ELSEVIER

Future Generation Computer Systems 16 (1999) 9–20

FGCS
FUTURE GENERATION COMPUTER SYSTEMS

An extensible information model for shared scientific data collections

Amarnath Gupta *, Chaitanya Baru

San Diego Supercomputer Center, University of California, San Diego, 9500 Gilman Drive, Building 109, La Jolla, CA 92093-0505, USA

Accepted 11 February 1999

Abstract

An information model is defined to support sharing of composite-media scientific data. The model consists of *data objects* and *links*. Data objects are associated with *descriptors* which contain all the metadata related to the object. A novel aspect of the information model is that both the data and metadata associated with a data object can be in structured or semistructured form. The links in the model are typed, and contain several built-in types such as *resolution* and *derivation* link types, to adequately model object relationships in scientific data. The model is extensible in the sense that users are allowed to define new object types and new link types. At the San Diego Supercomputer Center, we are currently investigating issues in providing the necessary infrastructure to implement this model. ©1999 Elsevier Science B.V. All rights reserved.

Keywords: Scientific information management; Extensible information model; Semistructured data model; XML

1. Introduction

There is growing consensus among computational scientists that observational data, results of computation and other forms of information produced by an individual or a research group need to be shared and used by other authorized groups across the world through the entire life cycle of the information [1]. Ideally, the scientist would have a technological infrastructure, implemented as a software architecture, which can be used to accomplish any mode of information sharing such as a database system that can be queried, a dissemination channel like the World Wide Web where information can be published and browsed, and collaboration software for simultaneous access. To create this information sharing infrastructure, it is necessary to develop a model to characterize both the nature of scientific information and the manner in which it is used. The model must be general enough to span the information sharing needs of multiple scientific disciplines, and yet efficient enough to satisfy any special requirements for a specific type of data. In particular, the model must be able to capture four interesting aspects of scientific information which set it apart from usual business information:

- Scientific information is typically *composite* in nature. In addition to regular, structured data, scientific information includes documents, images, videos, three-dimensional volumes, visualized simulation experiments and any combinations of these.
- In the scientific world, one needs to deal with both *data-centric* and *process-centric* views of information. In other words, while it is important to have

* Corresponding author
E-mail addresses: gupta@sdsc.edu (A. Gupta), baru@sdsc.edu (C. Baru)

PII: S0167-739X(99)00031-X

access to information, often it is also important to know how the information was derived.

- A given item of information is generally associated with several other diverse items of information, possibly distributed, which collectively forms a *heterogeneous context* of the original information. In biology, for example, the context of a protein structure may be an observed phenomenon in a cell line of an organ, a 2D electrophoretic separation and a mass-spectrometer plot.
- Scientists often require *integrated access* to information combining retrieval, computation and visualization of individual or multiple datasets.

The information model presented in this paper must support the mode of operation in scientific applications where information is often created by iterating through the steps of acquiring data, performing some 'procedure', analyzing the results, and then obtaining the next set of data. The data may be structured, e.g., information stored in a relational database, or semistructured/unstructured, e.g., text documents, multimedia information, or visualization of datasets. Procedures and analyses performed on the data may include a treatment protocol on a specimen, expert interpretation of observed data, and a computational procedure on an observed data set. The contributions of this paper are (i) a proposed 'hybrid' information model that integrates structured and semistructured data, and (ii) demonstration that the semistructured information model proposed by database researchers for other applications [2–4] is also well-suited for modeling composite-media scientific information.

The rest of the paper is organized as follows. In Section 2, we describe two typical usage scenarios for scientific information. In Section 3, we provide details of the information model with accompanying examples. Section 4 discusses issues related to the underlying infrastructure needed to implement this model. Section 5 concludes the paper with a note on our future directions.

2. Two usage scenarios

In this section, we describe two application scenarios demonstrating different aspects of scientists' approach to information.

2.1. Usage scenario I

Consider several groups of earth scientists who are jointly investigating biodiversity of a certain animal species in the state of California. Assume that each of these groups collects and studies a specific type of information such as natural habitats, land use, industrial pollution, effects of urbanization, climate and the population dynamics of the species of interest. Furthermore, the groups work with publicly available data from sources such as the Census Bureau and the Environmental Protection Agency at the Federal and state levels. In the course of their study, the scientists create derived maps from raw observations. For example, one map may show the increase in urbanization and traffic pollution from, say, 1996–1998, while another shows the change in the natural habitat of a specific species along with the change in its population density over the same period. These derived maps, created by compute intensive processing of raw datasets residing in distributed locations in heterogeneous databases, are stored for future access. Along with the maps, the system stores information about the attributes that are represented in the maps. It also stores the timestamp at which this map has been generated along with relevant information on accuracy limitations. The creator of the derived information may also store the maps at multiple levels of spatial resolution, where a coarser level of resolution provides an aggregated view of the data, which may highlight an aggregated property that is lost at finer resolutions. Hence, the coarse level information may be additionally associated with its own 'description' which is not present in the original, finer level map. A user accessing the maps may issue a query specifying the region of interest, attributes of interest and a desired level of resolution. If such a map has been computed and stored before, it is fetched from the database; otherwise the system computes the maps from raw information. If the user's request covers only a part of the mapped region, then only related subregions are extracted from the map when the results are presented to the user at the desired level of resolution. If a map satisfying the query is available, but at a finer level of resolution, then the system computes a coarse resolution map before presentation. After inspecting the presented map, the user may request a related piece of information, such as the latest SPOT data of the same region. In response, the system

issues a query to a corresponding, possibly remote, database and locates the information. Finally, the user may wish to compute the variation of a mapped variable over a period of time. A given map is submitted as part of a query to fetch other related maps and a computation is invoked to compute the variation. The results may be viewed through a presentation/visualization interface.

2.2. Usage scenario II

Consider several teams of biologists sharing information through a collaborative environment. Assume that one of the scientists is demonstrating an observation by performing telemicroscopy on a cell specimen. The microscope image is visible to other collaborators. The experimenting scientist takes a snapshot at a specific magnification and stores it along with the experimental settings. A region of interest is marked on the snapshot and a corresponding annotation is made, which is also stored. Next, the scientist wishes to view the region of interest at a higher magnification by taking a second snapshot. Again, upon identifying the specific site of interest, the scientist makes further annotations. At that point, a participant in the collaboration environment retrieves a previously stored tomographic reconstruction of the specific sub-cellular structure of interest. All the scientists now jointly view the original microscope snapshots, the experimenter's annotations and the 3D reconstruction. During the collaborative viewing and interpretation of this information, further measurements are taken, shapes are analyzed and inferences noted. All of this additional information is stored in the system for future reference and analysis. In fact, as part of the comparative analysis, similar information from past experiments may be retrieved and presented in the same environment. Information from past experiments is obtained by issuing queries based on attributes that describe the context of the experiments. Once retrieved, this information can be linked to the current experiment to provide context in the future. The search for information needed to facilitate this collaboration scenario may also reach out to the Internet. For example, a set of keywords stored in an annotation may trigger a bibliographic query to a MedLine service, and retrieve a ranked set of abstracts to be viewed together as part of the collaborative session.

2.3. Role of the information sharing infrastructure

The role and need for an information sharing infrastructure is clearly demonstrated from the above example scenarios.

These examples impose basic requirements on the information model including:

- support for multi-resolution information, especially for multimedia data;
- support for derivation ancestry, where a given item is derived in a known way from one or more other items in the lineage;
- the ability to define named sets of objects of a given type, along with 'context' information (i.e., a set of related or linked objects) for each object. This is useful for query processing and for exporting data to external processes, e.g., collaboration systems, and
- support for associating unstructured and semistructured objects, like free text annotations, with data objects.

The following section describes the information model used to support scientific applications such as the ones described in the previous subsections.

3. The information model

The information model is based on the abstractions of data *objects* and *links*. A data *object* consists of a data *item*, which is the actual observed datasets, or product of computation, and a data *descriptor*. The data descriptor contains a variety of metadata associated with the item, which can be queried. A data object is associated with a *type*, where the type of the object is defined by the schema of the descriptor associated with the object. In the scientific domain, examples of data objects are, LANDSAT images, results of a simulation experiment, and text documents of field observations. In each case, a descriptor schema can be specified to provide attribute-based access to the data objects. A *data descriptor* includes *assigned* (or, *non-derived*) metadata and *content* (or, *derived*) metadata. Assigned metadata includes metadata values assigned by external users and which are not derivable from the actual content of the data object. Examples include name of the author, title, and pixel resolution (for images). Content metadata examples include

number of pixels in an image, size of a file, and aspect ratio [1].

Data objects may be related to other data objects via typed links. Links may be manually assigned or computationally derived. Links represent instance-to-instance relationships and, unlike say the entity-relationship (ER) model, it is not mandatory for all instances of a given data type to have the same relationship with objects of another data type. Due to its navigational nature, we believe that the instance-based, or 'object-oriented', approach is suitable for modeling scientific information, in conjunction with a set-based approach for providing ad hoc querying capability.

3.1. Modeling data objects

As mentioned above, a data object consists of a data item and a data descriptor. The data descriptor is associated with a *schema*. The descriptor schema may be fully structured or partially structured. A fully structured schema has a fixed set of attributes, is non-nested, and is expressible as a relational schema which can be queried using a relational query language such as the Structured Query Language (SQL). A partially structured schema has a structured component as well as a semistructured component. The semistructured part is expressed in XML [5] and may or may not have an associated Document Type Definition (DTD). The *type* of an object is determined by the structured part of its schema.

The model supports the notion of a type hierarchy. Thus, if the type of an object **X** is a subtype of the type of some other object **Y**, then the structured schema associated with **X** is a *superset* of the structured schema associated with **Y**. The model provides a set constructor to allow the grouping of data objects of the same type into a named set. This provides the mechanism for users to create their own sets of data objects and to issue queries on those sets. In a typical implementation, a system will define a basic object type, or schema, for all objects (e.g., the set of attributes implied by the Dublin Core definition). Thus, this basic schema defines the so-called *base type* from which all other types are derived. For example, one may derive an image object type from this base type. The image object type defines a metadata schema that is common to all images regardless of the types of images (e.g., satellite versus microscopy) and regardless of the application domains to which they belong. From this image type, one could derive a subtype specific to, say, LANDSAT images. Further, different groups with collections of LANDSAT images may choose to employ different annotations for these images, at which point a semistructured data model may be employed to model annotation metadata associated with the images. In this example, third-party applications aware of the LANDSAT schema can query the data collections using that schema, while applications that understand the semistructured extensions may utilize the additional attributes specified in the semistructured part of the descriptor.

The notion of partially structured schemas is a unique capability that has not been discussed in previous related work, such as Lore [6,7]. The ability to define strict type-subtype hierarchies while simultaneously providing the flexibility to define semistructured descriptors is particularly useful in scientific information. As mentioned in Section 2.2, scientists often record inferences or annotations with raw observations and computations, on a per case basis. Thus, all instances of the same data object type may not share the same information structure for ad hoc observations. However, by representing this information in the form of ⟨attribute, value⟩ pairs, one can employ a semistructured model for the data. As mentioned earlier, even for the semistructured information, it is possible to register a DTD (i.e., schema). In this case, the DTD is used to support querying. Otherwise, data in the semistructured part can be navigated only if there is a recognizer that can parse the structure. If such a parser is not available, then the semistructured information is simply treated as a binary large object (BLOB). To support the semistructured model, we employ XML as the data specification language. For each object, the semistructured component is stored as a text block called **ssdata** [2], starting with

[1] Note that the distinction between assigned and derived metadata is not rigid. For example, if a user directly records a voice annotation associated with a video segment, it would be considered assigned metadata with respect to that video segment. However, if the audio annotation is extracted from the audio channel of the video, then it would be content metadata.

[2] If object **x** has a non-null value in **x.ssdata**, it has a semistructured component, else it is purely structured.

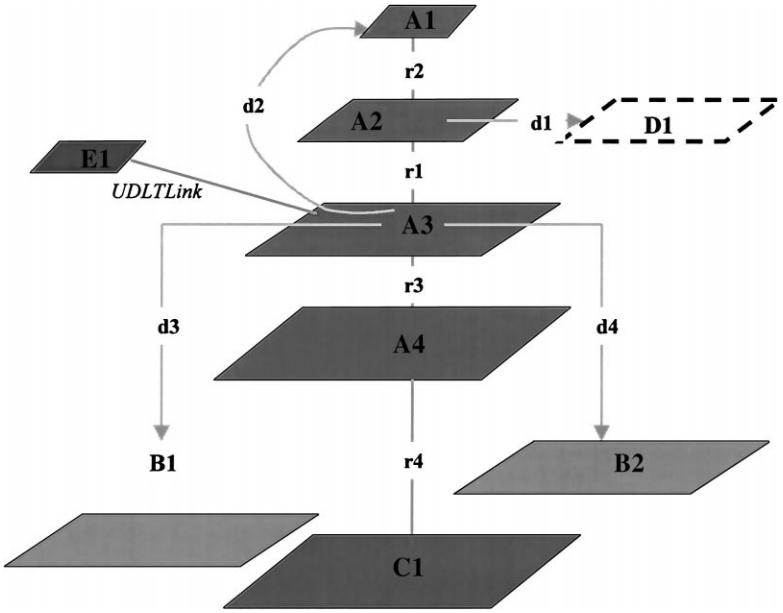

Fig. 1. Data model example.

the standard XML document prolog such as ⟨?xml version="1.0" encoding="UTF-8" ?⟩. While we do not provide a formal treatment of the model in this paper, it can easily be shown that, in the general case (if inheritance is supported), the semistructured data is equivalent to a directed acyclic graph. For example, the following semistructured description (without inheritance) results in the tree shown in Fig. 1.

⟨L1⟩ abc
⟨L2⟩ def ⟨/L2⟩
⟨L2⟩ ghi ⟨/L2⟩
⟨L3⟩ jkl
⟨L4⟩ mno ⟨/L4⟩
⟨/L3⟩
⟨/L1⟩

Operators to traverse from any node in the graph to its descendants or ancestors can be defined. One can further constrain the traversal by specifying selection conditions on the labels (i.e., attributes) and values of the destination nodes.

3.2. Modeling links

Links are associations that relate data objects in an information context. A link is minimally represented as the 5-tuple ⟨link_id, link_label, link_type, from_object, to_object⟩. The attribute link_label is a textual description of the semantic role of the link and can be queried like any textual attribute. Although the link is binary, it can always be used to model one-to-many links by first creating a group of target nodes, renaming it, and then connecting the source link to the group object.

3.2.1. Link types

Note that contrary to many graph-based data models [8,9], we support the notion of a link *type*. The concept of a link type allows the modeling of the structure and behavior of links and permits meaningful navigation along inter-object links. The link types also serve to constrain the relationship among data objects. The information model includes *built-in* link types and also provides support for *user-defined* link types. Built-in link types have predefined semantics. At this point, we recognize the following link types:

1. *Ordering Links*: This link type can be used to create ordered sequences of data objects. The link specifies a precedence relationship between two data objects of the same type. The ordering

link is implicit, effected through an ordering attribute from the metadata of the data object or by using a variable within the link itself. The choice to use one over the other is a matter of user preference and does not affect the semantics of any operation. Ordering links are described by the 6-tuple ⟨link_id, link_label, link_type, from_object, to_object, order_attrib⟩. The attribute link_type specifies whether the ordering attribute is in the metadata or in the link, and the last attribute refers to the ordering attribute. A typical ordering variable is *time*, which may be used to order a sequence of observations in an experiment. A more complex example is a sequence of images that represent a progression of slices through a 3D object, such as the coronal sections of a human brain.

2. *Resolution Links*: This is a special type of ordering link where the link is between two related image data objects of different resolutions. Resolution links always connect objects that are adjacent in the resolution dimension. However, a resolution link deviates from a simple ordering link in two ways. First, it is possible that the data that two data objects on two sides of a resolution link have are of different data types. For example, in Section 2.1, a resolution link may connect a LANDSAT image of a region under study to a much larger higher resolution SPOT image (a different data type) where further details of industrial occupancy of a region can be observed. Secondly, a higher resolution image may cover only a portion of a lower resolution image. A typical example is that of microscopy images where, as mentioned in Section 2.2, only regions containing interesting objects are zoomed. In this case, the resolution link additionally contains the spatial coordinates (only the two diagonal coordinate points for a rectangular region) in the low-resolution image corresponding to the high-resolution image.
3. *Derivation Links*: If data object o_2 is obtained by performing some operation on data object o_1, o_1 is connected to o_2 by a derivation link. The purpose of the derivation link is to establish the *lineage* of a data item by keeping enough information so that any data item can be traced back to other data objects that led to the current item. The operations themselves can have a wide variety, ranging from executing a computational procedure, to performing the next step in an experimental protocol. We model a derivation link by a link object and a corresponding operation object. The link object derives from the standard link and contains three additional attributes ⟨operation_id, operation_instance_id, materialized⟩. The first attribute refers to a registered operation such as a computational procedure known to the system. If the operation is unknown a special identifier is used. The operation instance identifier links the current derivation link to an instance of the operation object describing the operation. This operation object contains the parameters of the operation and may include a textual description or a reference to another data object. The *materialized* attribute specifies whether the data object is already materialized or needs to be computed at run-time by executing the operation. As an example of the above, consider that a classification procedure is executed on a LANDSAT image classifying each pixel into one of a finite set of categories (deep water, shallow water, asphalt, vegetation etc.). This classification procedure may need to specify parameters like minimum inter-class separation and minimum compactness of a class. Then the operation object identified by operation_id may have attributes ⟨operation_instance_id, interclass_separation, class_compactness, description, command⟩ where description is for annotations and comments and command is a method that uses the other attributes to create the actual command to execute the classification procedure. If multiple objects are used to derive a single data object, the data objects are grouped first and the derivation link connects the group to the derived object.
4. *Reference Links*: A reference link is a virtual connection to an object outside the scope of the infrastructure, such as an object at a website with a known URL. The reference link additionally contains a possibly null method to construct such a reference from the data object it attaches to. This can be used in constructing a query string to fill an HTML form and perform a search as mentioned in the example scenario of Section 2.2.
5. *User-defined Links*: To provide extensibility, the model supports user-defined links that can be reg-

istered by the application designer by using the *registerLinkType* function. The user first defines the new link type by inheriting from an existing link type. This process can be used not only to specify additional attributes but also to define constraints such as the data type of the participating nodes, whether group nodes are allowed on either ends of the link, and specific dependencies that needs to exist on the data items on both end of the link. For example, spatial adjacency links may be created on image data objects, where all connected images are captured from the neighboring fields of a single microscope slide.

3.2.2. Link operations

Once an object, or subset of objects, has been identified as the result of an ad hoc query, link operations allow navigation among data objects with a given object as the point of reference. As one navigates links to reach other objects the point of reference, or *currency*, changes to the new object. Examples of link operations are:

- Find all links at a node: the operation returns the link_ids of all links associated with that node, regardless of link type.
- Find all links of a particular type, at a node: similar to above, this operation returns all links of a particular type associated with the given node.
- Find end-points of each link (all links): returns the object IDs of all objects directly connected via a link to the current object.
- Get link metadata for a given link: retrieves all metadata attributes associated with a link.
- Get to the object at the other end of a link (i.e., traverse link): changes the currency to the object reached by traversing a given link.

The above is not meant to be an exhaustive set of possible link operations, but it provides examples of the types of navigational operations that are possible in the system.

3.3. Modeling example I

The information model can be formally expressed as a graph where the data objects correspond to nodes and the links correspond to edges. To support information sharing and collaboration, it is necessary to provide the ability to identify subgraphs that correspond to sets of linked objects that are of interest. The graph model of the data can itself be expressed as semistructured data using XML. Similarly, the subgraph representing objects of interest can also be expressed as semistructured data. Thus, when users query, share, or exchange information about objects, they use XML as the medium for doing so. We illustrate this with an example shown in Fig. 1 which depicts a set of data objects and different types of links among these objects. Assume that the objects labeled A1–A4 are, a set of LANDSAT images. With A3 as the reference, A1 and A2 are lower resolution images corresponding to A3. The resolution links, r1 and r2 capture this relationship. Similarly, A4 is a higher resolution image of a particular site and/or region in A3. Object C1 is also a higher resolution image of the same region. However, the type of C3 is different from the type of A1–A4. For example, while A1–A4 are all Landsat images (at multiple levels of resolution), C1 may be a SPOT data image.

B1 and B2 are derived from A3, and are objects of a different type than LANDSAT images. Thus, they are both linked to A3 via derivation links. Another example of a derivation link is the link between A2 and D1. While D1 is derived from A2, the dotted representation for D1 indicates that this object is not physically stored in the collection. Rather, it is generated on-the-fly by invoking the derivation method d3 on object A2. The link d2 from A3 to A1 indicates that the lower resolution image, A1, was derived from A3. The figure also shows a link between A3 and the object E1 via a link labelled, *UDLTLink*, which is an instance of a user-defined link.

The graph adjacency information for data object A3 can be represented as follows:

⟨GRAPH_OBJECT⟩
⟨DATA_OBJECT⟩
⟨DESCRIPTOR⟩
⟨STRUCTURED_METADATA⟩
⟨OBJECT_ID⟩ A3 ⟨/OBJECT_ID⟩
. . . all other structured metadata attributes (derived from supertypes as well as locally defined). . .
⟨/STRUCTURED_METADATA⟩
⟨MOREDATA⟩
. . . any semistructured metadata attributes, if they exist. . .

⟨/MOREDATA⟩
⟨/DESCRIPTOR⟩
⟨ITEM⟩
. . . contents of data item (or a reference to it). . .
⟨/ITEM⟩
⟨/DATA_OBJECT⟩
⟨LINKType = RESOLUTION, Id = r1⟩ Label info ⟨/LINK⟩
⟨LINKType = RESOLUTION, Id = r3⟩ Label info ⟨/LINK⟩
⟨LINKType = DERIVATION, Id = d2⟩ Label info ⟨/LINK⟩
⟨LINKType = DERIVATION, Id = d3⟩ Label info ⟨/LINK⟩
⟨LINKType = DERIVATION, Id = d4⟩ Label info ⟨/LINK⟩
⟨LINKType = UDT_LINK, Name = MyLink, Id = UDLTLink⟩ Label info ⟨/LINK⟩
⟨/*GRAPH_OBJECT*⟩

For a particular collaborative session, if a set of users are only interested in sharing specific derived information related to object A3 (i.e., A3 along with B1 and B2), they may wish to share the following subgraph of A3:

⟨SUBGRAPH_OBJECT⟩
⟨DATA_OBJECT⟩
⟨DESCRIPTOR⟩
⟨STRUCTURED_METADATA⟩
⟨OBJECT_ID⟩ A3 ⟨/OBJECT_ID⟩
. . . all other structured metadata attributes (derived from supertypes as well as locally defined). . .
⟨/STRUCTURED_METADATA⟩
⟨MOREDATA⟩
. . . any semistructured metadata attributes, if they exist. . .
⟨/MOREDATA⟩
⟨/DESCRIPTOR⟩
⟨ITEM⟩
. . . contents of data item (or a reference to it). . .
⟨/ITEM⟩
⟨/DATA_OBJECT⟩
⟨LINKType = DERIVATION, Id = d3⟩ Label info ⟨/*LINK*⟩
⟨LINKType = DERIVATION, Id = d4⟩ Label info ⟨/*LINK*⟩
⟨/SUBGRAPH_OBJECT⟩

3.4. Modeling example II

Next, we discuss an example where different approaches can be used to model the same information using the proposed information model. This results in modeling of information at different levels of detail and, correspondingly, different querying capabilities. The following discussion explains three different ways of modeling data, using video information related to a microscopy experiment as an example:

(a) *Video modeled as a BLOB with minimal (assigned) metadata*. The video information is modeled as a data object which belongs to the base type supported by the system. Thus, only minimal metadata is stored in the data descriptor. The data item itself is represented as a BLOB. With this model, one can query the minimal metadata, such as the set of Dublin Core attributes represented in the base type. However, it is not possible to issue a query that is specific to the contents of the video itself, since the video information is simply modeled as a BLOB. The only operations possible would be the basic *getobject*/*putobject* operations (i.e., play video and store video).

(b) *Video modeled as a BLOB with type-specific (assigned) metadata*. The schema for the data descriptor is defined as a video-specific subtype of the base type. Thus, the system is able to represent a variety of video-related metadata in addition to the basic metadata stored in case (a). For example, this metadata may include the duration of the video, the bit rate of MPEG encoding, the frame dimensions, and whether the video has audio and closed caption channels. This video subtype can be further specialized to include *feature* metadata that describes the content of the video. This is done by providing the ability to define *annotation tracks* which are tuple-structured attribute fields containing values for ⟨time_in, time_out, annotation_data⟩. The first two items specify a time interval in the video and the third item is the actual annotation which may be semistructured. A similar track-oriented video content description scheme was proposed by Weiss, Duda and Gifford [10] and has been commercially implemented in the Video Foundation Blade by Informix [11].

Given this model for the video object, the system can support retrieval of video segments based on queries such as, "video segments that are more than 1 min long, where the keywords 'monoclonal antibody' and 'Jurkat cells' are present in one or more annotation tracks". Using the time interval information in annotation tracks the system can also support the retrieval of only the desired segments of video. The sequence of frames that constitute the video is still represented as a BLOB.

(c) *Video modeled as an object with links to derived metadata*. The annotation track information can be enhanced to include key frames from the video designating particular event (e.g., cell death). Thus, the key frame is represented as derived information with respect to the particular video (or video segment). In addition, the key frame image may itself be further annotated to indicate, say, the boundary of the dying cell and the decreasing diameter of its nucleus. This latter annotation is derived information based on the key frame image object. Thus, rather than modeling the video as a BLOB, the information model now models the image contents of the video itself. The user may now issue a query based on where certain events occur within the video. This approach requires the user to model this information explicitly and to provide the neccesary data.

As in case (b), the data descriptor associated with the video has a subtype that is specific to video information. In addition, the descriptor may also inherit from a domain-specific type, for instance, it may inherit the schema that applies to information related to the subject 'cellular phenomena'. However, as mentioned before, since there can be high variability in the type of experimental information and annotation available for each observation, we allow this information to be represented using the semistructured representation.

4. Implementation issues

In this section, we discuss requirements on the underlying storage infrastructure needed to implement the information model introduced above. The infrastructure should provide efficient support for:

- storage and querying of metadata associated with data objects;
- navigation along inter-object links;
- storage of data objects including multimedia objects such as images, videos, geospatial, text;
- scalability and the ability to handle a wide range of collection and object sizes. Some collections may range in the millions of objects, while others may contain individual objects that may range up to terabytes (and even petabytes) in size;
- transparent distribution of and access to data stored across a wide area network and across heterogeneous storage resources; (An example of a system that provides this type of capability is the IBM Garlic project [12].)
- querying by object content;
- spatial access to data; (Many data objects are spatial in nature, e.g., remote-sensing data which may be accessed using latitude/longitude, and X-ray images which may be accessed using x and y coordinates and
- efficient on-the-fly materialization of derived objects.

In the following, we discuss some of the storage implementation issues in more detail.

4.1. Storage and querying of metadata

Ad hoc querying based on descriptor schemas is the fundamental access method for identifying data objects of interest. Thus, the ability to store and query metadata in a standard way is a basic requirement. For the structured part of the descriptor schema, we propose to use SQL as the standard query language. Relational databases are a mature technology and many implementation choices are available.

For managing the semistructured part of the descriptor schema, we need efficient mechanisms for storing and querying semistructured data. We will use XML as the standard language for representing this information. XML itself is an emerging standard and work has only just begun to provide database support for XML, and semistructured data in general [7]. As part of our future work, we will investigate issues in providing a schema definition and query language based on XML.

4.2. Storage of data

In many instances, scientific data sets are stored in ordinary UNIX files, though in some scientific fields,

e.g., molecular biology, the use of database management systems is more prevalent. Several options are available for storing data objects, as discussed below.

4.2.1. Use of database tables

If the data item associated with an object can itself be represented as a set of fields, or 'structured' columns (i.e., columns of type integer, float, character, etc.), then this information can be stored directly as a row in a database table. In this case, applications can query the object content simply by using the query language supported by the corresponding DBMS. If the data item is in semistructured form, the options are to convert this into an internal structured representation which can be implemented using a relational database, or to store the semistructured data in its native form. As mentioned before, the latter requires database systems that can efficiently manage semistructured data.

4.2.2. Use of BLOBs

The data item may be stored as a BLOB in a DBMS. This allows the application to exploit all the features provided by a typical DBMS (e.g., standard query language, query optimization, transactional capability, and backup/restore). Modern DBMSs also provide efficient techniques for handling BLOBs (e.g., segmentation, disk striping, separation of non-BLOB and BLOB data, and log/nolog options for BLOB data). However, if the data objects already exist in the form of files, it will be necessary to perform a one-time loading of this data into BLOBs in a database.

4.2.3. Use of files

Each data item may be stored simply as a separate file. Applications would then utilize the standard file I/O APIs to access these objects. However, unlike the BLOB case above, current filesystem generally do not support transaction capability nor do they provide backup/restore and recoverability capabilities.

4.2.4. Use of Hierarchical Storage Management (HSM)

The use of HSM systems can be considered in cases where a data collection contains a very large number of objects, or each individual data object is very large in size, or where a large fraction of a collection is not accessed (or, is accessed rarely). The HSM can employ high capacity media (e.g., magnetic tapes) to record this information. Systems such as HPSS [13], which support parallel I/O to tapes, would be able to handle very large object sizes. In addition, infrequently accessed data would migrate from disk to tape. For very large collections, HSMs also provide large storage capacities (e.g., ranging up to 100s of terabytes to a few petabytes).

From the above discussion, it is clear that a variety of options are available for storage of data. To simplify the implementation, we need a common interface to this variety of storage systems. A middleware system that provides such an interface is the SDSC Storage Resource Broker (SRB) [14]. This system provides a common API for storing/retrieving data from a variety of distributed, heterogeneous storage systems including filesystems, database system, and archival storage systems. Thus, the information model can be implemented on top of the SRB middleware.

4.3. Linking a data object with its metadata

Depending on the mechanism used for storing data objects, there are a variety of options for linking the data object with its related metadata. For example, if the data object is stored as a BLOB, then the metadata can be stored as columns in a database table with the data object being a 'BLOB column' in the same table. This provides a tight linkage between the metadata and the data, since both reside in the same row of a database table.

If the data object is implemented as a file, then there is a need to link the corresponding row(s) in the metadata database to the external file in a filesystem. One approach to providing this linkage is using the concept of Datalinks [15]. Datalinks provides the capability to implement a 'tight' link between the DBMS and the filesystem so that the data object and its corresponding metadata are always synchronized.

If the data objects are stored using an HSM, there are two options to consider. First, if the objects are relatively small in size (e.g., 10s to 100s of Kbytes)and second, there are also millions of objects, then the object can be stored as BLOBs in a DBMS and the DBMS can manage migration of data to a HSM. The advantage of this approach is that the DBMS can pack objects into a few large database 'containers' which

are stored in the HSM. Thus, the HSM is not burdened with managing millions of objects. In a typical DBMS, a container can be up to 2GB in size. Thus, a single container can store about two million objects of, say, 100KB each. The interactions between the DBMS and HSM are transparent to the user. This approach is being explored in a joint project between SDSC and IBM [16,17] and has been demonstrated in a project at SDSC [18].

If the data objects themselves are large and cannot be stored as BLOBs in a DBMS, then they may be stored directly as files in the HSM. In this case, the Datalinks approach, mentioned above, could be extended to link metadata stored in a DBMS to objects stored in a HSM.

5. Future work

San Diego Supercomputer Center and its NPACI partners have a large community of scientific researchers from diverse disciplines like molecular science, neuroscience, and earth systems science who generate large amounts of observational and simulation data and metadata. Our work has been motivated by the need to create a common infrastructure for sharing this data among cooperating groups. In developing this information model, our goal has been to address the general needs for information sharing, without bias to a specific scientific discipline. We currently have several building blocks to implement the model including storage broker middleware, distributed metadata service, automatic metadata extraction from multimedia data objects, and an XML based query language with view-definition capability. Our primary future work is to integrate these blocks into a complete application in an area such as molecular science.

References

[1] R. Williams (Ed.), Interfaces to Scientific Data Archives, Workshop Report, California Institute of Technology, Pasadena, 1998.

[2] P. Buneman, Semistructured data, in: Proc. 16th ACM SIGACT–SIGMOD–SIGART Symp. on Principles of Database Systems, Tucson, Arizona, 12–14 May 1997, ACM, New York, 1997, pp. 117–121.

[3] D. Quass, A. Rajaraman, Y. Sagiv, J.D. Ullman, J. Widom, Querying semistructured heterogeneous information, in: Proc. 4th Int. Conf. DOOD '95 on Deductive and Object-Oriented Databases, Singapore, 4–7 December 1995.

[4] P. Atzeni, G. Mecca, P. Merialdo, Semistructured und structured data in the web: going back and forth, SIGMOD Record 26(4) (1997) 16–23.

[5] Extensible Markup Language (XML) 1.0, http://www.w3.org/TR/REC-xml

[6] J. McHugh, S. Abiteboul, R. Goldman, D. Quass, J. Widom, Lore: a database management system for semistructured data, SIGMOD Record 26(3) (1997) 54–66.

[7] S. Abiteboul, D. Quass, J. McHugh, J. Widom, J.L. Wiener, The Lorel Query Language for semistructured data, Int. J. Digital Libraries 1(1) (1997) 68–88.

[8] J. Paredaens, P. Peelman, L. Tanca, G-Log: a graph-based query language, IEEE Trans. Knowledge and Data Eng. 7(3) (1995) 436–453 .

[9] M. Levene, G. Loizou, A graph-based data model and its ramifications, IEEE Trans. Knowledge and Data Eng. 7(5) (1995) 809–823.

[10] R. Weiss, A. Duda, D.K. Gifford, Composition and search with a video algebra, IEEE Multimedia 2(1) (1995) 12–25.

[11] URL: http://www.informix.com/informix/products/options/udo/datablade/dbmodule/informix2.htm

[12] W.F. Cody, L.M. Haas, W. Niblack, M. Arya, M.J. Carey, R. Fagin, D. Lee, D. Petkovic, P.M. Schwarz, J. Thomas, M. Tork Roth, J.H. Williams, E.L. Wimmers, Querying multimedia data from multiple repositories by content: the Garlic project, in: IFIP 2.6 3rd Working Conf. on Visual Database Systems (VDB-3), Lausanne, Switzerland, March 1995. http://www.almaden.ibm.com/cs/garlic/garlic_vdb95.html

[13] URL: http://www.sdsc.edu/HPSS

[14] C. Baru, R. Moore, A. Rajasekar, M. Wan, The SDSC storage resource broker, in: Proc. CASCON '98, Toronto, Canada, 30 November–3 December 1998, submitted for publication. http://www.npaci.edu/DICE/Pubs/srb.ps

[15] J. Davis, Datalinks: Managing External Data with DB2 Universal Database, http://www.software.ibm.com/data/pubs/papers/datalink.html

[16] The Massive Data Analysis Systems project, http://www.sdsc.edu/MDAS

[17] M. Lo, S. Padmanabhan, C.K. Baru, Integrating DB2 and HPSS, DICE group, San Diego Supercomputer Center, La Jolla, CA, in preparation.

[18] URL: http://www.sdsc.edu/DOCT

Dr. Chaitanya Baru received a B.Tech. in Electronics Engineering from the Indian Institute of Technology, Madras in 1979, and an M.E. and Ph.D. in Electrical Engineering from the University Of Florida in 1983 and 1985, respectively. He is the Senior Principal Scientist at the San Diego Supercomputer Center and Technical Lead of the Data Intensive Computing Environments group. Prior to joining SDSC in July 1996, he was with the IBM Toronto Labs and was a member of the Database Technology Institute (DBTI) at IBM Almaden, where he was one of the team leaders of DB2 Parallel Edition Version 1.0.

Dr. Amarnath Gupta received a B.Tech. in Mechanical Engineering from the Indian Institute of Technology, Kharagpur in 1984, and an MS degree in Biomedical Engineering from University of Texas at Arlington in 1987 and PhD (Engineering) in Computer Science from Jadavpur University, India in 1994. He is an Assistant Research Scientist at the Center for Computational Science and Engineering, UCSD. Before joining UCSD, he was a scientist at Virage, Inc., where he worked on multimedia information systems. Dr. Gupta's current research interests are in multimedia and spatiotemporal information systems, heterogeneous information integration and scientific databases.