

An Integrated Approach to Database Visualization

Dennis P. Groth Edward L. Robertson *

Computer Science, Indiana University,
Bloomington, IN 47405, USA.
Email: {dgroth, edrbtsn}@cs.indiana.edu

Abstract

We present an architecture that enables information visualization activities within a database environment. Our approach presents an abstraction of this transformation process, which we call mapping. The implementation of the mapping process is controlled by the end-user through a *Map*, which can be used to add order and scale to data.

1. INTRODUCTION

Supporting visualization activities within a database environment can focus on a variety of areas, ranging from query formulation to graphical representation. Database systems, architected to support storage, management and extraction of data, do not provide visualization methods. Extraction of data is specified by a query (typically SQL), with the results formatted in tabular fashion.

While a great deal of research has occurred in both scientific and information visualization, the main focus continues to be on the requirements of specific applications. Of course, there are a variety of general systems for visualizing data, as long as the data has the proper format. What is often hidden in most applications is the process of transforming the data into a form that can be visualized. The idea that data may need to be transformed is outlined in [2, pages 17–23]. A more specific review of the problem is described by Card and Mackinlay in [1]. Taxonomic approaches to describing data transformations is presented by Shneiderman in [5] and Chi in [3].

This research is expressly focused on the process of

*Both authors were supported by NSF Grant IIS-0082407

preparing and transforming data for the purposes of visualization. We present an abstraction of the transformation process, which we call a *Map*. By leveraging the strengths of database systems, we demonstrate that the process can be codified in way that provides control to the end user of the visualization system. Figure 1 depicts our proposed architecture supporting database visualization.[4]

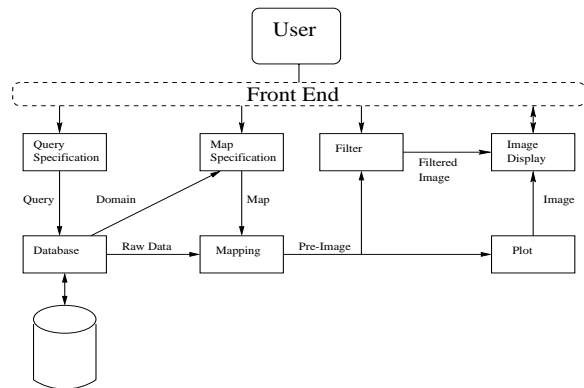


Figure 1: The architecture supporting visualization of database information.

The architecture supports specification of queries for extracting data, as well as the specification of the transformations necessary for visualizing the data. The key element of the architecture is the mapping component, which enables the user to define and implement the appropriate data transformations for their application. The plotting component constructs the graphical representation of the data. Filtering allows the user to select portions of the graphical display to be used in later visualizations. Natural extensions to the architecture to support new applications can be implemented via our approach.

2. SYSTEM IMPLEMENTATION

In this section we provide details on an implementation based on the previously described architecture. The system is written entirely in Java 1.3. Database access is provided via the JDBC API. The visualizations are generated using the Java 3D API. A client application pro-

vides access to the specification of data, queries, maps and visualizations. A separate, server component provides access to the mapping process, off-loading more costly data transformation processes to improve performance.

Users interact with the system through a graphical user interface that is designed using a desktop metaphor. Data, maps and visualizations are organized within an *Application* object. Using the application manager interface, a user adds various system objects to the desktop. A screen capture from the system is shown in Figure 2.

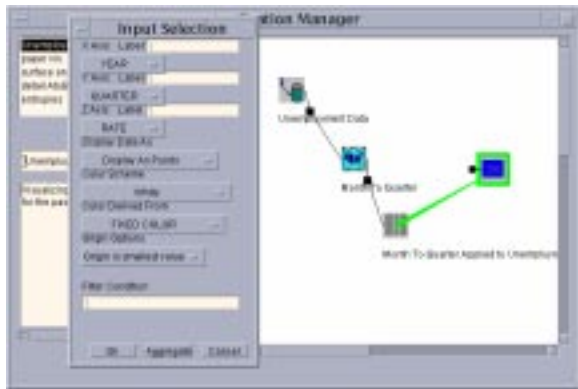


Figure 2: The application manager display. Objects are placed on the desktop and linked together to perform the appropriate transformations.

Feedback is provided to the user during this process by coloring the connecting line from the query to the map. If two application objects are connectable, the color of the line is green, otherwise it is red and the user cannot connect the objects. When the objects are connected the server component applies the map the input data. The desktop is updated to show the result of a mapping operation. In this case, a new object is added to the desktop that represents the input data transformed according to the map. The visualization shown in Figure 3 was constructed by mapping monthly unemployment data to quarters, thereby simplifying the display. The system supports a variety of output formats, including 1D, 2D and 3D forms.

The visualizations that we create allow for interactive manipulation by the user. Standard features, such as zooming, translation and rotation of the display are supported. The user can drill down into the visualization, using the mouse to display the underlying data values. Similar to brushing, portions of the visualization can be selected with the mouse. The selected points can be saved to the application desktop for further use. For example, the selected points can be viewed either in a different context, or in a different display type in seamless fashion.

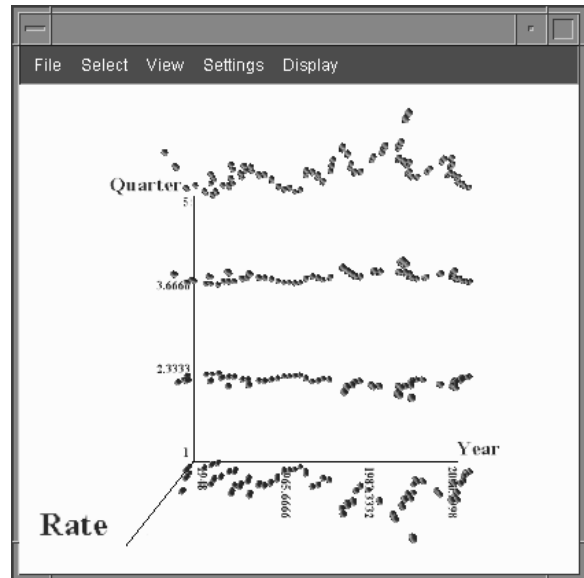


Figure 3: An example visualization generated through the system. The display shows U.S. unemployment data for the last 50 years.

3. CONCLUSION

We have presented an architecture and a system designed to support database visualization. Beyond just extracting data from a database and displaying it graphically, the architecture supports pre-visualization transformations of the data by leveraging existing database techniques, such as joins.

4. REFERENCES

- [1] CARD, S., AND MACKINLAY, J. The structure of the information visualization design space. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis '97)* (1997), pp. 92–99.
- [2] CARD, S. K., MACKINLAY, J. D., AND SHNEIDERMAN, B., Eds. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann Publishers, Inc., 1999.
- [3] CHI, E. A taxonomy of visualization techniques using the data state reference model. In *Proceedings of InfoVis 2000 (Salt Lake City UT, October 2000)* (2000), pp. 69–75.
- [4] GROTH, D. P., AND ROBERTSON, E. L. Architectural support for database visualization. In *Proceedings of the Workshop on New Paradigms in Information Visualization and Manipulation* (1998).
- [5] SHNEIDERMAN, B. The eyes have it: A task by data type taxonomy for information visualizations. In *Proc. IEEE Symp. Visual Languages, VL* (3–6 1996), pp. 336–343.