Polaris

A System for Query, Analysis and Visualization of Multi-dimensional Relational Databases

The problem

- Large databases are becoming common
  - Warehouses of historical data
  - Genome Project: *Identify and map all human genes*
  - Digital Sky Survey: *Photographic atlases of the night sky*

- How does one extract meaning from it?
  - Discover structures
  - Find patterns
  - Derive causal relations

- The exploratory analysis is unpredictable
  - Hypothesis
  - Experiments
  - Discoveries
  - Views need to change quickly

Introduction

State of the art

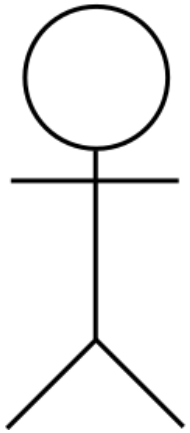
- The current solution
  1. Use data cubes to store multidimensional data
  2. Pivot tables allow data cube rotation
  3. Different dimensions used as rows and columns
  4. Others are aggregated
  5. Finally, graphs can be generated

Polaris

- Interface for exploration
- Extends pivot table to directly generate graphs
- Allows incremental building of visualizations

Use case

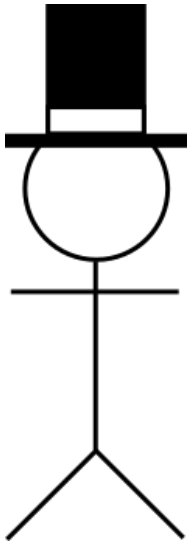
Imagine…



You

Use case

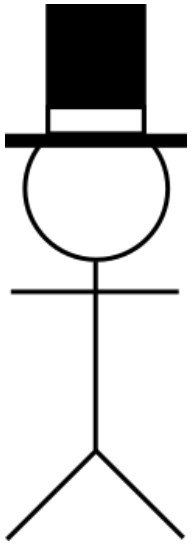
Imagine…



You are the CFO

Use case

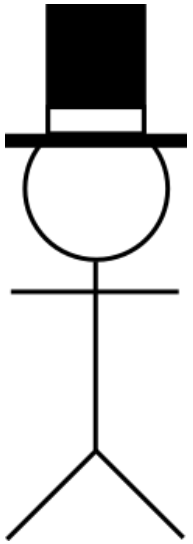
Imagine…



You are the CFO of a national coffee chain

Use case

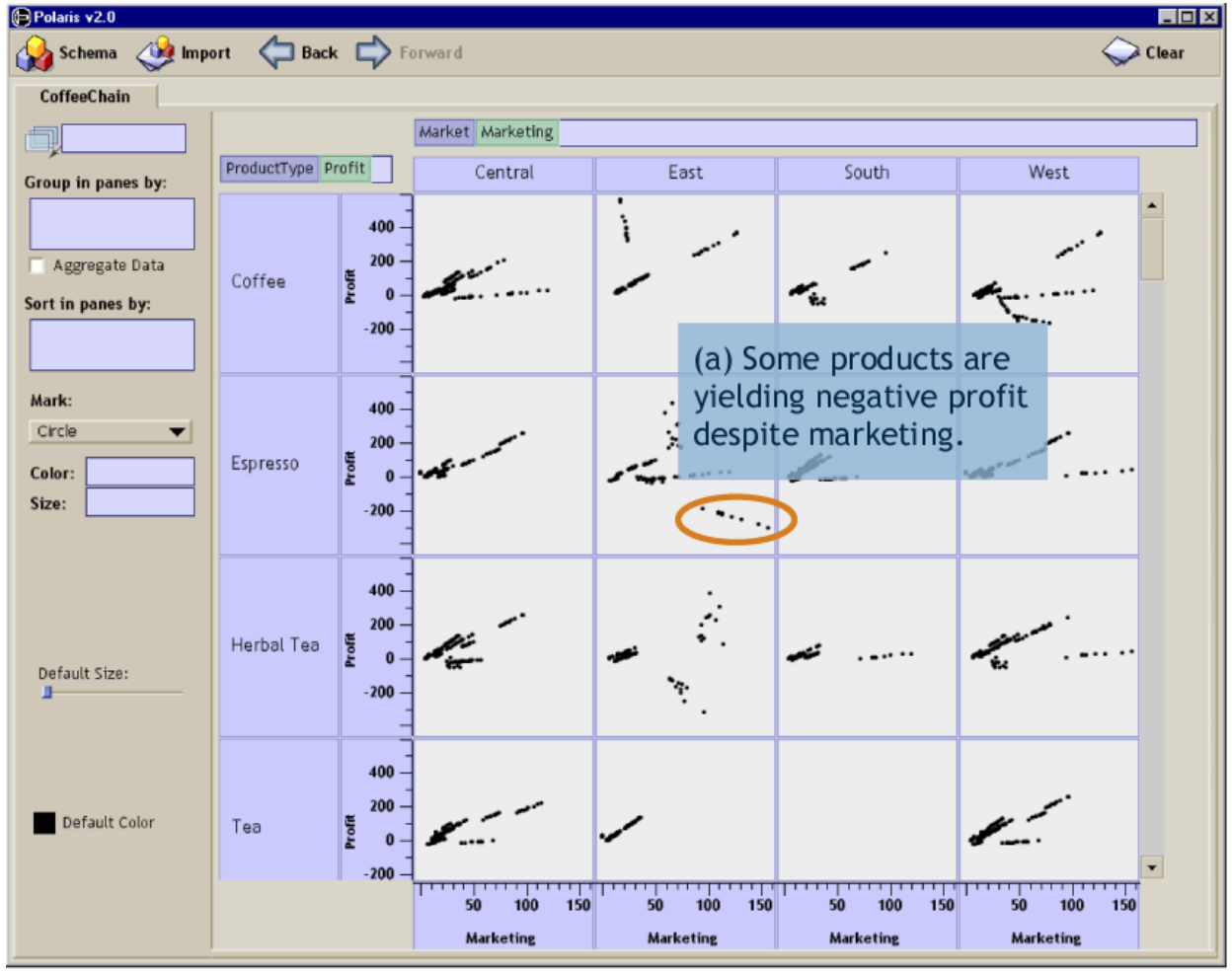
Imagine…



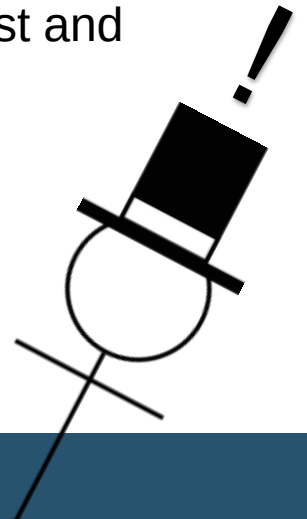
You are the CFO of a national coffee chain

And it is your job to cut expenses

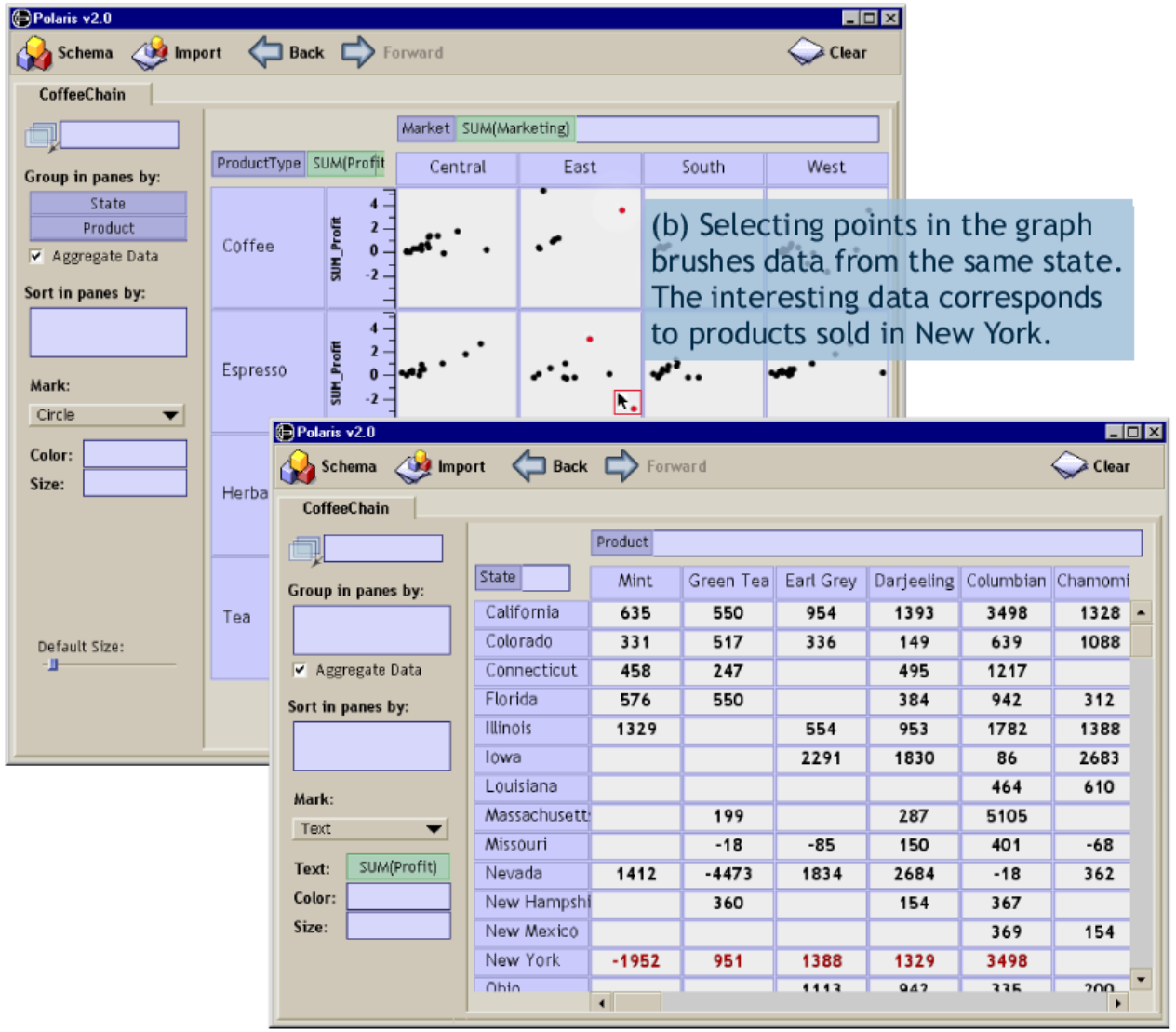
Use case



- Create table of scatterplots
- Table is *Product Type x Market*
- Each cell is a scatter plot *Profit x Cost*
- Each product entry is a mark on the chart
- Some products have high cost and negative profit

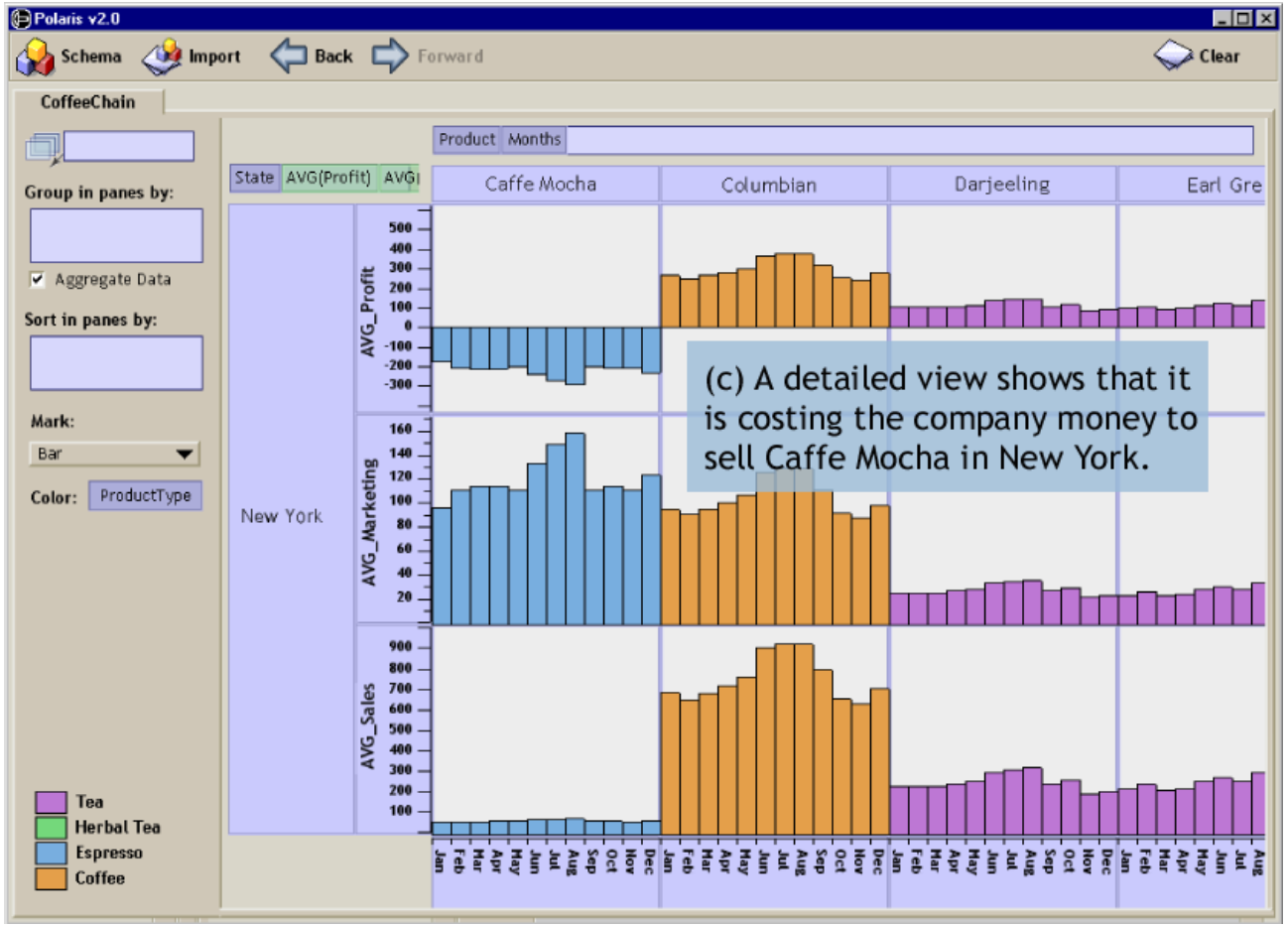


Use case



- Create linked displays
  - The same table grouped by state
  - A text table *State x Product*
- Selecting any entry highlights the corresponding entities on the other chart
- You see that in NY, some products have low profits and high marketing cost

Use case

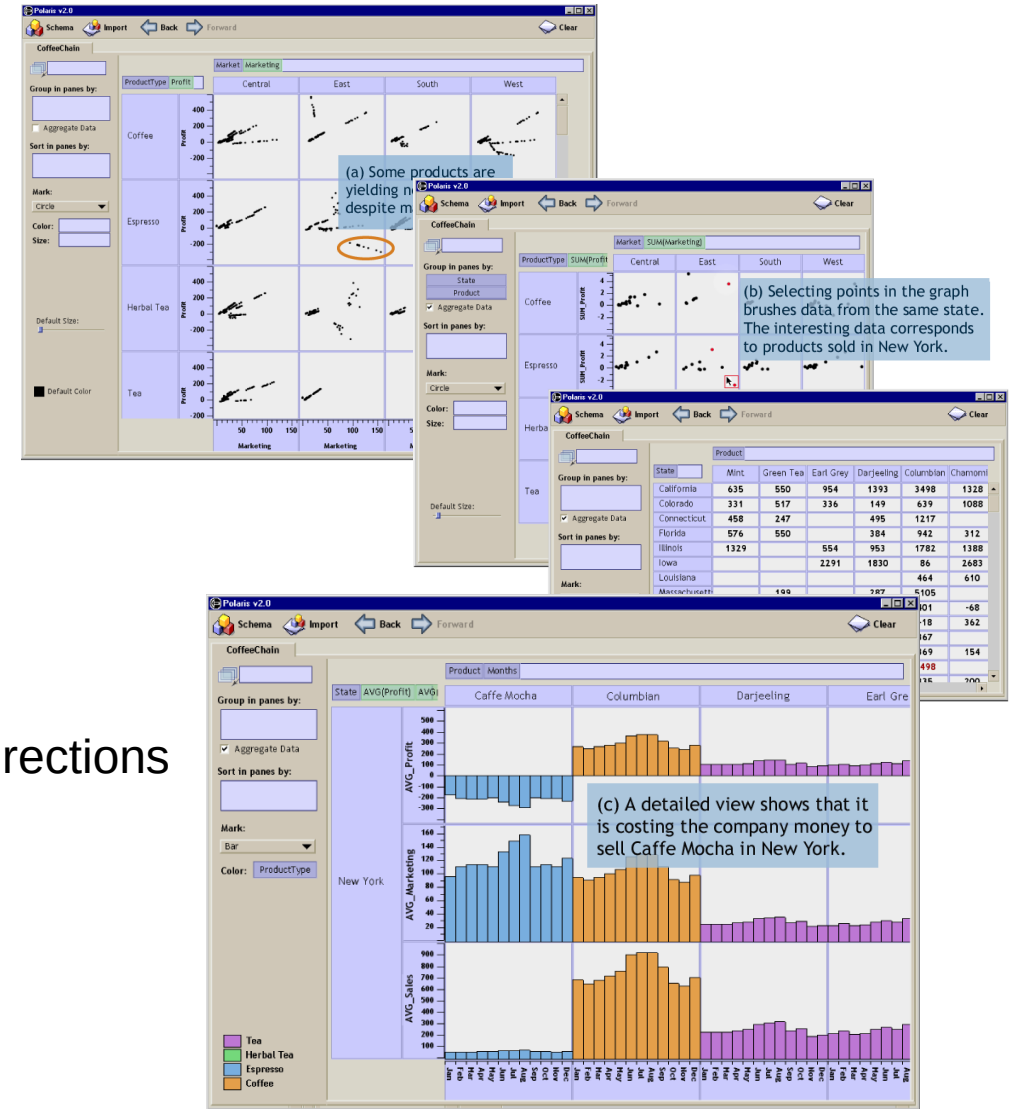


- Create a table of bar charts
- Shows profit, marketing and sales of each product in New York, by month
- You can see that the sales of Caffe Mocha in NY does not justify its marketing expenses

Use case

Important aspects of exploratory data analysis

- The data we want to see changes
- How we want to see it changes as well
- Hypotheses are created and experiments test them
- As we understand the data, we may drill down or go in other directions



Overview

Polaris supports interactive exploration of relational databases

Relational Databases In Polaris 

- Organize data into tables

- Each rows represents a basic entity
  - a.k.a. *tuple* or *record*
- Each column represents a property
  - a.k.a. *field*

- A database may contain multiple different tables

- Tables are interrelated

- Fields may be nominal, ordinal or quantitative

- Fields are divided into dimensions and measures

Overview

Polaris supports interactive exploration of relational databases

Overview

Polaris supports interactive exploration of relational databases

Which requires:

- **Data-dense displays:** Ability to visualize many dimensions of a large subset of the data

- **Multiple display types:** Generate displays for tasks such as discovering correlations, finding patterns, locating outliers and uncovering structure

- **Exploratory interface:** Need to rapidly change the data and the way the data is being displayed

Overview

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

Import:
Data from multiple data sources can be imported. Each data set maps to a different layer.

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.

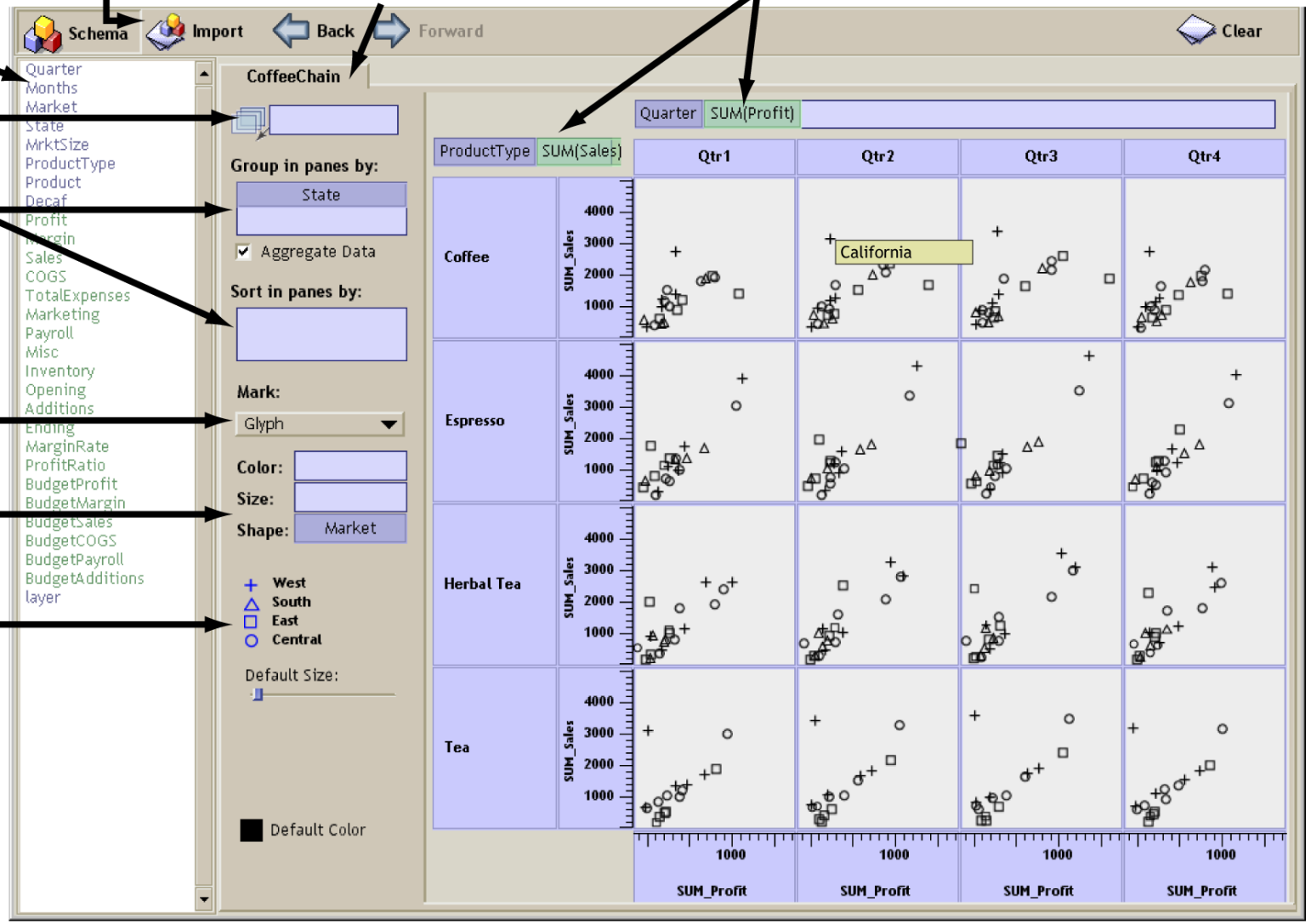
Layer Shelf:
The fields placed here determine how records are partitioned into layers.

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.

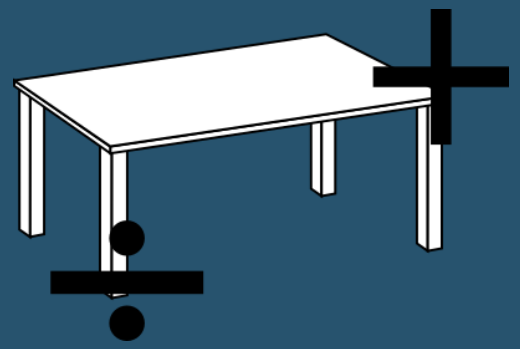


- Table-based displays
- A table has rows, columns and layers
- Each axis may have nested dimensions
- Each table entry (pane), has records encoded as marks
- Analyst can interact with displays

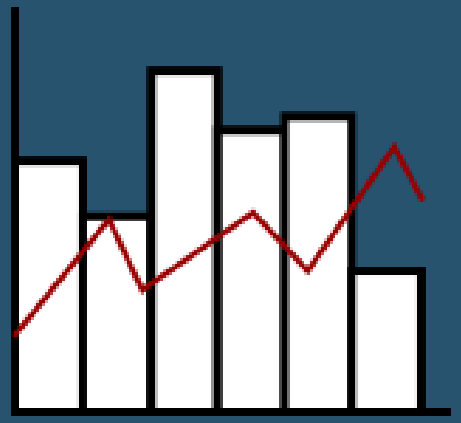
Generating Graphics

Visual specifications

Table Algebra



Types of Graphics



Visual Mappings

Generating Graphics

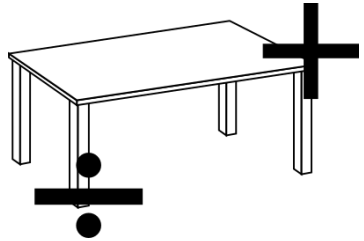


Table Algebra

- A complete table consists of 3 expressions
- The algebra has 3 operators (+, x, /)
  - Concatenation performs a union
  - Cross performs a Cartesian product
  - Nest is similar to cross but only on existing records

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	Coffee	Espresso	Herbal Tea	Tea	Coffee	Espresso	Herbal Tea	Tea	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

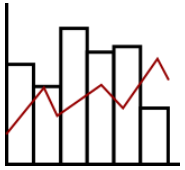
O×Q = Quarter x Profit = {(Qtr1,Profit), (Qtr2, Profit), (Qtr3, Profit), (Qtr4, Profit)}:

Qtr1	Qtr2	Qtr3	Qtr4
Profit (in thousands) 10 20 30 40 50 60	Profit (in thousands) 10 20 30 40 50 60	Profit (in thousands) 10 20 30 40 50 60	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)

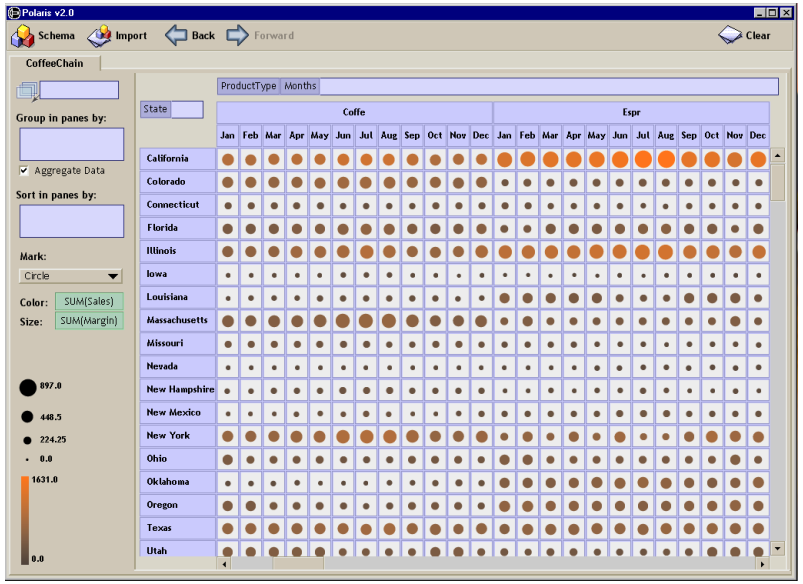
Generating Graphics



Type of Graphics

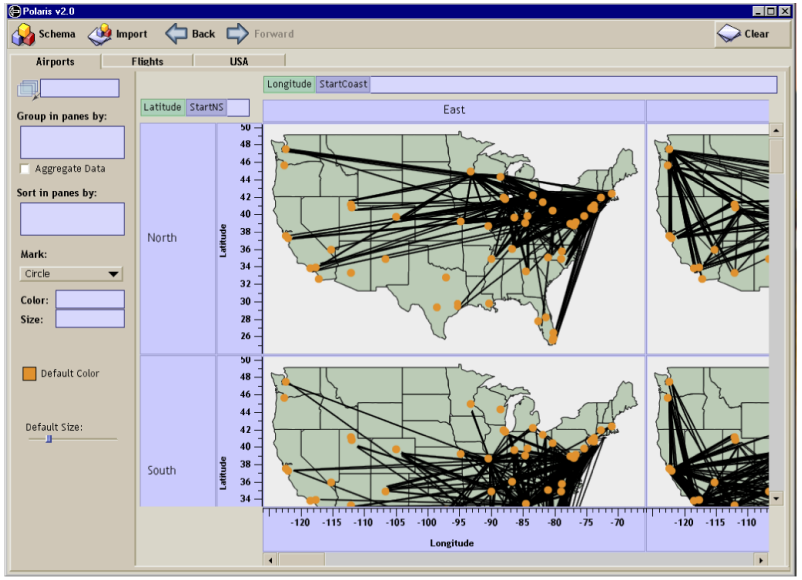
Implicitly defined during table configuration

Ordinal - Ordinal



Ordinal - Quantitative

Quantitative - Quantitative

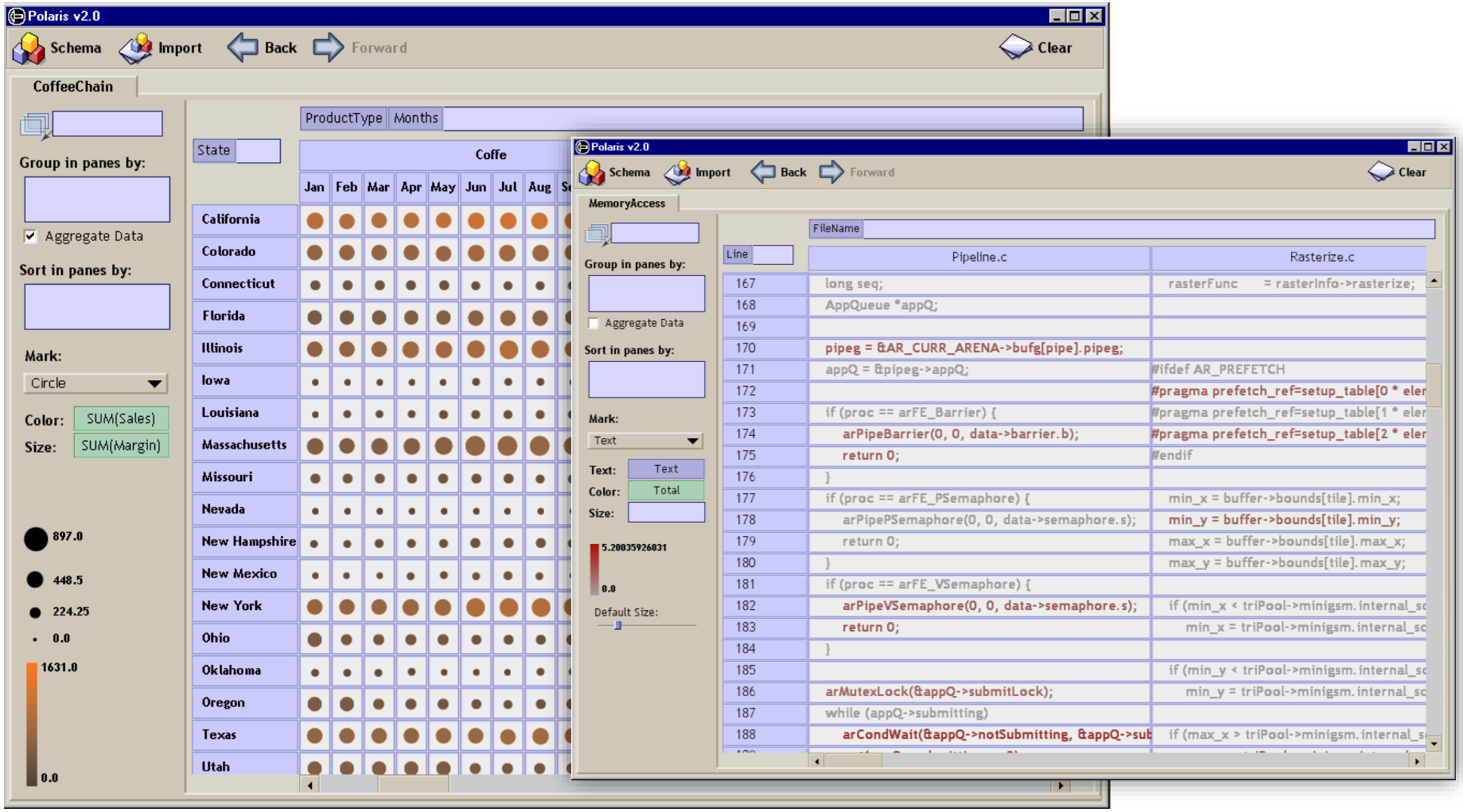


Generating Graphics

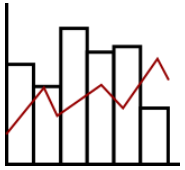
Type of Graphics

Ordinal - Ordinal

- Ex.: Table of numbers or marks
- Independent axis variables
- Number of records per pane has little effect over table structure



Generating Graphics



Type of Graphics

Ordinal - Quantitative

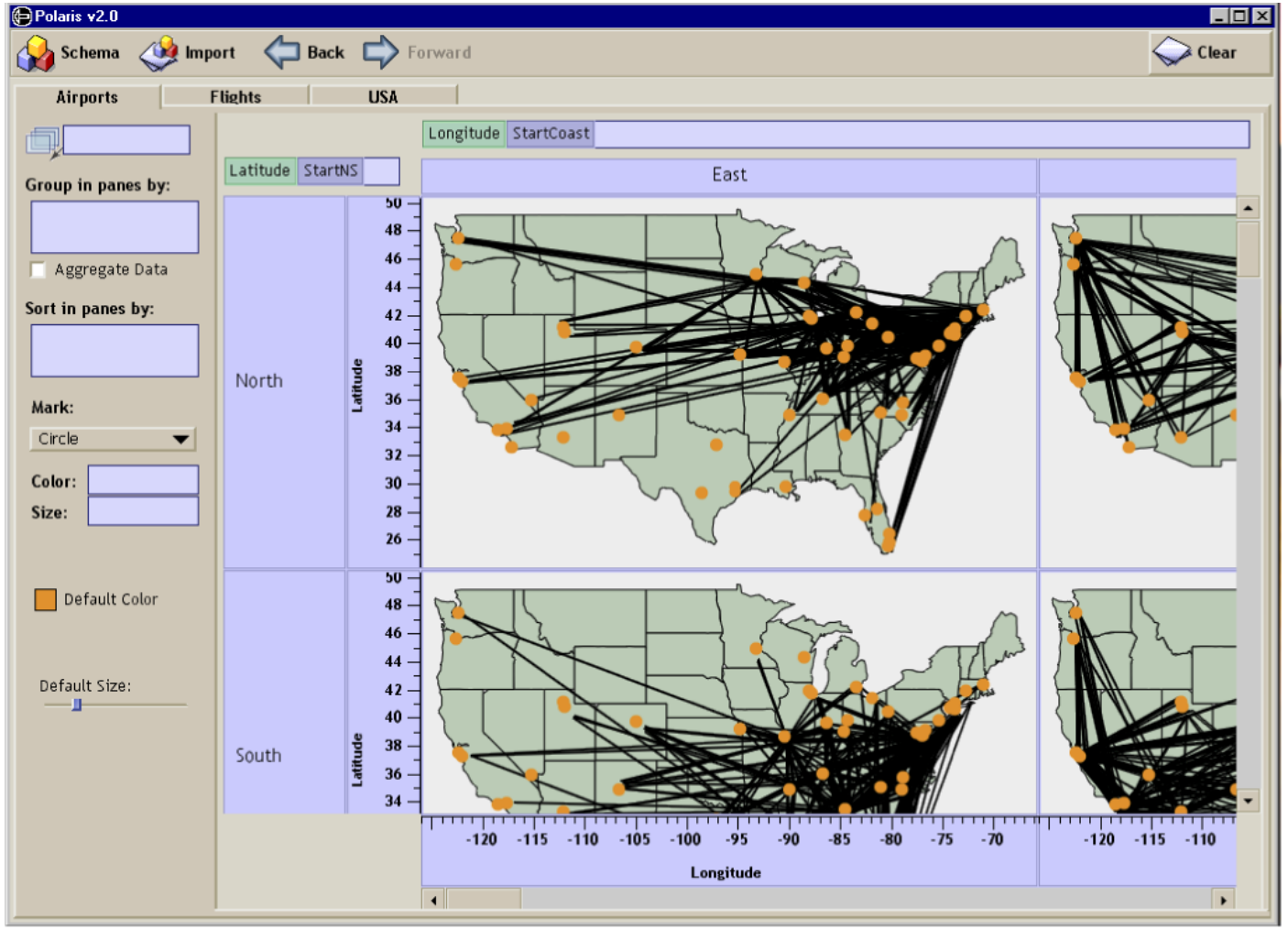
- Ex.: Bar charts and Gant charts
- Usually quantitative dependent on ordinal

Generating Graphics

Type of Graphics

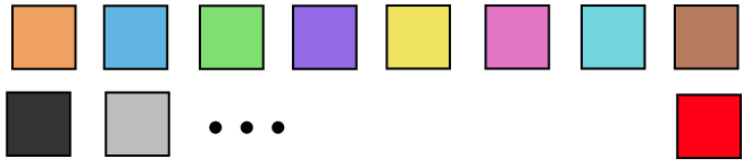
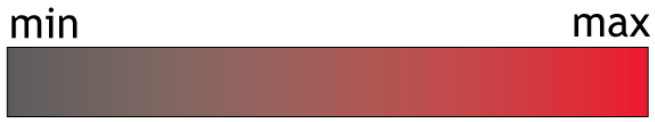
Quantitative - Quantitative

- Ex.: Scatter plots or map plots
- Discover relationships between variables



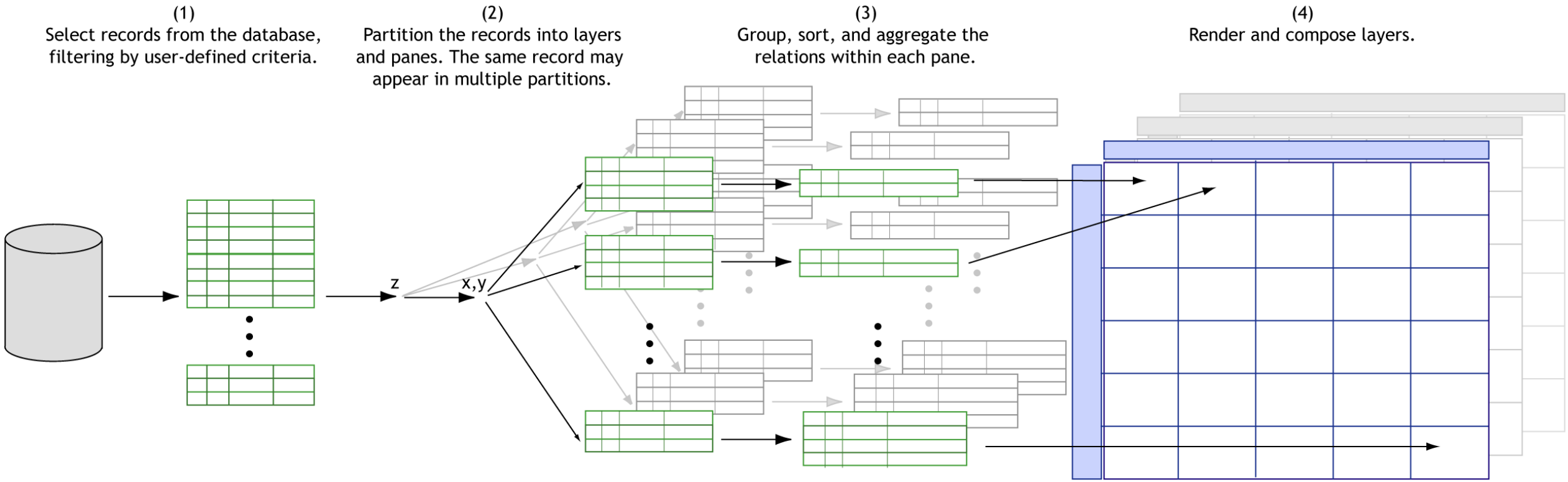
Visual Mappings

Based on Bertin's visual mappings

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		

Generating Queries

The data flow in Polaris can be precisely described using SQL queries

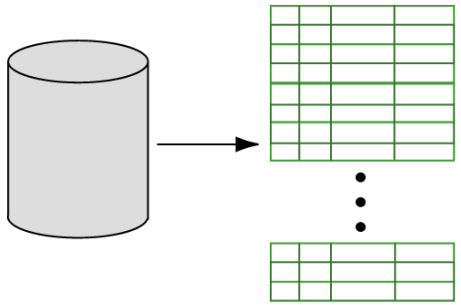


Generating Queries

Step 1: Selecting the Records

Retrieve records from database and apply user-defined filters

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



For an ordinal field, `filters` might be:

```
a in filter(A)
```

The user can specify the subset of the domain.

For a quantitative field, `filters` might be:

```
p >= min(P) and p <= max(P)
```

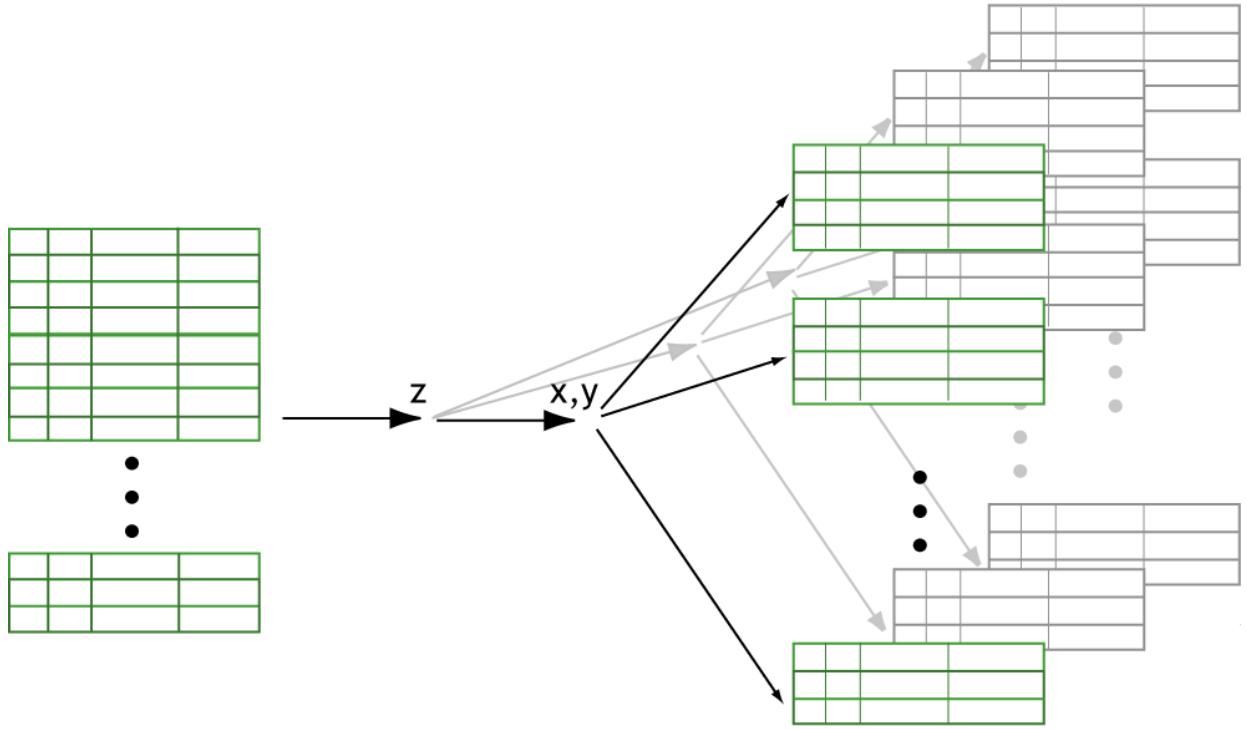
The user can specify the range of the domain.

Generating Queries

Step 2: Partitioning the records into panes

Partition into groups corresponding to each pane

```
For each pane{
        SELECT *
        WHERE  {Row(i) and
                Column(j) and
                Layer(k)}
}
```



Where *Row*, *Column*, and *Layer* represent selection criteria predicates

Generating Queries

Step 3: Transforming Records within the Panes

Assign aggregators to each measure

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

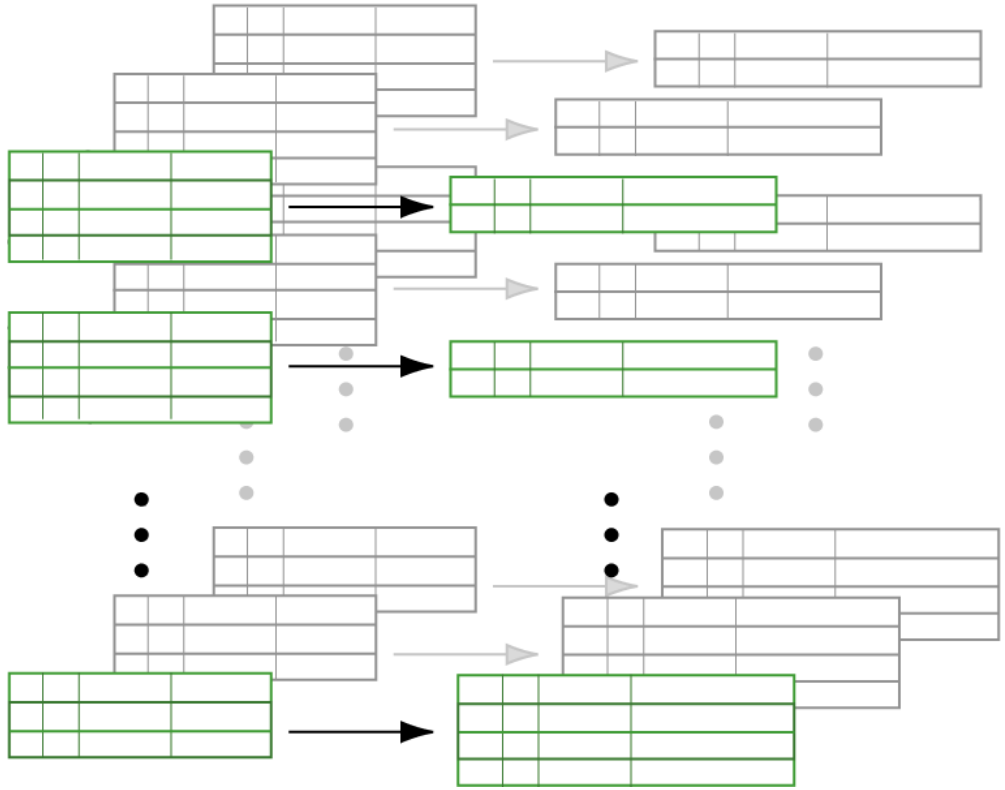
dim: *the dimensions in the database*

aggregates: *list of aggregations to be computed*

G: *the field names in the grouping shelf*

filters: *user-defined filters over the aggregations*

S: *the field names in the sorting shelf*



Conclusion

Polaris is an interface for the explorations and analysis of multi-dimensional databases.

Contributions

- Extends the Pivot Table interface by using a rich and expressive set of graphical displays
- Uses a precise sequence of operations to create visual specifications