Database Management for Data Visualization

Peter Kochevar
DEC/San Diego Supercomputer Center
La Jolla, CA

Abstract

Visualization management systems which integrate database management, data visualization, and graphical user-interface generation into one package are becoming essential tools for conducting science. Unfortunately, the link between data management and data visualization is not very well understood. To make matters worse, most database management systems available today are not well-suited for handling large, time-sequenced data sets that are common to many scientific disciplines. The reason for this shortfall is that most database systems do not use an appropriate data model, are not geared toward real-time operation, and they have inadequate user-interfaces. Hints as to how database systems can fix these problems are given so that effective visualization management systems can be constructed.

1 Introduction

Database management systems (DBMSs) are essential for doing research in fields which require the handling of large amounts of data. No where is this statement more true than in the Earth sciences where the study of global environmental change places great demands on information systems. To Earth scientists, the search for and ready access to data is as important as is their actual visualization. Frequently, scientists do not know precisely what they are looking for when conducting a study. The browsing of databases on the basis of data set contents is a much desired capability for many scientists and DBMSs, in principle, are particularly well-suited for this task.

Browsing requires close integration with data visualization systems. DBMSs are not very useful if the results of queries cannot be presented to users in an effective manner. Data visualization must address the presentation of all forms of data whether it be "raw" scientific data like satellite imagery, meta-data describing raw data, or a list of data sets that result from a database query.

For effective browsing, a visualization system must do more than just present information visually. Users must be able to interact with screen visuals as a way of customizing visualizations and interfacing with the DBMS. What is needed are *visualization management systems* which integrate DBMSs, data visualization systems, and graphical user-interface builders into one package.

From the standpoint of data management, *there are no good DBMSs for dealing with scientific data of the kind found in the Earth sciences.* The relational model is inadequate as a basis for a DBMS because most of the data that resides in scientific databases is simply not relational in nature. In the Earth sciences, much data is in the form of satellite imagery which does not easily fit the relational model of an unordered set of data elements. To store such data, relational DBMSs usually resort to the use of *binary large objects (blobs)* whose internal structure is completely opaque to a database's access machinery. Even worse is data associated with polyhedra, for example, river basins are often modeled as 3-space graphs in which data is associated with both graph vertices and graph edges. Relational systems are particularly bad at storing topological information required to fully describe such networks.

Object-oriented DBMSs are promising but it is too early to tell if they can serve as good scientific database managers. In general, most current DBMSs are lacking in their data models, in their ability to handle large, time-sequenced data sets at real-time rates, and they are lacking in their means of interacting with users. This paper begins to address some of these issues so that complete visualization management systems can be constructed which effectively and efficiently serve the scientific community.

2 A Data Model

An appropriate data model is an essential part of a DBMS. The fact that relational DBMSs use the notion of the relation as their theoretical underpinning is both their strength and their weakness. The mathematical notion of the relation gives rise to the relational calculus, has implications on storage strategies, and it has bearing on how data is to be visualized. But, the idea of the relation is too ackward a concept to adequately deal with a large variety of scientific data particularly when topological relationships exist between data elements.

For scientific data, the notion of the *fiber bundle* [3, 4, 7] is more appropriate as a basis for a data model than most other theoretical concepts. Fiber bundles contain the notion of the relation as a proper subset yet they are quite rich in their expressibility of varied data structures. The downside is that fiber bundles, with all their generality, are more complex entities than relations thus entailing a more complex set of manipulation operators.

Mathematically, a fiber bundle represents a space which is the Cartesian product of two other spaces; a *base space* and a *fiber space*. Conceptually, a copy of the fiber space is attached to each element of the base space. For each base space element, this copy is referred to as a *fiber*. A *fiber bundle section* or simply a *section* is the identification of one element from each fiber in the bundle. As an example, Figure 1 shows a 3D fiber space attached to a point in the spherical base space of a bundle. A section for this fiber bundle is illustrated in Figure 2 where each element of the section is a 3-vector.

Fiber bundles are a generalization of the concept of graphs of functions. In this sense, a base space of a fiber bundle is analogous to the independent variables of a function while its fiber space corresponds to the function's dependent variables. A fiber bundle section is represented by the function itself whereas the entire fiber bundle is the space where the function would be graphed.

Although it is often convenient to associate independent and dependent variables of functions with base and fiber spaces, the definition of fiber bundles makes no assumptions about the structure of these spaces. Both base and fiber spaces can be arbitrary topological spaces which may or may not be metric spaces, that is, there may or may not be any coordinates defined for them. For instance, a base or fiber space can just be a finite set of elements in which there is no implicit relationship between members of the set. With so little structure, the only meaningful operation on spaces of this type is the formation of subsets from among the elements.

In practice, each element of a fiber bundle section can be thought of as a simple scalar type, a record, or an entire fiber bundle section. Fields in a record can themselves represent a scalar, another record, or a fiber bundle section as well. Note that the *relation* which forms the theoretical underpinning of relational DBMSs is a special case of a fiber bundle. Such a structure has a base space consisting of an unordered set of elements and a fiber space which is the Cartesian

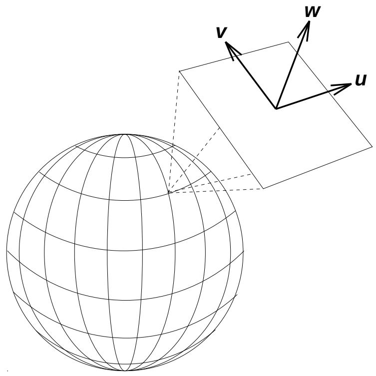

Figure 1: A fiber associated with a point in a base space.

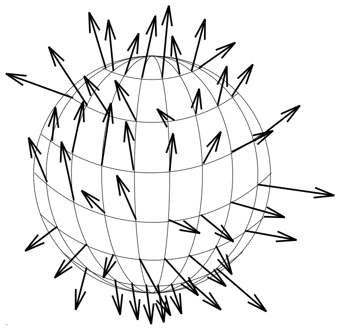

Figure 2: A fiber bundle section.

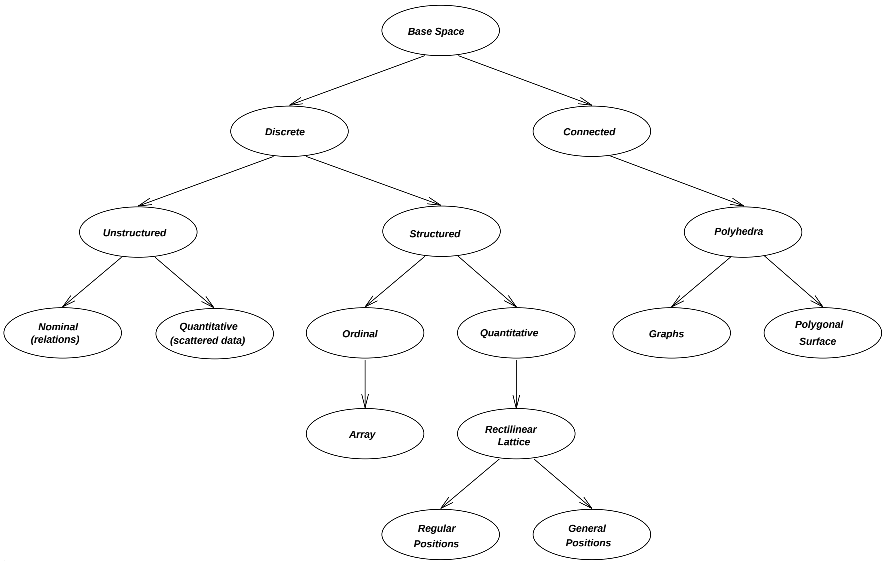

Figure 3: A partial taxonomy of fiber bundle base spaces.

product of the domains of the fields in a record.

A partial taxonomy of fiber bundles can be constructed which is based on the structure of their base spaces (see Figure 3). This taxonomy can be used to fashion an inheritance hierarchy from which an object-oriented implementation of fiber bundles can be made (see Figure 4). Depending on where a data set falls within the taxonomy, only a certain set of visualization techniques can apply. For instance, suppose a data set consists of a base space which is a 3D rectilinear lattice whose axes represent latitude, longitude, and altitude, and whose fibers represent the range of some scalar value. For such data sets, visualization techniques are limited to taking cutting planes, forming iso-surfaces, doing volume rendering, and so on.

3 Real-Time DBMSs

By its very nature, the field of global environmental change relies on and hence needs access to time-sequenced data. The most appropriate format for visualizing

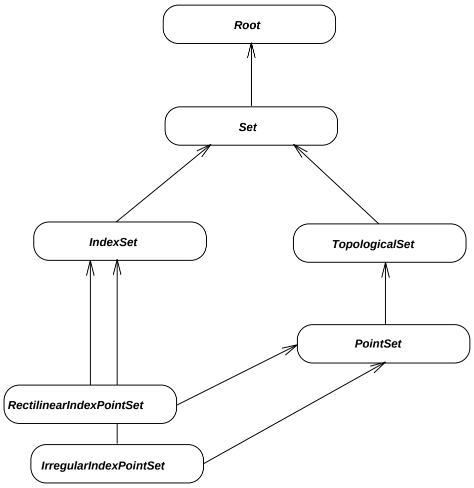

Figure 4: An inheritance graph for an object-oriented implementation of fiber-bundles.

such data is as animations so that environmental changes become readily apparent. To complicate matters, scientists want to visually navigate through the scenes representing the global data as the data is being animated. Therefore, it is not sufficient, in general, to simply replay an animation loop of pre-computed images.

Data comprising each time step of a time-sequenced data set can be very large since their coverage may range over the entire planet. To effect animation, database queries for these times slices must be handled many times per second. Unfortunately, most existing DBMSs are not equipped to process transactions at frame-rates for such large data items.

From a database, data must be sequenced into a set of rendering engines which construct images for eventual display on a monitor. To support the timely access to time-sequenced data, data must be distributed or "striped" across many interconnected storage servers. Nonetheless, these collections of servers must appear to the end-user, as well as applications programs, as a single, unified database. The DBMS must keep track of the whereabouts of the data, the compute servers, and the end-user's display system so as to minimize data traffic across the network.

To further maximize bandwidth, data should only be drawn out of a database at a level-of-detail that is compatible with the resolution of the display on which it is to be viewed. For instance, a 6K by 6K AVHRR satellite image is far too large to view altogether at its full resolution on a typical workstation. But, the entire image can be viewed at once if the viewpoint is pulled far enough back from the image. In this case, an appropriately sub-sampled rendition of the image suffices for its screen representation. As the viewpoint is brought closer to the image, a fuller resolution version of the image would need to be displayed. But to compensate in this case, large portions of the original image could be clipped away by the specification of a viewport commensurate with a display's resolution.

Level-of-detail management of the kind just described should be performed transparently to an end-user both during retrieval and during the storage of data. As a user navigates through a virtual environment representing data, level-of-detail swapping should occur automatically and seamlessly as data objects loom up and recede within a user's field of view [6]. In this regard, a visualization management system acts much like a flight simulator. At the heart of this simulator is the DBMS which must be able to support the access to varied levels-of-detail. To unburden users as much as possible, the DBMS should automatically sub-sample array data and it should automatically decimate polyhedral data when coarser levels-of-detail would suffice during a visualization.

Finally, real-time database support should not be limited to data retrievals since a DBMS must be able to ingest large data sets at real-time rates as well. For instance, sensor data such as satellite imagery is frequently generated in a continuous manner yet it still must be properly stored for future access. Similarly, output from simulation programs running on supercomputers or workstation farms can produce large data sets many times a second. Again, such data should be stored in a database so that the results can be analyzed by researchers at later times.

4 Interacting With a Database

Database interactions should fall under the aegis of visualization management systems which seek to integrate data rendering, data browsing, and data understanding into one package. The primary focus of such systems should be the creation of a methodology which lets non-visualization and non-database experts easily perform their own sophisticated interactive visualizations and complex database interactions. End-users should not have to write visualization programs to render every data set that results from a query. This idea should also hold not only for data sets that are used frequently and are familiar to a user, but also for data sets that are completely unfamiliar to them.

In Kochevar et al [8], a very simple visualization management system called the Visualization Executive is described which links the Postgres DBMS [11] with most any visualization package. The design of the Visualization Executive was intended to be somewhat analogous to that of the Apple Macintosh folder browser. On the Macintosh, a user traverses the folder tree until a document of interest is eventually located. The browsing of the folders proceeds visually by pointing and clicking on iconic representations of folders and documents. When a desired document is located, the user selects a representative icon and the document is then "visualized" within the application which created the document.

In a similar fashion, a user of the Visualization Executive navigates through the database by pointing and clicking on iconic representations of the database's contents. Once a data set of interest is located, a user selects a data set's icon and the corresponding data set is then visualized. The key idea which make this functionality possible is that information as to how to visualize each data set is stored in the database along with the data set itself.

To aid in the navigation of a database's contents, one example of a domain-specific, graphical query tool is provided within the Visualization Executive. The intent of this tool is to allow non-database experts to make complex queries of the database in a way that is more natural and intuitive given the kind of data that is being sought. For example, a frequent query of Earth science databases is for data sets that pertain to a particular region of the planet. Rather than forming a textual query in an arcane database query language, the Visualization Executive provides a world map on which a user can mark regions of interest.

Although far from a production system, the visualization management system that was developed was successful because of the insight that was gained from its design and implementation. Some of the things learned that have bearing on the interplay of visualization and database management include the following:

Visualize all forms of data. The results of general database queries should be visualizable just like any other scientific data. One appropriate visualization technique for data of this kind is the positioning of icons onto an "informational landscape". In this way, the visualization encodes additional information that can be utilized by a scientist. For example, queries for data sets pertaining to regions of the Earth might be positioned above a world map or globe so that a "light cone" highlights the exact area of a data set's coverage. In addition, the visualization of query results must be able to handle the screen clutter which can result when queries generate thousands of items. The hierarchical display of data, such as the case with folders on the Macintosh, is one visualization technique for dealing with the clutter.

Provide for interactivity. Interactivity is important in data visualization as a means for both exploring the information content of a chosen data set and for extending links back into the database. Objects that appear in a visualization can be thought of as "widgets" when various behaviors are assigned to screen visuals. These behaviors might be elicited when pointed at with a picking device via a callback subroutine, say. Such behaviors can trigger additional database interactions and visualizations, thus affording a mechanism with which to do database browsing. As an example, consider a query for all snow gauges in the Sierra Mountains of California which results in a collection of icons representing each gauge displayed on a map. When an icon is pointed at, a history of snow measurements for that site could be displayed as a 2-D graph with time as the independent axis and snow depth as the dependent axis. If, in addition, icons are provided to allow further database queries, a scientist might then request climate data for the same area and period of time in order to visually explore the possible correlation of mean temperature and snow depth.

Eliminate needless programming. Scientists should not be required to write visualization programs either themselves or through visualization experts acting as intermediaries. A scientist should only have to identify what is to be viewed via browsing and then let the visualization management system figure out how to do the visualization. To have this capability, a visualization management system requires built-in intelligence and access to a knowledge base that contains rules and definitions pertaining to data visualization. Simple systems have been developed which address the automatic visualization of certain kinds of data [2, 5, 9, 10] and these systems need to be extended to fully serve general scientific applications.

Account for end-user's tasks. A data set can be visualized in a variety of ways. When scientists visualize data they are making use of the large bandwidth of the human visual system to make a visual computation. Exactly how a scientist wishes to participate in the computation is information that can be used to structure an appropriate visualization for a particular task. At times scientists may wish to select one item from many possibilities, they may want to correlate or compare data in different data sets, they may want to search for patterns or anomalies, and so on. In general, a visualization should make the visual computation as efficient as possible. Having a list of predetermined visualization programs associated with a data set is too limiting in that it may hinder exploration and preclude the data reusability and sharability.

Utilize self-describing databases. Databases, augmented with knowledge bases, need to be self-describing, in other words, the schema design and a complete characterization of all data classes must be stored in the database along with the data. Having this information on-line allows a visualization management system to "learn" about the structure of a database thereby giving it the flexibility to dynamically create graphical interfaces tailored to the information that a user seeks. Information about classes of data and their relationship to one another should go beyond the mere description of syntactic structures. The semantics associated with classes of data is important information that needs to be stored in the data/knowledge base as well. Such information can be used to give a domain-specific look and feel to interactive visualizations.

Aid applications programmers. From a developers standpoint, it is preferable to interact with a DBMS by building persistence into existing programming languages. Within programs written in an extended programming language, designated data objects would continue to exist once the scope of the program ends. Such data objects would be used just like any traditional data object in a program except that their storage resides in a database managed by a DBMS. The ODE system from AT&T is one such DBMS which uses an extension of C++ as its interaction language [1].

5 Conclusion

Computation is an integral part of conducting science today. As a result many scientific disciplines, particularly the Earth sciences, are awash in data that must be managed. Unfortunately, database management systems currently available are inadequate for handling most forms of scientific data. Some of the areas in which database systems are lacking were discussed and hints as to how these shortcomings can be alleviated were offered. In general, those in the field of scientific visualization are among the first to grapple with the problem of managing scientific data. As such, the scientific visualization community must push the database field to satisfy its needs. This paper has been one small nudge in what is hoped is an appropriate direction.

References

[1] R. Agrawal and N. H. Gehani. ODE (Object Database and Environment): The language and the data model. In *Proc. ACM-SIGMOD 1989 Int'l Conf. Management of Data*, May 1989.

[2] Clifford Beshers and Steven Feiner. Automated design of virtual worlds for visualizing multivariate relations. In *Proceedings Visualization '92 Conference*, pages 283–290, 1992.

[3] D. M. Butler and M. H. Pendley. A visualization model based on the mathematics of fiber bundles. *Computers in Physics*, pages 45–51, Sep/Oct 1989.

[4] David M. Butler and Steve Bryson. Vector-bundle classes form powerful tool for scientific visualization. *Computers in Physics*, 6(6):576–584, Nov/Dec 1992.

[5] Stephen M. Casner. A task-analytic approach to the automated design of graphic presentations. *ACM Transactions on Graphics*, 10(2):111–151, April 1991.

[6] Thomas A. Funkhouser et al. Management of large amounts of data in interactive building walkthroughs. In *Proceedings Workshop on 3D Interactive Computer Graphics*, Boston, MA, 1992.

[7] R. B. Haber, B. Lucas, and N. Collins. A data model for scientific visualization with provisions for regular and irregular grids. In *Proceedings Visualization '91 Conference*, 1991.

[8] Peter Kochevar et al. Bridging the gap between visualization and data management: A simple visualization management system. In *Proceedings Visualization '93 Conference*, 1993.

[9] J. D. Mackinlay. Automating the design of graphical presentations of relational information. *ACM Transactions on Graphics*, 5(2):110–141, 1986.

[10] Hikmet Senay and Eve Ignatius. Vista: A knowledge based system for scientific data visualization. Technical Report GWU-IIST-92-10, George Washington University, March 1992.

[11] Michael Stonebraker et al. The implementation of POSTGRES. *IEEE transactions on knowledge and data engineering*, March 1990.