DOI:10.1145/1400214.1400233

Technical Perspective
The Polaris Tableau System

By Jim Gray

**Jim Gray nominated the Polaris paper for the Research Highlights section and wrote the first draft of this Technical Perspective in November 2006. David Patterson revised the essay in August 2008.*

DATA-INTENSIVE APPLICATIONS often have dozens of independent dimensions with a set of measurements for each. Such high-dimensional datasets involving many variables and measurements are increasingly common, but good tools to analyze them are not. If you have ever been frustrated when trying to plot a useful graph from a simple spreadsheet, you would appreciate the value of a system that allows users to create stunning graphs interactively and easily from large multidimensional datasets.

Stolte, Tang, and Hanrahan have done that with Polaris, a declarative visual query language that unifies the strengths of visualization and database communities. It allows users to visualize relationships between data using shape, size, orientation, color, and texture in all kinds of graphs, and leverages the advances in database systems to optimize performance of accesses to large datasets. This combination lets you interactively explore the raw data or perform data analysis. It is a major improvement over how analysis is currently done.

Their work makes three advances in parallel: First, they show how to automatically construct graphs, charts, maps, and timelines as table visualizations. While these ideas are implicit in many graphing packages, the authors unified several approaches into a simple algebra for graphical presentation of quantitative and categorical information. The unification makes it easy to switch from one representation to another and to change or add dimensions to a graphical presentation. Second, they unify this graphical language with the SQL query languages, producing a declarative visual query language in which a single "program" specifies both data retrieval and data presentation. The third advance is a GUI that "writes" the visual queries as you drag and drop dimensions or measurements in a data viewer. This combination makes it simple for users to ask "what if" questions for large multidimensional datasets.

Here is the "Visual SQL" to create a map by U.S. ZIP code of fundraising by the U.S. presidential candidates through May 2008. It places the results on a map using the latitude and longitude and lets the system pick the size of the circles representing the relative amounts of fundraising. It totals the amount of fundraising per candidate per ZIP code.

In fact the research for this work was conducted several years ago and incorporated as a commercial product—the Tableau system—that can analyze high-dimensional data from flat files, spreadsheets, and SQL data sources. The data algebra and graph algebra developed here is key to the success of the visual query language and Tableau.

Hence, this is a rare paper that explains both the basic premise and its real-world evaluation. The notion that a formal algebra of relationships between tables and visual encodings would help the exploratory nature of the system was indeed validated. However, they found that default values of the visual encodings were important since few users opted to choose the details of shape and color selection, since they were not trained graphic designers nor psychologists and would rather spend their time exploring the data. C

```
SELECT  Latitude ON ROWS,
        ( [Candidate Name] * Longitude ) ON COLUMNS,
        [Candidate Name] ON COLOR,
        SUM(Amount) ON SIZE,
        [Zip Code] ON LEVEL_OF_DETAIL
FROM [Contributions View]
WHERE [Candidate Name] = {"John McCain","Barack Obama"}
```

The VizSQL code above results in the following SQL code to collect the data:

```
SELECT ([Contributions View].[Candidate Name]),
       ([Contributions View].[Zip Code]),
       (SUM([Contributions View].[Amount]))
FROM [dbo].[Contributions View]
WHERE ((([Contributions View].[Candidate Name]) IN ('McCain, John S', 'Obama,
Barack'))
GROUP BY ([Contributions View].[Candidate Name]), ([Contributions View].[Zip
Code])
```

The graph here is the output of this query zoomed into Manhattan Island in New York City.