

Multiscale Visualization Using Data Cubes

A 2002 Paper by
Chris Stolte, Diane Tang, and Pat Hanrahan

Presented by
Curran Kelleher

Abstract

Most analysts start with an overview of the data before gradually refining their view to be more focused and detailed. Multiscale pan-and-zoom systems are effective because they directly support this approach. However, generating abstract overviews of large data sets is difficult, and most systems take advantage of only one type of abstraction: visual abstraction. Furthermore, these existing systems limit the analyst to a single viewing path on their data and thus a single set of abstract views.

This paper presents: (1) a formalism for describing multiscale visualizations of data cubes with both data and visual abstraction, and (2) a method for independently zooming along one or more dimensions by traversing a zoom graph with nodes at different levels of detail. As an example of how to design multiscale visualizations using our system, we describe four design patterns using our formalism. These design patterns show the effectiveness of multiscale visualization of general relational databases.

Abstract

Most analysts start with an overview of the data before gradually refining their view to be more focused and detailed. Multiscale pan-and-zoom systems are effective because they directly support this approach. However, generating abstract overviews of large data sets is difficult, and most systems take advantage of only one type of abstraction: visual abstraction. Furthermore, these existing systems limit the analyst to a single viewing path on their data and thus a single set of abstract views.

This paper presents: (1) a formalism for describing multiscale visualizations of data rather with both data and visual abstraction, and (2) a method for independently zooming along one or more dimensions by traversing a zoom graph with nodes at different levels of detail. As an example of how to design multiscale visualizations using our system, we describe four design patterns using our formalism. These design patterns show the effectiveness of multiscale visualization of general relational databases.

Abstract

Most analysts start with an overview of the data before gradually refining their view to be more focused and detailed. Multiview point-and-zoom systems are effective because they directly support this approach. However, generating abstract overviews of large data sets is difficult, and most systems take advantage of only one type of abstraction: visual abstraction. Furthermore, these existing systems limit the analyst to a single zooming path on their data and thus a single set of abstract views.

*This paper presents: (1) a formalism for describing multiview visualizations of data cubes with both data and visual abstraction, and (2) a method for independently zooming along one or more dimensions by traversing a zoom graph with nodes at different levels of detail. As an example of how to design multiview visualizations using our system, we describe *four design patterns using our formalism*. These design patterns show the effectiveness of multiview visualization of general relational databases.*

1 Introduction

When exploring large datasets, analysts often work through a process of “Overview first, zoom and filter, then details-on-demand” [14]. Multiscale visualizations are an effective technique for facilitating this process because they change the visual representation to present the data at different levels of abstraction as the user pans and zooms. At a high level, because a large amount of data needs to be displayed, it is highly abstracted. As the user zooms, the data density decreases and thus more detailed representations of individual data points can be shown.

The two types of abstraction performed in these multiscale visualizations are *data abstraction* and *visual abstraction*. Data abstractions (e.g., aggregation or selection) change the underlying data before mapping them to visual representations. Visual abstractions change the visual representation of data points (but not the underlying data itself) to provide more information as the user zooms, e.g., an image may morph from a simplified thumbnail to a full-scale

Data Abstraction

1 Introduction

When exploring large datasets, analysts often work through a process of “Overview first, zoom and filter, then details-on-demand” [14]. Multiscale visualizations are an effective technique for facilitating this process because they change the visual representation to present the data at different levels of abstraction as the user pans and zooms. At a high level, because a large amount of data needs to be displayed, it is highly abstracted. As the user zooms, the data density decreases and thus more detailed representations of individual data points can be shown.

The two types of abstraction performed in these multiscale visualizations are data abstraction and visual abstraction. Data abstractions (e.g., aggregation or selection) change the underlying data before mapping them to visual representations. Visual abstractions

change the visual representation of data points that are the underlying data itself to provide more information as the user zooms, e.g., an image may morph from a simplified thumbnail to a full-scale

2 Related Work

In this section, we review several existing multiscale visualization systems, focusing on how the systems perform both data and visual abstraction. *Data abstraction* refers to transformations applied to the data before being visually mapped, including aggregation, filtering, sampling, or statistical summarization.

Visual abstraction refers to abstractions that change the visual representation (e.g., a circle at an overview level versus a text string at a detailed level), change how data is encoded in the visual attributes of the glyphs (e.g., encoding data in the size and color of a glyph only in detailed views), or apply transformations to the set of visual representations (e.g., combining glyphs that overlap).

Multiscale Visualization in Cartography

Cartography is the source of many early examples of multiscale visualizations. Cartographic generalization [19] refers to the process of generating small scale maps by simplifying and abstracting large scale source material and consists of two steps: (1) combining

data abstractions limited to simple filtering and the ability to add or switch data sources. In addition, these systems primarily only allow for a single viewing path.

Our goal is to develop a system for describing and developing multiscale visualizations that support multiple view paths and both data and visual abstractions. We want to support multiple view paths because many large data sets today are organized using multiple hierarchies that define meaningful levels of aggregation (i.e., details).

Data cubes are a commonly accepted method for abstracting and summarizing relational databases. By representing the database with a data cube, we can switch between different levels of detail using a general mechanism applicable to many different data sets. Combining this general mechanism for performing meaningful data abstractions with traditional visual abstraction techniques enhances our ability to generate abstract views of large data sets, a difficult and challenging problem.

Previously, we presented Polaris, a tool for visually exploring relational databases [15] and later extended for hierarchically struc-

cannot. Finally, we present how we can create a query graph in Polaris specifications to describe a multiscale visualization of a hierarchical data set, as well as how we can easily implement such visualizations within our system.

3.1 Data Abstraction: Data Cubes

Not only are data cubes widely used, but they also provide a powerful mechanism for performing data abstraction that we can leverage. Specifically, data cubes quickly provide summaries of the underlying data at different meaningful levels of detail, rather than arbitrary summarizations such as aggregating every two records. This goal is achieved by building a lattice of data cubes to represent the data at different levels of detail according to a semantic hierarchy and providing mechanisms for then summarizing each cube. We first describe an individual data cube before describing the lattice.

Data cubes categorize information into two classes: dimensions and measures, corresponding to the independent and dependent

variables, respectively. For example, U.S. states are a dimension, while the population of each state is a measure. Within a cube, the data is abstractly structured as an n -dimensional data cube. Each axis corresponds to a dimension in the data cube and consists of every possible value for that dimension. For example, an axis corresponding to states would have fifty values, one for each state. Every "cell" in the data cube corresponds to a unique combination of values for the dimensions. For example, if we had two dimensions, State and Product, then there would be a cell for every unique combination of the two (e.g., one cell each for (California, Oranges), (California, Coffee), (Florida, Oranges), (Florida, Coffee), etc.). Each cell contains one value per measure of the data cube; e.g., if we wanted to know about product production and consumption, then each cell would contain two values, one for the number of products of each type consumed in that state, and one for the number of products of each type produced in that state.

Thus far, we have considered dimensions to be flat structures. However, most dimensions have a hierarchical structure. For exam-

variables, respectively. For example, U.S. states are a dimension, while the population of each state is a measure. Within a cube, the data is abstractly structured as an n-dimensional data cube. Each axis corresponds to a dimension in the data cube and consists of every possible value for that dimension. For example, an axis corresponding to states would have fifty values, one for each state. Every "cell" in the data cube corresponds to a unique combination of values for the dimensions. For example, if we had two dimensions, State and Product, then there would be a cell for every unique combination of the two (e.g., one cell each for (California, Oranges), (California, Coffee), (Florida, Oranges), (Florida, Coffee), etc.). Each cell contains one value per measure of the data cube, e.g., if we wanted to know about product production and consumption, then each cell would contain two values, one for the number of products of each type consumed in that state, and one for the number of products of each type produced in that state.

Thus far, we have considered dimensions to be flat structures. However, most dimensions have a hierarchical structure. For exam-

variables, respectively. For example, U.S. states are a dimension, while the population of each state is a measure. Within a cube, the data is abstractly structured as an n -dimensional data cube. Each axis corresponds to a dimension in the data cube and consists of every possible value for that dimension. For example, an axis corresponding to states would have fifty values, one for each state.

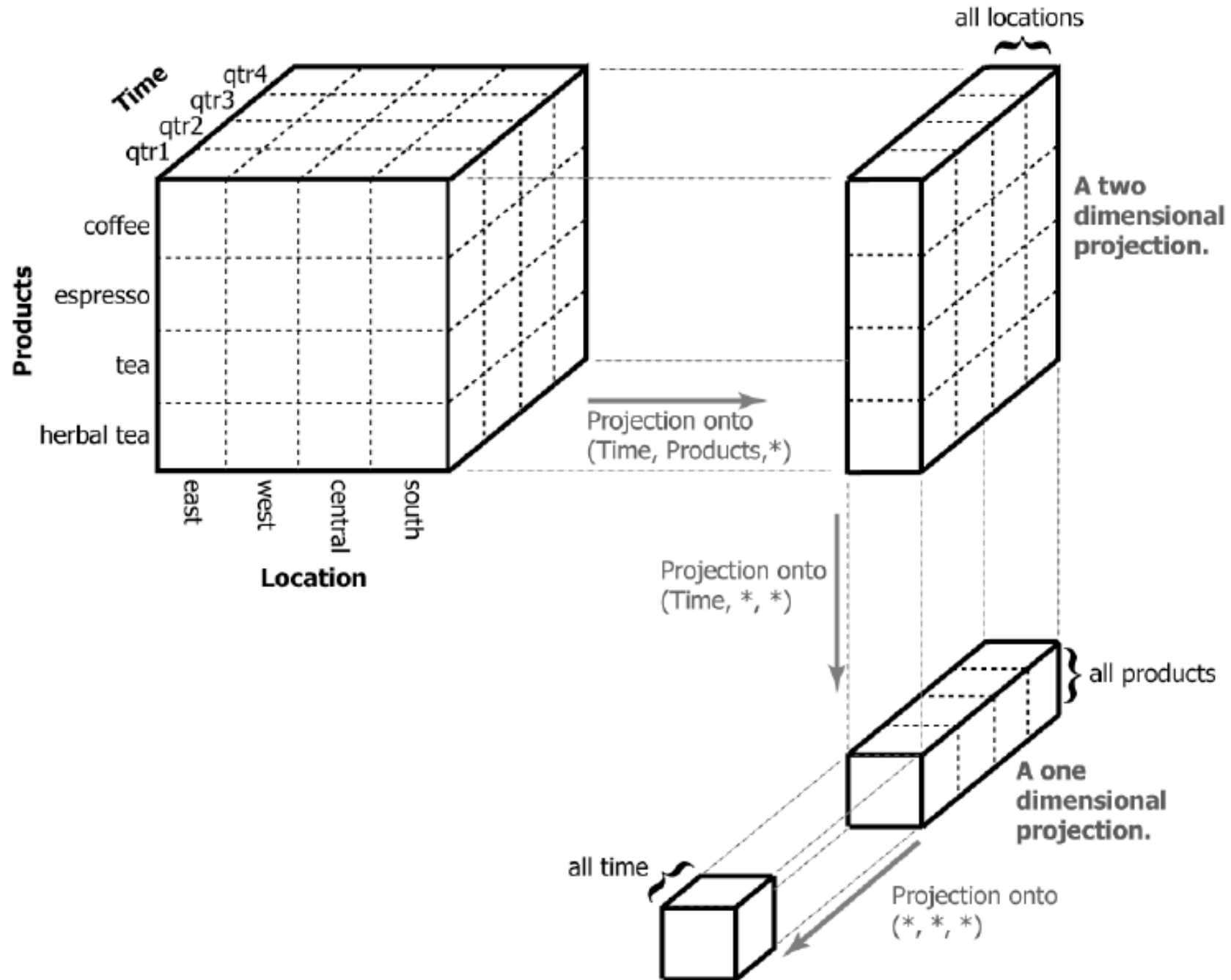
Every “cell” in the data cube corresponds to a unique combination of values for the dimensions.

For example, if we had two dimensions, State and Product, then there would be a cell for every unique combination of the two (e.g., one cell each for (California, Oranges), (California, Coffee), (Florida, Oranges), (Florida, Coffee), etc.).

Each cell contains one value per measure of the data cube (e.g., if we wanted to know about product production and consumption, then each cell would contain two values, one for the number of products of each type consumed in that state, and one for the number of products of each type produced in that state).

Thus far, we have considered dimensions to be flat structures. However, most dimensions have a hierarchical structure. For example, the dimension of U.S. states can be broken down into counties, cities, and zip codes.

Projecting a three dimensional data cube



Thus far, we have considered dimensions to be flat structures. However, most dimensions have a hierarchical structure.

For example, rather than having a single dimension "state", we may have a hierarchical dimension "location" that has levels for country, state, and county. If each dimension has a hierarchical structure, then the data must be structured as a lattice of data cubes, where each cube is defined by the combination of a level of detail for each dimension.

Data abstraction in this model means choosing a meaningful summary of the data. Choosing a data abstraction corresponds to choosing a particular projection in this lattice of data cubes: (a) which dimensions we currently consider relevant and (b) the appropriate level of detail for each relevant dimensional hierarchy. Specifying the level of detail identifies the cube in the lattice, while the relevant dimensions identifies which projection (from a dimension down to the number of relevant dimensions) of that cube is needed. Figure 1 shows a simple lattice and projection.

While identifying a specific projection in the data cube corresponds to specifying the desired data abstraction of the raw data, in

not as products of each type product as well as:

Thus far, we have considered dimensions to be flat structures. However, most dimensions have a hierarchical structure. For example, rather than having a single dimension “state”, we may have a hierarchical dimension “location” that has levels for country, state, and county.

If each dimension has a hierarchical structure, then the data must be structured as a lattice of data cubes, where each cube is defined by the combination of a level of detail for each dimension.

Data abstraction in this model means choosing a meaningful summary of the data. Choosing a data abstraction corresponds to choosing a particular projection in this lattice of data cubes: (a) which dimensions we currently consider relevant and (b) the appropriate level of detail for each relevant dimensional hierarchy. Specifying the level of detail identifies the cube in the lattice, while the relevant dimensions identifies which projection (from a dimension down to the number of relevant dimensions) of that cube is needed. Figure 1 shows a simple lattice and projection.

While identifying a specific projection in the data cube corresponds to computing the desired data abstraction of the raw data, in

not in products in each type product in each state.

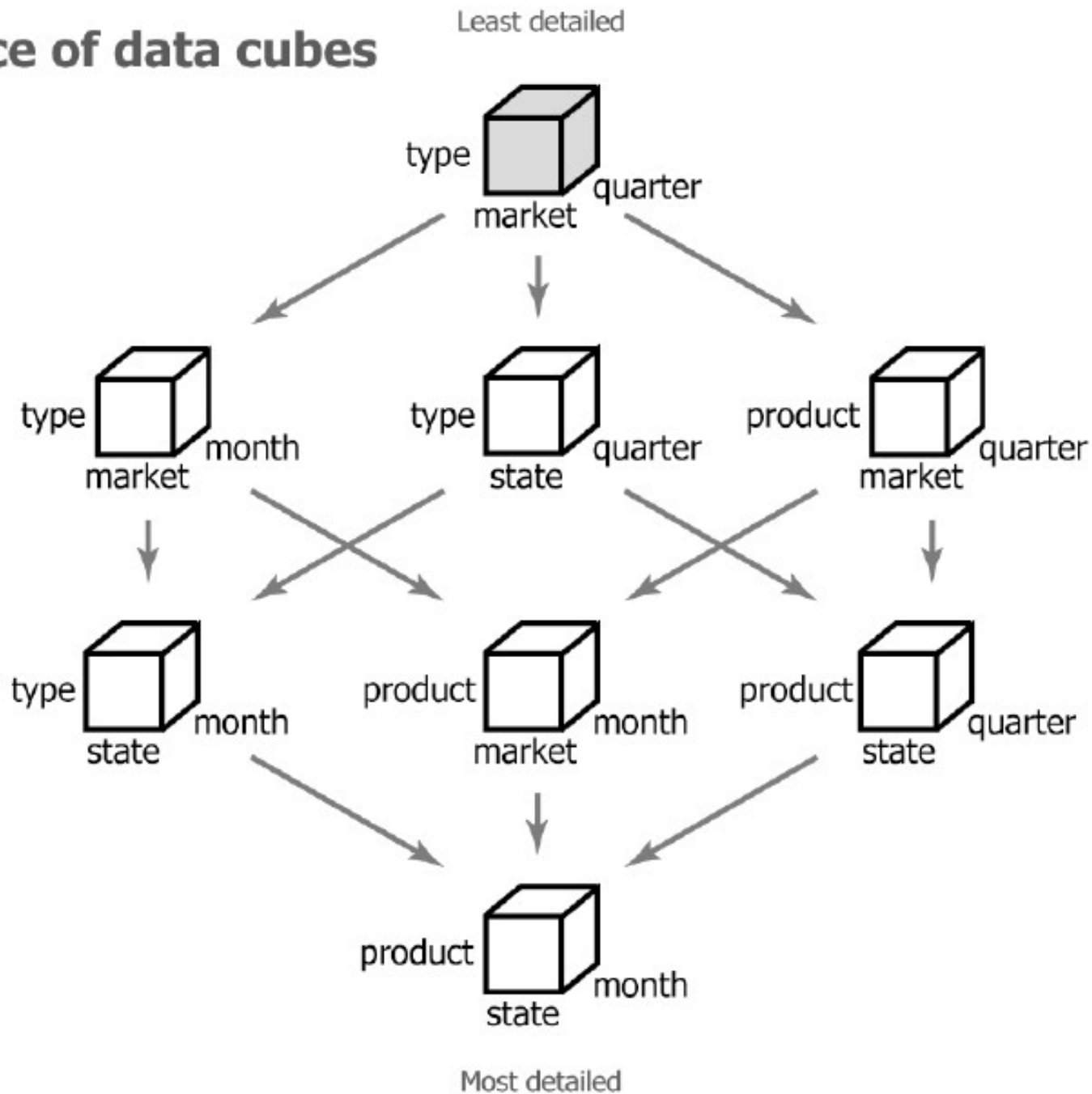
Thus far, we have considered dimensions to be flat structures. However, most dimensions have a hierarchical structure. For example, rather than having a single dimension "state", we may have a hierarchical dimension "location" that has levels for country, state, and county. If each dimension has a hierarchical structure, then the

data must be structured as a lattice of data cubes where each cube is defined by the combination of a level of detail for each dimension.

Data abstraction in this model means choosing a meaningful summary of the data. Choosing a data abstraction corresponds to choosing a particular projection in this lattice of data cubes: (a) which dimensions we currently consider relevant and (b) the appropriate level of detail for each relevant dimensional hierarchy. Specifying the level of detail identifies the cube in the lattice, while the relevant dimensions identifies which projection (from a dimension down to the number of relevant dimensions) of that cube is needed. Figure 1 shows a simple lattice and projection.

While identifying a specific projection in the data cube corresponds to specifying the desired data abstraction of the raw data, in

The lattice of data cubes



multidimensional data is a hierarchical structure. For example, rather than having a single dimensional "state", we may have a hierarchical dimensional "location" that has levels for country, state, and county. If each dimension has a hierarchical structure, then the data must be structured as a lattice of data cubes, where each cube is defined by the combination of a level of detail for each dimension.

Data abstraction in this model means choosing a meaningful summary of the data. Choosing a data abstraction corresponds to choosing a particular projection in this lattice of data cubes: (a) which dimensions we currently consider relevant and (b) the appropriate level of detail for each relevant dimensional hierarchy. Specifying the level of detail identifies the cube in the lattice, while the relevant dimensions identifies which projection (from a dimension down to the number of relevant dimensions) of that cube is needed. Figure 1 shows a simple lattice and projection.

While identifying a specific projection in the data cube corresponds to specifying the desired data abstraction of the raw data, in multiscale visualizations we need to specify both the data and visual abstractions. Both sets of information are contained in a *Projection*.

however, some dimensions may be hierarchical themselves. For example, rather than having a single dimensional "state", we may have a hierarchical dimensional "location" that has levels for country, state, and county. If each dimension has a hierarchical structure, then the data must be structured as a lattice of data cubes, where each cube is defined by the combination of a level of detail for each dimension.

Data abstraction in this model means choosing a meaningful summary of the data. Choosing a data abstraction corresponds to choosing a particular projection in this lattice of data cubes: (a) which dimensions we currently consider relevant and (b) the appropriate level of detail for each relevant dimensional hierarchy. Specifying the level of detail identifies the cube in the lattice, while the relevant dimensions identifies which projection (from a dimension down to the number of relevant dimensions) of that cube is needed. Figure 1 shows a simple lattice and projection.

While identifying a specific projection in the data cube corresponds to specifying the desired data abstraction of the raw data, in multiscale visualizations we need to specify both the data and visual abstractions: both sets of information are contained in a *projection*.

multidimensional, where each dimension has a hierarchical structure. For example, rather than having a single dimensional "state", we may have a hierarchical dimensional "location" that has levels for country, state, and county. If each dimension has a hierarchical structure, then the data must be structured as a lattice of data cubes, where each cube is defined by the combination of a level of detail for each dimension.

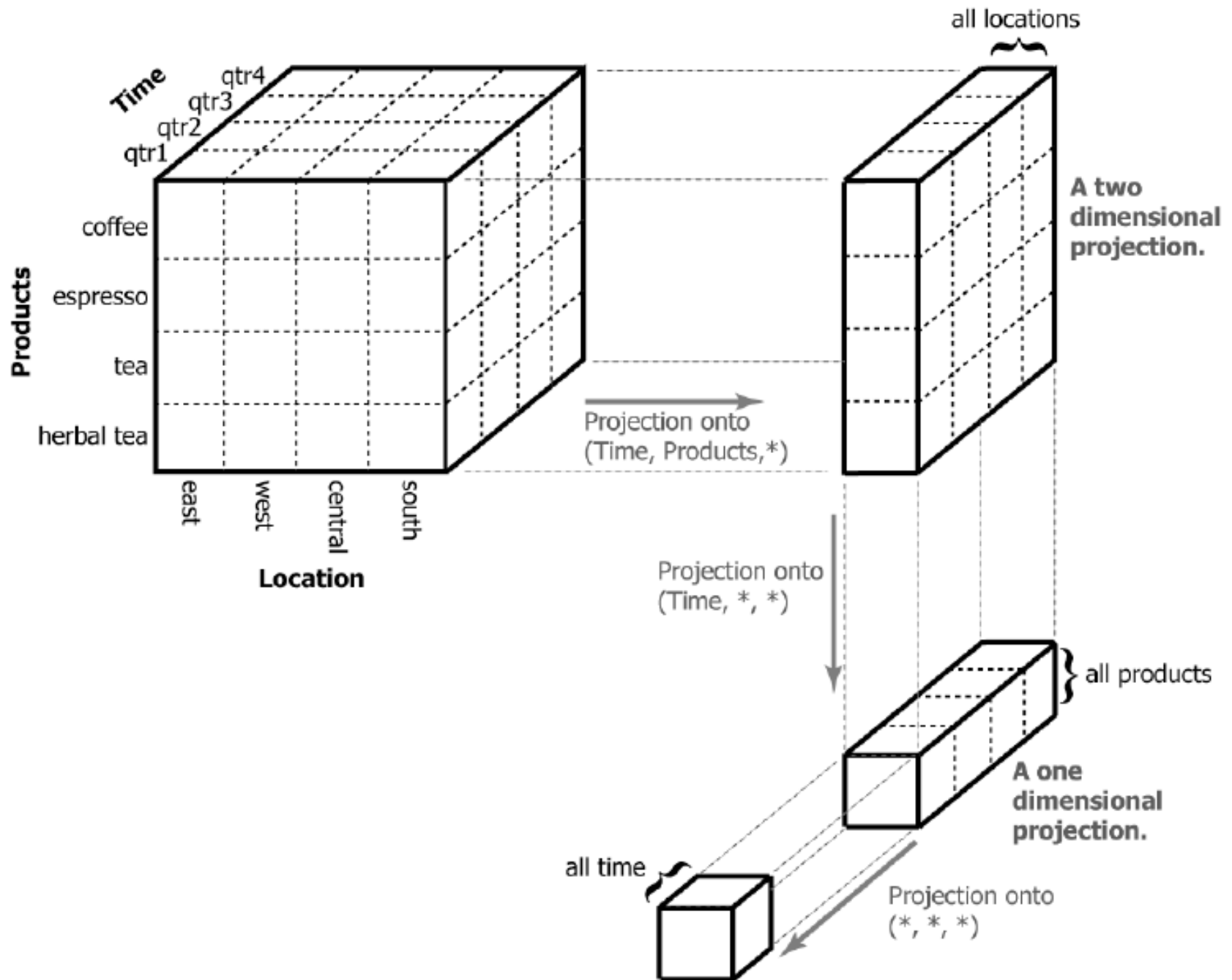
Data abstraction in this model means choosing a meaningful summary of the data.

Choosing a data abstraction corresponds to choosing a particular *projection* in this lattice of data cubes:

(a) which dimensions we currently consider relevant and (b) the appropriate level of detail for each relevant dimensional hierarchy. Specifying the level of detail identifies the cube in the lattice, while the relevant dimensions identifies which projection (from a dimension down to the number of relevant dimensions) of that cube is needed. Figure 1 shows a simple lattice and projection.

While identifying a specific projection in the data cube corresponds to specifying the desired data abstraction of the raw data, in multiscala visualizations we need to specify both the data and visual abstractions. Both sets of information are represented as a *projection*.

Projecting a three dimensional data cube



however, some dimensions may be hierarchical themselves. For example, rather than having a single dimensional "state", we may have a hierarchical dimensional "location" that has levels for country, state, and county. If each dimension has a hierarchical structure, then the data must be structured as a lattice of data cubes, where each cube is defined by the combination of a level of detail for each dimension.

Data abstraction in this model means choosing a meaningful summary of the data. Choosing a data abstraction corresponds to choosing a particular *projection* in this lattice of data cubes: (a) which dimensions we currently consider relevant and (b) the appropriate level of detail for each relevant dimensional hierarchy.

Specifying the level of detail identifies the cube in the lattice, while the relevant dimensions identifies which projection (from a dimension down to the number of relevant dimensions) of that cube is needed. Figure 1 shows a simple lattice and projection.

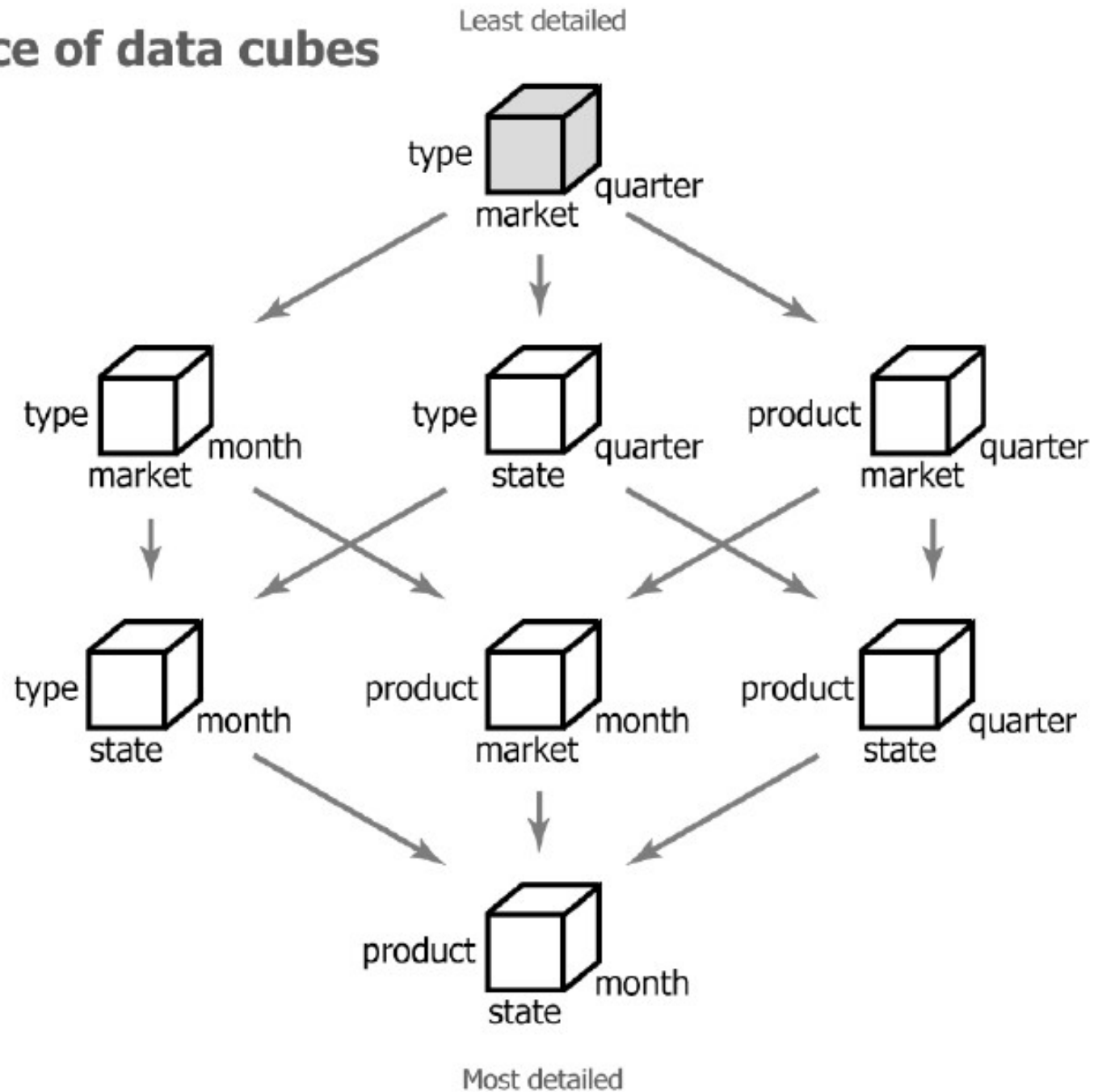
While identifying a specific projection in the data cube corresponds to specifying the desired data abstraction of the raw data, in multiscala visualizations we need to specify both the data and visual abstractions. Both sets of information are represented as a *projection*.

however, some dimensions have a hierarchical structure. For example, rather than having a single dimension "state", we may have a hierarchical dimension "location" that has levels for country, state, and county. If each dimension has a hierarchical structure, then the data must be structured as a lattice of data cubes, where each cube is defined by the combination of a level of detail for each dimension.

One abstraction in this model means choosing a meaningful summary of the data. Choosing a data abstraction corresponds to choosing a particular projection in this lattice of data cubes: (a) which dimensions we currently consider relevant and (b) the appropriate level of detail for each relevant dimensional hierarchy. Specifying the level of detail identifies the cube in the lattice while the relevant dimensions identifies which projection (from a dimension down to the number of relevant dimensions) of that cube is needed. Figure 1 shows a simple lattice and projection.

While identifying a specific projection in the data cube corresponds to specifying the desired data abstraction of the raw data, in multiscala visualizations we need to specify both the data and visual abstractions. Both sets of information are contained in a *projection*.

The lattice of data cubes



dimension, some dimensions may be hierarchical themselves. For example, rather than having a single dimension "state", we may have a hierarchical dimension "location" that has levels for country, state, and county. If each dimension has a hierarchical structure, then the data must be structured as a lattice of data cubes, where each cube is defined by the combination of a level of detail for each dimension.

Data abstraction in this model means choosing a meaningful summary of the data. Choosing a data abstraction corresponds to choosing a particular projection in this lattice of data cubes:

(a) which dimensions we currently consider relevant and the appropriate level of detail for each relevant dimensional hierarchy. Specifying the level of detail identifies the cube in the lattice,

while the relevant dimensions identifies which projection of that cube is needed. Figure 1 shows a simple lattice and projection.

While identifying a specific projection in the data cube corresponds to specifying the desired data abstraction of the raw data, in multiscale visualizations we need to specify both the data and visual abstractions. Both sets of information are represented as a *projection*.

Visual Abstraction

1 Introduction

When exploring large datasets, analysts often work through a process of “Overview first, zoom and filter, then details-on-demand” [14]. Multiscale visualizations are an effective technique for facilitating this process because they change the visual representation to present the data at different levels of abstraction as the user pans and zooms. At a high level, because a large amount of data needs to be displayed, it is highly abstracted. As the user zooms, the data density decreases and thus more detailed representations of individual data points can be shown.

The two types of abstraction performed in these multiscale visualizations are data abstraction and visual abstraction. Data abstractions (e.g., aggregation or selection) change the underlying data before mapping them to visual representations. Visual abstractions

change the visual representation of data points (but not the underlying data itself) to provide more information as the user zooms

(e.g., an image may morph from a simplified thumbnail to a full-scale

2 Related Work

In this section, we review several existing multiscale visualization systems, focusing on how the systems perform both data and visual abstractions. These abstractions refer to transformations applied to the data before being visually mapped, including aggregation, filtering, sampling, or statistical summarization. *Visual abstraction*

refers to abstractions that change the visual representation (e.g., a circle at an overview level versus a text string at a detailed level), change how data is encoded in the retinal attributes of the glyphs (e.g., encoding data in the size and color of a glyph only in detailed views), or apply transformations to the set of visual representations (e.g., combining glyphs that overlap).

Multiscale Visualization in Cartography

Cartography is the source of many early examples of multiscale visualization. Cartographic generalization [19] refers to the process of generating small scale maps by simplifying and abstracting large scale vector content and consists of two steps: (1) selection

3.2 Visual Abstraction: Polaris

Previously, we presented the Polaris database exploration tool [15], consisting of three parts: (1) a formal specification language for describing table-based visualizations, (2) a user interface for automatically generating instances of these specifications, and (3) a method for automatically generating the necessary database queries to retrieve the data to be visualized by a specification. We have extended all three parts to support hierarchically structured data cubes [16].

In this paper, we only use the specification language from the previous papers. We use this language to describe a node within the search graph identifying a multiscale visualization. In this section, we briefly review the components of a Polaris specification and introduce a graphical notation that succinctly captures the data and visual abstractions in table-based visualizations of hierarchically structured data.

A Polaris specification uses a formal table algebra to specify the table configuration of the visualization. Each expression in the table algebra defines an union of the tables from the table in directed con-

Visual Abstraction in Polaris

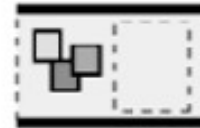
shape	color	size
-------	-------	------

Each layer has three encodings.

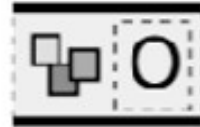
abc



blank means no encoding allowed



an empty slot indicates an optional data encoding



a slot containing a field type indicates a required data encoding



a primitive with no slot indicates a fixed value encoding

Primitives:

abc

= text



= point



= line



= polygon



= text or point

Color:



= ordinal palette



= quantitative ramp

Size:



= height



= width



= both

3.2 Visual Abstraction: Polaris

Previously, we presented the Polaris database exploration tool [15], consisting of three parts: (1) a formal specification language for describing table-based visualizations, (2) a user interface for automatically generating instances of these specifications, and (3) a method for automatically generating the necessary database queries to retrieve the data to be visualized by a specification. We later extended

all three parts to support hierarchically structured data values [16].

In this paper, we only use the specification language from the previous papers. We use this language to describe a node within the access graph identifying a node-based visualization. In this section, we briefly review the components of a Polaris specification and introduce a graphical notation that succinctly captures the data and visual abstractions in table-based visualizations of hierarchically structured data.

A Polaris specification uses a formal table algebra to specify the table configuration of the visualization. Each expression in the table algebra defines an access path to the data from the table in a direct way.

Abstract

Most analysts start with an overview of the data before gradually refining their view to be more focused and detailed. Multiscale pan-and-zoom systems are effective because they directly support this approach. However, generating abstract overviews of large data sets is difficult, and most systems take advantage of only one type of abstraction: visual abstraction. Furthermore, these existing systems limit the analyst to a single viewing path on their data and thus a single set of abstract views.

This paper presents: (1) a formalism for describing multiscale visualizations of data cubes with both data and visual abstraction, and (2) a method for independently zooming along one or more dimensions by traversing a zoom graph with nodes at different levels of detail. As an example of how to design multiscale visualizations using our system, we describe four design patterns using our formalism. These design patterns show the effectiveness of multiscale visualization of general relational databases.

3.2 Visual Abstraction: Polaris

Previously, we presented the Polaris database exploration tool [15], consisting of three parts: (1) a formal specification language for describing table-based visualizations, (2) a user interface for automatically generating instances of these specifications, and (3) a method for automatically generating the necessary database queries to retrieve the data to be visualized by a specification. We later extended all three parts to support hierarchically structured data cubes [16].

In this paper, we only use the specification language from the previous papers. We use this language to describe a node within the search graph identifying a multiscale visualization. In this section, we briefly review the components of a Polaris specification and introduce a graphical notation that succinctly captures the data and visual abstractions in table-based visualizations of hierarchically structured data.

A Polaris specification uses a formal table algebra to specify the table configuration of the visualization. Each expression in the table algebra defines one node of the search space. Every table in the search

Abstract

Most analysts start with an overview of the data before gradually refining their view to be more focused and detailed. Multiscale pan-and-zoom systems are effective because they directly support this approach. However, generating abstract overviews of large data sets is difficult, and most systems take advantage of only one type of abstraction: visual abstraction. Furthermore, these existing systems limit the analyst to a single viewing path on their data and thus a single set of abstract views.

This paper presents: (1) a formalism for describing multiscale visualizations of data rather with both data and visual abstraction, and (2) a method for independently zooming along one or more dimensions by traversing a zoom graph with nodes at different levels of detail. As an example of how to design multiscale visualizations using our system, we describe four design patterns using our formalism. These design patterns show the effectiveness of multiscale visualization of general relational databases.

3 Multiscale Visualizations

In this section, we present our system for describing multiscale visualizations that support multiple zoom paths and both data and visual abstraction.

Rather than considering multiscale visualizations as simply a series of linear zooms, we think of multiscale visualizations as a graph, where each node corresponds to a particular set of data and visual abstractions and each edge is a zoom. Zooming in a multiscale visualization is equivalent to traversing this graph. Each node in this graph can be described using a Polarix specification that identifies the visual representations and abstractions and can be mapped to a unique projection of the data cube, which is a data abstraction of the underlying relational data.

In the remainder of this section, we first review the two techniques we use to perform data abstraction (data cubes) and visual abstraction (Polarix). When reviewing Polarix, we also introduce a graphical notation for describing the key elements of a specification. Finally, we present how we can create a zoom graph of multiscale visualizations to describe a multiscale visualization of a

3 Multiscale Visualizations

In this section, we present our system for describing multiscale visualizations that support multiple zoom paths and both data and visual abstractions. Rather than considering multiscale visualizations as simply a series of linear zooms, we think of multiscale visualizations as a graph, where each node corresponds to a particular set of data and visual abstractions and each edge is a zoom. *Zooming* in a multiscale visualization is equivalent to traversing this graph. Each node in this graph can be described using a *Polaris* specification that identifies the visual representations and abstractions and can be mapped to a unique projection of the data cube, which is a data abstraction of the underlying relational data.

In the remainder of this section, we first review the two techniques we use to perform data abstraction (data cubes) and visual abstraction (*Polaris*). When reviewing *Polaris*, we also introduce a graphical notation for describing the key elements of a specification. Finally, we present how we can create a zoom graph of multiscale visualizations to describe a multiscale visualization of a

3 Multiscale Visualizations

In this section, we present our system for describing multiscale visualizations that support multiple zoom paths and both data and visual abstraction. Rather than considering multiscale visualizations as simply a series of linear zooms, we think of multiscale visualizations as a graph, where each node corresponds to a particular set of data and visual abstractions and each edge is a zoom. Zooming in a multiscale visualization is equivalent to traversing this graph. Each node in this graph can be described using a Polarix specification that identifies the visual representation and abstraction and can be mapped to a unique projection of the data cube, which is a data abstraction of the underlying relational data.

In the remainder of this section, we first review the two techniques we use to perform data abstraction (data cubes) and visual abstraction (Polarix). When reviewing Polarix, we also introduce a graphical notation for describing the key elements of a specification. Finally, we present how we can create a zoom graph of multiscale visualizations to describe a multiscale visualization of a set

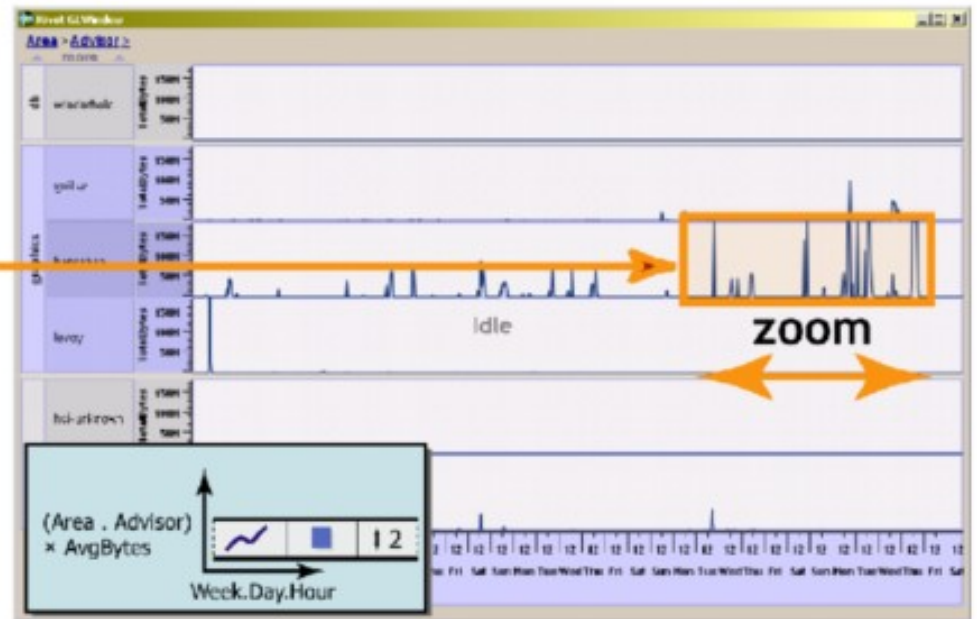
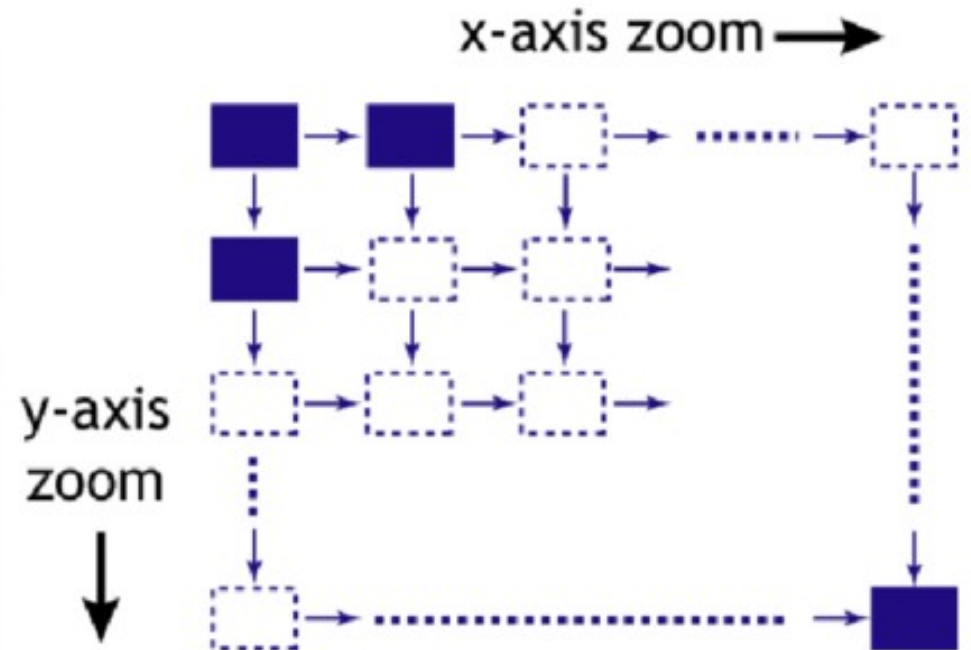
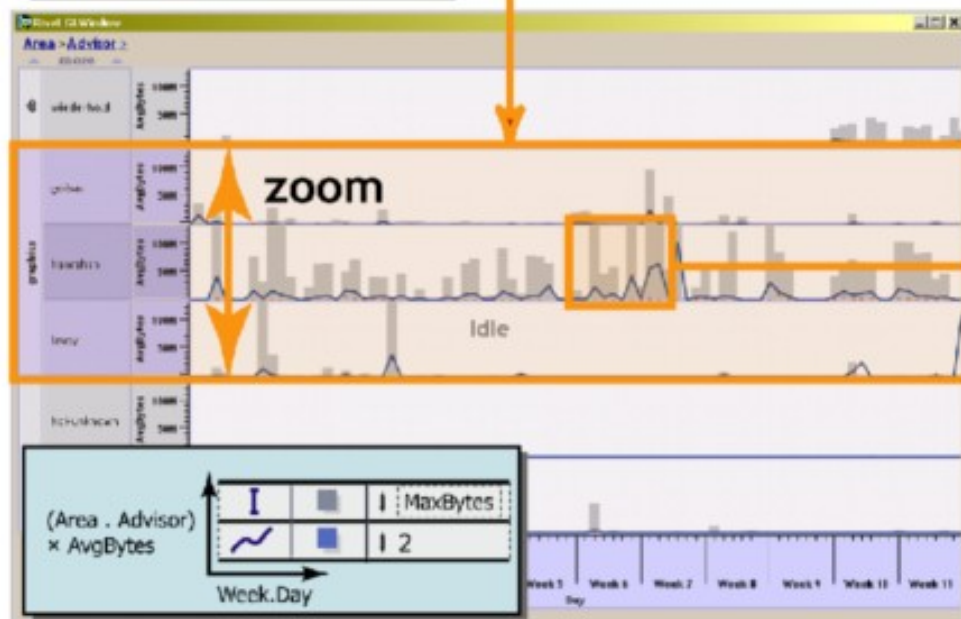
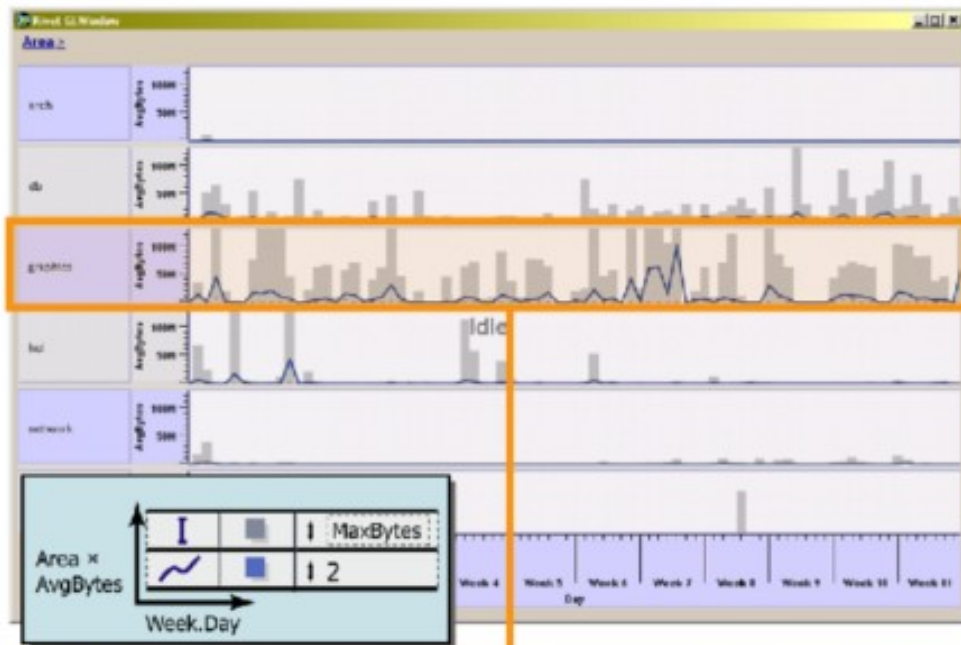
3 Multiscale Visualizations

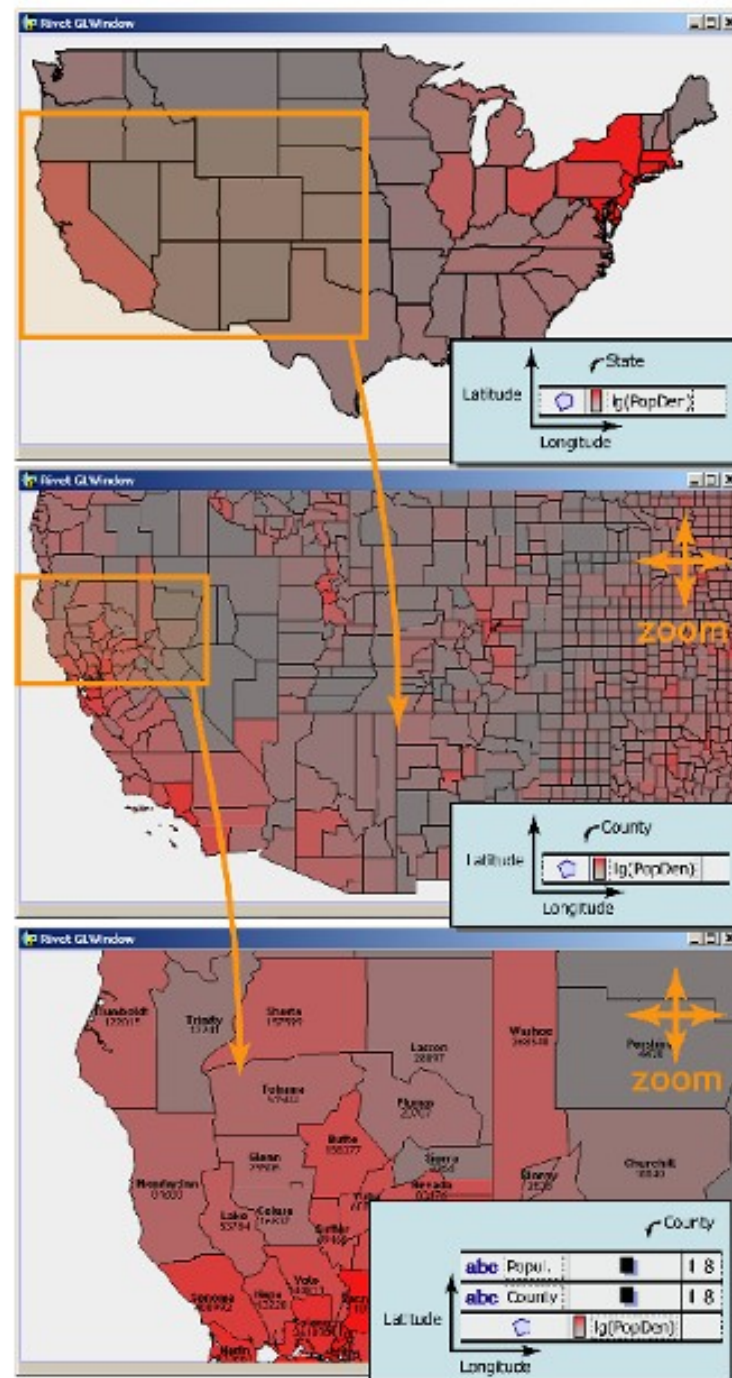
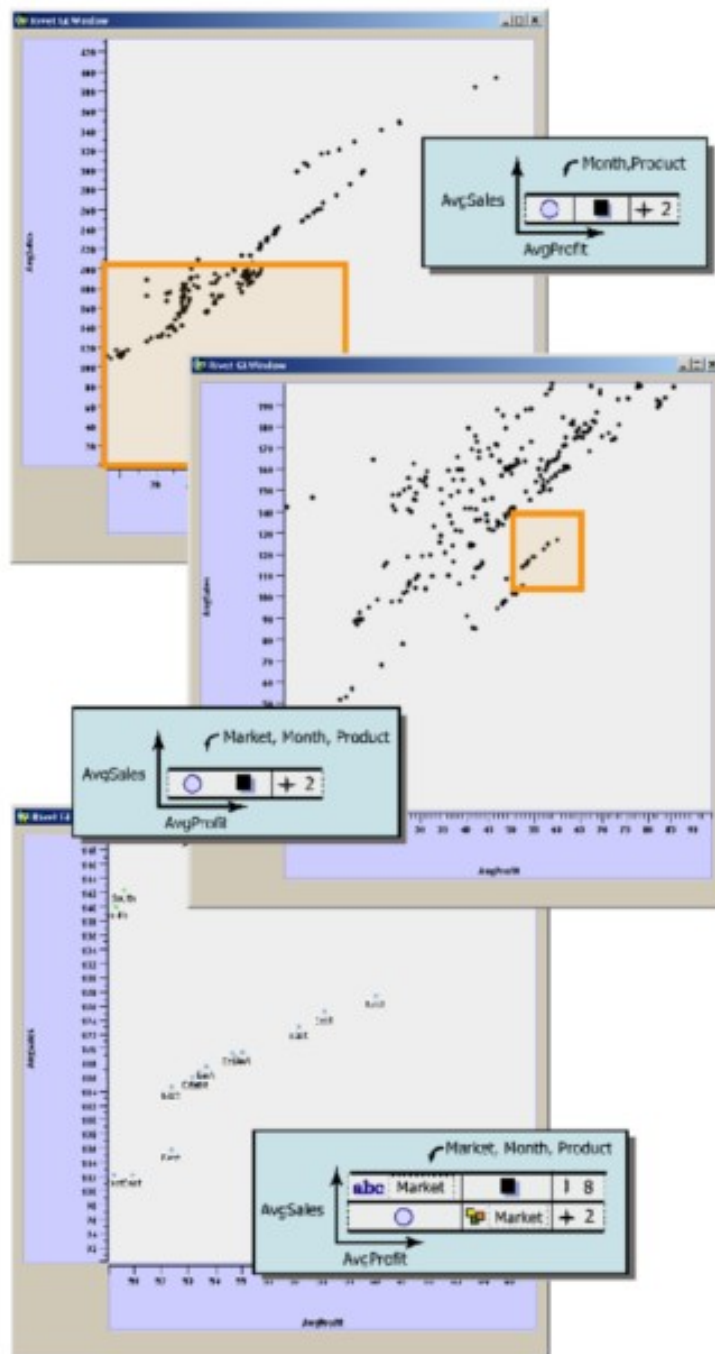
In this section, we present our system for describing multiscale visualizations that support multiple zoom paths and both data and visual abstraction. Rather than considering multiscale visualizations as simply a series of linear zooms, we think of multiscale visualizations as a graph, where each node corresponds to a particular set of data and visual abstractions and each edge is a zoom. Zooming in a multiscale visualization is equivalent to traversing this graph.

Each node in this graph can be described using a Polaris specification that identifies the visual representation and abstraction and can be mapped to a unique projection of the data cube, which is a data abstraction of the underlying relational data.

In the remainder of this section, we first review the two techniques we use to perform data abstraction (data cubes) and visual abstraction (Polaris). When reviewing Polaris, we also introduce a graphical notation for describing the key elements of a specification. Finally, we present how we can create a zoom graph of multiscale visualizations by describing a multiscale visualization of a

Traversing the Zoom Graph





The End.