9 A SEMANTIC DATA MODELING MECHANISM FOR MULTIMEDIA DATABASE

Qing Li and Jun Yang

Department of Computer Engineering and Information Technology
City University of Hong Kong

Yueting Zhuang

Department of Computer Science
Zhejiang University

Semantic data modeling of multimedia data is a fundamental problem in multimedia databases. The variety, context-dependency, and ambiguity of the semantics of multimedia data create the difficulty of applying conventional data modeling techniques, which usually deal with explicit, self-contained alphanumeric data. Many semantic data modeling techniques are either explicitly proposed for multimedia or can be used for modeling multimedia data. Specifically, we describe *MediaView* as a novel object-oriented view mechanism that can successfully capture the elusive nature of multimedia data.

In this chapter, we firstly describe the problem of semantic multimedia modeling and the dilemma of applying traditional data modeling techniques on multimedia data. We then present a brief review of the related techniques on semantic and multimedia data modeling. The *MediaView* mechanism is described in details and some of its applications are discussed.

9.1. Semantic Modeling of Multimedia

The explosive growth of multimedia data creates the need of manipulating them in an organized, efficient, and scalable manner, preferably, using multimedia databases. However, the problem of semantic data modeling, which is essential to a multimedia database, has not been fully understood and successfully solved. The difficulty comes from the following properties of multimedia objects (e.g., images, video clips, and audio segments) in regard of their semantics:

- **Variety**. Many multimedia objects can be semantically interpreted in a variety of different ways. That means an object[1] may have multiple latent meanings.

[1] If not indicated explicitly, "object" refers to "multimedia object" in this chapter.

- **Context-dependency**. The "appropriate" meaning of a multimedia object depends on the particular application or user who manipulates the object, the perspective from which it is interpreted, and the other objects that interact with it in the same environment. All these factors comprise the context indispensable for the interpretation of an object's semantics.

- **Ambiguity**. When considered in isolation rather than in a specific context, a multimedia object does not have "default" meaning — an explicit and dominant one among all of its latent meanings.

About the semantics of images, Santini et al. have made the following observation in [23]: "meaning is not an *intrinsic* property of the images that the database filters during the query, but an *emergent* property of the interaction between the user and the database". As a proof of this argument, consider the interpretation of van Gogh's famous painting "Sunflower", the leftmost image in both Figure 9-1 and Figure 9-2. When it is placed with the images in Figure 9-1, which are another four paintings of van Gogh, the meaning of "van Gogh's paintings" is implied. When the same image is interpreted in the context of Figure 9-2, however, the meaning of "flower" is more evident. In fact, this argument on the image semantics can be generalized to other types of media, i.e., a multimedia object must be interpreted in a *context* specific to the application and the database.

Figure 9-1. "Sunflower" in the context of "van Gogh's paintings"

Figure 9-2. "Sunflower" in the context of "flower"

The nature of multimedia data is fundamentally different from that of traditional data (e.g., numbers and text), whose semantics are explicit, objective, and self-contained. This large distinction accounts for the failure of applying conventional data modeling techniques on multimedia. For example, consider an object-oriented data model,

which is more powerful than other traditional models in term of semantic modeling capability. Conventionally, database designers rely on the conceptual schema—the middle level of the traditional ANSI/SPARC three-schema database architecture—to capture the semantics of multimedia data. A common way to do this is to define many classes, each of which represents an explicit semantic concept, and classify multimedia objects into appropriate classes. However, this approach conflicts with the nature of multimedia in several aspects: (1) No matter how many classes are defined, they are unlikely to cover all the eligible semantic concepts, which are simply unlimited. (2) It rules out the possibility that an object has various semantic interpretations (which imply that this object is eligible for more than one classes), given that multiple class membership is not permitted as is the case of a strongly typed object model. (3) As a consequence of (2), the data cannot be shared among applications that prefer different interpretations of the same data.

Alternatively, some data models choose to model only the primitive features of a multimedia object, such as the data size, format, and source, rather than its semantics. These models overcome the limitations mentioned in the last paragraph. Nevertheless, this approach is a sacrifice of modeling capability for generality and flexibility: Although in theory such data models are compatible with a broad range of applications, in practice no application is adequately supported due to the absence of data semantics. A typical multimedia application, say, authoring of an e-magazine, is unlikely to conduct queries like "find all JPEG images with size less than 200 KB", but is more reasonable to "find all the paintings drawn by van Gogh in his last year", which obviously goes beyond the capability of such data models.

The failure of using conceptual schema to model multimedia data can be ascribed to the absence of contexts in which their semantics can be properly interpreted. Therefore, it is natural and desirable to shift the task of semantic modeling from the conceptual level to the external (view) level, which is much closer to the applications. In this chapter, we propose *MediaView* as an object-oriented view mechanism to achieve this functional shift in the ANSI/SPARC three-level schema architecture. In this mechanism, the conceptual schema no longer characterizes the semantics of the multimedia data; instead, their semantics are described at the external level by a set of views, called *media views*, which provide the application-specific context for the interpretation of multimedia data. Specialized view derivation mechanisms, such as heuristic enumeration and object algebra, are proposed for the definition of media views. Media views can facilitate powerful multimedia-flavored queries and semantics-oriented navigation functionality, and suggest the semantic correlations between objects according to which the physical storage and indexing (of data) can be optimized.

9.2. Semantic and Multimedia Data Modeling Techniques

This section presents an brief review on the data modeling techniques related to semantic modeling of multimedia, which are roughly classified into three categories, as (1) multimedia database techniques, (2) previous object-oriented view mechanism, and (3) object-oriented data models with dynamic functions.

9.2.1. Multimedia Database Techniques

The proliferation of multimedia data imposes a great challenge on conventional database technology, which is inadequate to model their characteristics, such as the huge data size, spatial and temporal nature, etc. To address these limitations, many works have been proposed towards different aspects regarding the management of multimedia, including data models [2, 12, 17], presentation [13], indexing [9], and query processing [4].

Among all the data models, the object-oriented approach is generally regarded as the most suitable choice for modeling multimedia data, mainly because of its great modeling capacity and its extensibility (for new types of data) by means of inheritance. For example, OVID (Object Video Information Database) system [17] was proposed based on an object data model for video management. Similarly, the STORM [2] object-oriented database management system integrates structural and temporal aspects for managing different presentation of multimedia objects. The ORION project [12] proposed by Kim et al. also adopts object-oriented methodology for multimedia management.

Besides the data modeling, there are also some works addressing the retrieval and presentation issues in multimedia databases. For example, in [9] Faloutsos surveyed the works on multi-dimensional indexing and suggested a fast searching method for multimedia objects. The work of Klas et al. [13] focused on the presentation issue by proposing an efficient and generic playout service for multimedia application. Bertino et al. [4] presented an overview on issues concerning query processing of multimedia and described MULTOS as a multimedia server with advanced retrieval capability.

Although the aforementioned works are successful in their respective domains, they fall short of modeling the semantic aspect of multimedia data, which is of vital importance to MMIS applications. Motivated by this problem, *MediaView* provides an effective mechanism for semantic modeling of multimedia, which is orthogonal and complementary to all the previous works. In fact, it can be easily integrated with the existing techniques, provided that an object model is used.

9.2.2. Object-Oriented View Mechanism

There have been many successful previous works on view mechanism in object-oriented data models [1, 11, 12, 22, 24, 27]. Most of them utilize the query language defined for their respective object model to derive a view (or virtual class), e.g., Abiteboul et al. [1] proposes a general framework for view definition based on a clear semantics. These approaches differ from each other in the way that they treat the derived view in the global schema. For example, Heiler's approach [11] treats each view as a standalone object, rather than integrating into the schema. In Kim's work [12], the derived views are attached to the schema root as its direct subclasses. The approaches of Scholl et al. [24] and Tanaka et al. [27] address the issue of incorporating the derived views into the global schema. However, the consistency of

the schema is not guaranteed in their approaches. Rundensteiner's proposal of MutliView [22] is a more systematic solution on object view mechanism, which not only provides an algorithm for integrating the derived virtual classes into the global schema, but also allows the generation of multiple view schemata, with the consistency and closure property enforced by automatic tools. The issue of view materialization is also addressed in MultiView [15].

The fundamental distinction between *MediaView* and the past works is that all the existing view solutions are "information customization" mechanisms, i.e., hiding properties and instances of the base classes from a customized view, while *MediaView* is an "information augmentation" mechanism, in which objects are empowered with new properties when they are included into views.

9.2.3. "Dynamic" Object Models

There has also been a significant research direction on developing OODBMSs that can support advanced "dynamic functions", including *object evolution and migration*, *dynamic conceptual clustering*, and *multiple perspectives and representations*. Two radically different approaches towards supporting such dynamic functions in OODBMSs can be distinguished: one is based on language-independent or weakly-typed object models, and the other is based on strongly-typed object models extended with *role* facilities. These dynamic functions enhance the semantic modeling capability of an object model and make it more suitable for multimedia data.

- **Language-Independent and Weakly-Typed Models**

This approach supports the various dynamic functions for OODBMSs by employing a weakly-typed programming language (such as Smalltalk), where an object can have multiple classes, or by adopting a language-independent object model where the OODBMS defines its own object model and appropriate mappings are provided from languages to this object model. Examples of the former include GemStone [6], whereas OODAPLEX [7], O2 [8] and TIGUKAT [19] follow the latter. Consequently, these systems can support object migration as well as allow an object to be viewed from different perspectives. A main disadvantage of this approach is the inherent performance penalty due to the inefficiency of weakly typed languages.

- **Strongly Typed Models with Role Extensions**

Examples of work belonging to this group include conventional OODBMSs developed from strongly-typed programming languages such as C++. In particular, these systems use a static, classification-based approach in defining the structural and behavioral properties of data objects, where each object in the database is assumed to have exactly one class, namely, the one in which the object was created. Such an assumption, however, imposes some serious restrictions on modeling dynamic and/or multi-faceted real world objects, which is especially true for multimedia applications. To relax this modeling restriction, *role* facilities have been

proposed and/or incorporated recently into the "statically-typed" OODBMSs [16,20,21,25,26], resulting in improved modeling power and flexibility for conventional object-oriented models.

In the context of object-oriented databases, roles have been demonstrated to be effective in supporting *object evolution and migration* [20,26], as well as *conceptual clustering* modeling [16]. In particular, the work described in [20] devised a role model that facilitates describing the different behaviors throughout an object's evolution (and state-transitions). The work on *object migration patterns* [26] considered allowable patterns for object migration (i.e., change of class) represented as dynamic constraints, and used the valid "role set" histories of objects for this purpose. For the work on *conceptual clustering* models [16], roles have been used as "threads" through which objects are grouped together in forming ad hoc, dynamic collections (called *clusters*) as derived "meta-data". Roles have also been found useful in supporting multiple perspectives/representations of objects. Work in this direction includes *object specialization* [25], *aspects* [21] and *Fibonacci* [3], which all use somewhat different approaches. The work described in [25] took the combined approach of a prototype system and an object-oriented system, where classes are viewed (and realized) as individual objects (prototypes) auxiliary roles (perspectives), and objects define their own inheritance paths. By contrast, in [21], roles are attached to object classes directly as *extensions*, with each extension providing a specialized perspective (called an *aspect*) for the objects in the class. Independently, the Fibonacci system by Albano et al. [3] allows a role hierarchy to be associated to a base class so that its objects can play any role from that hierarchy.

9.3. *MediaView*: A Semantic Modeling Mechanism

MediaView is not the first object-oriented view mechanisms, which have been proposed in many previous works [1,11,12,22,24,27]. Nevertheless, *MediaView* is the first view mechanism (to the best of our knowledge) that is designed especially for multimedia databases. Consequently, the definition, derivation, and usage of media views are characteristic of a multimedia flavor and are fundamentally different from all the previous works.

9.3.1. An Overall Picture

As shown in Figure 9-3, the MediaView mechanism suggests a novel definition of the ANSI/SPARC three-level schema architecture for multimedia databases. This redefinition is based on the observation that a multimedia object can be characterized by features at two independent levels—primitive features at low level, as well as semantic features at high level. In MediaView, these two sets of features are respectively modeled at the conceptual level and at the external level. Note that this is a departure from the convention that the conceptual schema takes charge of all the semantic modeling tasks, while views are only used to define customized interface of the data.

The primitive features of multimedia are basic and context-independent features that are unlikely to have diverse interpretations in different contexts. Examples of primitive features include generic features such as data size and format, as well as media-specific features such as color histogram for images, motion vector for videos, and pitch contour for audios. The nature of primitive features determines that they can be suitably modeled by the conceptual schema, which provides an objective foundation for a broad range of applications. Specifically, these features are characterized through a collection of *base classes* defined in the conceptual schema during the initial schema definition process. For example, a typical base class defined is "*Bitmap Image*", with attributes "*color histogram*", "*texture*", etc.

In addition to primitive features, each multimedia object possesses semantic features that are usually subjective and context-dependent. Such features vary with the specific contexts in which an object is interpreted, e.g., the van Gogh's "Sunflower" can be described either as "flower" or "van Gogh's painting" depending on the specific situations. Therefore, we deem that the external (view) schema is in a better position than the conceptual schema to model such semantic features. By defining its own views, each application (user) can specify a customized context in which the multimedia data are interpreted for its specific purpose. Moreover, as stated by Rundensteiner [22], views in an object-oriented data model have additional advantages for semantic data modeling: (1) an object can maintain its unique identity even if it is included in more than one view, such that no new objects are created as the result of applying views. (2) type-specific attributes and operations can be assigned to objects when they are modeled by views, which are defined as virtual classes.

At the external level, the specific tool for modeling the semantic features of multimedia is a set of media views. On the surface, like many existing object-oriented view mechanisms, a media view is defined as a virtual class with specific attributes and operations. In essence, each media view corresponds to a semantic concept and is a cluster of the multimedia objects that can be interpreted by that semantic concept. Therefore, a media view can consist of heterogeneous but semantically related objects selected from different base classes, instead of homogeneous objects (from the same class) as in the traditional view mechanisms. On the other hand, while a multimedia object belongs to only one base class, it can be included into more than one media views simultaneously. This corresponds to the fact that a multimedia object may have multiple semantic interpretations, but its primitive features are uniform.

9.3.2. Fundamentals of *MediaView*

In the following, we firstly present a formal definition of media view and some related concepts. After that, we introduce a semantic graph model that describes the data reorganization as the outcome of applying media views.

Basic Concepts

Definition 1: *Set C as the set of base classes. A* ***base class*** $C_i \in C$ *has a unique class name, a type description, and a set of objects associated with it. The type of* C_i *is referred to as* ***type***(C_i)*, which defines a set of properties as the common interface of all the instances of* C_i*. The set of properties are referred to as* ***properties***(C_i)*, and each property p in it can be a value of a simple type, an instance of a certain class, or a method. The set of objects associated with* C_i *is defined as* ***extent***$(C_i)= \{o \mid o \in C_i\}$.

Definition 2: *A* ***media view*** MV_i *is a virtual class that has a unique view name, a type description, and a set of objects associated with it. The type of* MV_i *is referred to as* ***type***(MV_i)*, which defines a set of properties* ***properties*** (MV_i) *as the common interface of all its instances. Similarly, a property can be a value of a simple type, an instance of a media view, or a method. The set of objects associated with* MV_i *is defined as* ***extent***$(MV_i)= \{o \mid o \in MV_i\}$.

Definition 3: *A base class* C_i *is defined as a* ***subclass*** *of another base class* C_j *if and only if the following two conditions hold: (1)* ***properties***$(C_j) \subseteq$ ***properties***(C_i)*, and (2)* ***extent***$(C_i) \subseteq$ ***extent***(C_j)*. If* C_i *is the subclass of* C_j*, we also say that there is an* ***is-a*** *relationship from* C_i *to* C_j*. A* ***base schema*** *(BS) is a directed acylic graph G=(V, E), where V is a finite set of vertices and E is a finite set of edges as a binary relation defined on* $V \times V$*. Each element in V corresponds to a base class* C_i*. Each edge in the form of* $e=<C_i, C_j> \in E$ *represents an is-a relationship from* C_i *to* C_j *(or* C_i *is a subclass of* C_j*).*

Definition 4: *A media view MV_i is a* ***subview*** *of another media view MV_j (or there is an* ***is-a*** *relationship from MV_i to MV_j) if and only if* ***properties****(MV_j)* $\subseteq$ ***properties****(MV_i) and* ***extent****(MV_i)* $\subseteq$ ***extent****(MV_j). A* ***view schema*** *(VS) is a directed acylic graph G={V, E}, where a vertex in V corresponds to a media view MV_i, and an edge e=<MV_i,MV_j>* $\in$ *E represents an is-a relationship from MV_i* to *MV_j (or MV_i is a subview of MV_j).*

One can see a parallel between the definition of media view and base class, as well as between base schema and view schema. Actually, a media view behaves like a base class except the following difference: (1) the instances of a media view are selected from the base classes; we cannot create a view instance from scratch. (2) A media view may consist of heterogeneous objects selected from different base classes. (3) An object must belong to a unique base class, while it can be included into zero, one, or multiple media views.

In addition to the **is-a** relationship, conventional object-oriented data models also support the **composition** relationship between two base classes, which defines an instance of a class as a property of the instances of another class. Similarly, there is also **composition** relationship between two media views, through which we can create a complex view instance by referring to simple view instances as its properties.

Figure 9-4 exemplifies the basic concepts defined above, including (a) a base class "Image", (b) the media view "Impressionistic Artworks", (c) a base schema, and (d) a view schema, from which one can appreciate the main distinction between media views and base classes in regard of modeling functionality. Suppose both the class "Image" and the media view "Impressionistic Artworks" are used to model van Gogh's painting "Sunflower". The base class describes the painting by its low-level features, such as color histogram and dominant shape, while the media view interprets it from a higher level semantic perspective, say, its style. Accordingly, the

base schema is mainly constructed on the type or structural relationship, e.g., "JPEG Image" is a kind of "Image", and "Video Clip" refers to a set of "Image" and "Audio Clip" as its properties. In contrast, the view schema is based on semantic relationship, e.g., "Realistic Artworks" is one category of "Artworks".

The definition of media view differs from that proposed in conventional object-oriented view mechanisms [22] in several fundamental aspects: First, a media view can accommodate objects from different base classes (heterogeneous objects), while conventionally a view can only contain objects from a single class (homogeneous objects). This distinction is desirable in multimedia databases, since multimedia objects of different types may suggest the same semantic concept. Second, new properties can be defined for a media view, such that an object is empowered with these new properties (in addition to the properties defined in its base class) when it is identified into that media view. This offers a significant improvement in modeling capacity over the traditional view mechanisms, which only allows view to inherit or derive properties from base classes. Last but not the least, the view schema and the base schema, which model different aspects of multimedia data, are completely independent in the *MediaView* mechanism. In comparison, in previous works such as [22], views and base classes are usually integrated into a global schema.

Semantic Graph Model

Applying media views on the database corresponds to a semantics-based reorganization of multimedia objects, which are initially organized according to their types by base classes. As illustrated in Figure 9-5, a media view clusters multimedia objects of various types (typically, text, image, video, and audio) that have the same or closely related semantics as its instances. In this way, it defines an application-specific context in which the semantics of these objects can be interpreted. Since an object may have various semantic interpretations and thus belong to multiple media views, a media view may overlap or even subsume other views in terms of their instances.

As the objects of a media view are semantically correlated, we create a link between any pair of them to represent their semantic correlation. For each media view, such links constitute a complete graph among all its instances. Therefore, all the objects of the database can be interconnected into a huge graph (not necessarily a connected

graph) by a set of such complete sub-graphs corresponding to media views. This leads us to the following definition:

> ***Definition 5***: *The **semantic graph** (SG) is an undirected graph G={V, E}, where V is a finite set of vertices and E is a finite set of edges. Each element $V_i \in V$ corresponds to a multimedia object O_i in the database. E is a ternary relation defined on $V \times V \times N$. Each $e=<V_i, V_j, n> \in E$ represents a **semantic link** of degree n between object O_i and O_j, where n is the number of media views to which both objects belong. We define n as the **correlation factor** between O_i and O_j.*

> ***Definition 6***: *The **correlation matrix** $M=[m_{ij}]$ is the adjacency matrix of the semantic graph. Consequently, each element m_{ij} contains the correlation factor between O_i and O_j, with all the diagonal elements set to zero.*

The semantic graph model is illustrated by the example in Figure 9-6. Suppose the scenario of an e-publisher who plans to compose two special issues of an e-magazine on the topic of van Gogh and impressionistic artworks respectively. To collect related materials, the editor creates two media views, "van Gogh's Gallery" and "Impressionistic Artworks" (see Figure 9-6(a)), and inserts multimedia objects into them. For "van Gogh's Gallery", the materials include a biography of van Gogh, an audio guide, and two of his paintings "Potato Eaters" and "Sunflower". Both paintings are also eligible for the other media view, "Impressionistic Artworks", which also contains the works of other impressionistic artists. The fraction of the semantic graph corresponding to these two views is shown in Figure 9-6(a), along with the corresponding correlation matrix in Figure 9-6(b). The merit of the semantic graph and the correlation matrix lies in that they reveal the semantic correlation

Table 9-1. Media view operators

type-level	**v-overlap**	syntax	<*boolean*>:= **v-overlap** (<*media view1*, *media view2* >)
		semantics	**true**, if and only if ($\exists$ *o* ∈ *O*)(*o* ∈ **extent**(<*media view1*>) **and** *o* ∈ **extent**(<*media view2*>))
	cross	syntax	{<*object*>}:= **cross** (<*media view1*, *media view2* >)
		semantics	{<*object*>} := {*o* ∈ *O* | *o* ∈ **extent**(<*media view1*>) **and** *o* ∈ **extent**(<*media view2*>)}
	sum	syntax	{<*object*>}:= **sum** (<*media view1*, *meida-view2* >)
		semantics	{<*object*>} := {*o* ∈ *O* | *o* ∈ **extent**(<*media view1*>) **or** *o* ∈ **extent**(<*media view2*>)}
	subtract	syntax	{<*object*>}:= **subtract** (<*media view1*, *media view2*>)
		semantics	{<*object*>}:= {*o* ∈ *O* | *o* ∈ **extent**(<*media view1*>) **and** *o* ∉ **extent**(<*media view2*>)}
instance-level	**class**	syntax	<*base class*> := **class**(<*view instance*>)
		semantics	<*view instance*> is a instance of <*base class*>
	components	syntax	{<*object*>} := **components** (<*view instance*>)
		semantics	{<*object*>} := { *o* ∈ *O* | *o* is a component (direct or indirect) of <view instance>}
	i-overlap	syntax	<*boolean*> := **i-overlap** (<*view instnace1*>, <*view instance2*>)
		semantics	**true**, if and only if $\exists$ o ∈ O) (o ∈ components (<view instance1>) **and** o ∈ components(<view instance2>))

between objects with regard to a certain application. Later, we will demonstrate the use of them for view derivation and application in the same scenario of e-publishing.

9.3.3. View Operators

We define a set of operators that take media views and view instances as operands. Our intension is not to come up with a complete set of operators, but to focus on those that are indispensable in supporting query and navigation over multimedia objects. Table 9-1 summarizes these operators with a description of their syntax and semantics.

The view operators are classified into two categories according to the type of operands: *type-level* operators that manipulate media views (types) as operands, as well as *instance-level* operators with view instances (object) as operands. The type-level operators include **v-overlap**, which examines whether two media views has common instances, and three set operators as **cross**, **sum**, and **subtract** which find out objects that are instances of both of the two media views, instances of either of

the two views, or instances of the first view but not the second view. The instance-level operator **class** is used to retrieve the name of the base class of the specified view instance. As introduced in section 9.3.2, complex view instances can be composed from simple view instances by composition relationship as its components. The **components** operator is designed to find all the components (either direct or indirect) of a complex view instance, i.e., the objects that links to the specified view instance through one or more composition relationships. The **i-overlap** operator is a Boolean operator that returns true when two complex view instances have common components.

As can been seen, these view operators mainly manipulate the instances of media views and investigate the relationship between media views based on shared instances. The common objects shared by two media views suggest an ad hoc, implicit relationship between these two views, which cannot be explicitly defined or even expressed. As an example, consider the relationship between media view "van Gogh's Gallery" and "Impressionistic Artworks" (see Figure 9-6) through sharing of van Gogh's paintings. Such implicit relationships offer an important supplement to the explicitly defined is-a and composition relationship. The set of view operators proposed here provides a tool for making use of these relationships.

9.3.4. View Derivation Mechanism

We propose two approaches for the derivation of media view: heuristic enumeration and object algebra. In the former approach, the user identifies the instances of a media view by enumeration with the recommendation based on the content (low-level features) or semantic links (embodied in the semantic graph) among the candidate objects. In the latter approach, new media views are derived from existing media views by applying object algebra operators on them.

Heuristic Enumeration

To define a media view using this approach, a user, usually the application designer, needs to firstly specify the properties of the media view, and then enumerates all its instances. The enumeration process can be conducted manually, or with the commendation of promising candidates objects based on the content or semantic similarity between objects, or using a combination of the above strategies.

- **Blind enumeration**. Blind enumeration is a brute-force approach, which requires the user to exhaustively enumerate all the instances of a media view without any hints from the database. To create a media view, the user has to browse the database in a very haphazard manner, picks up the relevant objects, and labels them as the instances of the media view. Not surprisingly, this method is not scalable to large databases due to the great manual effort and processing time required.

- **Content-based enumeration**. This is a heuristic enumeration approach with the recommendation of candidate objects from the system based on the content

(expressed as primitive features) of the objects. In this approach, the user chooses a representative object (from the objects already identified as view instances or any other objects at hand) as a sample and searches for more objects that are similar to it. In this case, the similarity of the sample object with every object in the database is calculated based on primitive features. The primitive features and similarity functions can be whatever is defined in the base classes and are dependent on the type of the sample object. For example, for images we choose color histogram, dominant shape as the primitive features, and Euclidean distance as the similarity metric. After the similarity calculation is finished, a list of candidate objects ranked by their similarity score is returned to the user as the recommendation, from which he select relevant ones as view instances. The whole process is essentially the query-by-example (QBE) paradigm in content-based retrieval.

- **Semantics-based enumeration**. This approach is the counterpart of the content-based enumeration at the semantic level. Specifically, it takes the advantage of the semantic links among multimedia objects in the semantic graph (*SG* for short). A simple version of this approach works as follows: When the user identifies an object, say, O_i, as the instance of the media view being defined, we recommend the objects that are directly linked with O_i in *SG* (i.e., objects that appear together with O_i in some media views) as the candidate objects for further enumeration. The candidates are ranked by their correlation factor (viz., degree of the link) with the "seed" object O_i. To do this, we look into *i*th row of the correlation matrix *M* and sort the objects according to the value of each m_{ij}.

  The justification of the above approach is that two objects in the same media view are semantically correlated and therefore stand a greater chance (than a random pair of objects) to be selected together into another media view. Again, consider the scenario of e-publishing illustrated in **Figure 9-6**: Suppose that the editor want to create the third media view "van Gogh's paintings" based on the two defined views, "van Gogh's Gallery" and "Impressionistic Artworks". As the first step, he selects O_4, van Gogh's famous painting "Sunflower", into this new media view. Then, by examining the semantic graph, the system suggest candidate objects O_3, O_1, O_2, and O_5 to him, with O_3 appearing in the top since its correlation factor is the largest among the four. In this case, the editor can quickly confirm O_3, which is another masterpiece of van Gogh, as the member of the new media view.

The above approach is straightforward and intuitive. However, it suffers from an obvious setback: the objects that link to the seed object indirectly via intermediate objects are not considered for recommendation, which are likely to be promising candidates of the media view as well. To overcome this limitation, we introduce the following concepts:

> ***Definition 7.*** *The* ***n-level correlation matrix*** *M(n) is derived from* ***correlation matrix*** *M by the following formula:*

object algebra through which a media view can be derived by reusing the definition of existing views. Object algebra is a mechanism commonly used to derive views in many previously suggested object-oriented view mechanisms. However, while in traditional approaches a virtual class can be derived from either base classes or existing virtual classes, in our mechanism a media view can only be derived from existing views, due to the gap in semantics between media views and base classes. Table 9-2 displays a set of object algebra operators, which are adapted from those defined in [22].

As can be seen from Table 9-2, each operator specifies both the properties and the instances of a new media view on the top of existing ones. The select operator derives a new media view by filtering the properties and instances of the source view. The properties of the new view is specified by <property list>, and its instances are those in the source view that satisfy <predicate>. The intersection operator defines a media view with properties combined from both of the source views, and instances as common members of both views. On the contrary, the properties of the media view defined by union are the intersection of the properties of two source views, and the instances of the new view are the objects that belong to either of the source views. The media view defined by difference has the same properties with the first source

view, and its instances are those belong to the first source view but not in the second view.

Despite of the similarity in syntax, object algebra operators are fundamentally different from the view operators defined in section 9.3.3, mainly in the following two aspects: (1) Object algebra operators create new media views while view operators only returns simple values (e.g., Boolean) or set of objects. (2) Object algebra operators manipulate not only the instances but also the properties (type) of a media view, while view operators only deal with view instances. For example, the object algebra **intersection** is syntactically similar to the view operator **cross**, both of which identify the common member of two media views. However, **interaction** also derive a set of properties from two source views, based on which it finally define a new media view, while **cross** ends up with returning the common instances.

Object algebra operators provide useful tools for deriving media views on the top of existing views. For example, we can create a media view called "Impressionistic Paintings" through the query "select (name, artist, style) from 'Impressionistic Artworks' where type = 'painting'". As another example, a media view called "van Gogh's Artworks" can be derived by "intersection ('Impressionistic Artworks', 'van Gogh's Gallery')". This approach not only reduces manual efforts required for view definition, but also saves the storage cost for the definitions of media views.

Admittedly, both of the approaches described above demand human effort for view derivation. For the heuristic enumeration, even if highly relevant objects are recommended, the user needs to inspect the recommended objects and confirm them into the media view. Moreover, the semantics-based enumeration is based on the previously defined media views, whose definition in turn requires manual effort. For object algebra, since they are also applied on existing media views, manual effort are involved anyway. In fact, it is extremely difficult (if not impossible) to derive media views automatically, which are defined from application-dependent semantic perspectives. Fortunately, both of the proposed approaches are able to reuse the previously defined media views when deriving new media views, such that the view definition process becomes much easier and more efficient.

9.4. Applications of *MediaView*

In this section, we demonstrate how the *MediaView* mechanism facilitates flexible retrieval and browsing of multimedia objects in the database. Further, we discuss its potentials for optimizing the physical storage and indexing based on the semantic correlation between objects indicated in the graph model.

9.4.1. Multimedia Queries

In a multimedia database, a key factor to success lies on its data model's capacity in supporting multimedia-flavored queries, which are not well supported by traditional data models mainly due to their failure to capture the semantics of multimedia. In contrast, we demonstrate that view operators (see Table 9-1), coupled with the object

algebra operators (see Table 9-2), provide a strong basis for formulating rather powerful and sophisticated multimedia queries. In this subsection, we do not attempt to come up with a formal description of the retrieval capability of MediaView (e.g., in terms of a complete query language); instead, we provide some query examples here to illustrate the usage and expressive power of these operators as "querying constructs". For expository purpose, our query examples are all from the application domain of e-publishing. The "e-magazine", "e-journal", and "e-Newspaper" used in the examples are all media views, which have been already defined.

Example 1 - Locating a subject object.

Consider the following query: *Find out which e-magazine has published an issue in 1990s that contains a speech by Richard Nixon when he visited China in 1972.*

This query can be expressed by a combination of the following MediaView operators:

Select x.title, y.volume **from** e-magazine x

where y **in components**(x) **and** y.year **within** [1990, 2000]

and z in components(y) and class(z) = speech

and z.year = 1972 **and** z.venue = "China"

and z.speaker = "Richard Nixon";

Example 2 - Frequency checking

Consider the following query: Find out how many issues of the e-ReadersDigest magazine for the year of 2000 have enclosed a song related to sea.

This query can be expressed by a combination of the following MediaView operators:

Select x.volume **from** e-magazine x

where x.title = "e-ReadersDigest" **and** x.year = 2000

and y in components(x) and type(y) = song

and y.lyrics **contains** {"sea", "ocean"};

In the above query, contains is a "syntactic sugar" used for checking containment between a textual document (in this case, the lyrics of a song) and any keyword from a given set (in this case, "sea" or "ocean", or both).

Example 3 - Commonality checking

Consider the following query: F*ind out from the e-Newsweek and e-ReadersDigest any common objects (articles, photos, audio, or videos) in the issues of the year 1999.*

This query can be expressed by a combination of the following MediaView operators:

```
cross (( select x from e-magazine x
            where x.title = "e-ReadersDigest"),
        ( select y from e-magazine y
            where y.title = "e-Newsweek"));
```

Example 4 - Content-based retrieval

Consider the following query: *Find and rank all the e-journals whose cover pages are visually similar to that of DMKD ("Data Mining and Knowledge Discovery") by Kluwer Academic.*

This query can be expressed by a combination of the following MediaView operators:

```
Select x.title from e-journal x, y
where y.title = "Data Mining and Knowledge Discovery"
    and y.publisher = "Kluwer Academic"
order_by similarity(x.coverpage, y.coverpage);
```

In the above query, similarity is a content-based function used for calculating the low-level feature similarity between two media objects (in this case, two cover page images). Depending on the base class of the objects, this function can be implemented using visual features such as color, shape, and texture, or other primitive features such as keywords (if the objects are documents) and pitch contour (if the objects are audios).

Example 5 - Associative search

Consider the following query: Find from the e-Newspaper all the pairs so that not only both have similar headlines, but also common objects (articles, photos etc.) inside; rank the pairs according to their levels of similarity.

This query can be expressed by a combination of the following MediaView operators:

```
Select x.title, y.title from e-Newspaper x, y
where i-overlap (x, y) = true
order_by similarity(x.headline, y.headline);
```

Here, similarity is also a content-based function used for calculating the similarity between two sets of strings (in this case, the headlines). The i-overlap predicate is used for testing two instance objects to see if there is any common sub-object (cf. Table 1).

9.4.2. Navigation in "Media Map"

Besides the query operations, browsing and navigation are also among the most popular user behaviors in multimedia databases. As a part of the MediaView mechanism, a novel navigation paradigm—"media map"—is introduced based on the semantic graph model. The proposed paradigm facilitates the user to traverse the database in a semantics-oriented manner, which offers better user experience compared with traditional paradigm for navigation.

A tourist map usually supplies readers with the guides on the routes that they should follow to visit some of the sites. Our semantic graph model plays the similar role in a multimedia database: it defines the paths (semantic links) through which the user can traverse from an object to other semantically related objects. In this regard, we treat the semantic graph as a "media map" and propose the following scenario for navigation. When the user navigates to a certain multimedia object, he will be also given a set of candidate objects that he may traverse to from his current position. Typically, the candidates are the objects that link (directly or indirectly) to the object currently being visited. In a graphical interface, for each candidate object we display a small icon as a hint of its content, with its correlation factor with the current object shown alongside. After deciding the next object to visit, the user clicks the corresponding icon from the list and the system will lead him to the selected object.

In addition to traversal at object level, the semantic graph model also supports traversal at view level based on the relationship between media views, which is defined by the v-overlap operator (see Table 9-1) based on their common members. The scenario is similar to the object-level navigation: When the user is browsing the content of a certain media view, he is also suggested a list of related media views identified by applying v-overlap operator. These candidate views can be ranked according to the degree of their overlap with the current view in terms of common members.

The proposed navigation paradigm makes sense to multimedia databases because it captures the semantic correlations between objects or object clusters (media view). This semantics-oriented navigation offers smoother user experience than the traditional navigation paradigm, in which users browse the objects organized by their base classes. Suppose there is a user who is fascinated with van Gogh's painting "Sunflower" and therefore interested in reading his biography. Traditionally, he has to browse all the textual documents (as biography is a kind of textual document) in order to find the desired biography, which is a tedious and annoying work. In the "media map" paradigm, when the user is looking at "Sunflower", all other relevant materials (including van Gogh's biography) are recommended to him by the system, so that he can navigate to the interested biography without any trouble in the traditional scenario.

9.4.3. Optimization of Storage and Indexing

As a by-product of our mechanism, after many media views have been defined, we can improve the physical storage and indexing of multimedia objects based on a better knowledge of their mutual semantic correlation. Specifically, the (n-level) correlation factor between two objects reveals an inclination that the two objects are accessed together or successively, which provides a strong guideline for optimizing the storage and indexing tasks. For example, if two objects have a high correlation factor, they should be stored closer to each other physically (e.g., in the same disk block) to allow efficient access from the disk. On the other hand, we may create an index for a pair or a chain of objects with high mutual correlation, so that they can be retrieved much more efficiently.

Practical algorithms for the optimization of storage and indexing need to consider more factors, such as the structure of the schema and the characteristic user queries. This would be an interesting topic for further study, which is beyond the scope of the work presented in this chapter. Our intention here is to show the potentials of MediaView in supporting such "back links" from the view level to internal level for improving physical storage and indexing.

9.5. Summary

In this chapter, we have discussed the issue of semantic modeling of multimedia data and reviewed the related techniques on semantic and multimedia data modeling. Especially, we have presented *MediaView* as a novel object-oriented view mechanism for multimedia databases. Specifically, it consists of a collection of media views providing application-specific contexts in which the elusive semantics of multimedia data can be properly interpreted. *MediaView* supports sophisticated multimedia-flavored queries, promotes a novel paradigm for convenient navigation, and hints the semantic correlation between objects for the optimization of the physical storage and indexing.

Due to the great potentials of *MediaView*, many interesting future works can be done to extend its capability. For one thing, we attempt to define a set of "view algebras" with explicit semantic interpretations, in addition to the object algebra currently used. Since each media view corresponds to a semantic concept, it is reasonable to define view algebras from a purely semantic perspective, such as "synonymous", "antonymous", and "hyponym". If successfully defined, these view algebras will increase the capacity of *MediaView* in view derivation and querying. Another future work involves the efficiency aspect of the *MediaView* mechanism. Along this direction, new algorithms for indexing and physical storage of objects will be developed, in order to enhance the efficiency of object retrieval from media views. Schema structure and statistics of object access pattern have to be studied for this purpose.

Reference

1 Abiteboul, S., and Bonner, A., "Objects and Views", in *Proc. of ACM SIGMOD Conf. on the*

Management of Data, pp. 238—247, 1991.

2 Adiba, M., "STORM: Structural and Temporal Object-Oriented Multimedia Database Systems," *IEEE Int. Workshop on Multimedia DBMS*, NY, USA, August, 1995.

3 Albano, A., Bergamini, R., Ghelli, G., and Orsini, R., "An object data model with roles," in *Proc. of the Int. Conf. on Very Large Databases*, pp. 39--51, 1993.

4 Bertino, E., Catania, B., and Ferrari, E. "Query Processing". In *Multimedia Databases in Prespective*, pp. 181-217, Springer, 1997.

5 Brin, S., and Page, L., "The anatomy of a large-scale hypertextual web search engine", in *Proc. 7th Int. WWW Conf.,* 1998.

6 Butterworth, P., Otis, A., and Stein, J., "The GemStone object database management system." *Comm. of the ACM*, 34(10):64--77, 1991.

7 Dayal, U., "Queries and Views in an object-oriented data model." in *Proc. of 2nd Int. Workshop on Database Programming Languages*, pp. 90--102, Morgan Kaufmann, 1989.

8 Deux, O., et. al. "The O2 System." *Comm. of the ACM*, 34(10): 34--48, Oct. 1991.

9 Faloutsos, C. Indexing of Multimedia Data. Multimedia Databases in Prespective, pp. 219-245, Springer, 1997.

10 Haas, L. M., "Supporting Multi-Media Object Management in a Relational Database Management System", Technical report. IBM Almaden Research Center, 1989.

11 Heiler, S., and Zdonik, S. B., "Object views: Extending the vision", in *Proc. of IEEE Data Eng. Conf.,* pp. 86-93, Feb, 1990.

12 Kim, W., Garza, J. F., Ballou, N., and Woelk, D., "Architecture of the ORION Next-Generation Database System," *IEEE Trans. on Knowledge and Data Engineering*, Vol. 2, No.1, 1990, pp. 109-124.

13 Klas, W., Bool, S., and Lohr, M. "Integrated Database Services for Multimedia Presentation". In *Multimedia Information Storage and Management,* Kluwer Academic Publishers, USA, 1996.

14 Kleinberg, J., "Authoritative sources in a hyperlinked environment." in *Journal of ACM,* 46:5:604-632, 1999.

15 Kuno, H. A., Rundensteiner, E. A., "Materialized Object-Oriented Views in *MultiView*", in *Proc. RIDE-DOM,* IEEE CS Press, 1995.

16 Li, Q., and Smith, J., "A conceptual model for dynamic clustering in object databases," in *Proc. of the Int. Conf. on Very Large Databases*, pp. 457--468, Aug, 1992.

17 Oomoto, E., and Tanaka, K., "OVID: Design and Implementation of a Video-Object Database System", in *IEEE Trans. on Knowledge and Data Engineering*, vol.5, pp 629-641, 1993.

18 Orenstein, J. A., "A Comparison of Spatial Query Processing Techniques for Native and Parameter Spaces", In *Proc. of ACM SIGMOD Conf.* pp 343-352, 1990.

19 Ozsu, M. T., et. al. "TIGUKAT: A uniform behavioral objectbase management system." *VLDB Journal*, 4(3), VLDB Endowment, July 1995.

20 Pernici, B., "Objects with roles." in *Proc. of ACM Conf. on Office Information Systems*, pp 205—215, ACM, 1990.

21 Richardson, J., and Schwartz, P., "Aspects: Extending objects to support multiple, independent roles." in *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pp. 298--307, 1991.

22 Rundensteiner, E. A., "*MutiView*: A Methodology for Supporting Multiple Views in Object-Oriented Databases", in *Proc. of the 18th Conf. on Very Large Database*, Canada, 1992.

23 Santini, S., Gupta, A., and Jain, R., "Emergent Semantics through Interaction in Image Databases", in *IEEE Transactions on Knowledge and Data Engineering,* vol. 13, No. 3, 2001.

24 Scholl, M. H., Lassch, C., and Tresch, M., "Updateable Views in Object-Oriented Databases", in *Proc. of 2nd DOOD Conf.*, Germany, Dec. 1991.

25 Sciore, E., "Object specialization." in *ACM Trans. on Information Systems*, 7(2): 103--122, April, 1989.

26 Su, J., "Dynamic constraints and object migration." in *Proc. of the Int. Conf. on Very Large Data Bases*, pp 283-242, 1991.

27 Tanaka, K., Yoshikawa, M., and Ishihara, K., "Schema Virtualization in Object-Oriented Databases", in *Proc. of IEEE Data Eng. Conf.*, pp. 23-30, Feb, 1988.

28 Woelk, D., Kim, W., Luther, W., "An Object-Oriented Approach to Multimedia Databases", in *Proc. ACM SIGMOD Conference,* pp. 311-325, 1986.