, for example, image compression hardware, and digital signal processor for handling voices.

Multimedia information systems are to store, retrieve, and manage still images, video, and voices, etc. (*multimedia data*). For describing the structure and the semantics of multimedia data, in general, the object-oriented concept is regarded as suitable for data modeling framework. Although some multimedia (database) systems [14] to handle video data have been developed, there have been few reports which investigates data models suitable for video information management in depth.

Especially from the standpoint of the data modeling, the major features provided by OODB systems are

- representation and management of *complex objects*;
- handling *object identities*;
- *encapsulation* of data and associated procedures into *objects*; and
- *inheritance* of attribute structures and methods based on a *class hierarchy*.

These features are considered to be very suitable for data modeling and data management in CAD systems, office information systems, and so on. Our basic question is as follows: Is the modeling power offered by conventional OODB features enough for the multimedia data, especially for the video data? We believe that it is necessary to provide new modeling constructs for video database management because of the following reasons.

1) Video data itself is raw data, and it is created independently from how its contents and its database structure is described later.
2) Meaningful scenes in video data are identified and associated with their descriptional data incrementally and dynamically after the video data is stored as a database. Therefore, it is not easy to identify meaningful scene objects and define necessary attributes for describing them when the video database schema is defined. It is desirable that each scene object can have an arbitrary attribute structure suitable for describing the contents when it is identified as a meaningful scene. But most current OODB's require predefinition of those attribute structures, and they do not support enough *schema evolution* facilities such as adding and dropping attributes.
3) Meaningful scenes are sometimes overlapped or included by other meaningful scenes. It becomes important to provide a mechanism to share some descriptional data among meaningful scenes with the inclusion relationships. In order to realize such an inheritance mechanism, the inclusion relationships among instance objects should be considered. But many OODBMS's support only class-based inheritance.

This paper describes a data model called *Video Object Data Model* for modeling and retrieving video data, and the design and implementation of a video database prototype system, named **OVID**: Object-oriented Video Information Database. The notable features of our video object data model follow.

- We introduce the notion of *video object*. We can identify an arbitrary video frame sequence (a meaningful scene) as an independent object and describe its contents in a dynamic and incremental way. That is, each video object has its own attributes and attribute values to describe the contents.

Manuscript received June 1, 1992; revised December 1, 1992. This research was supported in part by Research Grant on Self-Organizing Information-Base System of the Science and Technology Agency.

E. Oomoto is with the Department of Information and Communication Sciences, Kyoto Sangyo University, Kitaku, Kyoto 603, Japan.

K. Tanaka is with the Department of Computer and Systems Engineering, Kobe University, Nada, Kobe 657, Japan.

IEEE Log Number 9209928.

- Our video object data model is *schemaless*. That is, a traditional class hierarchy of OODB systems is not assumed as a database schema.
- The *inheritance based on interval inclusion relationship* is introduced to share descriptional data among video objects. Intuitively, this means that, if a video frame sequence o_1 includes another video frame sequence o_2, then some of the o_1's attributes and attribute values are automatically inherited by o_2.
- A collection of composition operations for video objects is introduced in order to support editing, authoring and abstraction of video objects.

In Section II, we address several related works. In Section III, we describe several motivating problems and our basic ideas. In Section IV, we describe our video object data model together with illustrative examples. The features of the OVID system are described in Section V. Section VI includes concluding remarks.

II. Related Work

Parkes [14] introduced a mechanism for handling descriptive data for video information in his video-based CAI system. He introduced the notions of *events* and *settings*. An *"event"* is a hierarchical description of a video scene based on PART-OF relationships. Suppose a video scene A shows how to use a micro meter. The event "USING THE MICRO METER" is assigned to A. This is the root of the description. The event "USING THE MICRO METER" consists of 4 sub-events, that is, "REMOVE MICRO FROM CASE," "CLEAN MICRO," "MEASURE METAL," and "RECORD MEASURE." Each event corresponds to some portion of video A. The event "CLEAN MICRO" consists of 4 events, "HOLD MICRO," "LIFT CLOTH," "WIPE ROD," and "REPLACE CLOTH" and so on. The other notion, "settings," corresponds to different representations of the same object in the real world. The binary relations, for instance, `zoom in` or `zoom out`, etc., are defined between these settings. In summary, his model consists of PART-OF relationships among descriptional data and the binary relationships between settings. This model offers an interesting way for organizing a lot of descriptional data of video information, but they did not address the issue of inheritance of descriptional data.

Adiba and Quang [1] have developed the several notions to define, store and manipulate the historical multimedia data. For a given object X in a database, a history is a sequence of the successive values that X took across time. Their main subjects are concerned with historical data (containing *version*) management of general database systems or *multimedia* database systems, especially for multimedia documents including text, images and graphics. They do not focus on the mechanism for handling descriptional data of video information.

The composition of multimedia data is addressed by Little and Ghafoor [9]. The methods to compose multimedia data are classified into two categories: *spatial composition* is the notion of spatial arrangement of data; *temporal composition* is concerned with how to keep the synchronization of data in a process of data presentation. They introduced a *timed Petri net model* to synchronize the data presentation. Their model focuses on the description of *synchronization*[1] and they did not discuss about the mechanism for handling descriptional data of video information.

Woelk, Kim, and Luther [19], [20] discussed the management of multimedia data, especially, a collection of objects for capturing and controlling multimedia data in a conventional object-oriented framework.

MINOS [6] is a multimedia system using an optical disk. It manages text, voices and images. One of the characteristics of MINOS is that voice and text are managed symmetrically because they are both one-dimensional in nature.

Intermedia [16] is one of the most famous hypermedia systems. Although Intermedia has many features to retrieve, manage and present various multimedia document as a hypertext system, the facility for controlling, sharing, and composing video data description is relatively weak.

Hodges, Sasnett, and Ackerman [7], [8], [10] developed the *Athena Muse* system which is a multimedia environment for education. In these papers, they introduced the notion of *multidimensional information*. For an example of one dimensional data, they referred to the video data which is annotated by several text segments that switches with video image synchronously. Muse is an useful system to compose multimedia applications. But they addressed neither the data model issues nor its associated mechanism for handling descriptional data of video information.

EVA [10], [11] is a video annotator system developed by MIT. It provides software researchers with the annotation facility for video. Although EVA is a useful tool to analyze video data, the capability to share descriptional information among annotated video scene is relatively weak. It is not fully addressed what operations are needed to compose/decompose the annotated video scenes.

Allen [2] introduced the interval-based temporal logic as the framework to represent the knowledge and inference concerned with time. He addressed the notion of the constraint propagation. Intuitively, if a condition P holds during a time interval T, then P holds during any subinterval t of T. In our video object model, a similar but different propagation mechanism is introduced, called *inheritance based on interval inclusion relationships.* Some of the pairs of attribute/value attatched on a video-object are inherited by sub-video-objects, and the *inheritable* attributes specified by users.

PATRICIA tree [12] is an algorithm for retrieving arbitrary portions of a symbolic sequence with a given keyword. Users can specify any portions of a symbolic sequence as keywords, but PATRICIA does not provide the facility to describe the semantic contents of those portions as objects.

III. Motivating Problems and Basic Ideas

In this section, we first discuss motivating problems to describe the contents of a video data (for example, a video disc)

[1]We recognize the importance of the synchronization issues of multimedia data, but, in this paper, we focus only on the treatment of descriptional data of video information.

when using conventional OODBMS's. Next, we intuitively describe our approach.

3.1. Problems

3.1.1. Difficulties in Defining Attributes: If we wish to describe the contents of a video disc, and also wish to define a database schema by using conventional OODBMS's, first, we should decide what the basic *objects* are. A video disc consists of a sequence of video frames, and so, each video frame (a still image) can be regarded as a single object. However, usually, a (semantically) meaningful scene is a sequence of (not always continuous) video frames. So, it seems natural to define an object corresponding to each semantically meaningful sequence of video frames. This is possible in the case of using conventional OODBMS's. That is, the description for a semantically meaningful scene is represented as a tuple:

(*starting frame#, ending frame#, other attributes to describe the scene*).

However, we should note the following problems concerned with the difficulties in defining attributes in advance.

- Depending on the describer's viewpoints, the same scene (the same sequence of video frames) may be given different descriptions. That is, they may have different sets of attribute definitions.
- Since it is difficult to describe the whole contents of a video disc at a time, the descriptions for semantically meaningful scenes should be incrementally added. This also causes a difficulty in defining all attributes in advance.
- A semantically meaningful scene s_1 may contain another semantically meaningful scene s_2 as its subinterval. Suppose that the description of s_2 contains the same information as the one which is described for s_1. It is a very tedious task because a user must describe the same information on two different objects. If there is some mechanism, by which the common descriptional information is shared among scenes, based on time-interval inclusion relationship, then it would be useful for decreasing the effort of describing video scenes. But, conventional OODBMS's do not provide such an inheritance mechanism because they provide a mechanism only for the inheritance of attribute structures and methods among classes.

3.1.2. Difficulties in Querying: In conventional database systems, in general, a query is formulated by knowing a database schema such as table definitions or a class hierarchy. Users must know the attribute structures or class structures in order to retrieve desired objects. If we allow each object to have a different attribute structure then the following problems should be considered.

- Since each data object has its own structure, users must inspect the attribute definition of each object in order to know what attributes are defined for objects. It is very inconvenient, and so, some kind of a mechanism to cope with this problem is necessary.
- All the complete definitions of attributes and their values for objects must be described by users if there is no database schema. It may be a very hard task for users to describe many attribute definitions for objects. A mechanism to decrease the amount of descriptions of attribute definition for objects will be required.

3.1.3. Treatment of Composed Objects: Video data stored in video databases may be used for several editorial works. That is, video data are often cut and/or concatenated for preparing multimedia presentation. In order to do this by video database systems, those systems should provide a facility to retrieve video data and compose the resulting data into a new video object as well as a facility for several editiing operations. Also, these newly-composed video objects may need to be stored into the database. Conventional OODBMS's, however, do not provide enough capability to store the retrieved and/or composed objects into their object databases. This is because it needs a class for storing the retrieved and/or composed objects, but conventional OODBMS's are not good at generating a new class dynamically. Also, after composing a new video object from retrieved objects, it is desirable for the new video object to inherit some attribute information from the orginal objects. But, conventional OODBMS's does not support t he inheritance of attributes and their values among instance objects.

3.2. Basic Ideas

We consider that any portion of a video frame sequence is an independent entity, and so, we wish to make it possible to define an object, which has its own attributes and their values for arbitrary video frame sequences. We call that a *video object.* More generally, a video object corresponds to a certain set of video frame sequences, and it has its own attributes and their values to represent the content (meanings) of the corresponding video scene. Following are the features of our video object model:

- schemaless description of database;
- interval inclusion inheritance;
- composition of video objects based on is-a hierarchy.

The *schemaless description* is that we do not take an approach of assuming a specific database schema such as classes and a class hierarchy, so users can define any attribute structure for each video object. For example, broadcast stations such as the ABC Network have many varieties of video tape libraries. These may be news reports such as an airplane accident, movies or documentaries, and so on. We believe that it is very difficult to offer a common attribute structure for them. Also, users have different viewpoints to describe and/or retrieve video scenes, and many various requirements for retrieving from their various standpoints. Therefore, it is difficult to decide rigidly the attribute structures which denote the meanings of those video scenes. Also, they should be extensible freely and incrementally by users.

As new objects are defined in a database, new attributes to represent the contents of video objects may become needed. In our model, arbitrary attributes can be attached to each video object whenever it is necessary. For instance, when a user defines a video object over the video scene concerned with *John*, the user may want to describe his name, "John," as the attribute value of the *name* attribute. If the *name* attribute does not exist in the database, we can add it to this video object at any time.

We also introduce the notion of inheritance based on the *interval inclusion relationship*. By means of this notion, some descriptional data of video objects can be inherited to other video objects. For instance, consider that a night scene is defined as a video object *A* and another object *B* is defined over some portion of object *A*. The object *B* is also a night scene. If object *A* has attribute "Situation" and its value "night," then object *B* has the same attribute and its value by the *interval inclusion relationship.*

We will also define several operations, *interval projection, merge* and *overlap*, for video objects to compose new video objects. For example, in broadcast stations, video scenes are edited from source video. They are chopped in order to remove redundant scenes and/or concatenated with each other to make TV programs. Our operations for video objects correspond to this kind of video editing. They do not only synthesize a new video object, but also derive, based on the is-a hierarchy, the attributes and their values of the synthsized one from the original video objects.

IV. Video Object Data Model

In this section, we describe our *video object data model*, by which we have designed and implemented our prototype video-object database system.

4.1. Video-Objects

Intuitively, our notion of *video-object* is a descriptional data of a meaningful scene (motion picture), and it consists of 1) its *object identifier* (oid) 2) an *interval*, and 3) a collection of attribute/value pairs. Each video-object has a unique object identifier. Each video-object corresponds to a video-frame sequence, and the contents of the video-frame sequence is described by a collection of attribute/value pairs. An *interval* is represented by a pair of a starting frame# and an ending frame#, and denotes a continuous sequence of video-frames. In this paper, since we define a video-object which corresponds to more than one video-frame sequence, a set of *intervals* is associated with the corresponding video-object. So, note that a video-object does not necessarily correspond to a single continuous sequence of video frames. This is because a meaningful scene does not always correspond to a single continuous sequence of video frames.

In this paper, we assume the following mutually disjoint (countably infinite) sets: A set $\mathcal{D}$ of atomic values (numbers, strings and special symbols $\top$ and $\bot$), a set $\mathcal{ID}$ of *object identifiers*, a set $\mathcal{I}$ of intervals, and a set $\mathcal{A}$ of attribute names.

Definition 4.1: *Video-Object* An *video-object* is a triple (oid, I, v) where

1) *oid* is an object identifier which is an element of $\mathcal{ID}$;
2) I is a finite subset of $\mathcal{I}$;
3) v is an n-tuple $[a_1 : v_1, \ldots, a_n : v_n]$, where each a_i $(1 \leq i \leq n)$ is an attribute name in $\mathcal{A}$, and v_i is a value defined recursively in the following manner:
   a. each element $x \in \mathcal{D}$ is a value
   b. each interval $i \in \mathcal{I}$ is a value
   c. for values $v_1, \ldots, v_n (0 \leq n)$, $\{v_1, \ldots, v_n\}$ is a value called a *set value*
   d. each video-object is also a value.

For a given video-object, $o = (oid, I, v)$ with its value $v = [a_1 : v_1, \ldots, a_n : v_n]$, $attr(o)$ or $attr(v)$ denotes the set of all the attributes in v, and $value(o)$ denotes the value v, that is, $v = value(o)$. The value v_i is denoted by $o.a_i$ and/or $v.a_i$.

Hereafter, as a realistic example, we will use a documentary video disc[17] about `Prime Ministers of Japan`. Fig. 1 summarizes the whole example database.

4.2. Generalization Hierarchy for Values and Objects

A *generalization (is-a) hierarchy* for atomic values is assumed to be a lattice $G = (A, \succeq, \top, \bot)$, where A is a set of atomic values, $\succeq$ is an *is-a* relationship (a reflexive, transitive and anti-symmetric binary relation) among atomic values, $\top$ denotes *unknown*, and $\bot$ denotes *undefined.* The binary relation $\succeq$ denotes *more informative*. That is, if $a_1 \succeq a_2$ means that a_1 is-a a_2 and that the atomic value a_1 is more informative than the atomic value a_2. Fig. 1 shows an example generalization hierarchy of atomic values of our example database. In this figure, the symbol $\top$ is denoted by *nil*, and the symbol $\bot$ is omitted. In this example, the following holds:

- `Japanese statesman` $\succeq$ `statesman`,
- `Eisaku Sato` $\succeq$ `Japanese statesman`,
- `Eisaku Sato` $\succeq$ `statesman`.

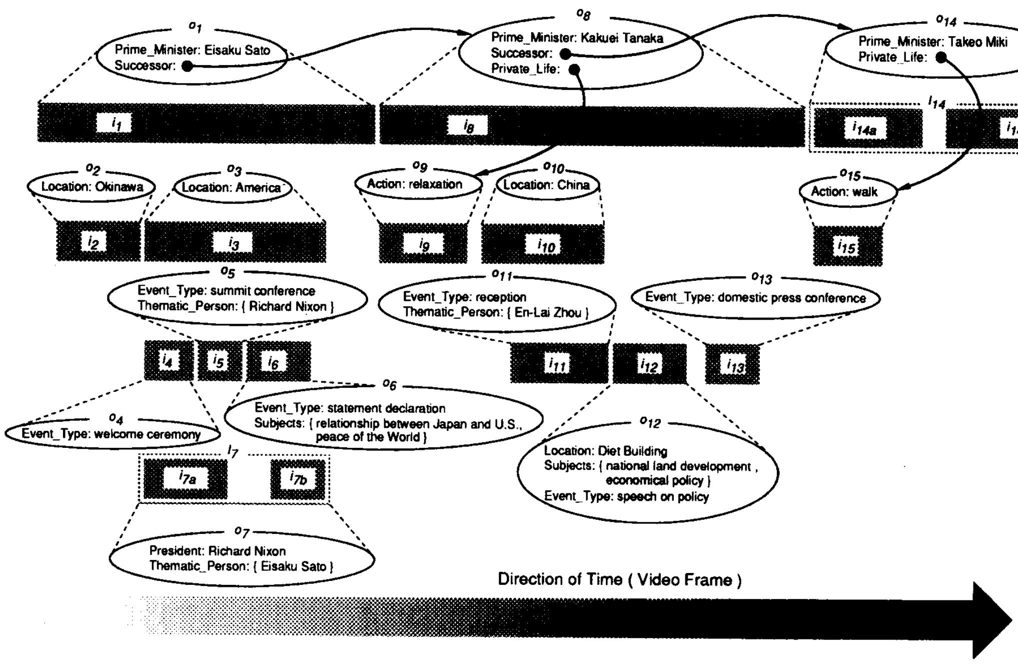
We assume that a *generalization* (*is-a*) [15] hierarchy for atomic values is given in advance by users.

The above *is-a* relationship can be extended to general values (including atomic values) and also to video-objects in the following manner: Suppose that we have two video-objects $o_1 = (oid_1, I_1, v_1)$ and $o_2 = (oid_2, I_2, v_2)$. First, given these two values v_1, v_2, two video-objects o_1, o_2 and a generalization hierarchy G, we extend the is-a relationship[2] in the following recursive manner.

- If both of v_1 and v_2 are set-type values, then v_1 is-a v_2 if for each element y in v_2, there exists an element x in v_1 such that x is-a y.
- Let v_1 and v_2 be two tuple-type values. Then v_1 is-a v_2 if $v_1.a$ is-a $v_2.a$ for each attribute a of v_2.
- For video-objects[3] $o_1 = (oid_1, I_1, v_1)$ and $o_2 = (oid_2, I_2, v_2)$, if v_1 is-a v_2, then o_1 is-a o_2.

[2]This extension follows the work by Khoshafian *et al.* [5]

[3]In this paper, we assume that the *is-a* relationship between video-objects has nothing to do with their object identifiers and intervals.



o_1
Prime_Minister: Eisaku Sato
Successor: ●
o_8
Prime_Minister: Kakuei Tanaka
Successor: ●
Private_Life: ●
o_{14}
Prime_Minister: Takeo Miki
Private_Life: ●
i_1
i_8
I_{14}
i_{14a}
i_{14b}
o_2
Location: Okinawa
o_3
Location: America
o_9
Action: relaxation
o_{10}
Location: China
o_{15}
Action: walk
i_2
i_3
i_9
i_{10}
i_{15}
o_5
Event_Type: summit conference
Thematic_Person: { Richard Nixon }
o_{11}
Event_Type: reception
Thematic_Person: { En-Lai Zhou }
o_{13}
Event_Type: domestic press conference
i_4
i_5
i_6
i_{11}
i_{12}
i_{13}
o_6
Event_Type: statement declaration
Subjects: { relationship between Japan and U.S., peace of the World }
o_4
Event_Type: welcome ceremony
I_7
i_{7a}
i_{7b}
o_{12}
Location: Diet Building
Subjects: { national land development, economical policy }
Event_Type: speech on policy
o_7
President: Richard Nixon
Thematic_Person: { Eisaku Sato }
Direction of Time (Video Frame)

Fig. 1. Example video object database.

Definition 4.2 Least Upper Bound of Values: For two values v_1 and v_2, $v = v_1 \vee v_2$ is the *least upper bound* of v_1 and v_2, if the following conditions hold.

1) v_1 is-a v,
2) v_2 is-a v, and
3) There does not exist any v' such that $v \neq v'$, v' is-a v, v_1 is-a v', and v_2 is-a v'.

For example, suppose that we have the following values:

v_1 = [**Prime_Minister**: Kakuei Tanaka, **Location**: Tokyo, **Action**: relaxation],

v_2 = [**Year**: 1974, **Location**: Tokyo, **Action**: walk].

v_2 = [**Year**: 1974, **Location**: Tokyo, **Action**: walk].

Also, assume the following is-a relationships:

relaxation is-a *daily life* and *walk* is-a *daily life.*

The least upper bound of v_1 and v_2 is shown as follows:

$v_1 \vee v_2$ = [**Location**: Tokyo, **Action**: daily life].

Intuitively, the obtained value $v_1 \vee v_2$ represents a ***maximum*** information that is common to both of the values v_1 and v_2. This notion will be used to define our composition operations for video-objects later.

Definition 4.3.:Greatest Lower Bound of Values: For two values v_1 and v_2, $v = v_1 \wedge v_2$ is the *greatest lower bound* of v_1 and v_2, if the following conditions hold:

1) v is-a v_1,
2) v is-a v_2, and
3) there does not exist any v' such that $v \neq v'$, v is-a v', v' is-a v_1 and v' is-a v_2.

Suppose that

v_1 = [**Location**: Tokyo, **Year**: 1972, **Subjects**: { national land development, financial problem }],

v_2 = [**Year**: 1972, **Subjects**: { financial problem, social welfare }].

The greatest lower bound of v_1 and v_2 is:

$v_1 \wedge v_2$ = [**Location**: Tokyo, **Year**: 1972, **Subjects**: { national land development, financial problem, social welfare }].

Intuitively, the obtained value $v_1 \wedge v_2$ represents a ***minimum*** information that contains both of the values v_1 and v_2. This notion will be also used to define our composition operations for video-objects later.

Example 4.1: In Fig. 1, we assume that the interval i_8 of a video-object o_8 corresponds to the scene of the Prime Minister `Kakuei Tanaka`, and that the interval set I_{14} of o_{14} corresponds to the scene concerned with Mr. `Takeo Miki`, who was the next prime minister, that is, *successor,* of `Kakuei Tanaka`. Also, suppose that the scene i_{15} shows that Mr. `Takeo Miki` is taking a walk around his house in his private time. We can define the following video-objects:

$$o_8 = (d_8\{i_8\}, v_8)$$

where

v_8 = [**Prime_Minister** : Kakuei Tanaka, **Successor** : o_{14}, **Private_Life** : o_9]

$o_9 = (d_9, \{i_9\}, v_9)$ where v_9 = [**Action** : relaxation]

$o_{14} = (d_{14}, I_{14} = \{i_{14a}, i_{14b}\}, v_{14})$, where

v_{14} =

[**Prime_Minister** : Takeo Miki, **Private_Life** : o_{15}]

$o_{15} = (d_{15}, \{i_{15}\}, v_{15}$, where v_{15} = [**Action** : walk].

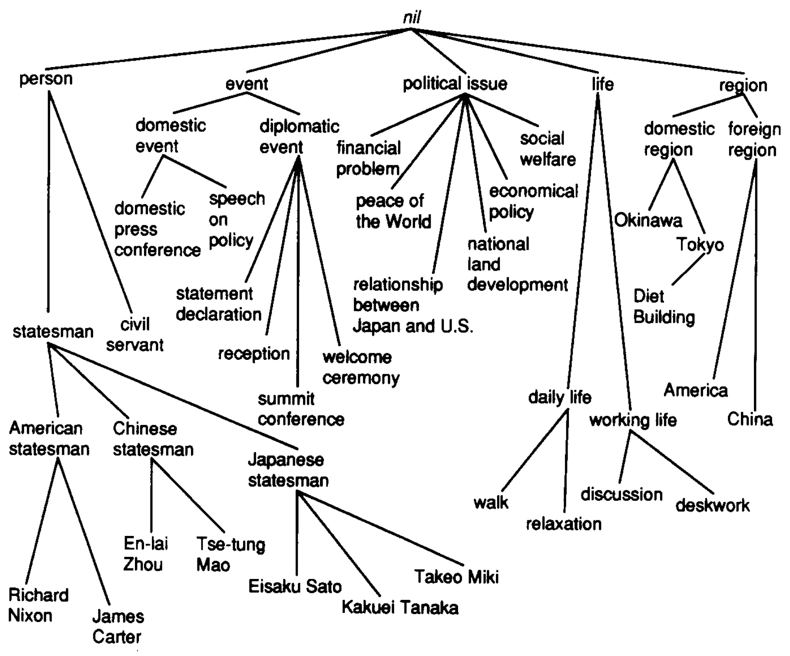


Fig. 2. Example generalization hierarchy of atomic values.

When describing the contents of a video scene, it is sometimes difficult to determine the value of an attribute in a textual form, for instance, the `private life` can be better described by a video image itself rather than textual sentences. Note that in our model, a video-object is allowed to be used as an attribute value (for example, the **Private_Life** value of o_8 is another video-object o_9).

4.3. Inheritance Based on Interval Inclusion Relationship

In this subsection, we will introduce a new inheritance mechanism, by which some attribute/value pairs of a video-object are inherited by another video-object if the former video-object's interval contains the latter video-object's interval. It should be noted that this type of inheritance is between *instances*, not between traditional classes and subclasses in OODB systems.

Since our video-object may contain more than one intervals as its component, first, we define the inclusion relationship between two sets of intervals. For given two interval sets, I_1 and I_2, the *inclusion relationship* between them is defined as:

- For each $i \in I_1$, if there exists $i' \in I_2$ such that $i \subseteq i'$, then I_1 is said to be *included* by the interval set I_2, denoted by $I_1 \sqsubseteq I_2$.

Suppose that we have video-objects $o_1 = (oid_1, I_1, v_1)$ and $o_2 = (oid_2, I_2, v_2)$ such that $I_1 \sqsubseteq I_2$ holds. Then, some of the attribute/value pairs of v_2 are inherited by the video-object o_1. As will be discussed later, not all attribute/value pairs should be inherited. The *inheritable attributes* should be defined in advance by users.

TABLE I
INHERITABLE ATTRIBUTES IN THE EXAMPLE VIDEO DATABASE

Inheritable Attributes	Event_Type, Location, President, Prime_Minister, Thematic_Person, Year
Non-inheritable attributes	Action, Private_Life, Subjects

Definition 4.4 Evaluation of Interval Inclusion Inheritance by a Single Object: Suppose that we have video-objects $o_1 = (oid_1, I_1, v_1)$ and $o_2 = (oid_2, I_2, v_2)$, such that $I_1 \sqsubseteq I_2$, and a set A of inheritable attributes. The result of applying the interval inclusion inheritance is called an *evaluation*. The *evaluation* of o_1 by o_2 and A, denoted by $eval(o_1, o_2, A)$, is a video-object $o'_1 = (oid_1, I_1, v'_1)$ such that

- $attr(v'_1) = attr(v_1) \cup (attr(v_2) \cap A)$,
- $v'_1.a = v_1.a$ if a is in attr(v_1), and
- $v'_1.a = v_2.a$ if a is in attr(v_2) $\cap A$, but not in attr(v_1).

It should be noted that the object identifiers and the interval sets of o_1 and o'_1 are the same. This means that the *evaluation* is not physically stored in the database and is dynamically calculated.

Definition 4.5 Evaluation of Interval Inclusion Inheritance by Multiple Objects: Suppose that we have a video-object $o_1 = (oid_1, I_1, v_1)$. Let $O = \{o_{21}, \ldots, o_{2m}\}$ be the set of all video-objects in the database such that $I_1 \sqsubseteq I_{2i}$ for each $o_{2i} = (oid_{2i}, I_{2i}, v_{2i})$. The *evaluation* of o_1 by $O - \{o_1\}$ and a set A of inheritable attributes, denoted by $eval(o_1, O, A)$, is a video-object $o'_1 = (oid_1, I_1, v'_1)$ such that

- $attr(v_1') = attr(v_1) \cup (attr(v_{21}) \cap A) \cup \ldots \cup (attr(v_{2m}) \cap A)$,
- $v_1'.a = v_1.a$ if a is in attr(v_1), and
- For each a in A, let $V_a = \{v_{2i}.a \mid a \in attr(v_{2i}), a \notin attr(v_1)$, and $1 \leq i \leq m\}$. For each nonempty set V_a, $v_1'.a =$ glb(V_a), where glb(V_a) denotes the greatest lower bound of all the elements in V_a.[4]

Example 4.2: Let's consider another example. Suppose that we have the following video-object:

$o_{12} = (oid)\{i_{12}\}, v_{12}$ such that
$v_{12} =$[**Event_Type**]: speech on policy,
Location: Diet Building,
Subjects: {national land development, economical policy}].

We do not consider that the attribute **Subjects** is an *inheritable* attribute because of the following reason. The attribute **Event_Type** denotes the kind of an event which occurred in the interval i_{12}, **Location** denotes the place where the event occured, and **Subjects** denotes the contents of a prime minister's speech: for example, { national land development, economical policy }. Assume that another object, namely o', is defined over the interval i' which is a subinterval of i_{12}, and that o' does not have any attributes in attr(v_{12}). The set-type value { national land development, economical policy } represents the whole contents of the speech. Since i' is a subportion of i_{12}, he might not refer to some of these subjects, for instance, `national land development`. Therefore, it is not suitable, we believe, that the attribute **Subjects** and its value of o_{12} are inherited by o'. On the other hand, it seems to be reasonable that the **Event_Type** and **Location** attributes and their values of o_{12} should be inherited by o' since i' is also a video scene of the speech in the Diet Building. Therefore, the attributes **Event_Type** and **Location** can be declared as inheritable attributes.

Intuitively, the inheritable attribute of a video-object o_i is the attribute such that its value is valid at an arbitrary time point in the interval (set) of o_i. We assume the inheritable attributes shown in Table 1 in our example video database.

4.4. Composition Operations for Video-Objects

In this subsection, we introduce basic operations to compose new video-objects from existing video-objects.

First, we define the *interval projection* operation.

Definition 4.6 Interval Projection Operation: Let $o = (oid, I, v)$ and I' be a given video-object and an interval set, respectively such that $I' \sqsubseteq I$ holds. Also, let A be a set of inheritable attributes. The *interval projection* of o onto I' is a video-object $o' = (oid', I', v')$ such that oid' is a new object identifier, and the value v' satisfies the following:

- $attr(v') = attr(v) \cap A$, and
- $v'.a = v.a$ for each attribute a in attr(v').

[4]Intuitively, V_a denotes a set of candidate values of attribute a, one of which is actually inherited by o_1'. In our definition, the greatest lower bound among V_a, that is, the most informative value is inherited by o_1'. In OVID system described in Section V, the evaluation is done according to this definition.

The interval projection operation is useful when defining a new video-object for a certain portion of a scene corresponding to an already existing video-object since the descriptional data of the existing video-object is automatically inherited and the amount of description can be decreased.

Next, we will define two important composition operations called *merge* and *overlap*. In order to describe these definitions, we need the notion of *merge* and *overlap* of two *interval sets*, which are defined as follows.

Definition 4.7 Merge and Overlap of Interval Sets: For two interval sets, I_1 and I_2, the *merge* and the *overlap* of I_1 and I_2, denoted by $I_1 \sqcup I_2$ and $I_1 \sqcap I_2$, respectively, are defined as:

- $I_1 \sqcup I_2$ is the minimal set of intervals such that
  1. For each interval i in I_1, there exists an interval i' in $I_1 \sqcup I_2$ such that $i' \supseteq i$.[5]
  2. For each interval i in I_2, there exists an interval i' in $I_1 \sqcup I_2$ such that $i' \supseteq i$.
  3. For every interval i_1 and i_2 in $I_1 \sqcup I_2$ such that $i_1 \neq i_2$, $i_1 \not\supseteq i_2$ and $i_2 \not\supseteq i_1$.
- $I_1 \sqcap I_2$ is the maximal subset of $\{i_1 \cap i_2 \mid i_1 \in I_1$ and $i_2 \in I_2\}$ such that $i_1' \not\supseteq i_2'$ and $i_2' \not\supseteq i_1'$ for arbitrary intervals i_1' and i_2' in $I_1 \sqcap I_2$.

Intuitively, the *merge* operation creates a new video-object o from existing video-objects o_1 and o_2 such that some descriptional data common to both of o_1 and o_2 is inherited by o, and that o's interval (set) is the union (merge) of the intervals (interval sets) of o_1 and o_2. In other term, the *merge* operation abstracts two existing video-objects into a new video-object. The *overlap* extracts the scene described by both of two existing video-objects as a new video-object.

Definition 4.8 Merge of Video-Objects: The *merge* of two video-objects $o_1 = (oid_1, I_1, v_1)$ and $o_2 = (oid_2, I_2, v_2)$, denoted by $o_1 \sqcup o_2$, is the video-object $o = (oid, I_1 \sqcup I_2, v)$ such that $v = [a_1 : v_1, \ldots, a_i : v_i, \ldots, a_n : v_n]$ where each a_i $(1 \leq i \leq n)$ is in $attr(v_1) \cap attr(v_2)$, and for each a_i,

- If both of $o_1.a_i$ and $o_2.a_i$ are values, then $v_i = o_1.a_i \vee o_2.a_i$.
- If both of $o_1.a_i$ and $o_2.a_i$ are video-objects then $v_i = o_1.a_i \sqcup o_2.a_i$.

Example 4.3: In our example database, we have:

$o_8 = (oid)_8, \{i_8\}$, [**Prime_Minister**: Kakuei Tanaka,
Successor: o_{14},
Private_Life: o_9])

$o_9 = (oid_9, \{i_9\}$, [**Action**: relaxation]),
$o_{14} = (oid_{14}, \{i_{14a}, i_{14b}\}$ [**Prime_Minister**: Takeo Miki,
Private_Life: o_{15}])
$o_{15} = (oid_{15}$, [**Action**: walk]).

Furthermore, assume the following *is-a* relationships:

`relaxation` *is − a* `daily life` and
`walk` *is − a* `daily life`.

[5]$i_1 \subseteq i_2$ for given two intervals i_1 and i_2 denotes that the interval i_1 is completely included by or equal to the interval i_2.

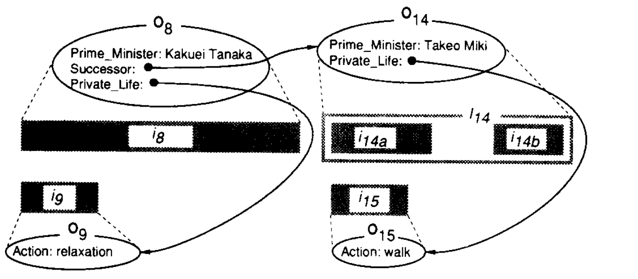


Fig. 3. An example of video objects.

The merge of o_8 and o_{14} is shown as:

$$o_8 \sqcup o_{14} = (oid', \{i_8, i_{14a}, i_{14b}\}, [\textbf{Prime\_Minister} : \text{Japanese statesman } \textbf{Private\_Life } o_9 \sqcup o_{15}])$$

where $o_9 \sqcup o_{15}$ = $(oid, \{i_9, i_{15}\}$, [**Action**: daily life])(See Fig. 2).

It should be noted that the *merge* operation is recursively applied to **Private_Life** attributes since they have video-objects as their values. Intuitively, this example's *merge* operation extracts the common information of two video-objects o_8 and o_{14}, and assigns it to a newly created, resulting video-object. This operation corresponds to an *editting* of films together with their descriptional data in movie companies or broadcast stations.

The final composition operation is *overlap*, which is intuitively to take an intersection of two video-objects. From two existing video-objects o_1 and o_2, this operation creates a new video-object o such that o's interval (set) is the intersection (overlap) of intervals (interval sets) of o_1 and o_2, and that o's descriptional data is roughly equal to the greatest lower bound of the descriptional data of o_1 and o_2.

Definition 4.9 Overlap of Video-Objects: For two video-objects $o_1 = (oid_1, I_1, v_1)$ and $o_2 = (oid_2, I_2, v_2)$ such that $I_1 \sqcap I_2$ is nonempty, the *overlap* of o_1 and o_2, denoted by $o_1 \sqcap o_2$, is a new video-object $o = (oid, I_1 \sqcap I_2, v)$ such that oid is a new object identifier and $v = [a_1 : v_1, \ldots, a_n : v_n]$, where

- each attribute a_i is in $attr(v_1) \cup attr(v_2)$,
- If both of $v_1.a_i$ and $v_2.a_i$ are values[6], then $v.a_i = v_1.a_i \wedge v_2.a_i$.
- If both $v_1.a_i$ and $v_2.a_i$ are video-objects, then $v_i = v_1.a_i \sqcap v_2.a_i$.

Example 4.4: As shown in Fig. 2, suppose that we have:

$o_5 = (oid_5, \{i_5\}$, [**Event-Type**: summit conference, **Thematic_Person**: {Richard Nixon}])

$o_7 = (oid_7, \{i_{7a}, i_{7b}\}$, [**President**: Richard Nixon, **Thematic_Person**: {Eisaku Sato}])

The overlap of o_5 and o_7 is:

$o_5 \sqcap o_7$ = (oid', $\{i_5 \cap i_{7a}\}$, [**Event_Type**: summit conference, **President**: Richard Nixon, **Thematic_Person**: { Richard Nixon, Eisaku Sato }])

[6]If either of $v_1.a_i$ or $v_2.a_i$ is not given in v_1 or v_2, then $\top$ is automatically assigned to $v_1.a_i$ or $v_2.a_i$.

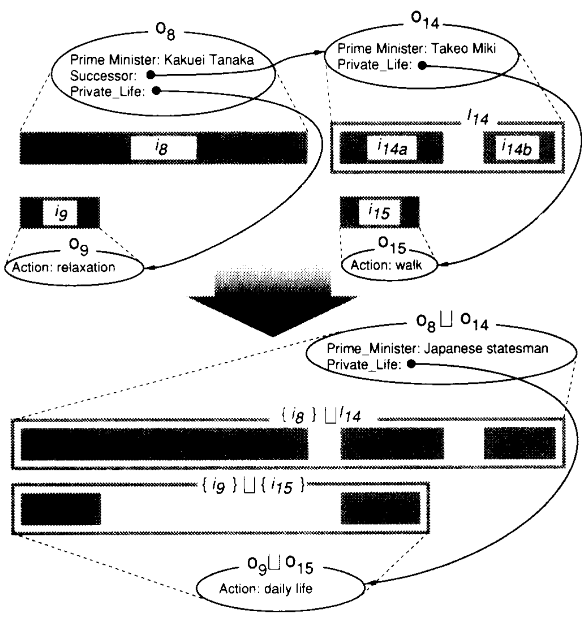
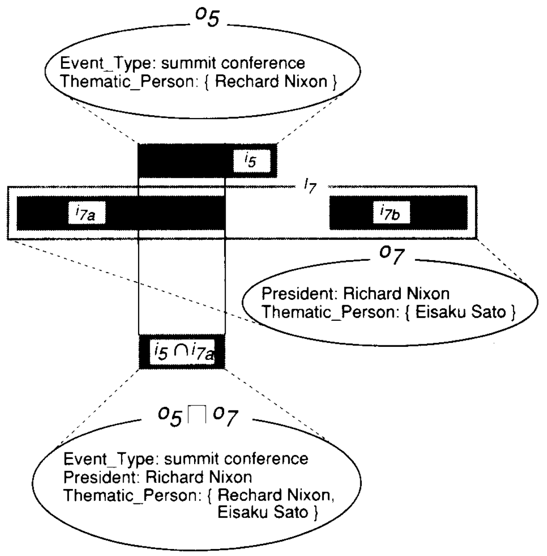


Fig. 4. Merging video-objects.



o_5
Event_Type: summit conference
Thematic_Person: { Rechard Nixon }
i_5
i_7
i_{7a}
i_{7b}
o_7
President: Richard Nixon
Thematic_Person: { Eisaku Sato }
$i_5 \cap i_{7a}$
$o_5 \sqcap o_7$
Event_Type: summit conference
President: Richard Nixon
Thematic_Person: { Rechard Nixon, Eisaku Sato }

Fig. 5. Overlaping video objects.

Note that in this example, the greatest lower bound of `Thematic_Person` values of o_5 and o_7 corresponds to the set union of `Thematic_Person` values of o_5 and o_7.

V. OVID: A VIDEO-OBJECT DATABASE SYSTEM

In this section, we will describe the overall configuration and the functionalities of our video-object database system

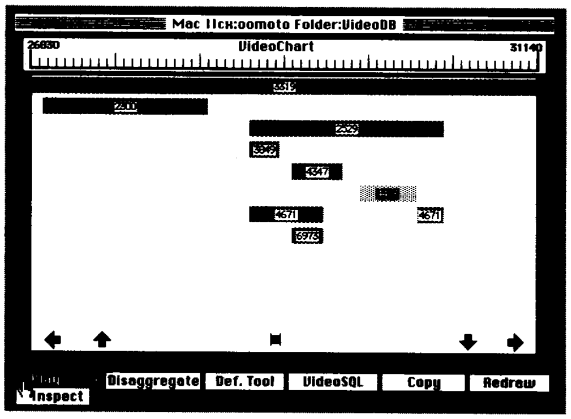
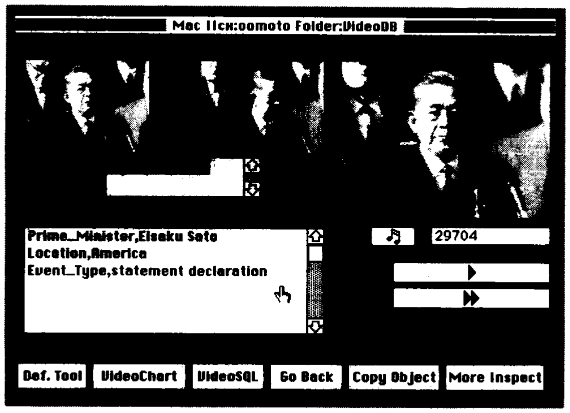


Fig. 6. VideoChart.



Prime_Minister,Eisaku Sato
Location,America
Event_Type,statement declaration

(a)

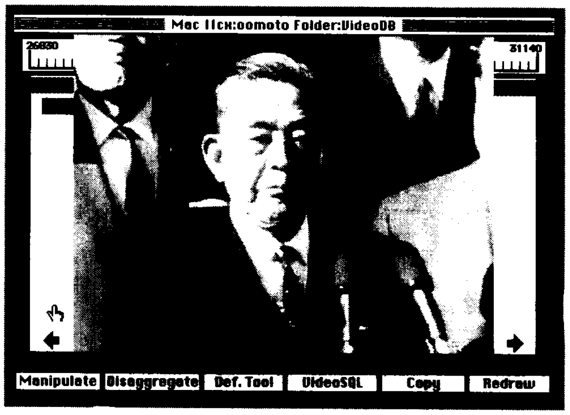
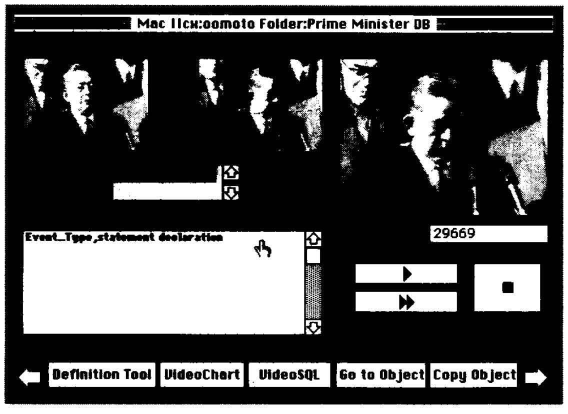


Fig. 7. *Play* operation for a video object.



Event_Type,statement declaration

(b)

Fig. 8. (a) Inspection of a video object. (b) The original definition of the object id: 4423.

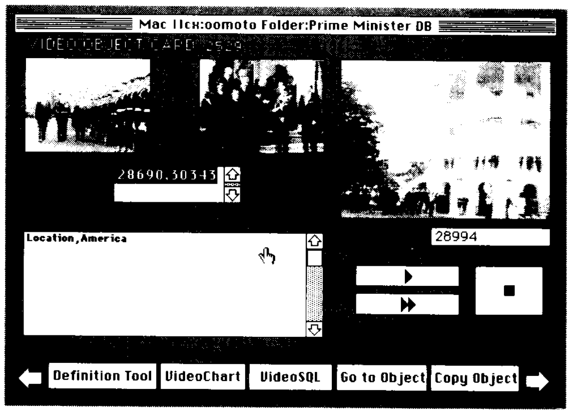
called OVID, which was developed based on our video object data model and is currently running.

5.1. Features of OVID

The following is a list of notable features of our OVID system.

- **Video Objects as Central Units** The *video-objects* described in Section IV are the central units of the OVID system. Each video-object intuitively consists of (1) the *oid* of the video-object, (2) a set of pairs of starting video frame number and an ending video frame number of the video scene, (3) a set of attribute/value pairs which describe the contents of the corresponding video frame sequence, and (4) basic methods such as `play`, `inspect`, and `disaggregate` operations.[7] Note that users can define two different video-objects for the same sequence of video frames since their *oid*'s can be different. Each video-object is represented as a bar chart in our OVID user interface called VideoChart. Through the VideoChart interface, users can play/inspect/decompose a video-object, and define/compose new video-objects.
- **Dynamic and Incremental Object Identification and Definition** Since our video object data model does not assume the existence of a database schema, users should be able to identify a meaningful scene at any time and define the meaningful scene with its descriptional data as a video-object. So, the system should facilitate users to add necessary attributes as well as to use attributes of predefined video-objects.
  Currently, the attributes and their values are given to each objects interactively in manual.
- **Video Object Composition** A meaningful scene does not always correspond to a single consecutive sequence of video frames. So, the system should facilitate definition of a video-object which corresponds to more than one consecutive sequence of video frames. This operation corresponds to the *merge* operation described in Section IV. In the case of merging two video-objects into another new video-object, some descriptional data of these two video-objects may also be able to be used as descriptional data of the newly merged object. This kind of inheritance

[7] In Section IV, we did not refer to methods of video-objects. In the OVID, the methods are encapsulated into each video-object.

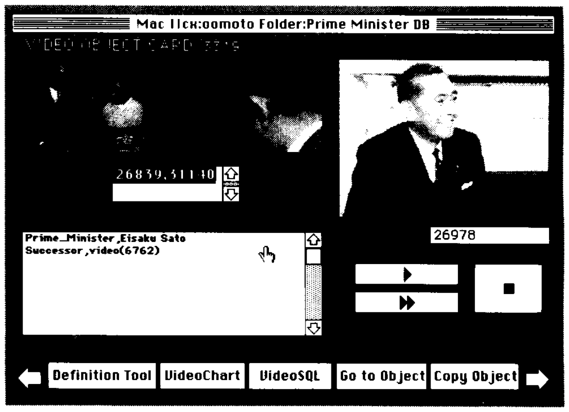


28690,30343

Location,America

28994

(c)



26839,31140

Prime_Minister,Eisaku Sato
Successor,video(6762)

26978

(d)

Fig. 8 (*cont'd.*) (c) The definition of the object with id: 2529. (d) The definition of the object with id:3319.

of descriptional data is achieved in our video object data model. Furthermore, OVID supports creation of a video-object by applying the *overlap* operation to predefined video-objects. Some descriptional data of the predefined video obejcts is also inherited by the *overlapped* video-object.

- **Browsing by Video Object Disaggregation** In order to browse a large[8] video-object, OVID provides a decomposition of a video-object into smaller video-objects. This decomposition corresponds to the *interval projection* operation described in Section 4. In OVID, the operation is called *disaggregation*, and the result of the disaggregation operation is a collection of smaller video-objects, each of which inherits some descriptional data from the original video-object. The disaggregation operation can be repeatedly applied, and so, this operation is useful to find a necessary scene contained by a large video-object in a navigational manner.
- **Generalization Hierarchy for Atomic Values** OVID supports the usage of a generalization (is-a) hierarchy, which consists of atomic values that are used as attribute values of video-objects, in both of the phases of video-object creation and retrieval of video-objects.
- **Video Objects as Attribute Values** As described in the video object model in Section IV, our model allows a video-object itself to be an attribute value of an other video-object. This is very useful when it is difficult to describe a meaningful scene by text.
- ***Ad hoc* Query Facility for Video Objects** OVID has an adhoc query facility, called *VideoSQL*. It facilitates retrieval of a collection of video-objects that satisfy a given condition. The inheritance based on the *interval inclusion relationships* is supported, and so, users can formulate their queries as if the stored video-objects had already inherited necessary descriptional data from larger video-objects. Also, VideoSQL supports queries to retrieve video-objects that contain a specified video-object as their attribute value.

The OVID system consists of the following components:

- VideoChart: A bar-chart type, visual interface for manipulating video-objects.
- VideoSQL: An adhoc query facility to retrieve video-objects.
- Video Object Definition Tool: A facility for object definition.

Each of these components will be described in the following subsections.

5.2. VideoChart

VideoChart has the following facilities:

1) Browsing video-objects in a bar-chart form.
2) Playing video-objects as live video.
3) Inspecting and updating video-objects.
4) Composing (merge and overlap) video-objects.
5) Decomposing (disaggregating) video-objects.
6) Moving to the video object definition tool and VideoSQL.

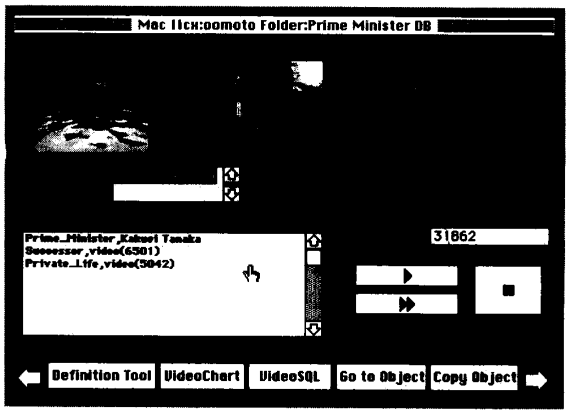
5.2.1. Browsing of a Video Database By VideoChart, we can view the contents of a video database[9] in a visual form.

The abscissa denotes the sequence of video frame ID's (time-axis), and the numerics placed at the top left and right corners denote the current range of the frame ID#. A collection of video-objects appearing in the specified range are displayed in bar-chart form. Each line denotes a single video-object. Fig. 2 denotes that the currently displayed range is from frame# 26 830 to frame#31 140. The frame range can be scrolled with the arrow button, and its scale can be changed to an arbitrary range.

The number located in the center of each bar denotes the object identifier of the video-object. If a video-object consists of more than one frame sequence (for instance the object id 4671), then those component video sequences are drawn in the same line.

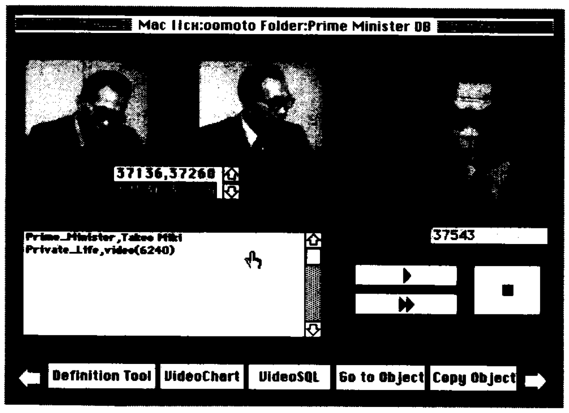
[8]Here, the term *large* means a video-object corresponding to a long sequence of video frames.

[9]Currently, OVID can handle one video database at a time. The name of a video database for the current session should be specified at system startup time.



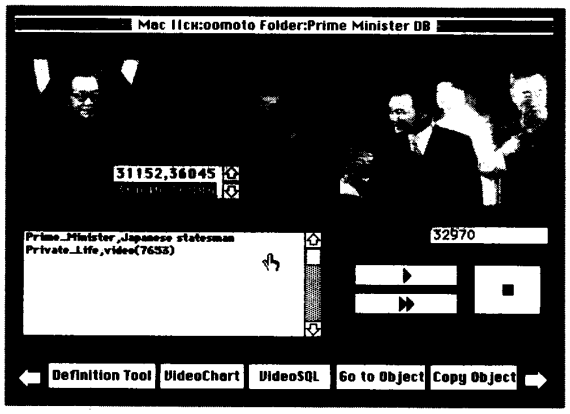
Prime_Minister,Kakuei Tanaka
Successor,video(6501)
Private_Life,video(5042)

(a)



Prime_Minister,Takeo Miki
Private_Life,video(6240)

(b)



Prime_Minister,Japanese statesman
Private_Life,video(7653)

(c)

Fig. 9. (a) Definition of the object with id:6762. (b) Definition of the object with id:6501. (c) Definition of the *merge* of two objects.

Users can select an arbitrary video-object among these objects by clicking with the pointing device. After a video-object is selected, several operations can be applied by clicking the buttons below the chart. If a user clicks the Copy button, then the selected video-object is copied into a buffer in order to form VideoSQL queries later or to define another video-object.

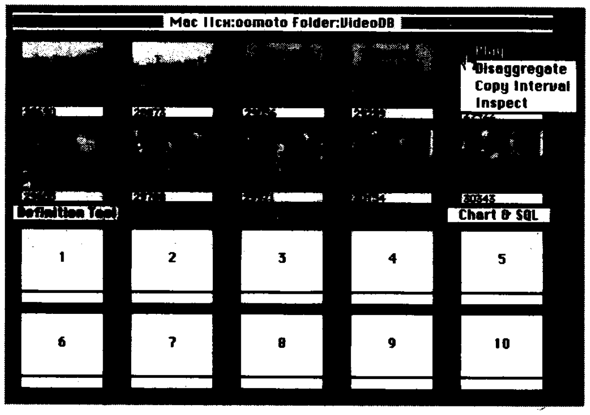


Fig. 10. *Disaggregation* of a video-object.

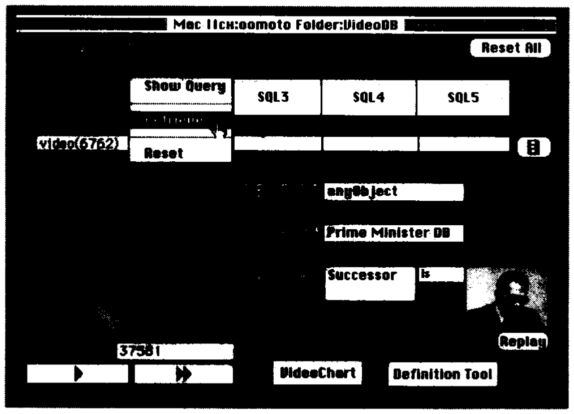


Fig. 11. Video SQL.

As the applicable operations (methods) for video-objects, VideoChart offers the `play`, `inspect`, and `disaggregate` operations.

Fig. 7 shows the example of the *play* operation for a certain video-object. The `play` operation just replays the selected video-object on the monitor screen as live video.

The `inspect` operation first *evaluates* the specified video-object based on *interval inclusion inheritance*, and displays the attributes and the attribute values (including the inherited ones) of the video-object. For example, as shown in Fig. 8(a), the video-object with id:4423 is defined for the range from frame# 29 635 to frame# 30 110. The window entitled `Evaluated Value` shows a list of attribute/value pairs. Each line in this window denotes an attribute name followed by its value. In this example, the evaluation of this video-object has three attributes, `Prime_Minister`, `Location` and `Event_Type`. Fig. 8(b) shows the original attribute/value pair which was defined for this video-object. Originally, this video-object has one attribute `Event_Type` and its value `statement declaration`. The interval of this video-object is included by video-objects with id:2529 and id:3319. The contents of the video-objects with id:2529 and with id:3319 are shown in Fig. 8(c), and Fig. 8(d), respectively. The object with id:2529

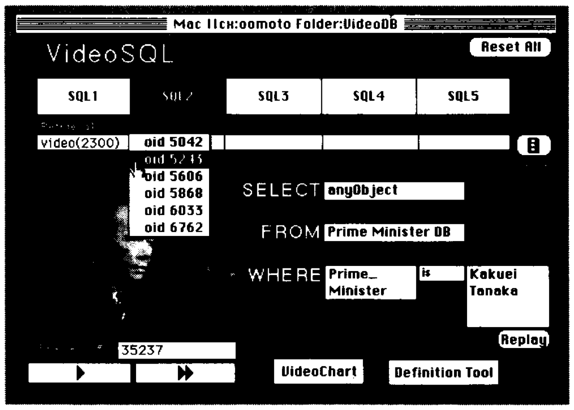


Fig. 12. The selection of one of the retrievals.

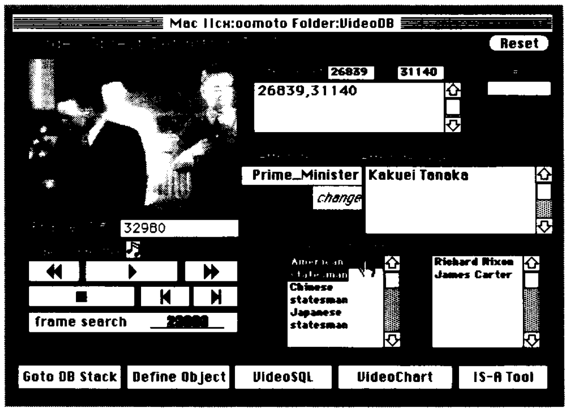


Fig. 13. Video object definition tool.

has the attribute Location, which is defined as *inheritable*[10]. The object with id: 3319 has the attributes Prime_Minister and Successor, where Successor is *noninheritable*, and Prime_Minister is *inheritable*. So, the attribute/value pairs of Location and Prime_Minister are inherited to the video-object with id:4423. This example is the actual example of *interval inclusion inheritance* referred to in Section IV.

5.2.2. Merge and Overlap Operations in VideoChart: After a user selects two arbitrary video-objects, the user can apply the operations *merge* and *overlap* to those video-objects. Suppose that a user select the two video-objects with id:6501 and id:6762, and apply the merge operation to them. Then, as the result of merging these two video-objects, a new video-object, in this case, with id:7757, is created. This example corresponds to Example 4.3 in Section IV.

The video-objects with id:6762 (concerned with Mr. Kakuei Tanaka) and with id:6501 (concerned with Mr. Takeo Miki) are shown in Fig. 9(a) and Fig. 9(b), respectively. The former video-object(id:6762) has three attributes Prime_Minister, Successor and Private_Life. Its Successor value is another video-object(id:6501), and it is denoted by a string expression: video(6501). Also, its Private_Life value is the video-object with id:5042. The latter video-object (id:6501) has attributes Prime_Minister and Private_Life. Its Private_Life value is also the video-object with id:6240. The resulting video-object (with id:7757) of *merging* the objects with id:6762 and with id:6501 is shown in Fig. 9(c). Note that the Prime_Minister value of this video-object becomes Japanese statesman based on the generalization hierarchy shown in Section 4 since the Prime_Minister values of the original video-objects are Kakuei Tanaka and Takeo Miki. Since the {merge} operation is applied to attribute values recursively, the objects with id:5042 and with id:6240 are also merged and a new video-object, in this case, with id:7653, is automatically created. Intuitively speaking, the video-object with id:7757 is a video scene concerned with two Japanese prime ministers, and the video-object with id:7653 is the private life of these two Japanese prime ministers.

Users can also apply the *overlap* operation to objects in a similar manner.

5.2.3. Disaggregation of a Video-Object: In OVID, an arbitrary video-object is automatically divided into ten sub-video-objects when the operation disaggregate is applied to it. Fig. 10 shows that Video Disaggregation System as just launched. In this example, a video-object defined over the frame interval from frame#:28 690 to frame#:30 343 is disaggregated into ten sub-video-objects. The system displays the ten sub-video-object icons, each of which denotes a resultant of *interval projection* of the original video-object onto ten overlapping intervals. (Here, the frame interval of the original video-object) is simply (mechanically) divided.) Since these sub-video-objects can be treated as ordinary video-objects, user can apply several operations to them further. When a user selects an arbitrary icon and applies the play method to it with the menu selection, the corresponding live video is replayed on the region of the icon. If the disaggregate operation is applied to the sub-video-object further, it is divided into the smaller sub-sub-objects on the lower part of the screen. Further operations can be applied to these sub-sub-video-objects in a similar manner.

These sub-video-objects are treated as ordinary video-objects. That is, these sub-video-objects have no attribute/value pairs, but they are *evaluated* based on the *interval inclusion inheritance*.

VideoChart can also select and manipulate an arbitrary portion of the video frame sequence as an video-object. In such cases, the specified portion is considered to have no attribute/value pairs. Users can manipulate it as if it were an ordinary video-object. Therefore, the play, inspect and disaggregate operations can be applied to it. If disaggregate is applied, then the disaggregation system divides it into sub-video sequence in a similar manner.

We consider that the proposed *disaggregation* facility is suitable for the following case: Suppose that we are going to define a new video-object over some portion of a large video sequence or to take out some appropriate video sequence for another purpose, for example, editing multimedia documents or desktop presentations. It may be complicated to search the

[10]A collection of inheritable attributes are declared by our object definition tool, which will be described later.

whole sequence of the source video in order to pick up a desirable scene, especially, in the case when the source is very large, such as a two-hour movie. Even if users have an opportunity to see all the contents of the source video, it is not easy to remember the exact position of the target scene by means of the frame ID or the absolute time. It is usual only to remember the approximate order of several scenes. Disaggregation enables us to browse the whole video sequence from the abstract level view, and approach to the detail scene gradually and, finally, to select the desirable sequence by this disaggregation facility.

5.3. VideoSQL

VideoSQL is a query language of OVID for retrieving video-objects. Fig. 11 shows an example query formulated by VideoSQL. Users can formulate VideoSQL queries in the fill-in-the-blank manner. The result of a VideoSQL query is a collection of video-objects that satisfy the specified condition. Before evaluating queries, the target video-objects are first *evaluated* based on the mechanism of the interval inclusion inheritance. Then, for each *evaluated* video-object, the specified condition of the query is examined. A VideoSQL query consists of the following clauses:

- **SELECT** clause: This clause is quite diffrent from ordinary SQL. It specifies only the category of the resulting video-objects, that is, `Continuous`, `Incontinuous`, or `anyObject`.
  - `Continuous` denotes that only the video-objects, consisting of a single continuous video frame sequence, are retrieved.
  - `Incontinuous` denotes that video-objects, consisting of more than one continuous video frame sequence, are retrieved.
  - `anyObject` denotes that the system retrieves all types of video objects independently from whether they are continuous or not.

  For instance, assume that a video-object o_1 consists of more than one video frame sequence, $\{i_1, i_2, i_3\}$, and o_2 consists of $\{i_4\}$. If `Continuous` is specified in the **SELECT** clause, then o_2 may be retrieved, but o_1 is not contained in the retrieval result. On the other hand, o_2 is not retrieved if `Incontinuous` is specified.
- **FROM** clause: This clause is used to specify the name of the video database.
- **WHERE** clause: This clause is used to specify the condition, consisiting of attribute/value pairs and comparison operators. The video frame# also can be used in the qualification condition. The user can select the necessary attribute names by means of a pop-up menu. Currently, the user can specify the following condition:

  1. `[attribute] is [value | video object]`
     This condition denotes to return video-objects which have the specified attribute value or video-object. For instance, Fig. 11 is an example for retrieving video-objects, each of which has a certain video-object as the value of the `Successor` attribute. Users can paste the video-object into the WHERE clause,[11] which was already copied by the VideoChart `Copy` command. The pasted object is expressed in the string expression `video(`*`object id`*`)`. The atomic values used in OVID form a generalization hierarchy as a rooted tree (see Fig. 2). A user can specify an arbitrary node in the WHERE clause of his query. By means of this facility, we can formulate a query at a more abstract level. For instance, the following is the usage example of an abstract value. The *is-a* relationships, "walk *is-a* daily life" and "relaxation *is-a* daily life," are predefined. The query,
     ```
     Action is daily life
     ```
     retrieves the video-objects such that the attribute value of `Action` in the evaluation is any of `walk`, `relaxation` or `daily life`.
  2. `[attribute] contains [value | video object]`
     This condition is concerned with set-type attributes. It denotes to return video-objects which contain the specified value or the specified video-object in the set value of the specified attribute.
  3. `definedOver [ video sequence | video frame ]`
     This condition denotes to return video-objects that are defined over the specified video frame sequence or frame. For example,
     ```
     definedOver frame(15000)
     ```
     denotes to retrieve the video-objects which includes the video frame with frame#:15 000 in their defined video sequence.

VideoSQL can formulate more than one query and save their retrieval results at the same time. The retrieved video-objects are able not only to be replayed, but also to become operands of the `Merge` operation. The display for each query is switched when a `SQL-X` button is clicked, where `X` denotes a query number. The oid's of retrieved video-objects for each query is listed under the corresponding `SQL-X` button, and users can select the desired video-object from the list (see Fig. 12) . The selected video-object for each query is entered into the `Retrieval` field. Clicking the arrow button on the lower left corner, the highlighted video-object is replayed in the window.

VideoSQL offers the facility to `Merge` the retrieved video-objects for multiple queries. After executing several queries and selecting one video-object from the answer list for each query, they can be merged into a new video-object by means of clicking the `Merge` button.

5.4. Video Object Definition Tool

The video object definition tool, shown in Fig. 13, is to define or to update video-objects. There are several functions

[11] In order to make sure of the scene, the pasted object can be replayed in the query field by clicking the "Replay" button.

to control the video disc device and the facilities to search an arbitrary video frame with the frame ID in the left part. In the right part of the definition tool, the facilities for video object definition are arranged, and the object definition is accomplished by filling in the blanks with attribute values by a pointing device. Attribute names are selected by a pull-down menu. Atomic values can be specified by selecting an appropriate value from the scroll window of the `Generalization Browser` in the lower right part. Fig. 13 denotes that "Richard Nixon *is-a* American statesman" and "James Carter *is-a* American statesman" and the current selected node in the hierarchy is "American Statesman."

To maintain the dictionary of the generalization hierarchy, a visual tool is provided. With this tool, users can browse and navigate in the generalization hierarchy and add arbitrary nodes. The maintenance of attribute names is also supported by this tool. The *inheritability* of attributes is globally effective over the whole database.

5.5. *Implementation*

Our current OVID system was implemented on Macintosh with HyperCard[3]. We choose the platform of prototyping for the OVID because of the following reason: HyperCard provide the rich facilities to construct the interactive and graphical user interface, and decrease the amount of the effort for implementation of the user interface. It also have the capability to build simple databases in its native facilities.

Each video-object is represented by a "card" of HyperCard, and a database, which is the set of video-objects, is represented by a set of cards, namely, a "stack." Currently, a single database can be specified in each session. The card id of each card is used as the object identifier because it is unique in one stack. The process procedures to derive the attribute value on the merge and overlap operations are provided in the system as the program codes of HyperTalk which is the programming language for HyperCard. The video images are stored on a video disc, which is controlled through an RS232C serial interface. A video image are input into the video overlay board, and displayed on the monitor screen. Therefore, the video images are physically stored in a separate place from the place storing video-objects.

VI. Concluding Remarks

In this paper, we introduced a video object data model for video data, and define several operations for video-objects. The major characteristics of our video object data model are the following:

1) The notion of **video-object** is introduced to provide a descriptional data for an arbitrary portion of time-sequential data, especially of video data.
2) **Interval inclusion inheritance**, which enables us to share information among video-objects in the database in order to decrease the effort of providing descriptional data for video information.
3) A collection of operations *interval projection*, *merge*, *overlap* are introduced in order to compose video-objects from predefined video-objects, which are, we believe, important to edit video data. It should be noted that these composition operations contain the abstraction operation based on generalization hierarchy or the above interval inclusion inheritance operation.

Also, we described our prototype video database system, named **OVID**, based on our video object model. It was developed on Macintosh with HyperCard. Major features of our prototype system are the following.

- **VideoChart**: a visual interface to browse a video database and to inspect/manipulate video-objects. Each video-object is represented as (a sequence of) a bar, and the following operations are prepared to manipulate video-objects: play, inspect, disaggregate, merge, and overlap video-objects.
- **VideoSQL**: a query language to retrieve video-objects, which facilitates a video-object to be specified in the qualification clause.

The following problems need further research.

- **Formalization and Generalization of Model**: Expressive power and several mathematical properties of our model should be further inverstigated.
  Furthermore, our video object data model can be generalized into the one which treats not only video, but also general spatial or temporal objects. Indeed, the notion of *inheritance based on inclusion relationships* can be applied not only to one dimensional data such as video frame sequences, but also to two dimensional data such as maps or three dimensional data such as solid modeller objects. That is, in general, the descriptional information for some "region" object may be inherited by "sub-region" objects. Suppose that there is an area object in a geographical database, it has an attribute and its value, `zip_code:65 121`, and there exists a town object inside of the area. Then, generally, the town object should have the same zip code as the area's. We consider that the inheritance based on inclusion relationships is useful to share the `zip_code` attribute and its value between the area and the town object.
- **Efficient implementation of inclusion-based inheritance**: Our current OVID system was implemented throughly by using HyperCard, which is not suitable for managing a lot of objects efficiently. Also, our inclusion-based inheritance needs a faster implementation. For example, some physical storage structure to support the inclusion-based inheritance needs to be developed.
- **Automatic Support Facility for Object Description**: In current OVID, all attributes and attribute values are manually given to each video-object. Several researches[13] have challenged for characterizing and indexing video data automatically. Since the methods developed by such researches are not incompatible with our OVID system, the descriptive power for video-objects will be enriched by introducing those contributions.
- **Extension to a Hypermedia Database System**: Our video object model should be extended to more general data modeling framework for sound, still images, text, etc. The inheritance based on inclusion relationship will be

useful in such applications. As the future step, we intend to extend the OVID to a hypermedia system including a hypertext system and a geographical database system. We are also planning to apply our system to more practical applications.

ACKNOWLEDGMENT

We would like to thank anonymous referees for many valuable comments and suggestions.

REFERENCES

[1] M. Adiba and N. B. Quang, "Historical multi-media databases," in*Proc. 12th Int. Conf. on Very Large Data Bases* Aug. pp. 63–70, 1986.

[2] J. F. Allen, "Maintaining knowledge about temporal intervals," *Commun. ACM,* vol. 26, pp. 832–843, Nov. 1983.

[3] Apple Computer Inc., "HyperCard Script Language Guide: The Hyper-Talk Language," Addison-Wesley Publishing Company, Inc., Cupertino, 1988.

[4] M. Atkinson, F. Bancilhon, D. DeWitt, K. Dittrich, D. Maier, and S. Zdonic, "The object-oriented database system manifesto," in *Proc. First Int. Conf. Deductive and Object-Oriented Databases*, pp. 40–57, Dec. 1989.

[5] F. Bancilohon and S. Khoshafian, "A calculus for complex objects," in *Proc. ACM PODS,* pp. 53–59, Mar. 1986.

[6] S. Christodoulakis, F. Ho, and M. Theodoridou, "The multimedia object presentation manager of MINOS: A systematic approach," in *Proc. ACM SIGMOD Int. Conf.*, pp. 295–310, May 1986 .

[7] M. E. Hodges, R. M. Sasnettand, and M. S. Ackerman, "A construction set for multimedia applications," *IEEE Software,* vol. 6, pp. 37–43, Jan. 1989.

[8] M. E. Hodges and R. M. Sasnett, "Plastic editors for multimedia documents," in *Proc. USENIX Summer '91* pp. 463–471, June 1991.

[9] T. D. C. Little and A. Ghafoor, "Multimedia object models for synchronization and databases," in *Proc. Sixth Int. Conf. Data Engineering,* Feb. 1990, pp. 20–27.

[10] W. E. Mackay, "EVA: An experimental video annotator for symbolic analysis of video data," *SIGCHI Bulletin,* vol. 21, pp. 68–71, Oct. 1989.

[11] W. E. Mackay and G. Davenport "Virtual video editing in interactive multimedia applications," *Commun. ACM,* vol. 32, pp. 802–810, July 1989.

[12] D. R. Morrison, "PATRICIA—Practical algorithm to retrieve information coded in alphanumeric," *J. ACM,* vol. 15, pp. 514–534, Oct. 1968.

[13] A. Nagasaka and Y. Tanaka, "Automatic video indexing and full-video search for object appearances," *Visual Database Systems, II,* E. Knuth and L.M. Wegner, Eds. Amsterdam: North-Holland, 1992, pp. 113–127.

[14] A. P. Parkes, "CLORIS: A prototype video-based intelligent computer-assisted instruction system," in *Proc. R.I.A.O. '88 Conf.*, pp. 24–50, Oct. 1987.

[15] J. Smith and D. C. P Smith, "Database abstractions: Aggregation and generalization," *ACM Trans.Database Syst.*, vol. 2, pp. 105–133, 1977.

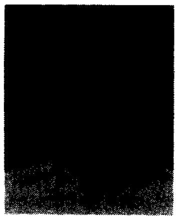
[16] K. E. Smith and S. B. Zdonic, "Intermedia: A case study of difference between relational and object-oriented database systems," in *Proc. OOPSLA '87,* pp. 452–465, Oct. 1987.

[17] NHK Service Center, *Showashi 9 Saisho Retsuden*, PIONEER LDC Corp., Japan, 1987.

[18] K. Tanaka and M. Yoshikawa, "Toward abstracting complex database objects: Generalization, reduction and unification of set-type objects (Extended Abstract)," in *Proc. 2nd Int. Conf. Database Theory*, Aug. 1988, Lecture Notes in Computer Science 326, Springer-Verlag, pp. 252–266.

[19] D. Woelk, W. Kim, and W. Luther, "An object-oriented approach to multimedia databases," in *Proc. ACM SIGMOD '86*, pp. 311–325, May 1986.

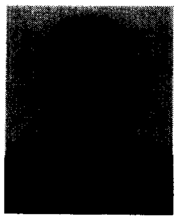
[20] D. Woelk and W. Kim, "Multimedia information management in an object-oriented database system," in *Proc. 13th Int. Conf. Very Large Data Bases*, Sept. 1987, pp. 319–329.



Eitetsu Oomoto received the B.E. degree in 1988 and the M.E. degree in 1990 in instrumentation engineering from Kobe University, Japan.

In 1993, he joined the Department of Information and Communication Sciences, Faculty of Engineering, Kyoto Sangyo University, where he is now a lecturer. His interests include database systems, multimedia systems, user interfaces, and artificial intelligence.

Mr. Oomoto is a member of the Information Processing Society of Japan.



Katsumi Tanaka received the B.S., M.S., and Ph.D degrees in information science from Kyoto University, in 1974, 1976, and 1981 respectively.

In 1986, he joined the Department of Instrumentation Engineering at Kobe University, where he is now an Associate Professor. His interests include object-oriented databases, historical database models, and hypermedia systems.

Dr. Tanaka is a member of the ACMand the Information Processing Society of Japan.