Polaris: A System for Query, Analysis, and Visualization of Multidimensional Relational Databases

Chris Stolte, Diane Tang, and Pat Hanrahan

Abstract—In the last several years, large multidimensional databases have become common in a variety of applications such as data warehousing and scientific computing. Analysis and exploration tasks place significant demands on the interfaces to these databases. Because of the size of the data sets, dense graphical representations are more effective for exploration than spreadsheets and charts. Furthermore, because of the exploratory nature of the analysis, it must be possible for the analysts to change visualizations rapidly as they pursue a cycle involving first hypothesis and then experimentation. In this paper, we present Polaris, an interface for exploring large multidimensional databases that extends the well-known Pivot Table interface. The novel features of Polaris include an interface for constructing visual specifications of table-based graphical displays and the ability to generate a precise set of relational queries from the visual specifications. The visual specifications can be rapidly and incrementally developed, giving the analyst visual feedback as they construct complex queries and visualizations.

Index Terms—Database visualization, database analysis, visualization formalism, multidimensional databases.

1 Introduction

In the last several years, large databases have become common in a variety of applications. Corporations are creating large data warehouses of historical data on key aspects of their operations. International research projects such as the Human Genome Project [20] and Digital Sky Survey [31] are generating massive databases of scientific data.

A major challenge with these databases is to extract meaning from the data they contain: to discover structure, find patterns, and derive causal relationships. The analysis and exploration necessary to uncover this hidden information places significant demands on the human-computer interfaces to these databases. The exploratory analysis process is one of hypothesis, experiment, and discovery. The path of exploration is unpredictable and the analysts need to be able to rapidly change both what data they are viewing and how they are viewing that data.

The current trend is to treat multidimensional databases as n-dimensional data cubes [16]. Each dimension in these data cubes corresponds to one dimension in the relational schema. Perhaps the most popular interface to multidimensional databases is the Pivot Table [15]. Pivot Tables allow the data cube to be rotated, or pivoted, so that different dimensions of the dataset may be encoded as rows or columns of the table. The remaining dimensions are aggregated and displayed as numbers in the cells of the table. Cross-tabulations and summaries are then added to the resulting table of numbers. Finally, graphs may be generated from the resulting tables. Visual Insights recently released a new interface for visually exploring projections of data cubes using linked views of bar charts, scatterplots, and parallel coordinate displays [14].

In this paper, we present Polaris, an interface for the exploration of multidimensional databases that extends the Pivot Table interface to directly generate a rich, expressive set of graphical displays. Polaris builds tables using an algebraic formalism involving the fields of the database. Each table consists of layers and panes and each pane may contain a different graphic. The use of tables to organize multiple graphs on a display is a technique often used by statisticians in their analysis of data [5], [11], [38].

The Polaris interface is simple and expressive because it is built upon a formalism for constructing graphs and building data transformations. We interpret the state of the interface as a visual specification of the analysis task and automatically compile it into data and graphical transformations. This allows us to combine statistical analysis and visualization. Furthermore, all intermediate specifications that can be created in the visual language are valid and can be interpreted to create visualizations. Therefore, analysts can incrementally construct complex queries, receiving visual feedback as they assemble and alter the specifications.

2 Related Work

The related work to Polaris can be divided into three categories: formal graphical specifications, table-based data displays, and database exploration tools.

2.1 Formal Graphical Specifications

Bertin's *Semiology of Graphics* [6] is one of the earliest attempts at formalizing graphing techniques. Bertin

- *The authors are with the Computer Science Department, Stanford University, Stanford, CA 94305.*
E-mail: {cstolte, hanrahan}@graphics.stanford.edu, dtang@cs.stanford.edu.

Manuscript received 17 Apr. 2001; accepted 10 July 2001.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number 114503.

developed a vocabulary for describing data and the techniques for encoding data in a graphic. One of his important contributions is the identification of the retinal variables (position, color, size, etc.) in which data can be encoded. Cleveland [11], [12] used theoretical and experimental results to determine how well people can use these different retinal properties to compare quantitative variations.

Mackinlay's APT system [26] is one of the first applications of formal graphical specifications to computer-generated displays. APT uses a set of graphical languages and composition rules to automatically generate 2D displays of relational data. The Sage system [29] extends the concepts of APT, providing a richer set of data characterizations and generating a wider range of displays.

Livny et al. [25] describe a visualization model that provides a foundation for database-style processing of visual queries. Within this model, the relational queries and graphical mappings necessary to generate visualizations are defined by a set of relational operators. The Rivet visualization environment [9] applies similar concepts to provide a flexible database visualization tool.

Wilkinson [41] recently developed a comprehensive language for describing traditional statistical graphs and proposed a simple interface for generating a subset of the specifications expressible within his language. We have extended Wilkinson's ideas to develop a specification that can be directly mapped to an interactive interface and that is tightly integrated with the relational data model. The differences between our work and Wilkinson's will be further discussed in Section 8.

2.2 Table-Based Displays

Another area of related work is visualization systems that use table-based displays. Static table displays, such as scatterplot matrices [18] and Trellis [3] displays, have been used extensively in statistical data analysis. Recently, several interactive table displays have been developed. Pivot Tables [15] allow analysts to explore different projections of large multidimensional datasets by interactively specifying assignments of fields to the table axes, but are limited to text-based displays. Systems such as the Table Lens [27] and FOCUS [32] visualization systems provide table displays that present data in a relational table view, using simple graphics in the cells to communicate quantitative values.

2.3 Database Exploration Tools

The final area of related work is visual query and database exploration tools. Projects such as VQE [13], Visage [30], DEVise [25], and Tioga-2 [2] have focused on developing visualization environments that directly support interactive database exploration through *visual queries*. Users can construct queries and visualizations directly through their interactions with the visualization system interface. These systems have flexible mechanisms for mapping query results to graphs and all of the systems support mapping database records to retinal properties of the marks in the graphs. However, none of these systems leverages table-based organizations of their visualizations.

Other existing systems, such as XmdvTool [40], Spotfire [33], and XGobi [10], have taken the approach of providing a set of predefined visualizations, such as scatterplots and parallel coordinates, for exploring multidimensional data sets. These views are augmented with extensive interaction techniques, such as brushing and zooming, that can be used to refine the queries. We feel that this approach is much more limiting than providing the user with a set of building blocks that can be used to interactively construct and refine a wide range of displays to suit an analysis task.

Another significant database visualization system is VisDB [22], which focuses on displaying as many data items as possible to provide feedback as users refine their queries. This system even displays tuples that do not meet the query and indicates their "distance" from the query criteria using spatial encodings and color. This approach helps the user avoid missing important data points just outside of the selected query parameters. In contrast, Polaris provides extensive ability to drill-down and roll-up data, allowing the analyst to get a complete overview of the data set before focusing on detailed portions of the database.

3 Overview

Polaris has been designed to support the interactive exploration of large multidimensional relational databases. Relational databases organize data into tables where each row in a table corresponds to a basic entity or fact and each column represents a property of that entity [36]. We refer to a row in a relational table as a *tuple* or *record* and a column in the table as a *field*. A single relational database will contain many heterogeneous but interrelated tables.

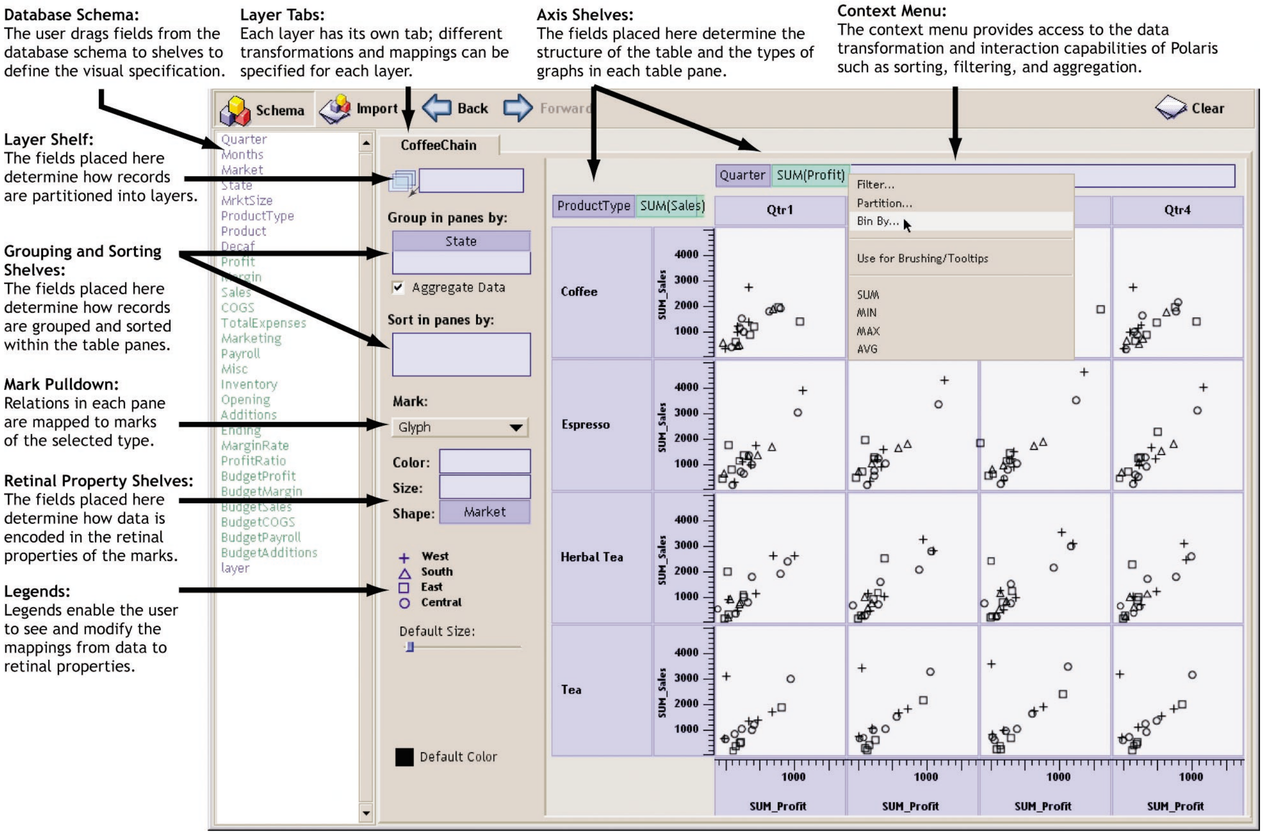
We can characterize fields in a database as nominal, ordinal, quantitative, or interval [6], [34]. Polaris reduces this categorization to ordinal and quantitative by treating intervals as quantitative and assigning an ordering to the nominal fields and subsequently treating them as ordinal.

The fields within a relational table can also be partitioned into two types: dimensions and measures. Dimensions and measures are similar to independent and dependent variables in traditional analysis. For example, a product name or type would be a dimension of a product and the product price or size would be a measure. The current implementation of Polaris treats all ordinal fields as dimensions and all quantitative and interval fields as measures.

In many important business and scientific databases, there are often many dimensions identifying a single entity. For example, a transaction within a store may be identified by the time of the sale, the location of the store, the type of product, and the customer. In most data warehouses, these multidimensional databases are structured as n-dimensional data cubes [36]. Each dimension in the data cube corresponds to one dimension in the relational schema.

To effectively support the analysis process in large multidimensional databases, an analysis tool must meet several demands:

- **Data-dense displays:** The databases typically contain a large number of records and dimensions. Analysts need to be able to create visualizations that



Database Schema: The user drags fields from the database schema to shelves to define the visual specification.

Layer Tabs: Each layer has its own tab; different transformations and mappings can be specified for each layer.

Axis Shelves: The fields placed here determine the structure of the table and the types of graphs in each table pane.

Context Menu: The context menu provides access to the data transformation and interaction capabilities of Polaris such as sorting, filtering, and aggregation.

Layer Shelf: The fields placed here determine how records are partitioned into layers.

Grouping and Sorting Shelves: The fields placed here determine how records are grouped and sorted within the table panes.

Mark Pulldown: Relations in each pane are mapped to marks of the selected type.

Retinal Property Shelves: The fields placed here determine how data is encoded in the retinal properties of the marks.

Legends: Legends enable the user to see and modify the mappings from data to retinal properties.

Fig. 1. The Polaris user interface. Analysts construct table-based displays of relational data by dragging fields from the database schema onto shelves throughout the display. A given configuration of fields on shelves is called a visual specification. The specification unambiguously defines the analysis and visualization operations to be performed by the system to generate the display.

will simultaneously display many dimensions of large subsets of the data.

- **Multiple display types:** Analysis consists of many different tasks such as discovering correlations between variables, finding patterns in the data, locating outliers, and uncovering structure. An analysis tool must be able to generate displays suited to each of these tasks.
- **Exploratory interface:** The analysis process is often an unpredictable exploration of the data. Analysts must be able to rapidly change what data they are viewing and how they are viewing that data.

Polaris addresses these demands by providing an interface for rapidly and incrementally generating table-based displays. In Polaris, a table consists of a number of rows, columns, and layers. Each table axis may contain multiple nested dimensions. Each table entry, or *pane*, contains a set of records that are visually encoded as a set of marks to create a graphic.

Several characteristics of tables make them particularly effective for displaying multidimensional data:

- **Multivariate:** Multiple dimensions of the data can be explicitly encoded in the structure of the table, enabling the display of high-dimensional data.
- **Comparative:** Tables generate *small-multiple* displays of information, which, as Tufte [38] explains, are easily compared, exposing patterns and trends across dimensions of the data.
- **Familiar:** Table-based displays have an extensive history. Statisticians are accustomed to using tabular displays of graphs, such as scatterplot matrices and Trellis displays, for analysis. Pivot Tables are a common interface to large data warehouses.

Fig. 1 shows the user interface presented by Polaris. In this example, the analyst has constructed a matrix of scatterplots showing sales versus profit for different product types in different quarters. The primary interaction technique is to drag-and-drop fields from the database schema onto shelves throughout the display. We call a given configuration of fields on shelves a *visual specification*. The specification determines the analysis and visualization operations to be performed by the system, defining:

- The mapping of data sources to layers. Multiple data sources may be combined in a single Polaris visualization. Each data source maps to a separate layer or set of layers.
- The number of rows, columns, and layers in the table and their relative orders (left to right as well as back to front). The database dimensions assigned to rows are specified by the fields on the *y* shelf, columns by fields on the *x* shelf, and layers by fields on the *layer* (z) shelf. Multiple fields may be dragged onto each shelf to show categorical relationships.
- The selection of records from the database and the partitioning of records into different layers and panes.

- The grouping of data within a pane and the computation of statistical properties, aggregates, and other derived fields. Records may also be sorted into a given drawing order.
- The type of graphic displayed in each pane of the table. Each graphic consists of a set of marks, one mark per record in that pane.
- The mapping of data fields to retinal properties of the marks in the graphics. The mappings used for any given visualization are shown in a set of automatically generated legends.

Analysts can interact with the resulting visualizations in several ways. Selecting a single mark in a graphic by clicking on it pops up a detail window that displays user-specified field values for the tuples corresponding to that mark. Analysts can draw rubberbands around a set of marks to brush records. Brushing, discussed in more detail in Section 5.3, can be performed within a single table or between multiple Polaris displays.

In the next section, we describe how the visual specification is used to generate graphics. In Section 5, we describe the support Polaris provides for visually querying and transforming the data through filters, sorting, and data transformations. Then, in Section 6, we discuss how the visual specification is used to generate the database queries and statistical analysis.

4 Generating Graphics

The visual specification consists of three components: 1) the specification of the different table configurations, 2) the type of graphic inside each pane, and 3) the details of the visual encodings. We discuss each of these in turn.

4.1 Table Algebra

We need a formal mechanism to specify table configurations and we have defined an algebra for this purpose. When the analysts place fields on the axis shelves, as shown in Fig. 1, they are implicitly creating expressions in this algebra.

A complete table configuration consists of three separate expressions in this table algebra. Two of the expressions define the configuration of the x and y axes of the table, partitioning the table into rows and columns. The third expression defines the z axis of the table, which partitions the display into *layers*.

The operands in this table algebra are the names of the ordinal and quantitative fields of the database. We will use A, B, and C to represent ordinal fields and P, Q, and R to represent quantitative fields. We assign ordered sets to each field symbol in the following manner: To ordinal fields, we assign the members of the ordered domain of the field and, to quantitative fields, we assign the single element set containing the field name.

$$\begin{aligned}\mathtt{A} &= \mathtt{domain(A)} = \{\mathtt{a_1},\ldots,\mathtt{a_n}\}\\ \mathtt{P} &= \{\mathtt{P}\}.\end{aligned}$$

This assignment of sets to symbols reflects the difference in how the two types of fields will be encoded in the structure of the tables. Ordinal fields will partition the table into rows and columns, whereas quantitative fields will be spatially encoded as axes within the panes.

A valid expression in our algebra is an ordered sequence of one or more symbols with operators between each pair of adjacent symbols and with parentheses used to alter the precedence of the operators. The operators in the algebra are cross (×), nest (/), and concatenation (+), listed in order of precedence. The precise semantics of each operator is defined in terms of its effects on the operand sets.

4.1.1 Concatenation

The concatenation operator performs an ordered union of the sets of the two symbols:

$$\begin{aligned}\mathtt{A}+\mathtt{B} &= \{\mathtt{a_1},\ldots,\mathtt{a_n}\}+\{\mathtt{b_1},\ldots,\mathtt{b_m}\}\\ &= \{\mathtt{a_1},\ldots,\mathtt{a_n},\mathtt{b_1},\ldots,\mathtt{b_m}\}\\ \mathtt{A}+\mathtt{P} &= \{\mathtt{a_1},\ldots,\mathtt{a_n}\}+\{\mathtt{P}\}\\ &= \{\mathtt{a_1},\ldots,\mathtt{a_n},\mathtt{P}\}\\ \mathtt{P}+\mathtt{Q} &= \{\mathtt{P}\}+\{\mathtt{Q}\}\\ &= \{\mathtt{P},\mathtt{Q}\}.\end{aligned}$$

4.1.2 Cross

The cross operator performs a Cartesian product of the sets of the two symbols:

$$\begin{aligned}\mathtt{A}\times\mathtt{B} &= \{\mathtt{a_1},\ldots,\mathtt{a_n}\}\times\{\mathtt{b_1},\ldots,\mathtt{b_m}\}\\ &= \{\mathtt{a_1b_1},\ldots,\mathtt{a_1},\mathtt{b_m},\\ &\qquad \mathtt{a_2b_1},\ldots,\mathtt{a_2b_n},\ldots,\\ &\qquad \mathtt{a_nb_1},\ldots,\mathtt{a_nb_m}\}\\ \mathtt{A}\times\mathtt{P} &= \{\mathtt{a_1},\ldots,\mathtt{a_n}\}\times\mathtt{P}\\ &= \{\mathtt{a_1},\mathtt{P},\ldots,\mathtt{a_nP}\}.\end{aligned}$$

4.1.3 Nest

The nest operator is similar to the cross operator, but it only creates set entries for which there exist records with those domain values. If we define R to be the dataset being analyzed, r to be a record, and $A(r)$ to be the value of the field A for the record r, then we can define the nest operator as follows:

$$\begin{aligned}\mathtt{A/B} = \{\mathtt{a_ib_j} \mid \exists \mathtt{r}\in\mathtt{R}\ \mathtt{st}\\ \mathtt{A(r)} = \mathtt{a_i}\ \&\ \mathtt{B(r)} = \mathtt{b_i}\}.\end{aligned}$$

The intuitive interpretation of the *nest* operator is "B within A." For example, given the fields *quarter* and *month*, the expression *quarter/month* would be interpreted as those months within each quarter, resulting in three entries for each quarter. In contrast, *quarter* × *month* would result in 12 entries for each quarter.

Using the above set semantics for each operator, every expression in the algebra can be reduced to a single set, with each entry in the set being an ordered concatenation of zero or more ordinal values with zero or more quantitative field names. We call this set evaluation of an expression the *normalized set form*. The normalized set form of an expression determines one axis of the table: The table axis is partitioned into columns (or rows or layers) so that there is a one-to-one correspondence between set entries in the

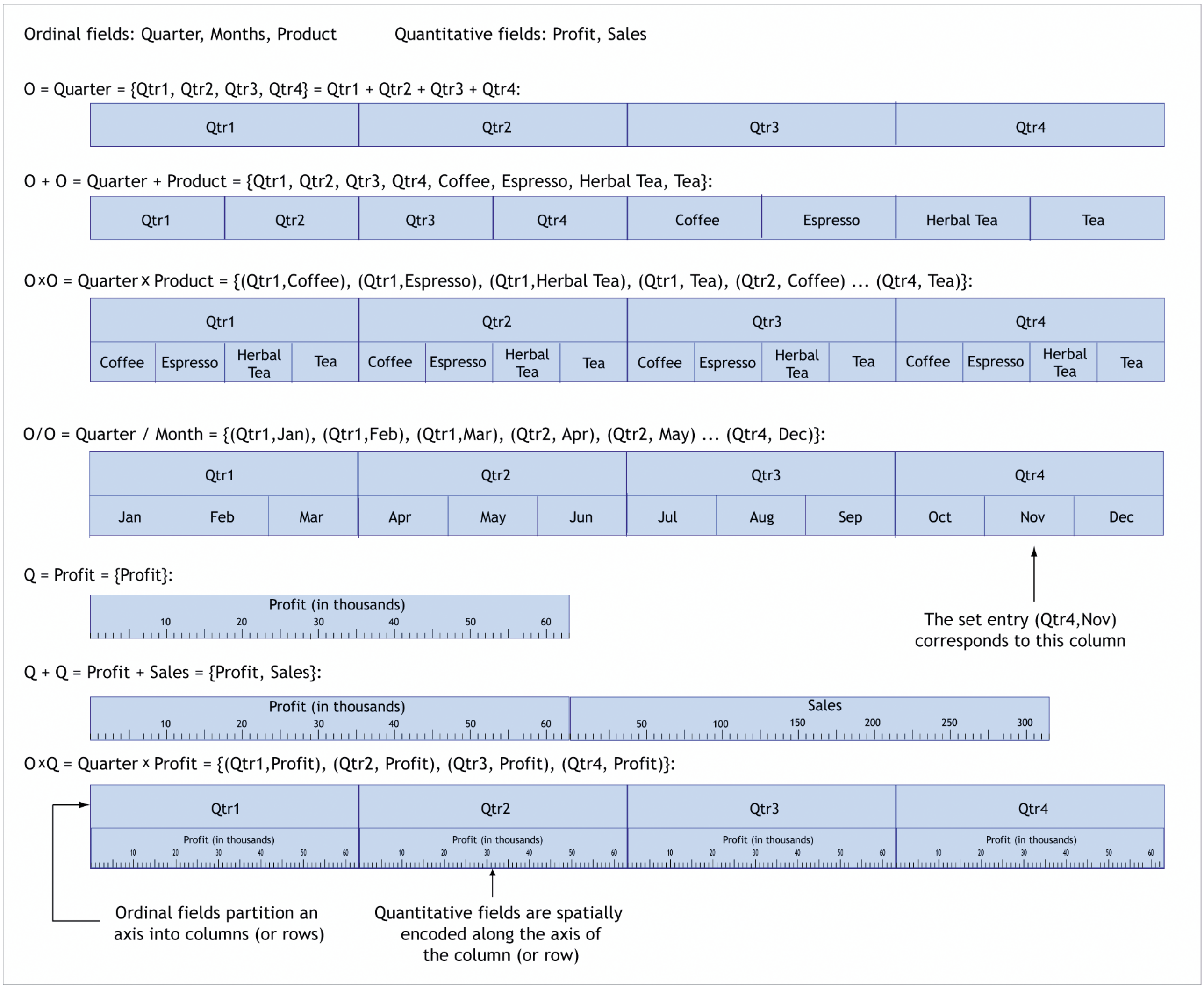

Ordinal fields: Quarter, Months, Product
Quantitative fields: Profit, Sales
O = Quarter = {Qtr1, Qtr2, Qtr3, Qtr4} = Qtr1 + Qtr2 + Qtr3 + Qtr4:
Qtr1 Qtr2 Qtr3 Qtr4
O + O = Quarter + Product = {Qtr1, Qtr2, Qtr3, Qtr4, Coffee, Espresso, Herbal Tea, Tea}:
Qtr1 Qtr2 Qtr3 Qtr4 Coffee Espresso Herbal Tea Tea
O×O = Quarter x Product = {(Qtr1,Coffee), (Qtr1,Espresso), (Qtr1,Herbal Tea), (Qtr1, Tea), (Qtr2, Coffee) ... (Qtr4, Tea)}:
Qtr1 Qtr2 Qtr3 Qtr4
Coffee Espresso Herbal Tea Tea
O/O = Quarter / Month = {(Qtr1,Jan), (Qtr1,Feb), (Qtr1,Mar), (Qtr2, Apr), (Qtr2, May) ... (Qtr4, Dec)}:
Qtr1 Qtr2 Qtr3 Qtr4
Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
Q = Profit = {Profit}:
Profit (in thousands)
10 20 30 40 50 60
The set entry (Qtr4,Nov) corresponds to this column
Q + Q = Profit + Sales = {Profit, Sales}:
Profit (in thousands)
10 20 30 40 50 60
Sales
50 100 150 200 250 300
O×Q = Quarter x Profit = {(Qtr1,Profit), (Qtr2, Profit), (Qtr3, Profit), (Qtr4, Profit)}:
Qtr1 Qtr2 Qtr3 Qtr4
Profit (in thousands)
10 20 30 40 50 60
Ordinal fields partition an axis into columns (or rows)
Quantitative fields are spatially encoded along the axis of the column (or row)

Fig. 2. The graphical interpretation of several expressions in the table algebra. Each expression in the table algebra can be reduced to a single set of terms and that set can then be directly mapped into a configuration for an axis of the table.

normalized set and columns. Fig. 2 illustrates the configurations resulting from a number of expressions.

Analysts can also combine multiple data sources in a single Polaris visualization. When multiple data sources are imported, each data source is mapped to a distinct layer (or set of layers). While all data sources and all layers share the same configuration for the x and y axes of the table, each data source can have a different expression for partitioning its data into layers. Fig. 3b and Fig. 3c illustrate the use of layers to compose multiple distinct data sources into a single visualization.

4.2 Types of Graphics

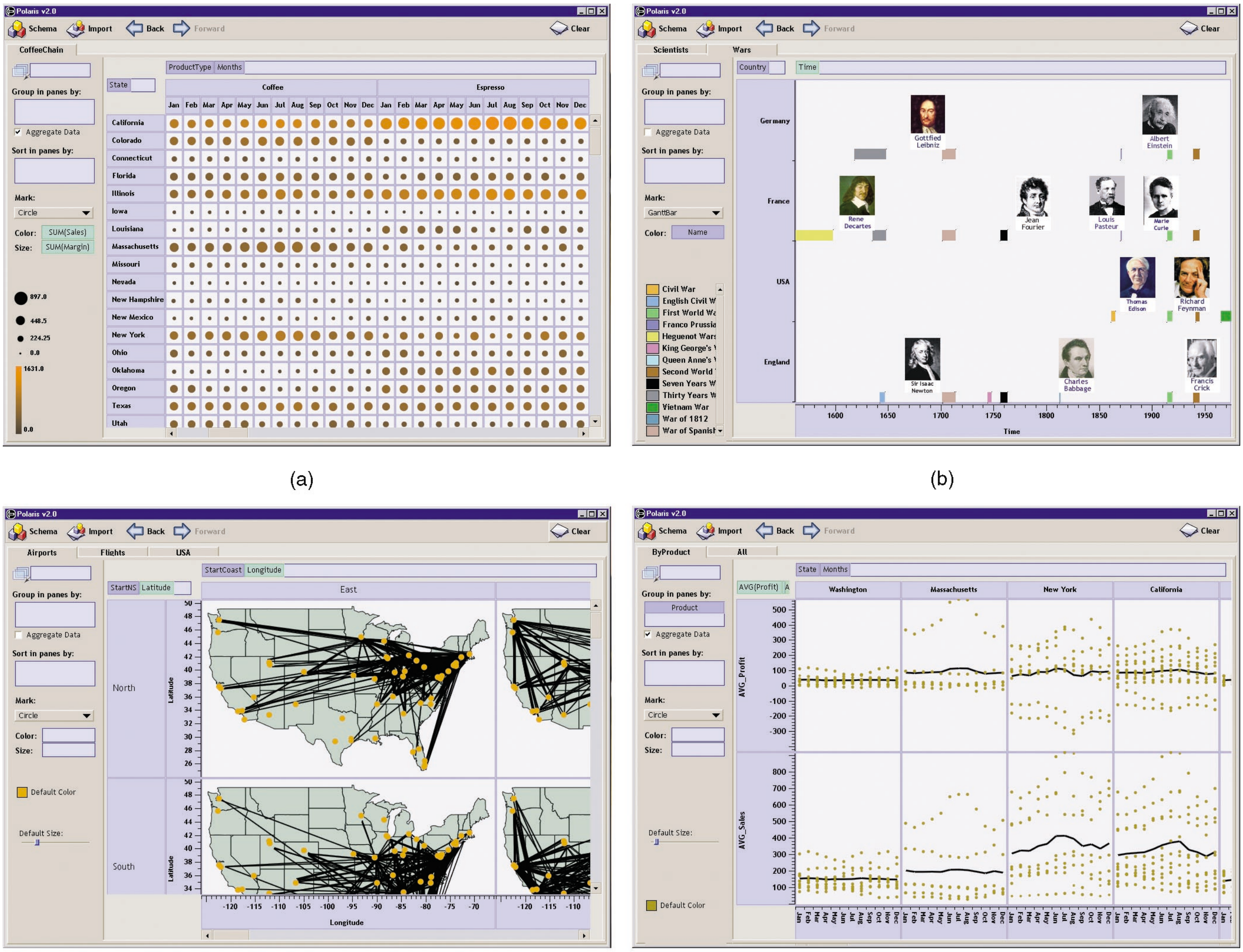
After the table configuration is specified, the next step is to specify the type of graphic in each pane. One option, typical of most charting and reporting tools, is to have the user select a chart type from a predefined set of charts. Polaris allows analysts to flexibly construct graphics by specifying the individual components of the graphics. However, for this approach to be effective, the specification must balance flexibility with succinctness. We have developed a taxonomy of graphics that results in an intuitive and concise specification of graphic types.

When specifying the table configuration, the user also implicitly specifies the axes associated with each pane. We have structured the space of graphics into three families by the type of fields assigned to their axes:

- ordinal-ordinal,
- ordinal-quantitative,
- quantitative-quantitative.

Each family contains a number of variants depending on how records are mapped to marks. For example, selecting a bar in an ordinal-quantitative pane will result in a bar chart, whereas selecting a line mark results in a line chart. The mark set currently supported in Polaris includes the rectangle, circle, glyph, text, Gantt bar, line, polygon, and image.

Following Cleveland [12], we further structure the space of graphics by the number of independent and dependent variables. For example, a graphic where both axes encode independent variables is different than a graphic where one axis encodes an independent variable and the other encodes



(a)

(b)

(c)

(d)

Fig. 3. (a) A standard Pivot Table except with graphical marks rather than textual numbers in each cell, showing the total sales in each state organized by product type and month. By using a graphical mark with the total sales encoded as the size of the circle, trends may be easier to spot than by scanning a table of numbers. (b) Gantt charts showing the correspondence between wars and major scientists, organized by country. Note that this display is using two different data sources, one for the wars and one for the scientists, each corresponding to a different layer. Because both data sources have the country and time dimensions, Polaris can be used to visually join the two data sets into a single display with common *x* and *y* table axes. (c) Maps showing flights across the United States. This display has multiple data sources corresponding to different layers. The "USA" data source contains data corresponding to the underlying state outlines. The "Airport" data source contains the coordinates of numerous airports in the US. Finally, the "Flights" data source describes flights in the continental United States, including data about the region in which the flight originated. When a field that only appears in a single data set (flights) is used to partition a layered display, the data from the other data sources (airports and states) is replicated into each pane formed by the partitioning. Thus, each pane displays all states and airports, but only a subset of the flights. (d) A small-multiple display of line charts overlaying dot plots. Each chart displays the profit and sales over time for a hypothetical coffee chain, orgainized by state. The display is constructed using two layers, where each layer contains a copy of the same data set. The layers share the same table structure, but use different marks and data transformations. As a result, the line chart and dot plot can be at different levels of detail. In this case, each line chart shows average profit across all products per month, whereas the dot plot has an additional grouping transform specified that results in separate dots displaying average profit per product per month.

a dependent variable ($y = f(x)$). By default, dimensions of the database are interpreted as independent variables and measures as dependent variables.

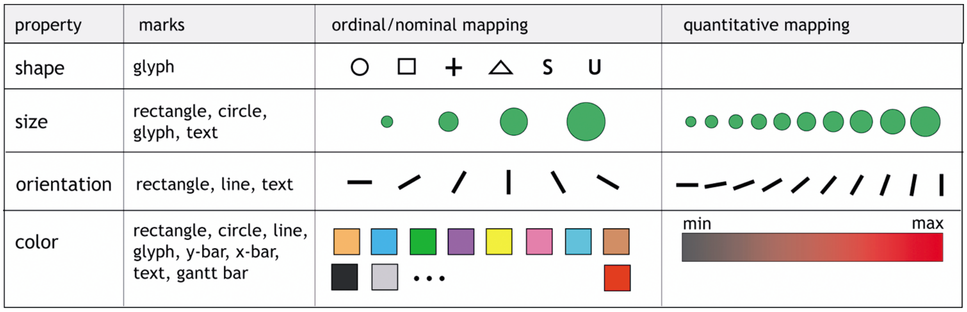
Finally, the precise form of the data transformations, in particular, how records are grouped and whether aggregates are formed, can affect the type of graphic. Some graphic types best encode a single record, whereas others can encode an arbitrary number of records.

We briefly discuss the defining characteristics of the three families within our categorization.

4.2.1 Ordinal-Ordinal Graphics

The characteristic member of this family is the table, either of numbers or of marks encoding attributes of the source records.

In ordinal-ordinal graphics, the axis variables are typically independent of each other and the task is focused on understanding patterns and trends in some function $f(O_x, O_y) \rightarrow R$, where R represents the fields encoded in the retinal properties of the marks. This can be seen in Fig. 3a, where the analyst is studying sales and margin as a function



property	marks	ordinal/nominal mapping	quantitative mapping
shape	glyph	○ □ + △ S U	
size	rectangle, circle, glyph, text	● ● ● ●	● ● ● ● ● ● ● ● ● ●
orientation	rectangle, line, text	— ⁄ / | \ ⟍	— — ⁄ ⁄ ⁄ / / / | |
color	rectangle, circle, line, glyph, y-bar, x-bar, text, gantt bar	🟧 🟦 🟩 🟪 🟨 🟪 🟦 🟫 ⬛ ⬜ ••• 🟥	min max

Fig. 4. The different retinal properties that can be used to encode fields of the data and examples of the default mappings that are generated when a given type of data field is encoded in each of the retinal properties.

of product type, month, and state for the items sold by a hypothetical coffee chain. Fig. 7a presents another example of an ordinal-ordinal graphic. In this figure, the analyst is investigating the performance of a graphics rendering library. The number of cache misses attributable to each line of source code has been encoded in the color of the line.

The cardinality of the record set in each pane has little effect on the overall structure of the table. When there is more than one record per pane, multiple marks are shown in each display, with a one-to-one correspondence of mark to record. The marks are stacked in a specified drawing order and the spatial placement of the marks within the pane conveys no additional information about the record's data.

4.2.2 Ordinal-Quantitative Graphics

Well-known representatives of this family of graphics are the bar chart, possibly clustered or stacked, the dot plot, and the Gantt chart.

In an ordinal-quantitative graphic, the quantitative variable is often dependent on the ordinal variable and the analyst is trying to understand or compare the properties of some set of functions $f(O) \rightarrow Q$. Fig. 6c illustrates a case where a matrix of bar charts is used to study several functions of the independent variables product and month. The cardinality of the record set does affect the structure of the graphics in this family. When the cardinality of the record set is one, the graphics are simple bar charts or dot plots. When the cardinality is greater than one, additional structure may be introduced to accommodate the additional records (e.g., a stacked bar chart).

The ordinal and quantitative values may be independent variables, such as in a Gantt chart. Here, each pane represents all the events in a category; each event has a type as well as a begin and end time. In Fig. 3b, major wars over the last 500 years are displayed as Gantt charts, categorized by country. An additional layer in that figure displays pictures of major scientists plotted as a function of the independent variables country of birth and date of birth. Fig. 7c shows a table of Gantt charts, with each chart displaying the thread scheduling and locking activity on a CPU within a multiprocessor computer. To support Gantt charts, we need to support intervals as an additional type of field that exists in the meta-data only, allowing one field name to map to a pair of columns in the database.

4.2.3 Quantitative-Quantitative Graphics

Graphics of this type are used to understand the distribution of data as a function of one or both quantitative variables and to discover causal relationships between the two quantitative variables. Fig. 6a illustrates a matrix of scatterplot graphics used to understand the relationships between a number of attributes of different products sold by a coffee chain.

Fig. 3c illustrates another example of a quantitative-quantitative graphic: the map. In this figure, the analyst is studying how flight scheduling varies with the region of the country in which the flight originated. Data about a number of flights between major airports has been plotted as a function of latitude and longitude; this data has been composed with two other layers that plot the location of airports and display the geography of each state as a polygon.

It is extremely rare to use this type of graph with a cardinality of one, not because it is not meaningful, but because the density of information in such a graphic is very low.

4.3 Visual Mappings

Each record in a pane is mapped to a mark. There are two components to the visual mapping. The first component, described in the previous section, determines the type of graphic and mark. The second component encodes fields of the records into visual or retinal properties of the selected mark. The visual properties in Polaris are based on Bertin's retinal variables [6]: shape, size, orientation, color (value and hue), and texture (not supported in the current version of Polaris).

Allowing analysts to explicitly encode different fields of the data to retinal properties of the display greatly enhances the data density and the variety of displays that can be generated. However, in order to keep the specification succinct, analysts should not be required to construct the mappings. Instead, they should be able to simply specify that a field be encoded as a visual property. The system should then generate an effective mapping from the domain of the field to the range of the visual property. These mappings are generated in a manner similar to other visualization systems. We discuss how this is done for the purpose of completeness. The default mappings are illustrated in Fig. 4.

Shape: Polaris uses the set of shapes recommended by Cleveland for encoding ordinal data [11]. We have extended this set of shapes to include several additional shapes to allow a larger domain of values to be encoded.

Size: Analysts can use size to encode either an ordinal or quantitative field. When encoding a quantitative domain as size, a linear map from the field domain to the area of the mark is created. The minimum size is chosen so that all visual properties of a mark with the minimum size can be perceived [23]. If an ordinal field is encoded as size, the domain needs to be small, at most four or five values, so that the analyst can discriminate between different categories [6].

Orientation: A key principle in generating mappings of ordinal fields to orientation is that the orientation needs to vary by at least 30° between categories [23], thus constraining the automatically generated mapping to a domain of at most six categories. For quantitative fields, the orientation varies linearly with the domain of the field.

Color: When encoding an ordinal domain, we use a predefined palette to select the color for each domain entry. The colors in the palette are well separated in the color spectrum, predominantly on hue [37]. We have ordered the colors to avoid adjacent colors with different brightness or substantially different wavelengths in an attempt to include harmonious sets of colors in each palette [6], [23], [37]. We additionally reserve a saturated red for highlighting items that have been selected or brushed.

When encoding a quantitative variable, it is important to vary only one psychophysical variable, such as hue or value. The default palette we use for encoding quantitative data is the isomorphic colormap developed by Rogowitz and Treinish [28].

5 Data Transformations and Visual Queries

The ability to rapidly change the table configuration, type of graphic, and visual encodings used to visualize a data set is necessary for interactive exploration. However, it is not sufficient; additional interactivity is needed. The resulting display must be manipulable. The analysts must be able to sort, filter, and transform the data to uncover useful relationships and information and then they must be able to form ad hoc groupings and partitions that reflect this newly uncovered information [5].

In this section, we describe four interaction techniques Polaris provides to support analysis within the resulting visualizations: deriving additional fields, sorting and filtering, brushing and tooltips, and undo and redo.

5.1 Deriving Additional Fields

While analyzing data, one of the most important interactions needed is the ability to derive additional fields from the original data. Typically, these generated fields are aggregates or statistical summaries. Polaris currently provides five methods for deriving additional fields: simple aggregation of quantitative measures, counting of distinct values in ordinal dimensions, discrete partitioning of quantitative measures, ad hoc grouping within ordinal dimensions, and threshold aggregation.

Simple Aggregation refers to basic aggregation operations, such as summation, average, minimum, and maximum, that are applied to a single quantitative field. A default aggregation operation is applied to a quantitative measure when it is dragged to one of the x or y-axis (or layer); shelves and aggregation is enabled. The user can change which aggregation function is applied by right-clicking on the field and choosing a different aggregation function from the popup context menu. Polaris can be easily extended to provide any statistical aggregate that can be generated from relational data.

Counting of Ordinal Dimensions refers to the counting of distinct values for an ordinal field within the data set. This aggregation function can be applied to an ordinal field by right-clicking on the field and choosing the CNT (count) aggregation function. Unlike simple aggregation, applying the count operator will change the field type (to quantitative) and thus change the table configuration and graph type in each pane.

Discrete Partitioning is used to discretize a continuous domain. Polaris provides two discretization methods: binning and partitioning. Binning allows the analyst to specify a regular bin size in which to aggregate the data; binning will not change the graph type since the resulting derived field is also quantitative, just at the specified granularity. Partitioning allows the user to individually specify the size and name of each bin. Partitioning of a quantitative field will result in an ordinal field, thus changing the graph types and table configuration. Binning is useful for creating graphs, such as histograms, in which there are many regularly sized bins, while partitioning is useful for encoding additional categorizations into the data, either ad hoc or derived from known domain information. Both can be applied by right-clicking on the field name and choosing either the "Bin by..." or "Partition..." option.

Ad hoc Grouping is the ordinal version of quantitative partitioning, where the user can choose to group together different ordinal values for the purpose of display and data transformations. For example, a user may choose to group California and Florida together into an "Orange provider" partition. This type of arbitrary grouping and aggregation is powerful since it allows the analyst to add his own domain knowledge to the analysis and to change the groupings as the exploration uncovers additional patterns. The user can apply ad hoc grouping by right-clicking on the field name and choosing the "Partition..." option. This transformation derives an ordinal field from an ordinal field and, thus, the graph type does not change.

Threshold Aggregation is the last type of derived field we support and it differs from the rest since it is derived from two source fields: an ordinal field and a quantitative field. If the quantitative field is less than a certain threshold value for any values of an ordinal field, those values are aggregated together to form an "Other" category. This allows the user to specify threshold values below which the data is considered uninteresting. One challenge in supporting threshold aggregation is that it can require two aggregation passes if the quantitative field desired is itself a derived field (e.g., the average of the quantitative profit field).

Adding derived fields on the fly is necessary as part of the exploration and analysis process. As the analyst explores the data, relationships within the data are discovered and groupings can be used to encode and reflect this discovered information. Furthermore, aggregations and statistics can be used to hide uninteresting data and to reduce large data sets in size so that they are tractable for understanding and analysis.

5.2 Sorting and Filtering

Sorting and filtering data play a key role in analysis. It is often only by reordering rows and columns and filtering out unimportant information that patterns and trends in the data emerge. Polaris provides extensive support for both of these analysis techniques.

Filtering allows the user to choose which values to display so that he can focus on and devote more screen space and attention to the areas of interest. For all fields, the user can right-click on the field name to bring up a menu and choose the "Filter..." option.

For ordinal fields, a listbox with all possible values is shown and the user can check or uncheck each value to display it or not. For quantitative fields, a dynamic query slider [1] allows the user to choose a new domain. Additionally, there are textboxes showing the chosen minimum and maximum values that the user can use to directly enter a new domain. In using Polaris, we discovered that we needed to provide a slightly larger domain than the actual data domain for the user to select from since the user may want some buffer space in the graphs.

Note that applying a filter changes the interpretation of the field in the algebra. For ordinal fields, it reduces the domain to just the filtered values, rather than all values. It does not change how a quantitative field is interpreted in the table algebra.

Sorting allows the user to to uncover hidden patterns and trends and to group together interesting values by changing the order of values within a field's domain or the ordering of tuples in the data. Changes to the ordering of values within a field's domain can affect the ordering of the panes within a table, the ordering of values along an axis (such as in a bar chart), and the composite ordering of layers. The ordering of tuples affects the drawing order of marks within a pane. The drawing order is most relevant in graphs where a single primitive encodes multiple tuples, such as a line or polygon primitive, or where marks overlap and the drawing order thus determines the front-to-back ordering and occlusion of marks.

Polaris provides three ways for a user to sort the domain. First, the user can bring up the filter window and drag-and-drop the values within that window to reorder the domain. Second, if the field has been used to partition the table into rows or columns, the user can drag-and-drop the table row or column headers to reorder the domain values. Finally, Polaris provides programmatic sorting, allowing the user to sort one field based on the value in another field. For example, the user may want to sort the State field by the Profit field. The ordering of tuples within a pane is determined by which fields the analyst has placed in the Sort shelf.

5.3 Brushing and Tooltips

Many times, when exploring the data, analysts want to directly interact with the data, visually querying the data to highlight correlated marks or getting more details on demand. Polaris provides both brushing and tooltips for this purpose.

Brushing allows the user to choose a set of interesting data points by drawing a rubberband around them. The user selects a single field whose values are then used to identify related marks and tuples. All marks corresponding to tuples sharing selected field values with the selected tuples are subsequently highlighted in all other panes or linked Polaris views. Brushing allows the analyst to choose data in one display and highlight it in other displays, allowing correlation between different projections of the same data set or relationships between distinct data sets.

Tooltips allow the user to get more details on demand. If the user hovers over a data point or pane, additional details, such as specific field values for the tuple corresponding to the selected mark, are shown. Analysts can use tooltips to understand the relationship between the graphical marks and the underlying data.

5.4 Undo and Redo

The final interaction technique we provide in Polaris is unlimited undo and redo within an analysis session. Users can use the "Back" and "Forward" buttons on the top toolbar to either return to a previous visual specification or to move forward again. This functionality is critical for interactive exploration since it allows the user to quickly back out of changes (or redo them if he goes too far back) and try a different exploration path without having to rebuild the desired starting point from scratch. Support for undo also promotes more experimentation and exploration as there is no fear of losing work done thus far. If the user does want a clean canvas, Polaris also provides a "Clear" button.

By using the formalism and visual specification, implementing Undo and Redo is trivial. We simply need to save the visual specification at each stage and moving backward or forward is done by simply updating the display to reflect the saved visual specification.

6 Generating Database Queries

In addition to determining the appearance of the visualization, the visual specification also generates queries to the database that 1) select subsets of the data for analysis, then 2) filter, sort, and group the results into panes, and then, finally, 3) group, sort, and aggregate the data before passing it to the graphics encoding process.

Fig. 5 shows the overall data flow in Polaris. We can precisely describe the transformations in each of the three phases using SQL queries.

6.1 Step 1: Selecting the Records

The first phase of the data flow retrieves records from the database, applying user-defined filters to select subsets of the database.

For an ordinal field *A*, the user may specify a subset of the domain of the field as valid. If *filter(A)* is the user-selected

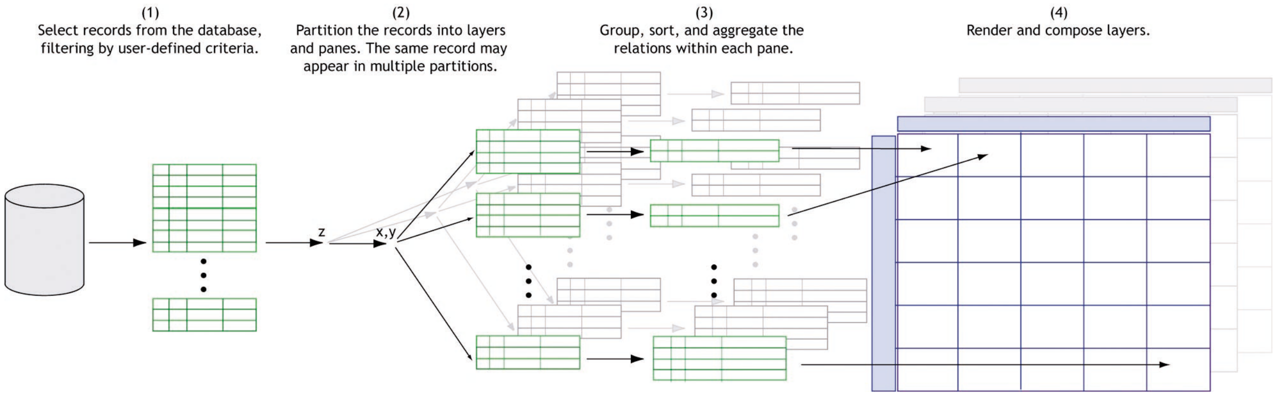

(1)
Select records from the database, filtering by user-defined criteria.
(2)
Partition the records into layers and panes. The same record may appear in multiple partitions.
(3)
Group, sort, and aggregate the relations within each pane.
(4)
Render and compose layers.
z
x,y

Fig. 5. The transformations and data flow within Polaris. The visual specification generates queries to the database to select subsets of the data for analysis, then to filter, sort, and group the results into panes, and then, finally, to group, sort and aggregate the data within panes.

subset, then a relational predicate expressing the filter for *A* is:

$$\texttt{A in filter (A)}.$$

For a quantitative field *P*, the user may define a subset of the field's domain as valid. If *min(P)* and *max(P)* are the user-defined extents of this subset, then a relational predicate expressing the filter for *P* is:

$$(\texttt{P} \geq \texttt{min (P) and P} \leq \texttt{max (P)}).$$

We can define the relational predicate *filters* as the conjunction of all of the individual field filters. Then, the first stage of the data transformation network is equivalent to the SQL statement:

```
SELECT *
WHERE {filters}.
```

6.2 Step 2: Partitioning the Records into Panes

The second phase of the data flow partitions the retrieved records into groups corresponding to each pane in the table. As we discussed in Section 4.1, the *normalized set form* of the table axis expressions determines the table configuration. The table is partitioned into rows, columns, and layers corresponding to the entries in these sets.

The ordinal values in each set entry define the criteria by which records will be sorted into each row, column, and layer. Let *Row(i)* be the predicate that represents the selection criteria for the *i*th row, *Column(j)* be the predicate for the *j*th column, and *Layer(k)* the predicate for the *k*th layer. For example, if the *y*-axis of the table is defined by the normalized set:

$$\{\mathtt{a_1b_1P,\ a_1b_2P,\ a_2b_1P,\ a_2b_2P}\},$$

then there are four rows in the table, each defined by an entry in this set, and *Row* would be defined as:

$$\begin{aligned}
\texttt{Row(1)} &= (\texttt{A} = \mathtt{a_1} \texttt{ and } \texttt{B} = \mathtt{b_1})\\
\texttt{Row(2)} &= (\texttt{A} = \mathtt{a_1} \texttt{ and } \texttt{B} = \mathtt{b_2})\\
\texttt{Row(3)} &= (\texttt{A} = \mathtt{a_2} \texttt{ and } \texttt{B} = \mathtt{b_1})\\
\texttt{Row(4)} &= (\texttt{A} = \mathtt{a_2} \texttt{ and } \texttt{B} = \mathtt{b_2}).
\end{aligned}$$

Given these definitions, the records to be partitioned into the pane at the intersection of the *i*th row, the *j*th column, and the *k*th layer can be retrieved with the following query:

```
SELECT *
WHERE {Row(i) and Column(j) and
          Layer(k)}.
```

To generate the groups of records corresponding to each of the panes, we must iterate over the table, executing this SELECT statement for each pane. There is no standard SQL statement that enables us to perform this partitioning in a single query. We note that this same problem motivated the CUBE [16] operator; we will revisit this issue in the discussion section.

6.3 Step 3: Transforming Records within the Panes

The last phase of the data flow is the transformation of the records in each pane. If the visual specification includes aggregation, then each measure in the database schema must be assigned an aggregation operator. If the user has not selected an aggregation operator for a measure, that measure is assigned the default aggregation operator (SUM). We define the term *aggregates* as the list of the aggregations that need to be computed. For example, if the database contains the quantitative fields Profit, Sales, and Payroll and the user has explicitly specified that the average of Sales should be computed, then *aggregates* is defined as:

```
aggregates *
SUM(Profit), AVG(Sales), SUM(Payroll).
```

Aggregate field filters (for example, SUM(Profit) > 500) could not be evaluated in Step 1 with all of the other filters because the aggregates had not yet been computed. Thus, those filters must be applied in this phase. We define the relational predicate *filters* as in Step 1 for unaggregated fields.

Additionally, we define the following lists:

G: the field names in the grouping shelf,
S: the field names in the sorting shelf, and
dim: the dimensions in the database.

The necessary transformation can then be expressed by the SQL statement:

```
SELECT {dim}, {aggregates}
GROUP BY {G}
HAVING {filters}
ORDER BY {S}.
```

If no aggregate fields are included in the visual specification, then the remaining transformation simply sorts the records into drawing order:

```
SELECT *
ORDER BY {S}.
```

7 Results

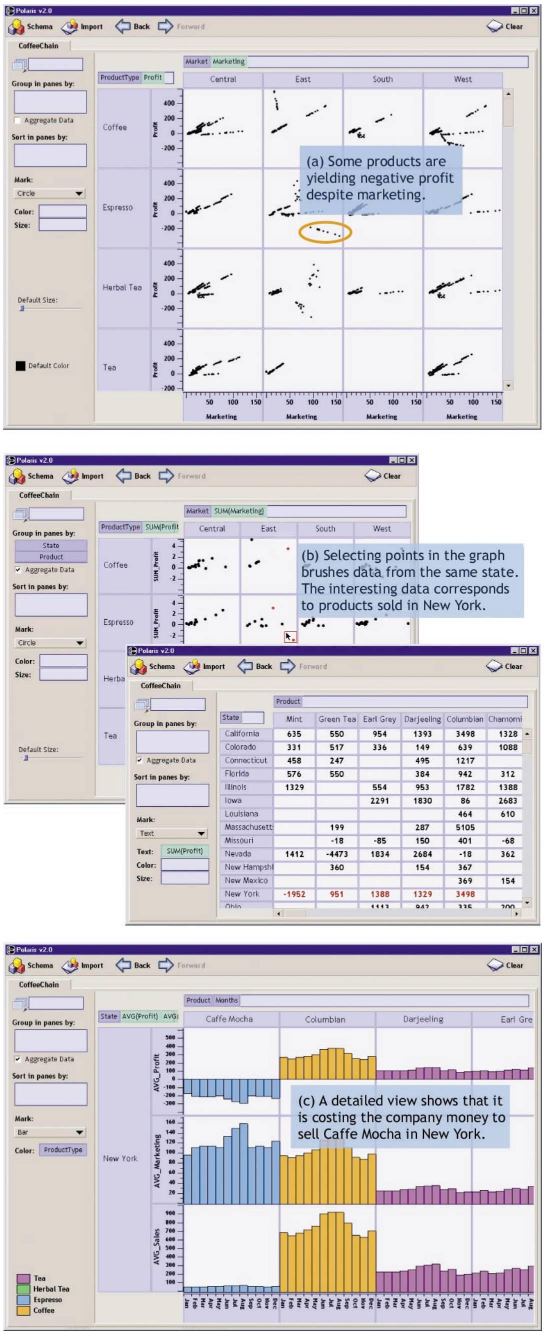
Polaris is useful for performing the type of exploratory data analysis advocated by statisticians such as Bertin [5] and Cleveland [12]. We demonstrate the capabilities of Polaris as an exploratory interface to multidimensional databases by considering the following two scenarios, performed using a Polaris prototype implemented in the Rivet visualization environment [9].

7.1 Scenario 1: Commercial Database Analysis

The chief financial officer (CFO) of a national coffee store chain has just been told to cut expenses. To get an initial understanding of the situation, the CFO creates a table of scatterplots showing the relationship between marketing costs and profit categorized by product type and market (Fig. 6a). After studying the graphics, the CFO notices an interesting trend: Certain products have high marketing costs with little or no return in the form of profit.

To further investigate, the CFO creates two linked displays: a table of scatterplots showing profit and marketing costs for each state and a text table itemizing profit by product and state (Fig. 6b). The two views are linked by the state field: If records are selected in either display, then all records with the same state value as the selected records are highlighted. The CFO is able to use these linked views to determine that, in New York, several products are offering very little return despite high expenditures.

The CFO then creates a third display (Fig. 6c): a set of bar charts showing profit, sales, and marketing for each product sold in New York, broken down by month. In this view, the CFO can clearly see that Caffé Mocha's profit margin does not justify its marketing expenses. With this data, the CFO can change the company's marketing and sales strategies in this state.



(a) Some products are yielding negative profit despite marketing.

(b) Selecting points in the graph brushes data from the same state. The interesting data corresponds to products sold in New York.

State	Mint	Green Tea	Earl Grey	Darjeeling	Columbian	Chamomi
California	635	550	954	1393	3498	1328
Colorado	331	517	336	149	639	1088
Connecticut	458	247		495	1217	
Florida	576	550		384	942	312
Illinois	1329		554	953	1782	1388
Iowa			2291	1830	86	2683
Louisiana					464	610
Massachusett		199		287	5105	
Missouri		-18	-85	150	401	-68
Nevada	1412	-4473	1834	2684	-18	362
New Hampshi		360		154	367	
New Mexico					369	154
New York	-1952	951	1388	1329	3498	
Ohio			1113	942	335	200

(c) A detailed view shows that it is costing the company money to sell Caffe Mocha in New York.

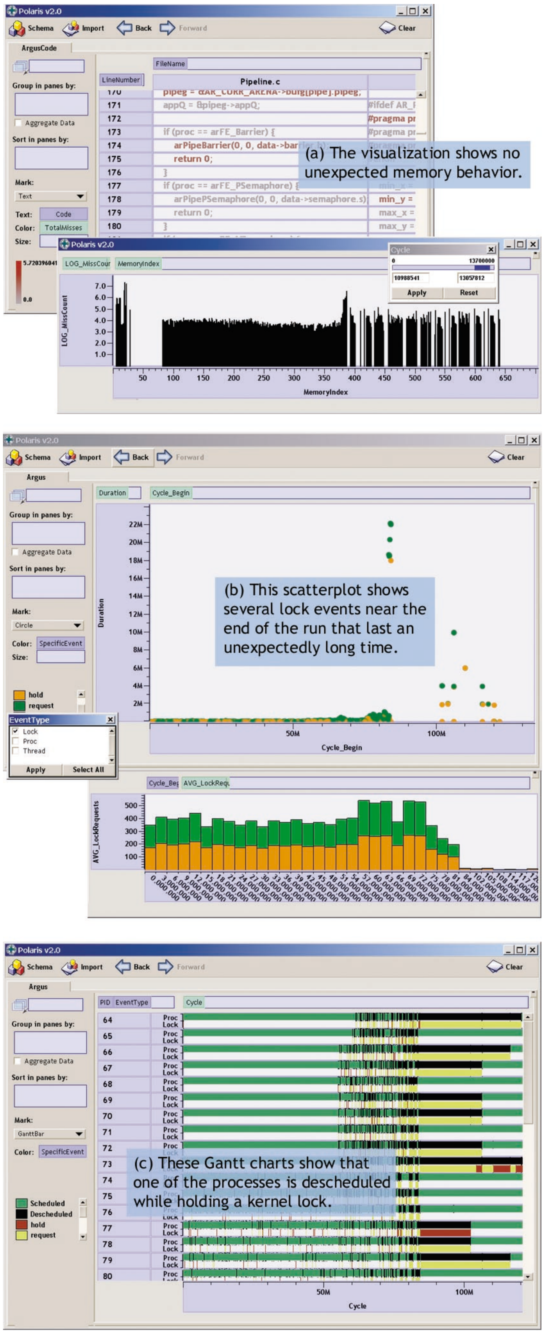
Fig. 6. An example scenario demonstrating the capabilities of Polaris for exploratory analysis of multidimensional databases. The data displayed is for a hypothetical coffee store chain and the analyst is searching for ways to reduce the company's expenses.

7.2 Scenario 2: Computer Systems Analysis

At Stanford, researchers developing Argus [21], a parallel graphics library, found that its performance had linear speedup when using up to 31 processors, after which its performance diminished rapidly. Using Polaris, we recreate the analysis they performed using a custom-built visualization tool [8].

Initially, the developers hypothesized that the diminishing performance was a result of too many remote memory accesses, a common performance problem in parallel programs. They collected and visualized detailed memory statistics to test this hypothesis. Fig. 7a shows a visualization constructed to display this data. The visualization is composed of two linked Polaris instances. One displays a histogram of cache misses by virtual address, the other displays source-code, with each line's hue encoding the number of cache misses suffered by that line. Upon seeing these displays, they could tell that memory was not in fact the problem.

The developers next hypothesized that lock contention might be a problem, so they reran Argus and collected detailed lock and scheduling information. The data is shown in Fig. 7b using two instances of Polaris to create a composite visualization with two linked projections of the



```
pipeg = &AR_CORR_ARENA->bufg[pipe].pipeg;
appQ = &pipeg->appQ;

if (proc == arFE_Barrier) {
    arPipeBarrier(0, 0, data->barrier.b);
    return 0;
}
if (proc == arFE_PSemaphore) {
    arPipePSemaphore(0, 0, data->semaphore.s);
    return 0;
}
```

(a) The visualization shows no unexpected memory behavior.

(b) This scatterplot shows several lock events near the end of the run that last an unexpectedly long time.

(c) These Gantt charts show that one of the processes is descheduled while holding a kernel lock.

Fig. 7. A scenario demonstrating how researchers use Polaris to analyze the performance problems of a parallel graphics library.

same data. One projection shows a scatterplot of the start cycle versus cycle duration for the lock events (requests and holds). The second shows a histogram over time of initiated lock events. The scatterplot shows that toward the end of the run, the duration of lock events (both holds and requests) were taking an unexpectedly long time. That observation correlated with the histogram showing that the number of lock requests peaked and then tailed off toward the end of the run, indicating that this might be a fruitful area for further investigation.

A third visualization, shown in Fig. 7c, shows the same data using Gantt charts to display both lock events and process scheduling events. This display shows that the long lock requests correspond to descheduled periods for most processes. One process, however, has a descheduled period corresponding to a period during which the lock was held. This behavior, which was due to a bug in the operating system, was the source of the performance issues.

7.3 Summary

These two examples illustrate several important points about the exploratory process. Throughout the analyses, both what the data users want to see and how they want to see it change continually. Analysts first form hypotheses about the data and then create new views to perform tests and experiments to validate or disprove those hypotheses. Certain displays enable an understanding of overall trends, whereas others show causal relationships. As the analysts better understand the data, they may want to drill-down in the visible dimensions or display entirely different dimensions.

Polaris supports this exploratory process through its visual interface. By formally categorizing the types of graphics, Polaris is able to provide a simple interface for rapidly generating a wide range of displays. This allows analysts to focus on the analysis task rather than the steps needed to retrieve and display the data.

8 Discussion

In this section, we focus on three points of discussion. First, we discuss how our work compares to that of Wilkinson [41], second, the interpretation of our visual specifications as database queries, and, finally, the interactivity and performance of Polaris.

Several of the ideas in our specification are extensions of Wilkinson's [41] efforts to develop a grammar for statistical graphics. His grammar encapsulates both the statistical transformation of datasets and their mapping to graphic representations.

The primary distinctions between Wilkinson's system and ours arise because of differences in the data models. We chose to focus on developing a tool for multidimensional relational databases and we decided to build as much of the system as possible using relational algebra. All of the data transformations required by our visual specifications can be precisely interpreted as standard SQL queries to OLAP servers. Wilkinson instead intentionally uses a data model that is not relational, citing shortcomings in the relational model's support for statistical analysis. Consequently, his specification defines operations and function in terms of his own data model, consisting of variable sets and indexed variables.

The differences in design are most apparent in the table algebra. As in our system, Wilkinson's table algebra performs two functions: it provides database services such as set operations and it specifies the layout of the tables and graphs. Since we use relational algebra for all our database services, our algebra is different. For example, his blend operator performs both set union and may partition the axes of a table; our concatenation operation is different since it just performs partitioning. Another difference is in his cross and nest operators: cross generates a 2D graphic and nest only a 1D graphic. We use a different mechanism (shelves) to specify axes of the graphic. Overall, whether our system is better than Wilkinson's is hard to judge completely, and will require more experience using the systems to solve practical problems. One major advantage of our approach is that it leverages existing database systems and as a result was very easy to implement.

Another interesting issue is the interpretation of our visual specifications as database queries. When database queries are generated from the visual specifications in Polaris, it is necessary to generate a SQL query per table pane. This problem is similar to the one that motivated Gray et al. to develop the CUBE operator [16]. The CUBE operator generalizes the queries necessary to develop cross-tab and Pivot Table displays of relational data into a single, more efficient operator. However, the CUBE operator cannot be applied in our situation because it assumes that the sets of relations partitioned into each table pane do not overlap. In several possible Polaris table configurations, such as scatterplot matrices, there can be considerable overlap between the relations partitioned into each pane. One can imagine generalizing the CUBE operator to handle these overlapping partitions.

Another major limitation of the CUBE operator is its method for computing aggregates. Usually, only aggregates based on sums are allowed. More complex aggregation operators requiring ranking, such as computation of medians and modes, are not part of the current specification, although they are available in some commercial systems. These operators are very useful for data mining applications.

A final point of discussion is the interactivity and performance of Polaris. Although Polaris is designed to be an interactive and exploratory interface to large data warehouses, our research has focused on the techniques, semantics, and formalism needed to provide an effective exploratory interface rather than on attaining interactive query times. While we would like the system to be reasonably responsive as the user modifies the visual specification, our experience has been that the query response time does not need to be real-time in order to maintain a feeling of exploration: The query can even take several tens of seconds. Within this constraint, Polaris can currently be used with many large databases, especially if a large subset of the views can be materialized a priori [39]. Furthermore, it is important to note that many queries on data warehouses, such as those generated with existing Pivot Table tools, will be returning a small number of tuples and, thus, the most relevant constraint on performance is server-side query time and not client-side drawing or data manipulation.

We have used Polaris with two reasonably large data sets: 1) a subset of a packet trace of a mobile network over a 13 week period [35] that has over 6 million tuples and 2) a subset of the data collected from the Sloan Digital Sky Survey (approx. 650 MB), both stored in Microsoft's SQLServer. With both data sets, we are able to get reasonably responsive performance and maintain a sense of exploration for the queries we ran. We intend to pursue database performance issues in order to scale to much larger data sets as part of our future research. There are many techniques that can be used to improve the performance of the queries, including existing techniques such as materialized views [39], progressively refined queries that provide intermediate feedback [19], and sampled queries [17], [24]. We also think that substantial performance benefits can be gained by leveraging the coherence between successive queries generated by visualization systems using both caching and prefetching.

9 Conclusions and Future Work

We have presented Polaris, an interface for the exploration and analysis of large multidimensional databases. Polaris extends the well-known Pivot Table interface to display relational query results using a rich, expressive set of graphical displays. A second contribution of this system is a succinct visual specification for describing table-based graphical displays of relational data and the interpretation of these visual specifications as a precise sequence of relational database operations.

We have many plans for future work in extending this system. As stated above, one area of future work is exploring database performance issues. A related area is expanding Polaris to expose the hierarchical structure of data cubes. Each level of the hierarchy can be thought of as a different level of detail. One idea, a la Pad++ [4], is to change the visual representation as we change the level of detail; the visual representation will be determined in part by the available screen space. In order to make this type of system interactive, especially given the large quantities of data involved, we are also exploring prefetching and caching strategies to achieve real-time interactivity.

Another area of future work is to leverage the direct correspondance of graphical marks in Polaris to tuples in the relational databases in order to generate database tables from a selected set of graphical marks. This technique can be used to develop lenses, similar to the Magic Lens [7], that can perform much more complex transformations because they operate in data space rather than image space. This technique can also be used to compose Polaris displays, using a selected mark set in one display as the data input to another. We are exploring these techniques and believe it is possible to develop a relational spreadsheet by composing Polaris displays in this manner.

Extending the x, y, and layer shelves to include an animation shelf would enable analysts to partition the data on those fields and create animated displays that sequence through the data. For example, in the coffee chain data set, dropping the Month field on the animation shelf would create an animation showing how the data changes over time.

Acknowledgments

The authors especially thank Robert Bosch for his contribution to the design of Polaris, his review of manuscript drafts, and for many useful discussions. The authors also thank Maneesh Agrawala for his reviews and discussions on early drafts of this paper. The air traffic data is courtesy of William F. Eddy and Shingo Oue. The coffee chain data is courtesy of Stephen Eick and Visual Insights. This work was supported by the US Department of Energy through the ASCI Level 1 Alliance with Stanford University.

References

[1] C. Ahlberg, C. Williamson, and B. Shneiderman, "Dynamic Queries for Information Exploration: An Implementation and Evaluation," *Proc. SIGCHI 1992,* p. 619, 1992.

[2] A. Aiken, J. Chen, M. Stonebraker, and A. Woodruff, "Tioga-2: A Direct Manipulation Database Visualization Environment," *Proc. 12th Int'l Conf. Data Eng.,* pp. 208-217, Feb. 1996.

[3] R. Becker, W.S. Cleveland, and R.D. Martin, "Trellis Graphics Displays: A Multi-Dimensional Data Visualization Tool for Data Mining," *Proc. Third Ann. Conf. Knowledge Discovery in Databases,* Aug. 1997.

[4] B.B. Bederson, L. Stead, and J.D. Hollan, "Pad++: Advances in Multiscale Interfaces," *Proc. SIGCHI,* 1994.

[5] J. Bertin, *Graphics and Graphic Information Processing.* Berlin: Walter de Gruyter, 1980.

[6] J. Bertin, *Semiology of Graphics.* Madison, Wis.: Univ. of Wisconsin Press, 1983. Translated by W.J. Berg.

[7] E.A. Bier, M. Stone, K. Pier, W. Buxton, and T. DeRose, "Toolglass and Magic Lenses: The See-Through Interface," *SIGGRAPH '93 Proc.,* pp. 73-80, Aug. 1993.

[8] R. Bosch, C. Stolte, G. Stoll, M. Rosenblum, and P. Hanrahan, "Performance Analysis and Visualization of Parallel Systems Using SimOS and Rivet: A Case Study," *Proc. Sixth Ann. Symp. High-Performance Computer Architecture,* p. 360-371, Jan. 2000.

[9] R. Bosch, C. Stolte, D. Tang, J. Gerth, M. Rosenblum, and P. Hanrahan, "Rivet: A Flexible Environment for Computer Systems Visualization," *Computer Graphics,* pp. 68-73, Feb. 2000.

[10] A. Buja, D. Cook, and D.F. Swayne, "Interactive High-Dimensional Data Visualization," *J. Computational and Graphical Statistics,* vol. 5, no. 1, pp. 78-99, 1996.

[11] W.S. Cleveland, *The Elements of Graphing Data.* Pacific Grove, Calif.: Wadsworth Advanced Books and Software, 1985.

[12] W.S. Cleveland, *Visualizing Data.* Hobart Press, 1993.

[13] M. Derthick, J. Kolojejchick, and S.F. Roth, "An Interactive Visualization Environment for Data Exploration," *Proc. Knowledge Discovery in Databases,* pp. 2-9, Aug. 1997.

[14] S. Eick, "Visualizing Multi-Dimensional Data," *Computer Graphics,* pp. 61-67, Feb. 2000.

[15] *Microsoft Excel—User's Guide.* Redmond, Wash.: Microsoft, 1995.

[16] J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, H. Pirahesh, and F. Pellow, "Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals," *Proc. 12th Int'l Conf. Data Eng.,* pp. 152-159, Feb. 1996.

[17] P.B. Gibbons, Y. Matias, and V. Poosala, "Aqua Project White Paper," technical report, Bell Laboratories, Murray Hill, N.J., Dec. 1997.

[18] J.A. Hartigan, "Printer Graphics for Clustering," *J. Statistical Computation and Simulation,* vol. 4, pp. 187-213, year?

[19] J.M. Hellerstein, P.J. Haas, and H.J. Wang, "Online Aggregation," *Proc. ACM SIGMOD,* pp. 171-82, June 1997.

[20] "Human Genome Project,"available at http://www.ornl.gov/hgmis/project/about.html, Sept. 2001.

[21] H. Igehy, G. Stoll, and P. Hanrahan, "The Design of a Parallel Graphics Interface," *Proc. SIGGRAPH 1998,* pp. 141-150, 1998.

[22] D. Keim and H.-P. Kriegel, "VisDB: Database Exploration Using Multidimensional Visualization," *IEEE Computer Graphics and Applications,* vol. 14, no. 5, pp. 40-49, 1994.

[23] S.M. Kosslyn, *Elements of Graph Design.* New York: W.H. Freeman and Co., 1994.

[24] R.J. Lipton, J.F. Naughton, D.A. Schneider, and S. Seshadri, "Efficient Sampling Strategies for Relational Database Operations," *Theoretical Computer Science,* vol. 116, nos. 1-2, pp. 195-226, 1993.

[25] M. Livny, R. Ramakrishnan, K. Beyer, G. Chen, D. Donjerkovic, S. Lawande, J. Myllymaki, and K. Wenger, "DEVise: Integrated Querying and Visual Exploration of Large Datasets," *Proc. ACM SIGMOD,* May 1997.

[26] J.D. Mackinlay, "Automating the Design of Graphical Presentations of Relational Information," *ACM Trans. Graphics,* pp. 110-141, Apr. 1986.

[27] R. Rao and S. Card, "The Table Lens: Merging Graphical and Symbolic Representations in an Interactive Focus+Context Visualization for Tabular Information," *Proc. SIGCHI '94,* pp. 318-322, 1994.

[28] B. Rogowitz and L. Treinish, "How NOT to Lie with Visualization," *Computers in Physics,* pp. 268-274, May/June 1996.

[29] S.F. Roth, J. Kolojejchick, J. Mattis, and J. Goldstein, "Interactive Graphic Design Using Automatic Presentation Knowledge," *Proc. SIGCHI '94.* pp. 112-117, Apr. 1994.

[30] S.F. Roth, P. Lucas, J.A. Senn, C.C. Gomberg, M.B. Burks, P.J. Stroffolino, J. Kolojejchick, and C. Dunmire, "Visage: A User Interface Environment for Exploring Information," *Proc. Information Visualization,* pp. 3-12, Oct. 1996.

[31] "Sloan Digital Sky Survey,"available: http://www.sdss.org/, Sept. 2001.

[32] M. Spenke, C. Beilken, and T. Berlage, "FOCUS: The Interactive Table for Product Comparison and Selection," *Proc. ACM Symp. User Interface Software and Technology,* Nov. 1996.

[33] Spotfire Inc., available: http://www.spotfire.com, Sept. 2001.

[34] S.S. Stevens, "On the Theory of Scales of Measurement," *Science,* vol. 103, pp. 677-680, year?

[35] D. Tang and M. Baker, "Analysis of a Local-Area Wireless Network," *Mobicom,* 2000.

[36] E. Thomsen, *OLAP Solutions: Building Multidimensional Information Systems.* New York: Wiley Computer Publishing, 1997.

[37] D. Travis, *Effective Color Displays: Theory and Practice.* London: Academic Press, 1991.

[38] E.R. Tufte, *The Visual Display of Quantitative Information.* Chesire, Conn.: Graphics Press, 1983.

[39] J. Ullman, "Efficient Implementation of Data Cubes via Materialized Views," *Proc. Knowledge Discovery in Databases,* 1996.

[40] M. Ward, "XmdvTool: Integrating Multiple Methods for Visualizing Multivariate Data," *Proc. Visualization,* pp. 326-331, Oct. 1994.

[41] L. Wilkinson, *The Grammar of Graphics.* New York: Springer, 1999.

Chris Stolte received the BSc degree in computer science in 1996 from Simon Fraser University. He is currently a PhD student in the Department of Computer Science at Stanford University. His research interests include information visualization and computer graphics. His current research projects include the Rivet project on visualizations for studying computer systems and the Polaris project on visual interfaces for the exploration and analysis of large relational databases.

Diane Tang received the AB degree in computer science in 1995 from Harvard/Radcliffe University and the MS and PhD degrees in computer science in 2000 from Stanford University. She is a research associate in the Department of Computer Science at Stanford University. Her interests include information visualization, especially with regards to level-of-detail issues, mobile networking, and distributed systems. She is currently working on the Rivet and Polaris projects on interactive visual exploration of large datasets. She is a recipient of the NPSC Fellowship.

Pat Hanrahan is the CANON USA Professor of Computer Science and Electrical Engineering at Stanford University, where he teaches computer graphics. His current research involves visualization, image synthesis, and graphics systems and architectures. Before joining Stanford, he was a faculty member at Princeton University. He has also worked at Pixar, where he developed volume rendering software and was the chief architect of the RenderMan(TM) Interface—a protocol that allows modeling programs to describe scenes to high quality rendering programs. Prior to working for Pixar, he directed the 3D computer graphics group in the Computer Graphics Laboratory at the New York Institute of Technology. Professor Hanrahan has received three university teaching awards. In 1993, he received an Academy Award for Science and Technology, the Spirit of America Creativity Award, the SIGGRAPH Computer Graphics Achievement Award, and he was recently elected to the National Academy of Engineering.