Visual exploration of large data sets

Miron Livny Raghu Ramakrishnan Jussi Myllymaki

Computer Sciences Department
University of Wisconsin–Madison
Madison, WI 53706
{miron,raghu,jussi}@cs.wisc.edu

ABSTRACT

`DEVise` is a data visualization and exploration system capable of handling large data sets using off-the-shelf hardware with minimal memory requirements. Data can be large in volume, complex in structure (multi-dimensional and/or hierarchical), and may be imported from different sources such as database servers, external programs, and World Wide Web resources. Commercial and scientific databases can also be linked to `DEVise` to allow the user to visualize and analyze related information from heterogeneous sources. Associations between data sources are developed interactively as the user gains more knowledge of the data being explored.

To assist in handling large data sets, `DEVise` allows a user to logically split the data into more manageable units at different levels. The user selects a data source, a data stream within a data source (e.g. a time series), attributes of a stream, and a mapping of attributes to graphical objects. At each step, the selections made by the user reduce the data volume. `DEVise` takes advantage of this form of "data compression" to optimize its caching strategies and to minimize the accesses needed to fetch data from tertiary storage, for example.

`DEVise` supports users with different expertise levels by automating most tasks performed by a novice user and by also providing a programming interface that allows new data sources to be defined, new graphical objects to be used, and custom storage policies to be employed.

Keywords: data exploration, data visualization, tertiary storage

1 INTRODUCTION

Visual exploration of large, multidimensional data sets is becoming increasingly important for business and scientific communities alike. Data streams are being continuously produced by remote sensors, program trace generators, simulators, and the stock market, to name a few. Interpreting such data in text form is virtually an impossible task due to the quantity of data. Analyzing the data statistically may be impossible if it is not fully understood what should be looked for. A visual representation, on the other hand, can often be used to find interesting trends and patterns that cannot be expressed in exact statistical terms.

Large and complex data sets present a challenging task for visualization systems. The size of data streams can easily reach to tens or hundreds of megabytes and each record in the stream may contain hundreds of data attributes. It is important for visualization systems to avoid overwhelming the main memory or virtual memory of a workstation, and to allow the user to navigate through the data and metadata without being buried in it. Standard & Poor's CompuStat database,[1] for instance, contains financial data of roughly 5,000 companies. The total size of the database is over 1 gigabyte and each data record has over 350 attributes. The Institute for the Study of Security Markets (ISSM) distributes an annual database[2] containing trade and quote transactions of 5,000 stocks. Each stock is represented as a separate time series. The single largest time series in the 1992 ISSM database is for the IBM stock which recorded almost a million trade and quote transactions, totalling over 20 megabytes in size.

We have developed a data visualization and exploration system called `DEVise` that attempts to solve both the data size and metadata complexity issues. The system uses off-the-shelf hardware and has minimal main memory requirements. It can handle data that is large in volume, complex in structure (multi-dimensional and/or hierarchical), and that may be imported from different sources such as database servers, external programs, and World Wide Web resources. Commercial and scientific databases can also be linked to `DEVise` to allow the user to visualize and analyze related information from heterogeneous sources.

This paper is organized as follows. In Section 2 we review related literature on data exploration and visualization. A review of the `DEVise` system is provided in Section 3. Problems related to managing large data sets and techniques to solve them are discussed in Section 4. The essential features of `DEVise` are demonstrated via a case study in Section 5. Section 6 concludes the paper.

2 RELATED WORK

An introduction to existing visualization software can be found in the surveys by Kornbluh[3] and Braham.[4] There is a large body of literature on techniques used for exploratory data analysis and analysis of multidimensional data.[5–8] Two widely used visualization systems today are AVS[9] and Khoros[10] which are quite successful at visualizing data from physical phenomena such as weather patterns, terrain maps, and medical images. For techniques used to visualize trace files, we refer the reader to Traceview[11] and ParaGraph.[12]

Our focus is on visualizing *streams* of data, that is, record-oriented data sets such as time series or trace files. For work on visualizing a general data structure consisting of arrays and structures, we refer the reader to the VIS-5D and VIS-AD systems.[13] Our focus on stream data allows the `DEVise`[14,15] system to provide a more flexible mapping mechanism to the user, and to incorporate performance optimizations specific to stream data. It also lets us efficiently interface `DEVise` to record-based data servers such as SQL relational databases or a SEQ[16,17] sequence database server.

Common to exploratory data analysis and trace file visualization is the need for a powerful browser. Many visualization systems assume that the data set being visualized will fit in main memory, or assume that virtual memory can be used as a safeguard when the data set is larger than main memory. Our contribution is in making the data browsing aspects of visualization flexible, and in making a first attempt at visualizing large data sets without resorting to virtual memory. The flexibility we offer users comes partly from the ability to define dynamically mappings from data to graphical attributes. Our choice of graphical attributes is inspired by Bertin's work on graphics information processing.[18] The performance limitations of the virtual memory are to a large extent bypassed with the design of the buffer manager and the query processor in the `DEVise` system and with the mechanisms for caching data in various forms. These features make it possible to effectively visualize and explore large quantities of data with limited main memory sizes.

3 SYSTEM DESCRIPTION

In this section, we briefly review the model of visualization used in the `DEVise` system and highlight some of the software and hardware features of the system. The visualization aspect of `DEVise` is described in more detail in earlier papers.[14,15]

In `DEVise` , visualization is structured into a four-stage process, as shown in Figure 1. A *data stream* consists of binary or ASCII records where each record contains one or more *attributes*. A data stream is combined with a *schema* to produce *TData*. The schema describes the data attributes in the data stream.

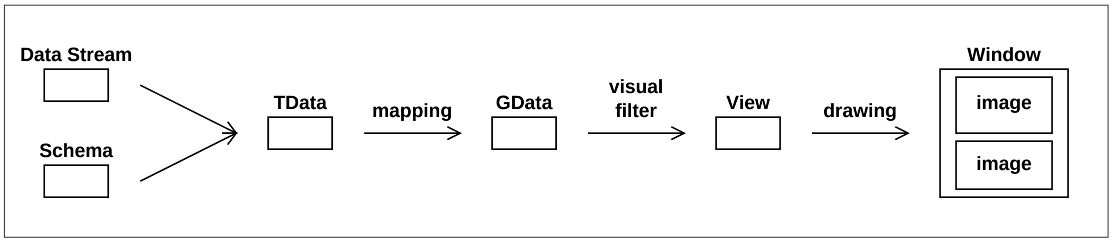

Figure 1: Model of Visualization

Each TData record is mapped to a *GData* record where each GData record consists of the graphical attributes *X*, *Y*, *size*, *color*, *pattern*, *orientation* and *shape* (Figure 2). Each shape (vector, rectangle, bar, polygon, or oval) can have optional parameters specifying, for example, the width and height individually. This definition of a *mapping* gives the user control over the attributes of graphical representation down to the record level. It is up to the user to decide which graphical attributes have constant values and which take their values from the TData record.

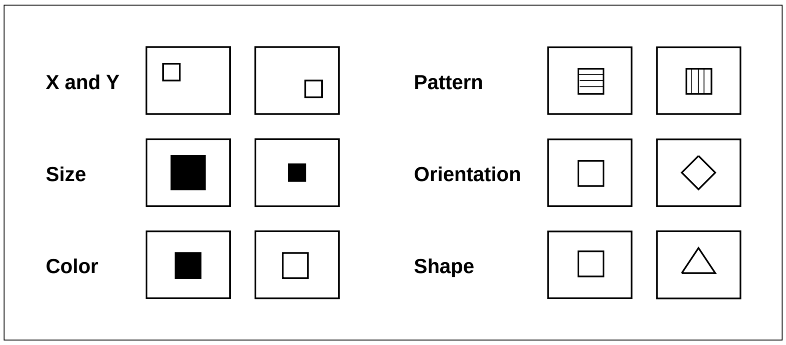

Figure 2: Graphical Data Attributes

A *visual filter* determines the portion of GData to be displayed by defining a query over the graphical attributes of GData. Scrolling and zooming are accomplished by defining a range query over the X and Y attributes of the visual filter. The filter can also be an arbitrary query over all attributes of GData, such as "Retrieve all GData where color = 'red' and shape = 'rectangle' and ...".

A *view* takes a collection of GData records selected by the visual filter and produces an image (a pixel map). A view can also display statistics of the data. For instance, the average or minimum/maximum can be displayed graphically together with the data. A Kiviat graph can be used to display such statistics of multiple views in concise form.

One or more views are drawn in a *window* which is the physical output device. The window provides the screen real estate, and the user can control the real estate by moving and resizing the windows. Various view layouts are available: a window can arrange multiple views vertically, horizontally, or in a tile layout. In automatic mode, the system picks the most appropriate layout whenever the user resizes the window.

A `DEVise` data display window may be embedded in a window of another application. This feature allows any external program to exploit the graphing features of `DEVise` and merge the resulting graphs seamlessly into its user interface. Such an external program merely allocates screen real estate in its own window and instructs `DEVise` to perform the visualization in that screen area.

A *visual link* makes two or more views share some graphical attributes of their visual filter. An X link, for instance, is used to display the same X range in all views sharing that link; zooming or scrolling in the X direction in any of the views zooms or scrolls the others as well. Or, linking two views by the color attribute and selecting only red shapes in one view (via the visual filter) would perform the same selection in the other view as well.

A visual *cursor* associates two views in a different way. A user may display the same GData using two different visual filters in separate views. For instance, one may have a *global view* displayed in one view, containing all GData records, while another view displays a *focus view*, containing a smaller subrange of the GData records. A cursor associates the two views so that a cursor symbol appears in the global view, indicating the X or Y ranges displayed in the focus view. Moving a cursor in the global view with the mouse updates the visual filter of the focus view to correspond to the new location of the cursor. The typical application of this feature is that the user recognizes interesting trends or events in the global view, and moves the cursor to cover the interesting area, which is then displayed in more detail in the focus view. The size and location of the cursor symbol is determined by the zoom level and scroll position of the focus view.

4 MANAGING LARGE DATA SETS

The focus of this section is to describe ways with which large and/or complex data sets can be presented to and manipulated by the user. The key components are managing external data sources, manipulating data streams, defining and using schemas, and managing the mappings of stream data to graphical representations. We also describe a session template mechanism which allows the user to save and reuse the configuration of a visualization session. Section 5 demonstrates many of the features discussed in this section via a case study.

4.1 Manipulating Data Streams

Sometimes an analyst wishes to visualize only a few interesting data streams out of a database containing thousands of them, and it is easy to be buried in the overflow of data. For instance, many security market databases contain time series of thousands of companies or stocks, yet most of the companies are of little interest to an analyst of a particular industry. `DEVise` lets the user to pick interesting data streams and name them for future reference. As a result, the user is not forced to deal with the vast portion of the database he or she is never going to access in the daily work.

Bringing data in from different sources may create another obstacle if the user cannot manipulate the data in a uniform way and using a common user interface. In `DEVise` , after picking interesting data streams from external data sources and giving each of them a unique name, the user never again has to consider where the data is actually coming from. Data streams from all possible sources are viewed as a single collection of streams the user can visualize.

4.2 Defining and Using Schemas

A schema describes a data stream and has two parts: the physical and logical schema. The physical schema is a description of the data attributes and their order in the data stream. The description includes the name and type (integer or floating point number, string, or date) of each attribute. The physical schema also indicates which (if any) attributes are stored in sort-order, as `DEVise` can take advantage of this information to improve performance.

The logical schema lists the names of the attributes the user wants to operate on when exploring data. Different users may have different sets of attributes they are interested in even if they use the same data stream, and the logical schema lets them hide some or most of the attributes. Attributes can also be grouped into an attribute hierarchy which simplifies the task of selecting interesting attributes from large or complex schemas. For instance, the schema for the CompuStat database, described in the next section, contains over 350 attributes for each data record. By grouping and eliminating attributes, a user interested in only a particular subset of the available attributes can reduce the amount of information he or she has to deal with in the daily work.

4.3 Managing Mappings

A mapping, which defines the visual appearance of TData records, is an entity that exists by itself. A mapping is defined based on a particular schema but is not tied to any single TData. The user can *apply* a mapping to any number of TData's that have the same schema. The separation of mappings and TData's is a powerful concept for two reasons. First, it allows one person (such as a system administrator) to provide a common set of mappings, while another person (an analyst) applies the predefined mappings to any data streams of interest. This fits well in environments or fields of science where a standard set of mappings is commonly used. For instance, some attributes may always be displayed with a certain color or shape, and some X-Y attributes pairs may be more common than others.

The second reason why it is important to separate mappings and TData is that it allows the user to look at a given TData from different viewpoints by iterating through all mappings defined for that schema, or to look at many TData's from one viewpoint by iterating through all TData's that use the same schema. The system may even show a thumbnail portfolio of all mappings applied to all TData's and let the user pick interesting ones for closer perusal.

4.4 Interfaces to External Data Sources

Sometimes large data sets are distributed to analysts in a proprietary, foreign format over which the analysts have no control. We cannot assume that data is readily available in the format expected by the visualization system. Data in a foreign format must either be converted to a format the visualization system can use, or the visualization system must be expanded to understand the foreign format and perform the conversion on the fly.

Several problems make the former approach unattractive, particularly when large quantities of data are involved. Converting data and copying it to another tape or disk may consume precious time and storage space. The convert-and-copy step introduces a time lag and may become a nuisance if the analyst receives new data on a regular basis and has to go through the process frequently. Copying in itself may become an issue if copyrighted data is involved. Data in the foreign format may be highly compressed, so uncompressing and copying it may make the use of the data inconvenient if the data no longer fits on a single tape. Updates made to the original data are not reflected in the converted copy automatically which may lead to inconsistencies over time.

We have chosen to follow the latter approach and hence extended `DEVise` by interfacing it to external data protocols and formats (Figure 3). While the native data stream format used by `DEVise` is a regular file, several other data transport mechanisms and formats are supported. Data can be imported via network (socket interface) or via interprocess communication (pipe interface). The data can come from a SQL database server or a SEQ[16,17] sequence database server. The system can also access data stored in the World Wide Web (WWW).

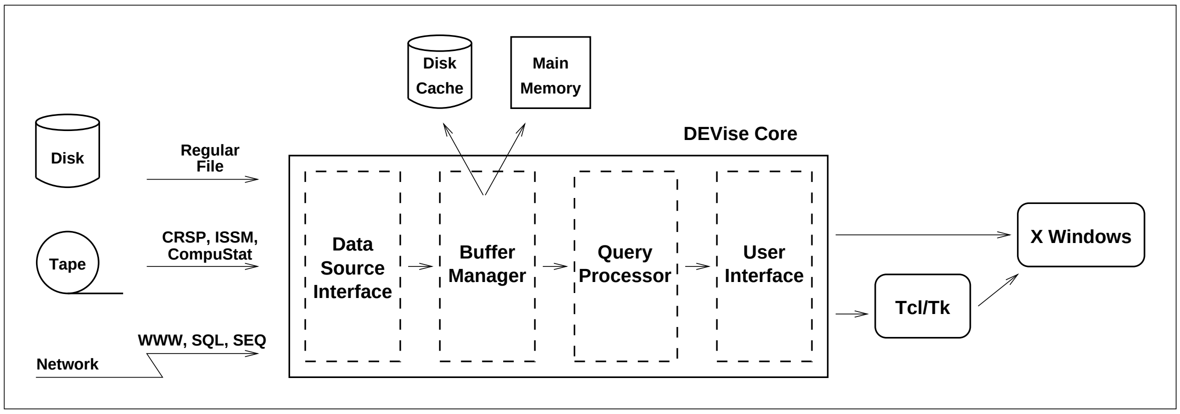

Figure 3: `DEVise` System Architecture

Our collaboration with the School of Business at the University of Wisconsin has produced converters for three databases commonly used by analysts of security markets and other financial instruments. `DEVise` can access data directly on the distribution media (typically magnetic tape) of the following databases:

- Center for Research in Security Prices (CRSP) database[19]
- Standard & Poor's CompuStat database[1]
- Institute for the Study of Security Markets (ISSM) database[2]

Adding support for an external data source is accomplished by defining two software interfaces to it. One interface allows the user to select one or more data streams (typically time series) from the external source. For static databases such as the security market databases, a utility scans the database once and produces an index of all data streams (companies or stocks) available in the database. The user interface reads this index at start-up and lets the user select one or more data streams for visualization. For other types of external data sources, such as SQL databases and WWW resources, there is no index of data streams available, rather, the user defines a query or command which is executed and materialized at a later point when the actual visualization takes place.

The other interface needed to integrate an external data source to `DEVise` is the data extraction routine. With the security market data sources, this routine takes the names of companies or stock symbols selected by a user, sorts them according to their physical order on the storage media (magnetic tape), and then extracts some part or all of the data streams requested. For other types of external data sources, the extraction routines makes a (network) connection to the appropriate server and executes the query or command defined by the user. The resulting output is cached on disk and used for visualization.

4.5 Session Templates

Despite the reduced complexity in manipulating and visualizing large data sets, creating and organizing a visualization session still involves some manual work which one would like to save and return to in the future. `DEVise` allows the user to save everything he or she did during the visualization session, including window positions and sizes, mappings (whether they are displayed on screen or not), view layouts, links, cursors, axis and view title settings, and a history of all visual filters used by the user during the session. The user can later resume a session which restores all settings, and also redraws the last images displayed in the views from pixel images stored as part of the session file.

An alternate way to use the session file is to open it as a template and substitute different TData's in place of the TData's which were used when the session was created and saved. This allows the user to browse a different company or stock, for example, using exactly the same windows and views as he or she used to visualize an earlier company or stock.

Data streams can be switched on the fly as well, without using the session file as an intermediate step. The user merely selects a new data stream (one that uses the same schema) and the system redraws the views using data from the new stream.

5 CASE STUDY: SECURITY PRICES

In this section, we demonstrate some of the features of `DEVise` via a case study using the ISSM and CompuStat databases. The ISSM database contains information about every trading and quote transaction that has occurred in the NYSE and AMEX stock exchanges in a particular year. The CompuStat database contains financial information such as annual and quarterly company earnings and balance sheets over a 20-year period. While some of the discussion in this section is specific to the ISSM and CompuStat databases, most features discussed have general applicability.

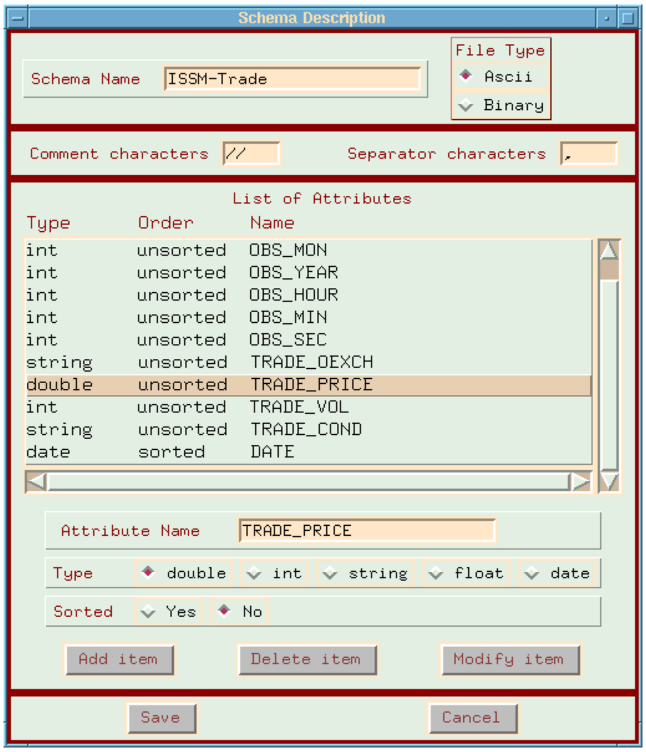

Figure 4: Physical Schema Editor

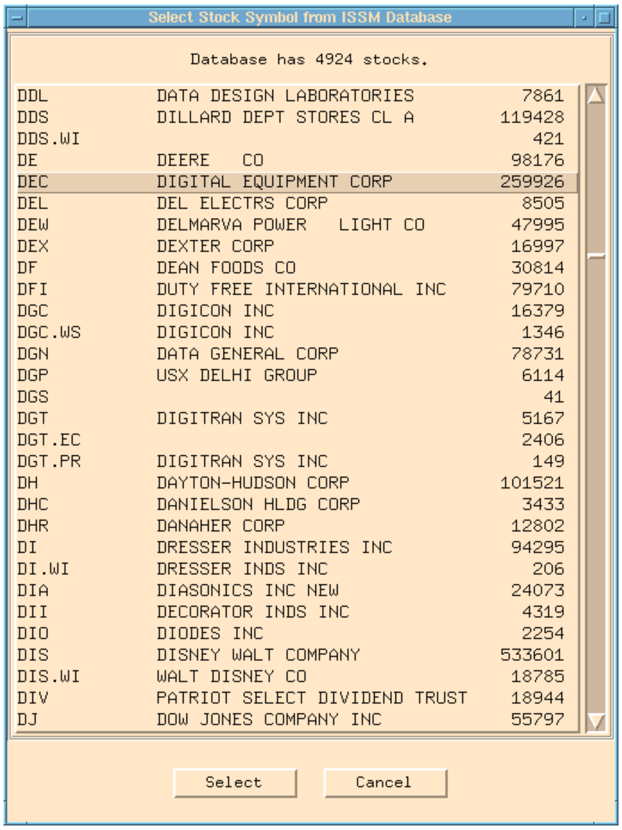

Figure 5: Stock Selection

The ISSM database is distributed annually on magnetic tape, and in this example we will use the 1992 database. For each of the roughly 5,000 companies included in this database, there is a time series containing, among other attributes, the trading price of the company's stock and the quantity of shares traded. The total size of the ISSM database is just over one gigabyte. The largest time series in this database is the IBM stock which had almost a million trades and quotes in 1992, or, on the average, one trade or quote transaction roughly every 30 seconds throughout the year.

The CompuStat database is also one gigabyte in size but has a much more complex schema than the ISSM database. The CompuStat schema has over 350 attributes, so it is important for the user to be able navigate through the metadata in a structured, hierarchical way.

5.1 Defining a Schema

When a new type of data stream is to be visualized in `DEVise` , the data provider defines the physical schema via a schema editing tool (Figure 4). The physical schema is defined only once for each type of data stream, and a descriptive name is associated with the schema for later references. An end user can define a logical schema with the same schema editor. The logical schema is based on a physical schema, but the user is able to customize the schema according to her or his own preferences. The user can remove attributes he or she is not interested in, and can also group attributes into a hierarchy that most closely corresponds to a grouping that exists in the real world, regardless of the physical order or position of the attributes in the data stream (Figures 6 and 7).

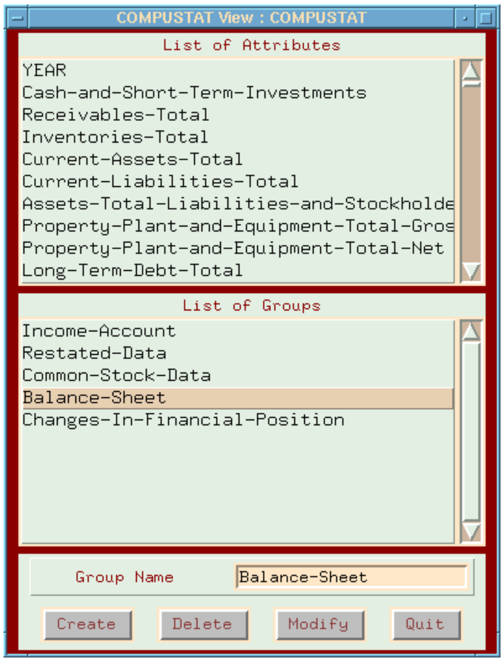

Figure 6: Logical Schema Editor

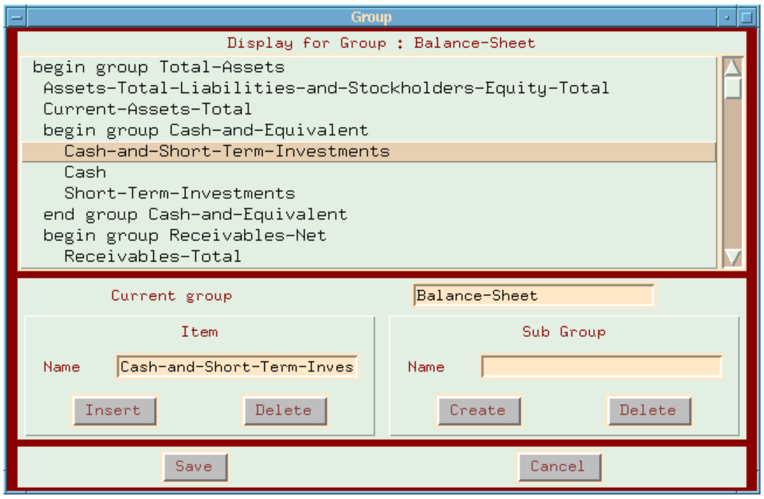

Figure 7: Attribute Grouping

5.2 Defining a Data Stream

The next step involves selecting interesting data streams from the external data source and associating a descriptive name with each stream. Since the ISSM and CompuStat databases are stored on magnetic tape, `DEVise` needs to cache the data streams on disk at various times when visualization is performed. The system also needs to employ a replacement policy for selecting streams to remove from the disk cache when it is running out of disk space. `DEVise` lets the user specify the "value" of each data stream for this purpose; the user may decide that one particular stock should always be kept on disk while some other stocks are less frequently used and therefore an extra tape retrieve delay is acceptable.

Stocks in the ISSM database are indexed by the stock symbol but a company name is included in the header of each stock's time series. Visualizing a new data stream from the ISSM database involves selecting a company from a list (Figure 5) and defining the other characteristics of the data stream (Figure 8). The user then selects one or more data streams (from any data sources) from a central catalog that accumulates and grows over time as users define more and more data streams for visualization (Figure 9).

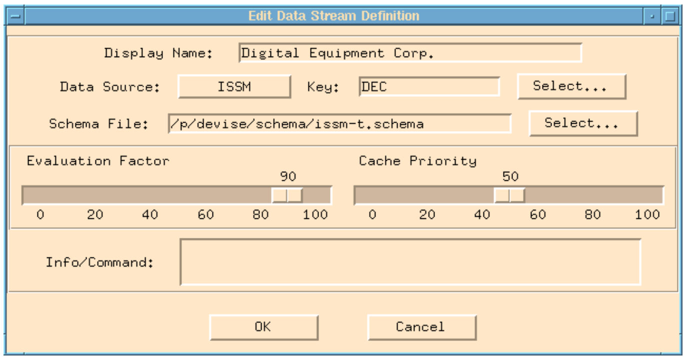

Figure 8: Stream Definition

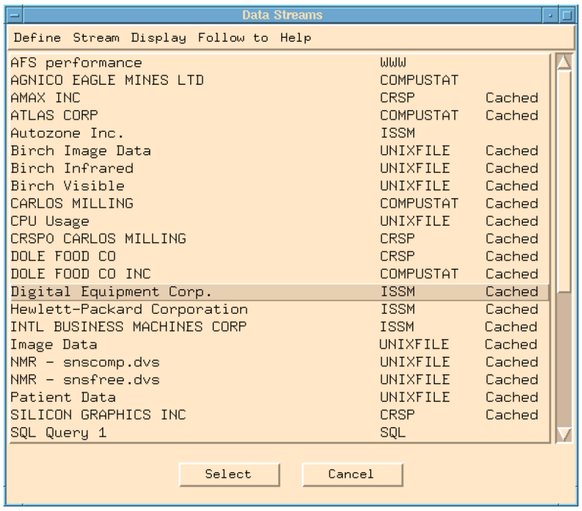

Figure 9: Stream Catalog

5.3 Defining Mappings

Once a user has selected a data stream for visualization, he or she defines mappings, windows, visual filters, and links, which together determine the graphical appearance of the data. A manual procedure lets the user define each of these components individually, but a labor-saving utility called the *automatic mapping utility* lets the user simply select a common X for one or more Y attributes (Figure 10) and also indicate the window where the views will be placed, the layout of the views (vertical, horizontal, and tiled layouts are currently available), and the type of the graphs (scatter plot, image, or bar chart). At this point, the user can also turn on or off the X links, X and Y axes, and view titles.

5.4 Visual Links and Cursors

Figure 12 shows a typical layout of views where one window contains a global view of the stock price plotted over time and another window contains two focus views. One focus view displays the stock price over time for a small time range (6 weeks in this case) while the other focus view displays the volatility (number of shares traded) over the same time period. The two focus views are linked in the X axis so that zooming in or out or scrolling left or right in one view updates the other view as well.

The global and focus stock price views are associated with a cursor in the X axis. An inverted rectangle

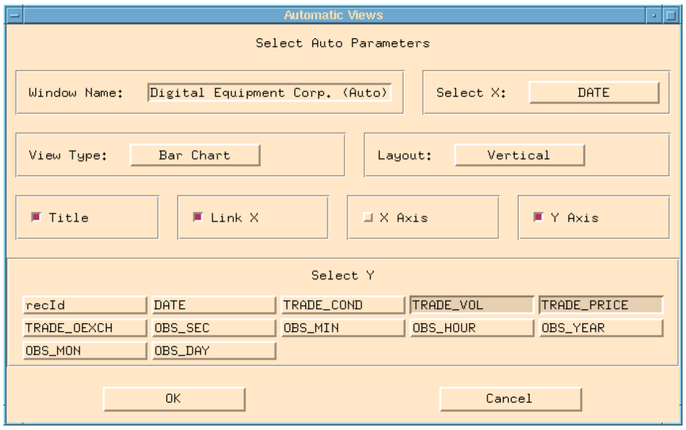

Figure 10: Automatic Mappings

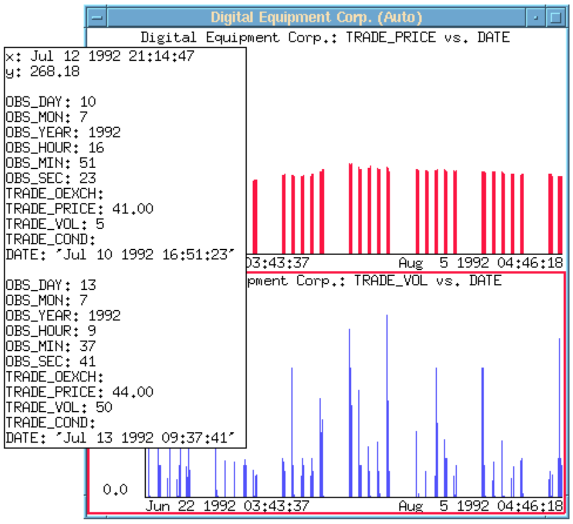

Figure 11: Pop-Up Data Record

appears in the global view to indicate the current zoom level and scroll position of the focus view. The user can move the cursor with the mouse which then updates both focus views because they are linked. Zooming in or out of either focus view causes the cursor to change size, and scrolling in a focus view causes the cursor to move.

5.5 Querying TData Records

Each GData record is displayed as a graphical symbol according to the mapping definition. Clicking the mouse on a symbol pops up a window which displays the complete TData record (Figure 11). In this example, the user clicked at X = July 12th and Y = 268 in the trade volume view, and the system displays the nearest two records in the pop-up window. The top record represents the last trade on the preceding Friday, and the bottom record is the first trade on the following Monday.

5.6 Switching Databases

As described in Section 4.4, our prototype system is integrated with three security price and financial information databases. One of the system features is the ability to associate data streams in one external data source with data streams in another data source. For instance, there is a time series for IBM in all three financial databases but the company is indexed differently in all three. A utility accessible to the user via the menus in `DEVise` has automatically associated each occurrence of IBM in the different databases so that the user can easily compare related time series from two or more databases.

6 CONCLUSION

In this paper we have discussed problems associated with visualizing large data sets and suggested ways to solve some of the problems. Large data sets involve both large data volumes and complex structures (multi-dimensional and/or hierarchical data). Both aspects can bury the user in an overflow of data unless the user has

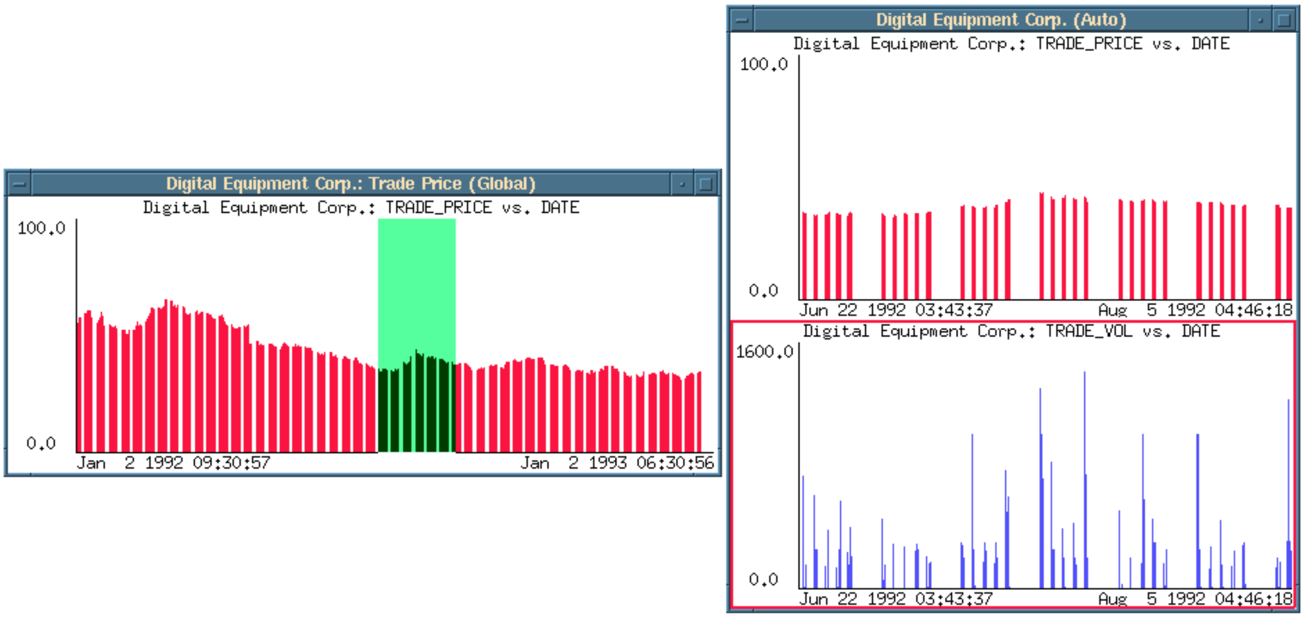

Figure 12: Visual Link and Cursor

tools for manipulating data and metadata in a structured way. Just as importantly, performance may become a serious bottleneck and limit the usefulness of a visualization system if the volume of data exceeds main memory sizes and virtual memory is used as a safeguard.

`DEVise` is a data visualization and exploration system designed to handle large data sets using off-the-shelf hardware with minimal memory requirements. It attempts to solve the data management issue by letting the user focus on just the data streams and data attributes he or she is interested in, and hiding the rest. Logical schemas with attribute groups provide a means for hiding the complexity (and uninteresting attributes) of physical data structures. Named data streams let the user identify interesting data sequences in large databases (perhaps consisting of thousands of time series), and provide a common interface to dissimilar external data sources.

`DEVise` further simplifies data management tasks by providing a template mechanism that lets the user design a visualization session using one or more data streams and then reapply the same visualization definitions (including those of windows, mappings, and views) to other data streams. An special utility automates many tasks involved in designing a visualization session, reducing the learning curve of novice users.

To remove memory size constraints and improve overall performance, `DEVise` borrows query processing and buffer management concepts and techniques from database management systems. Data stored in tertiary storage, such as magnetic tapes, can be visualized in `DEVise` either directly or via a secondary storage cache.

7 ACKNOWLEDGEMENTS

The work presented in this paper was supported by NASA grant NAGW-3921. Raghu Ramakrishnan's work was additionally supported by a Packard Foundation Fellowship in Science and Engineering, a PYI Award with matching grants from DEC, HP, IBM, Tandem, and Xerox, and NSF grant IRI-9011563.

8 REFERENCES

[1] Standard & Poor's. *CompuStat Database Manual*, 1993.

[2] Institute for the Study of Security Markets. *ISSM Database Manual*, 1993.

[3] Ken Kornbluh. Active data analysis: Advanced software for the 90's. *IEEE Spectrum*, 31(11):57–83, November 1994.

[4] Robert Braham. Math & visualization: New tools, new frontiers. *IEEE Spectrum*, 32(11):19–36, November 1995.

[5] Leo Breiman and Jerome Friedman. Tools for large data set analysis. In Edward J. Wegman and James G. Smith, editors, *Statistical Signal Processing*, pages 191–197. Marcel Dekker, Inc., 1994.

[6] David J. Lubinsky and Forrest W. Young. Guiding data analysis. In *Proc. Statistical Computing Section*, pages 205–210, 1990.

[7] Paul F. Velleman. Computing and modern (exploratory) data analysis. In *Proc. Statistical Computing Section*, pages 46–54, 1990.

[8] S.H.C. du Toit, A.G.W Steyn, and R.H. Stumpf. *Graphical Exploratory Data Analysis.* Springer-Verlag, New York, 1986.

[9] Craig Upson, Thomas Faulhaber Jr., David Kamins, David Laidlaw, David Schlegel, Jeffrey Vroom, Robert Gurwitz, and Andris van Dam. The Application Visualization System: A computational environment for scientific visualization. *IEEE Computer Graphics & Applications*, 9(4):30–42, July 1989.

[10] The Khoros Group. *Khoros Manual.* University of New Mexico, Albuquerque, NM 87131, 1991.

[11] Allen D. Malony, David H. Hammerslag, and David J. Jablonowski. Traceview: A trace visualization tool. *IEEE Software*, pages 19–28, September 1991.

[12] Michael T. Heath and Jennifer A. Etheridge. Visualizing the performance of parallel programs. *IEEE Software*, pages 29–39, September 1991.

[13] William Hibbard, Brian E. Paul, David Santek, Charles Dyer, Andre Battaiola, and Marie-Francoise Voidrot-Martinez. Interactive visualization of earth and space science computation. *IEEE Computer*, pages 65–72, 1994.

[14] Michael Cheng, Miron Livny, and Raghu Ramakrishnan. Visual analysis of stream data. In *Proc. of SPIE – Int. Soc. Opt. Eng.*, volume 2410, pages 108–119, San Jose, CA, April 1995.

[15] Michael Cheng. *Visual Exploration of Large Amounts of Record Based Sequence Data.* PhD thesis, University of Wisconsin–Madison, August 1995.

[16] Praveen Seshadri, Miron Livny, and Raghu Ramakrishnan. Sequence query processing. In *Proc. ACM SIGMOD*, pages 430–441, Minneapolis, MN, May 1994.

[17] Raghu Ramakrishnan, Michael Cheng, Miron Livny, and Praveen Seshadri. What's next? Sequence queries. In *Proc. Int. Conf. Management of Data*, December 1994.

[18] Jacques Bertin. *Graphics and Graphic Information-Processing.* Walter de Gruyter, Berlin, 1981.

[19] Center for Research in Security Prices. *CRSP Stock File Guide.* Graduate School of Business, University of Chicago, 1992.