

**Panel: Can We Meaningfully
Integrate Drawing, Text, Image
and Voice with Structured Data?**

Chair: Amit P. Sheth, *UNISYS
Systems Development Group*

James Larson, *Honeywell*

Willis Luther, *MCC*

Brian Phillips, *Tektronix*

Robert A. Yost, *IBM Almaden Research Center*

Managing and Integrating Unstructured and Structured Data: Problems of Representation, Features, and Abstraction

A Position Paper

Amit Sheth
UNISYS Systems Development Group

The majority of work in database management has been on developing DBMSs for traditional business applications, such as airline reservations and banking, that use well formatted or structured data. Many nontraditional applications require management of unstructured data such as drawings, images, text and voice. Examples of nontraditional applications are office information systems, factory information systems (including CAD/CAM), scientific information systems, library automation environments, and geographic databases. It has been widely recognized that the existing commercial DBMSs are not well suited for efficient storage and manipulation of unstructured data. They also do not provide all the features and functions required for the nontraditional applications. Integration of unstructured data is required for sharing different types of data among different applications in factories and offices. Efforts pursuing computer integrated manufacturing and multimedia office systems find this particularly necessary.

In addition, integration of structured and unstructured data can also help in efficient processing of unstructured data. Unstructured data can be managed efficiently by abstracting some knowledge from the data, storing it in a structured form, and then processing the subset of data stored in the structured form.

There is a good amount of R & D activity in constructing extended DBMSs which allow efficient representation of unstructured data and provide various functions required for nontraditional applications (e.g., [Christodoulakis et al 84] [Stonebraker and Rowe 85], etc.). However, relatively little effort has been spent in evolving systems that integrate existing traditional DBMSs with applications using unstructured data and the systems used for such data.

Three important issues need to be addressed when integrating and managing structured and unstructured data: representation, features, and abstraction.

1. **REPRESENTATION:** Popular data models such as network and relational models have been shown to be inadequate for representing unstructured data. They are inadequate in representing complex objects with explicit hierarchy or other relationships. Many applications require flexibility in the choice of representation. Existing DBMSs also lack the user interfaces required to display the different types of unstructured data. More recently, researchers and developers have sought to extend the relational model [Lorie and Plouffe 83, Stonebraker and Rowe 85, Schwarz et al 86] or to use an object model [Dayal et al 85, Woelk et al 86] for representation of unstructured data. I agree with the opinion of some that extensions to existing DBMSs to handle different data types and applications may result in complex and inefficient systems [Christodoulakis 85]. However, the debate to decide which approach is better is likely to remain inconclusive until the performance of different types of systems can be compared. The object model has found tremendous popularity, but there does not seem to be one

standard definition of an object that is suitable for all applications. Researchers have sought to amend or extend the objects found in a programming environment [Luther et al 87]. An important question is whether there is one data model that is best suited for all applications and representation of different data types. Furthermore, there has been very little discussion on what to do when integrating multiple preexisting systems specializing in supporting different applications and data types. Use of different conceptual/cannonical models (e.g., E-R model) have been investigated for integrating systems with traditional structured data types and models, but not for systems with both structured and unstructured data types.

2. **FEATURES:** The commercially available DBMSs have been found to be lacking in many features required for applications using unstructured data. For example, an engineering design application requires features such as version control, configuration control, release control, support for complex objects, abstract data types, long fields, long transactions, and many others [Balza et al 83, Katz 85, Kim et al 87]. Other sets of features are required for managing image, voice, text, etc. Requirements for some features in systems that manage unstructured data are different from systems that manage only structured data (e.g., transaction management [Kim et al 85]). An important question to discuss is whether there is a basic set of features that should be provided by a system that integrates or manages different data types. For example, the transitive closure operation, a feature missing in traditional DBMSs, is shown to be very important in many nontraditional applications.
3. **ABSTRACTION:** Many nontraditional applications require very large repositories of data. Certain information may be extracted from these repositories, and possibly transformed and correlated [Christodoulakis 85]. This extracted information may be more structured and hence stored and manipulated differently. The problem of automatically extracting information is very important for efficiently managing unstructured data. It is also a very difficult problem because it depends on the data type and often on the data semantics (i.e., its application dependent use). Examples of abstraction processes are order preserving transformations for text and automatic edge detections and segmentation processes for images [Christodoulakis et al 84, Benn and Radig 84]. A knowledge-based approach for capturing and using semantic information has a good potential. For example, extraction and use of semantics associated with hierarchical structure found in CAD/CAM databases has been discussed in [Rosenthal et al 84]. More work needs to be done to understand what information to extract and how to store and process such information. I feel that the problem of abstraction is extremely important, but has received the least attention as compared to the problems of representation and features.

In this panel, we will address application as well as system aspects of the panel topic by discussing the following issues:

1. When do we need to integrate unstructured and structured data? What is a meaningful integration of unstructured and structured data?
2. What information can be extracted from very large repositories of unstructured data for use by other systems and to support efficient management of unstructured data? How much of the extraction can be done automatically? What is the role of traditional structured data management systems? What features and functions are required to integrate different data types and to efficiently manage unstructured data? What are the user interface requirements? Which of the representation, features and abstraction requirements can be met today? If not now, when will it be possible to meet them?
3. Is the object model a panacea? How much of the problem of meaningful integration is solved by the object-oriented approach? What properties should an object management system have for this purpose?

References:

- [Benn and Radig 84] W. Benn and B. Radig, "Retrieval of Relational Structures for Image Sequence Analysis," *Proc. of the Tenth Intl. Conf. on Very Large Data Bases*, August, 1984.
- [Christodoulakis et al 84] S. Christodoulakis, J. Vanderbroek, J. Li, T. Li, S. Wan, Y. Wang, M. Papa, and E. Bertino, "Development of a Multimedia Information System for an Office Environment," *Proc. of the Tenth Intl. Conf. on Very Large Data Bases*, August, 1984.
- [Christodoulakis 85] S. Christodoulakis, "Multimedia Data Base Management: Applications and Problems (position paper)," *Proc. of the ACM SIGMOD Conf. on the Management of Data*, 1985.
- [Dayal et al 85] U. Dayal, A. Buchmann, D. Goldhirsch, S. Heiler, F. Manola, J. Orenstein, and A. Rosenthal, "PROBE- A Research Project in Knowledge-Oriented Database Systems: Preliminary Analysis," Computer Corporation of America Technical Report CCA-85-03, July, 1985.
- [Katz 85] R.H. Katz, *Information Management for Engineering Design*, Springer-Verlag, 1985.
- [Kim et al 85] W. Kim, R. Lorie, D. McNabb, and W. Plouffe, "A Transaction Mechanism for Engineering Design Databases," *Proc. of the Tenth Intl. Conf. on Very Large Data Bases*, August, 1984.
- [Kim et al 87] W. Kim, D. Woelk, J. Garza, H. Chou, J. Banerjee, and N. Ballou, "Enhancing the Object-Oriented Concepts for Database Support," *Proc. of the Third Intl. Conf. on Data Engineering*, February, 1987.
- [Lorie and Plouffe 83] R. Lorie and W. Plouffe, "Relational Databases for Engineering Data," *Proc. of the Second Intl. Conf. on Foundations of Computer Aided Process Design*, June, 1983.
- [Rosenthal et al 84] A. Rosenthal, S. Heiler, and F. Manola, "An Example of Knowledge-Based Query Processing in a CAD-CAM DBMS," *Proc. of the Tenth Intl. Conf. on Very Large Data Bases*, August, 1984.
- [Schwarz et al 86] P. Schwarz, W. Chang, J.C. Freytag, G. Lohman, J. McPherson, C. Mohan, and H. Pirahesh, "Extensibility in the Starburst Database Systems," *Proc. 1986 International Workshop on Object-Oriented Database Systems*, September, 1986.
- [Stonebraker and Rowe 85] M. Stonebraker and L. Rowe, "The Design of Postgres," Memo. No. UCB/ERL 85/95, Electronics Research Lab., Univ. of California, November, 1985.
- [Woelk et al 86] D. Woelk, W. Kim and W. Luther, "An Object Oriented Approach to Multimedia Databases," *Proc. ACM SIGMOD Conf. on the Management of Data*, May, 1986.

Browsing Unstructured Data

James A. Larson
Honeywell, Inc.

Computers have demonstrated that structured data such as tables and can be stored, accessed, and manipulated accurately and efficiently by computers. Unfortunately, structured data is but a small part of information needed by database users. Unstructured data in the form of text, drawings, images, and voice recordings represent a significant base of information needed to make decisions and control a wide variety of processes.

Structured and unstructured data needs to be integrated in order to be useful. For example, an analyst doing research on the environmental impact of a certain chemical will need to retrieve textual data describing the effects of the chemical, tabular data containing results of experiments involving the chemical, pictorial data showing the effects of the chemical in various environments, and perhaps a verbal recording of several experts discussing the chemical. The analyst will need to display data objects in some convenient arrangement in order to correlate the data and extract the precise information needed.

Currently humans are the largest class of unstructured data users. Many issues need to be resolved about how unstructured data can be presented in an integrated manner for human use. Not until sophisticated knowledge extraction techniques are economically viable will non-human users be able to effectively use unstructured data.

A visual approach to browsing heterogeneous databases is one way for users to integrate information from the various databases. Users perform four basic operations when browsing: structuring, choosing a structure of the objects to be examined; filtering, selecting instances of the objects to be examined; panning, examining neighboring object instances; and zooming, determining the level of detail for examining object instances.

Structuring: The structure of data objects (how data objects logically relate to each other) can be determined by human experts using knowledge acquisition techniques.

Human experts may establish the structure of data objects. Scientists create taxonomies describing relationships of the subjects or phenomenon they research; librarians create cataloguing schemes for organizing books on library shelves; and database designers create schemas which describe how database records may relate to each other.

However, without training humans generally do a mediocre job of extracting semantics, building cross references, and establishing the structure for diverse data objects. Knowledge acquisition techniques are needed to help humans establish the structure of objects. Expert systems are needed extract knowledge from text and images.

Clustering algorithms are useful for establishing taxonomies. Image analysis and text understanding systems promise to extract and generate subject categories and keywords descriptive of images and text. Automated classification techniques can be used to identify subject classes to which a new data object belongs. Unfortunately, different expert systems will be needed to extract knowledge from each type of data for each subject domain.

The structure of objects can be defined either before their display or during their display. The organization and taxonomy inherent in most textbooks can be used to structure objects before the objects are displayed to the user. Knowledge acquisition and representation techniques are needed by knowledge engineers to define appropriate taxonomies and structures.

Windowing systems are useful for simultaneously displaying of different types of data. Users are able to visually correlate different types of data in a manner convenient to their immediate needs and discover new relationships among displayed objects, and record those relationships for future use.

Filtering: The time and effort expended searching can be dramatically decreased by restricting the search space to a characterization of the class of relevant data objects. Query languages can be used to project and restrict irrelevant data objects leaving only potentially interesting data objects for browsing.

Panning: Often the number of data objects, even in a filtered search space, is too large to display on a terminal screen. Panning techniques, including paging and both horizontal and vertical scrolling, permit users to scan large numbers of data objects.

Zooming: Users navigate through a structure of data objects by zooming. Zooming can be used to change the level of abstraction of data objects being displayed. "Zooming in" causes more detail to be displayed; "zooming out" causes less detail to be displayed. Opening and closing windows is a type of zooming in which a highly abstract object (window name) and a more concrete object (contents of the opened window) are interchanged. Zooming can also be used to change context: users can zoom to a related data object or class of data objects. Zooming is the user interface equivalent of the programming language command "goto."

Object-oriented approaches appear promising as an approach for representing and manipulating unstructured data types. However, by themselves, object-oriented approaches do not solve the problems of extracting and relating knowledge from unstructured data types and presenting data from unstructured data types in a manner useful to human users of that data.

Multimedia Systems and Text

Brian Phillips

Teklabs, Tektronix, Inc., Oregon

Multimedia Systems

Current technology supports flexible multimedia interactions. Text and still and animated pictures can be viewed and created on bit-mapped displays. Signal processing hardware provides voice access. Hyperdata systems such as Apple's HyperCard give freedom in accessing multimedia data.

Advantages of Multimedia Systems

Mixing media has benefits – although it can be used just to prettify an application.

Electronic books parallel the text and pictures of the printed form. Animation enhances understanding an algorithm. Voice annotation of documents frees the annotator from the keyboard and keeps the comments separate from the draft.

An alternative presentation extends the power of a user. A net-list contains the same information as a diagram of a circuit. An on-line diagram, however, allows the user to transcend reality. The values of components can be directly manipulated by moving a "slider" for example. The physical system is much harder to reconfigure.

An architect could give a client the drawings of a building. It would be more informative to have a "video-tour" of the structure in the manner of the MIT Media Lab construction of a tour of Aspen.

Each medium has its advantages and these need to be understood. Environmental, as well as technical conditions must be considered. Electronic mail may be thought equivalent to telephoning with the advantage of avoiding telephone tag but there are socially conditioned situations where email is unacceptable.

Integrating the Media

A key issue in utilizing multimedia is the integration of the various forms. Changes in a circuit diagram should update the net-list and vice-versa. The client's modification from the pictorial presentation of the building should modify the architectural drawings. As humans, we have this integrative capability. As we read instruction manuals, we can link text to figures and diagrams. Such meaningful integration is a basis for maintaining data integrity and transformation among the media.

An alternative to maintaining several representations is to have a single underlying description from which the others can be derived. Transmittal of data shows a disadvantage of retaining media-specific structures. A bit map of a circle may not achieve the intended purpose due to different aspect-ratios on the destination display. If the concept of a circle is transmitted; the recipient then can display it under the local conditions. Moreover, if the recipient did not have graphical display capabilities, other means for presenting the information could be used – verbal output if a telephone were the device.

If the goal is to have canonical representations then automatic means should be developed to build it. Efforts in automatic natural language processing contribute to this goal.

Textual Analysis

The many efforts to build universal meaning representations of the content of text are limited in scale. Structural analyses at the level of a book or a manual are not practical. Large volume processing is by keywords which does not have the granularity for meaningful integration.

A design for progress will be a hybrid system. Cheap processes, keyword analysis and script-based skimming, establish contexts and structural analyses operate within the domains so identified.

Object Databases for Integration

We have begun to build a "project encyclopedia" using the object metaphor. A semantic network is being built as the conceptual framework for data. Text and graphic material is stored but not structurally analyzed. The encyclopedia is intended to provide an environment for experimenting with large volume text processing and its integration to other media.

Developments in object-oriented databases will enhance the system. With permanent objects, the semantic network will be able to take advantage of object communication at the program level. The Smalltalk notion of multiple views of the same model is compatible with the design of a single canonical representation.

Can image data be integrated with structured data?

Robert A. Yost

IBM Almaden Research Center, San Jose, California 95120

The first page of an 8.5 by 11 inch employment application form is fed into an optical scanner. Over five million points on the surface of the document are sampled. Since this is a simple black and white document, each sample point produces either a "1" or "0", depending on its relative darkness. Over five million bits (about 650K bytes) of data pour through compression hardware, producing 200K bytes of "compressed" data. As this data is routed through a nearby computer to some direct access storage device, the next page is scanned, and the process continues.

Image data always has some structured data too

The digital data produced by scanning a document, such as this employment application form, is represented as a bag-of-bits that can be used to reproduce an image of the original document on either a printer or a bit-mapped display. However, without some associated structured data (e.g., character strings, integers, floating point numbers, etc.), this bag-of-bits is useless. Some structured data would normally be captured as the document is scanned (such as the resolution of the scanned image, the compression encoding technique, the dimensions of the original document, the date and time the document was scanned, the identification of the scanner, the number of bytes in the bag-of-bits, etc.). Other structured data might be derived from the content of the image itself. Such data might be captured manually by viewing the document and entering data therefrom, or it might be captured automatically by invoking special pattern matching programs (e.g., optical character recognition functions) to extract the relevant information. So any data management system designed to handle image data must also handle a certain amount of descriptive, structured data as well.

To integrate or not to integrate

An interesting question, then, is whether to integrate the structured data with the unstructured, image data. Should a single data manager, say an SQL relational database management system (DBMS), be used to access both image and related structured data? Suppose that we are writing an application program which makes use of an SQL DBMS to implement a Job Applicant Information System. This system contains information about all job applicants, including the image data resulting from scanning the original application forms. In good relational style, we decide that each row (tuple) in a table (relation) called APPS should contain facts (attributes) about an individual applicant (e.g., Name, date of birth, sex...). Given this model, it makes sense to logically place the bag-of-bits representing the scanned application form in the same row as all the other facts about the applicant. In this way the image data is logically related to all the other facts about the applicant, and all those facts can be used to relate to other facts in the database, using the normal relational commands as defined by SQL.

On the other hand, image data is so different from structured data that maybe it should be handled by special-purpose image data management systems in order to get good performance. Image data is different for many reasons, including (1) the possible need to avoid copying the image in order to reduce CPU and real main

storage consumption, (2) the fact that application programs may not need to own the value of the bag-of-bits associated with an image, (3) the recovery protocols, (4) the space management policies on secondary storage, and (5) special clustering requirements to keep related images close to one another on magnetic or optical media. With this line of reasoning, one can quickly become convinced that standard data management systems are not cost effective when compared with special-purpose image data managers. However, this performance advantage increases the complexity of application programs that use image data. For instance, when writing the Job Applicant Information System, we must deal with two data managers - SQL for the structured data and a new image programming interface for the image data. Our application program first inserts images into the image database, and then inserts descriptors for these images into the rows of the APPS table in the SQL database. The application program is now responsible for this association, because neither the SQL DBMS nor the image data manager are aware of one another.

Note: We assume that there is a powerful transaction management component that can handle a related set of changes against the two independent data managers in an atomic fashion.

By separating the two data managers, we have moved some of the responsibility for data integrity from the data managers to the application program. If the application deletes a row from the APPS table, it will have to delete the images identified in the image descriptors contained in that row. If it fails to do so, there will be some "dangling images" left in the image database. Likewise, if the application program drops the entire APPS table, it will have to find and drop all images described in all rows contained therein. There are other unfortunate consequences of this separated approach that have to do with the data definition function, integrity constraints, authorization, and distribution.

So, what is the answer?

One solution is to have a separate image management component that deals well with the special performance-related aspects of image data, but to only expose these functions through the interface presented by the structured data manager. We intend to take this approach in the Starburst (experimental SQL DBMS) project [SCHW86]. Image data will appear to be logically in the relation model, but in the underlying physical representation, image data will be managed by a special-purpose data manager. In this way we expect to get the advantages of integration and good performance.

Bibliography

- [SCHW86] P. Schwarz, W. Chang, J.C. Freytag, G. Lohman, J. McPherson, C. Mohan, and H. Pirahesh, Extensibility in the Starburst Database System, *Proceedings 1986 International Workshop on Object-Oriented Database Systems*, Pacific Grove, CA (September 1986) pp. 85-92.

Can We Meaningfully Integrate Drawings, Text, Images, and Voice
with Structured Data?

Jack A. Orenstein
Computer Corporation of America
4 Cambridge Center, Cambridge MA 02142
jack@cca.cca.com

The answer depends on who you ask.

**Designer of Data Models and Query Languages: YES,
(but ...)**

Attempts to implement "non-traditional" applications on top of existing database systems have not been entirely successful. Schemas for the representation of complex datatypes (drawings, text, etc.) are usually easy to develop, but it is very difficult to implement even the most trivial operations over these representations using only a query language. On the other hand, database systems extended with application-specific datatypes and operations are very narrowly focussed.

Recent work has combined ideas from object-oriented programming languages with ideas from database systems. The idea behind *object-oriented database systems* is to add *extensibility* to a database system, not an application-specific extension. An object-oriented database system has an open-ended data model. An application-specific extension can be incorporated in the form of an object class. An *object class* provides a datatype and functions that can be applied to instances of that datatype. (e.g. polygons and operations on polygons). The database stores instances of the datatype and queries can use the operators of the query language in conjunction with functions of the object class. With object-oriented data models, alphanumeric data and "unformatted" data can be integrated. Queries can freely mix references to the two kinds of data.

So here's why the answer is "yes": Any property or function of a drawing, text, image, or voice sample, can be related to other objects in arbitrarily complex ways. If "meaningful" properties or functions are supplied with an object class, then "meaningful" integration is possible. The "(but ...)" represents the fact that object-oriented database system research is very new, and that there hasn't much real-world testing of the data models.

Implementer of a Database System: SORT OF

Common to all work on object-oriented database systems is some form of extensibility at the physical level. There are two basic requirements here. First, there is a need to be able to store unformatted data of arbitrary length. Second, there is a need to incorporate subprograms of user-supplied object classes. These facilities are related: an instance of an object class, e.g. a polygon, can be stored in the database as a data structure in some host language. The code of the object class provides the means by which the data structure is manipulated. Several commercially available database systems have a "long field" type. Work on the efficient storage and retrieval of long fields has been carried out in the EXODUS project.

Specialized access methods are needed to support efficient retrieval for non-traditional datatypes. Access methods for specific domains have been developed, e.g. signature files for document retrieval, and quadtrees for image data. Most object-oriented database systems provide a way to incorporate these access methods. EXODUS offers language support for the construction of new access methods. However, there has been little work on more general-purpose access methods that span application areas. Our own work on the PROBE project is one attempt in this direction, providing a set of algorithms for spatial data that are independent of dimension and representation.

One of the most interesting implementation problems is query optimization for object-oriented database systems. The usual rules of thumb are no longer adequate. For example, many query optimizers assume that selection with the equality predicate is highly selective. No such assumptions can be made about user-supplied predicates, so one approach is to provide a way to describe such hints. Users also have to supply rules that indicate valid transformations of the functions provided with the object classes.

The answer here is "sort of" because some important implementation problems have been perceived, and because research on solutions has started. The answer will be more positive when we're further along with the solutions.

Domain Expert: NO

A *domain expert* is responsible for implementing an application-specific object class. While a data model designer is concerned with functions and properties of objects in the abstract, a domain expert is concerned with very specific functions and properties of a certain type of object. In many fields, the state of the art does not include the ability to extract much "meaning" from raw data, so many domain experts would answer the question of the title in the negative. For example, there isn't much processing of texts that is meaningful: there are techniques for extracting keywords, and formatting, but extraction of meaning is very difficult. The answer here is "no" until more meaning can be extracted from the raw data.