47

SpreadCube: A Visualization Tool for Exploratory Data Analysis

by

Tichomir Gospodinov Tenev

Submitted to the Department of Electrical Engineering and Computer Science in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

January 29, 1997

Author ____
Department of Electrical Engineering and Computer Science
January 30, 1997

Certified by ____
Ronald Mac Neil
Thesis Supervisor

Accepted by ____
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

SpreadCube: A Visualization Tool for Exploratory Data Analysis

by

Tichomir Gospodinov Tenev

Submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the requirements for the Degree of Master of Engineering in Electrical Engineering and Computer Science.

Abstract

SpreadCube is a tool for visualizing and making sense of large multidimensional data sets. It integrates the use of visualization techniques such as focus+context (fisheye view) with On-Line Analytical Processing (OLAP) techniques for dealing with multidimensional data models. SpreadCube fuses textual, graphical and audio representation of data to produce a powerful tool for Exploratory Data Analysis (EDA). Thus, a column of numbers can be at the same time a scatter plot; similarly, a matrix of numbers is simultaneously viewed as a surface in three dimensional space. The symbiosis between data and graphics naturally leads to a direct manipulation and context sensitive user interface. SpreadCube is a demonstration how the appropriate combination of simple presentation and data analytical techniques results in a tool which is more than simply the sum of its components.

Thesis Supervisor: Ronald Mac Neil (Massachusetts Institute of Technology)
Title: Principal Research Associate

Thesis Supervisor: Ramana Rao (Xerox Palo Alto Research Center)
Title: Member of the Research Staff

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 What is "Exploratory Data Analysis"?

This thesis is about a tool for making sense of data.

The idea of using tools to aid our understanding of information, is at least as old as the invention of letters and numerals. Ever since, humanity has been creating preconceptions on how information should be represented and manipulated. For example, simply writing down a column of numbers to "get a feel about them" constitutes the most basic act of information visualization. Sorting the numbers in the column is an example of a manipulation that changes the particular way in which information has been represented. By performing similar kind of manipulations one hopes to arrive at a representation which is more revealing about what it has to say. This process is known "Exploratory Data Analysis" or EDA.

Not surprisingly, the field of Exploratory Data Analysis, spun off from the field of statistics. Indeed both areas deal with analyzing data. The principle difference, however, is the following: While statistics is about measuring the truthfulness of a hypothesis, exploratory data analysis is what formulates that hypothesis in the first place. This, in fact, is the essence of the scientific method according to which one states a hypothesis and then proves it. That is why EDA is always done in the context or as a prelude to some scientific undertaking, rather than for its own sake. The various techniques, however, which people have used for data exploration across different areas of science, have experienced a convergent evolution. The endeavor to formalize and generalize such techniques have given rise to the field of Exploratory Data Analysis.

In the dawn of the electronic age, EDA was done like this:

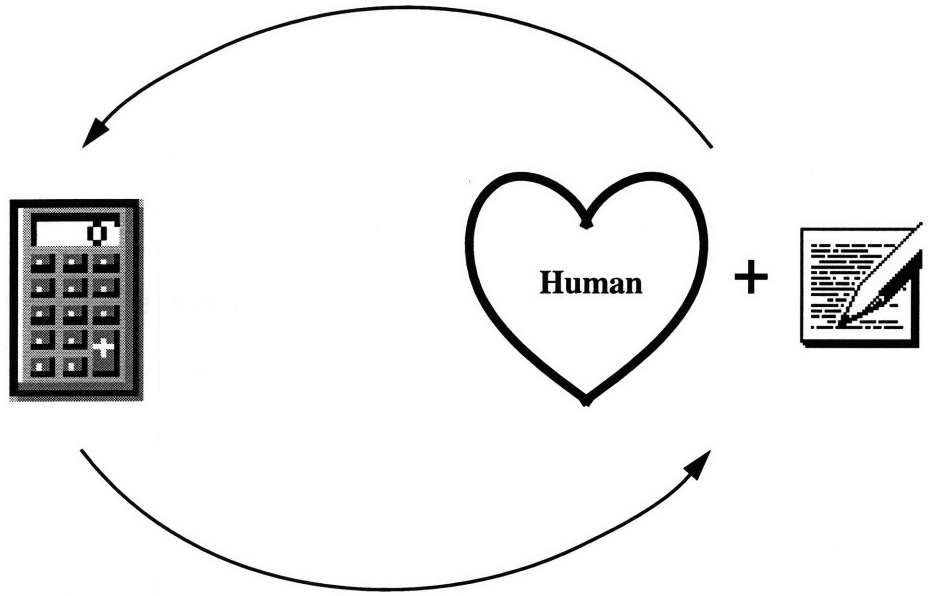


Figure 1.1: A simple tool for doing EDA

The bulk of the process represented on figure 1.1 is performed by the human operator who would often use a pencil and paper to store or sketch the intermediate results. The language used for communication between the operator and the calculator is primitive. That is why the information flow, represented by the two arrows on the figure, is of very limited bandwidth.

Naturally, people dreamt of better ways of analyzing data, in a technological age, where machines will do most of the thinking:

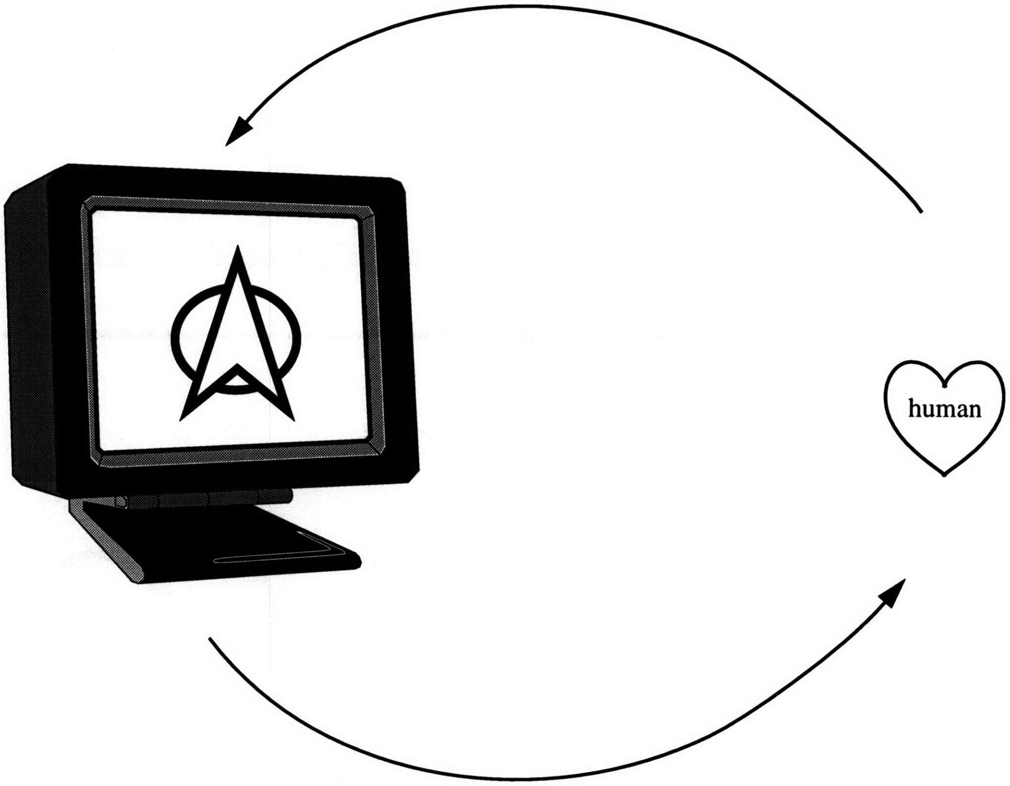


Figure 1.2: An intelligent tool for doing EDA

In the process depicted on figure 1.2, most of the work is done by the computer. In fact the computer is so intelligent, that it only needs to convey the high level ideas about its discoveries to the human. The human too needs only convey brief instructions specifying the direction of exploration.

EDA techniques, however, are only but a part of the data exploration process. In reality, the success of this process ultimately depends on the subjective judgement of the people who are involved in it. It is up to their intuition and experience to formulate reasonable theories about the observed data. This fact is illustrated on both figure 1.1 and figure 1.2: the human is always a part of the picture. Regardless of how intelligent the tool is, it needs human guidance to direct its research in order to produce results and formulate hypothesis which are meaningful to humans.

Therefore rather than seeking to create a tool which can solve all our problems of information understanding, it is more reasonable to invest into perfecting the human-computer symbiosis. The real goal should be to achieve a relationship, in which the computational power of the machine is used to enhance the human intuition, while at the same time, human intuition is used to direct the computational power of the machine:

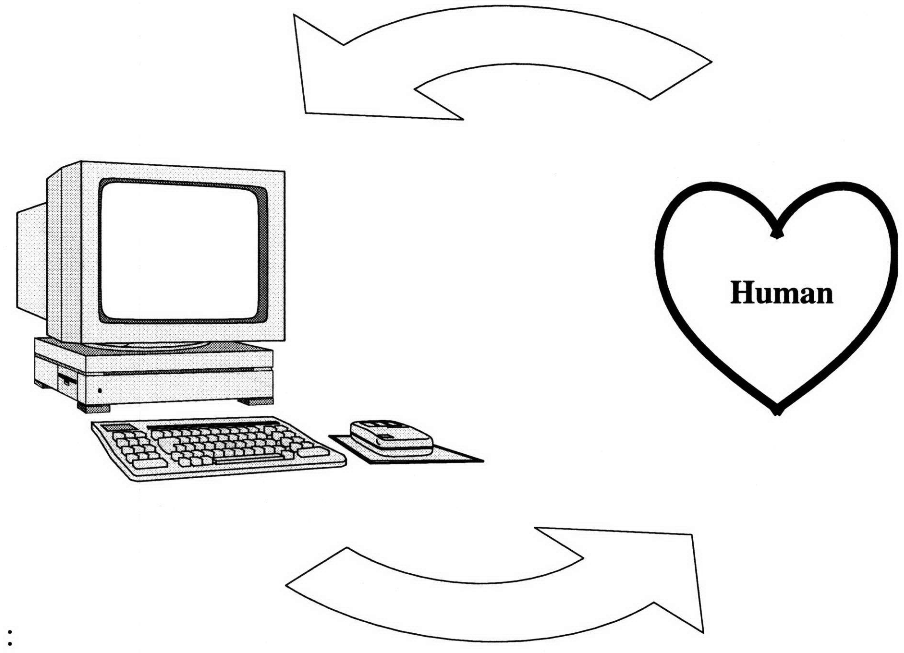


Figure 1.3: EDA through intense computer human interaction

In the scheme depicted on figure 1.3, both the computer and the human operator share significant parts of the work. Furthermore, the relationship is highly interactive compared to the one illustrated by figure 1.1 or figure 1.2. The operator initiates the process by invoking a representation of the data set. The representation, created by the machine, lets the operator to formulate hypotheses; the operator in turn, directs the machine to create new representations based on these hypotheses and so on. The thick arrows on figure 1.3 represent the rich flow of information. The top represents the user interface, and the bottom one represents the data visualization.

The next two section introduces some general issues about designing suitable visualizations and user interfaces.

1.2 Designing Visualizations and User Interfaces.

This thesis is also about data visualization and computer-human interaction. When designing visualizations one must consider the following factors:

People's preconceptions about what data should look like. The visualization is going to be more self-explanatory if it explores familiar representations of data. For example, most people are used to data being represented in tabular or graph format. Therefore, it is easiest for them to quickly find their way around when presented with a graph or a table. Even though, each of these representations may not be the richest possible one, it serves to provide a backward compatibility between the data explorer and the representation format.

The capabilities of the visualization media. Both a sheet of paper and a computer screen can be used as a visualization media. The two, however, have different strengths and weaknesses and because of that, they require different visualization techniques in order to be used most effectively. For example, the computer image does not have the high resolution of the paper image. However, it can be updated much faster and can be animated. That is why, often it is not the best approach to apply visualization techniques which work best in one medium, e.g. paper, directly to another medium, e.g. a computer screen.

The dynamics of the visualization. Human perception is not a static, snapshot-like process. We can detect both spatial and temporal correlations within a representation. Therefore, certain dependences within the visualized data set can be expressed as cause and effect relationships over time. An example of exploring this idea, could be demonstrating how a portion of the visualization perturbs as a result of perturbing another portion.

Increased information content without clutter. When designing visualizations one must be aware of the types of information representations which can co-exist without obscuring each other. Thus, by using multiple such representation, one will be able to increase the content of the represented information without at the same time increasing clutter.

Responding to and directing human's attention. Maintaining a high bandwidth of communication between the computer and the human requires that the gates of human consciousness remain open and facing the right direction. The components of the visualization which are in the focus of attention should reveal greater detail, at the expense of the rest. As the focus of human attention changes, the visualization should be able to respond accordingly. In addition interesting visual patterns which are likely to capture the attention of the human should correspond to interesting information patterns, or else they will be a distraction.

Directness of manipulation. The human directs the operations of the computer through a set of user interface controls. Therefore, to increase human's responsiveness, the user interface must allow for quick and intuitive translation from human's intentions to the appropriate UI actions.

1.3 The SpreadCube Approach

"SpreadCube" is the exploratory data analysis tool which is the subject of this thesis. The approach to data analysis which SpreadCube undertakes, can be generalized over a wide range of data management and data analysis tools. It is summarized in the following sequence of steps:

1.3.1 Define the Abstract Data Model

First, we conceive of an abstract data object which will be the placeholder for the data to be analyzed. For example, in the case of a spreadsheet based EDA tool, the abstract object is an infinite two dimensional array of cells some of which are filled with data values.

The so conceived abstract data object is an instance of a data model. A data model is a particular way of organizing the data, which is expressive of certain semantic relationships within that data. The choice of data model depends on the type of semantic relationships which we are interested to make expressible, the complexity of the model which we can handle, and the degree of generality which we need to maintain.

Some semantic relationships within the data are more apparent than others and can be readily expressed within the structure of the data model. The goal of EDA, in the context of a data model, is to manipulate the abstract data object in such a way that more of the useful information implicit in the data becomes explicit and expressible as part of the data model structure.

1.3.2 Define Operations on the Data Model

Once we have an idea of how data should be organized according to a data model, we must define the set operations on that model which access and reorganize the data.

In general, there are two types of operations: schema operations and data interpretation operations. The former are not concerned with the actual data values, but only with rearranging and accessing them. The latter depend on the contents of the data to modify or extend the structure of the data object.

Both the SpreadCube abstract data model and the operations which it supports are discussed in greater detail in chapter 3.

1.3.3 Define a Visual Model and Its Operations

In most cases, the structure of the abstract data model, could turn out to be unsuitable for direct representation on a visual media. For example, in the case of SpreadCube the abstract data model is *N*-dimensional, where *N* can be larger than 3. Therefore, one must construct a less abstract representation of the data model which relates closely to the abstract model, but at the same time can be directly rendered on a visual medium.

Such less abstract counterpart of the data model is the visual model. The abstract data model is to the visual model what the Earth's globe is to its stereographic projection. Any operation on the elements of the data model has its counterpart operation on the elements of the visual model. But just like a stereographic projection which introduces misleading information about the earth's geometry, similarly a visual model introduces false relationships and even extra structure which does not exist in the abstract model.

The operations on the visual model are invoked by human actions through the user interface and ultimately result in operations on the abstract data model. A more detailed description of the visual model is presented in chapter 4.

1.3.4 Create a Visualization and User Interface

The visual model describes how the components of the abstract model should be laid out on the visual medium. A "visualization" is an instance of rendering those components onto the medium thus making them visible and accessible through the user interface.

The user interface translates human actions, like mouse clicks and keyboard strokes, into operations on the visual model. It uses the current visualization as a context for providing the arguments to those operations.

The SpreadCube visualization and interface are discussed in greater detail in chapters 5 and 6.

1.3.5 Worry About Data Input

Even though, the input of the data into the EDA tool is not part of the EDA itself, it is nevertheless an important practical issue to be dealt with. Just like, an exploratory expedition does not expect to find roads in the jungle, a data exploration tool should not rely on fixed input format. That is why, for example, SpreadCube provides a syntax for describing the layout of the input data file.

1.4 Overview

1.4.1 World View

SpreadCube's abstract data model is based on the view of the world which it adopts. According to this view, the "world" is some complex system, which can be in any one of a set of states, and in any state one can observe the values of a fixed set of parameters.

The data model uses a fixed number of *dimensions* to refer to a particular state of the system, and a fixed number of *variables* to describe the observations at each state. The ultimate goal of exploratory data analysis, according to this model, is to understand the relationships amongst variables and the relationships between variables and dimensions. Each dimension contains a set of *dimension keys* and each variable is a set of *values*. Thus, a N-tuple of dimension keys uniquely identifies a state of the system in a N-dimensional data model. Similarly, a K-tuple of values describes the observations at each state for a K-variable model.

1.4.2 The Big Picture

The diagram on figure 1.3 describes the realization of exploratory data analysis as a computer-human interaction loop. The SpreadCube's implementation of this loop is sketched on figure 1.4.

The abstract data model is a multidimensional array of values, which are indexed by the dimension keys. That's why it is pictured as a cube on figure 1.4. The abstract data model is represented by a visual model.

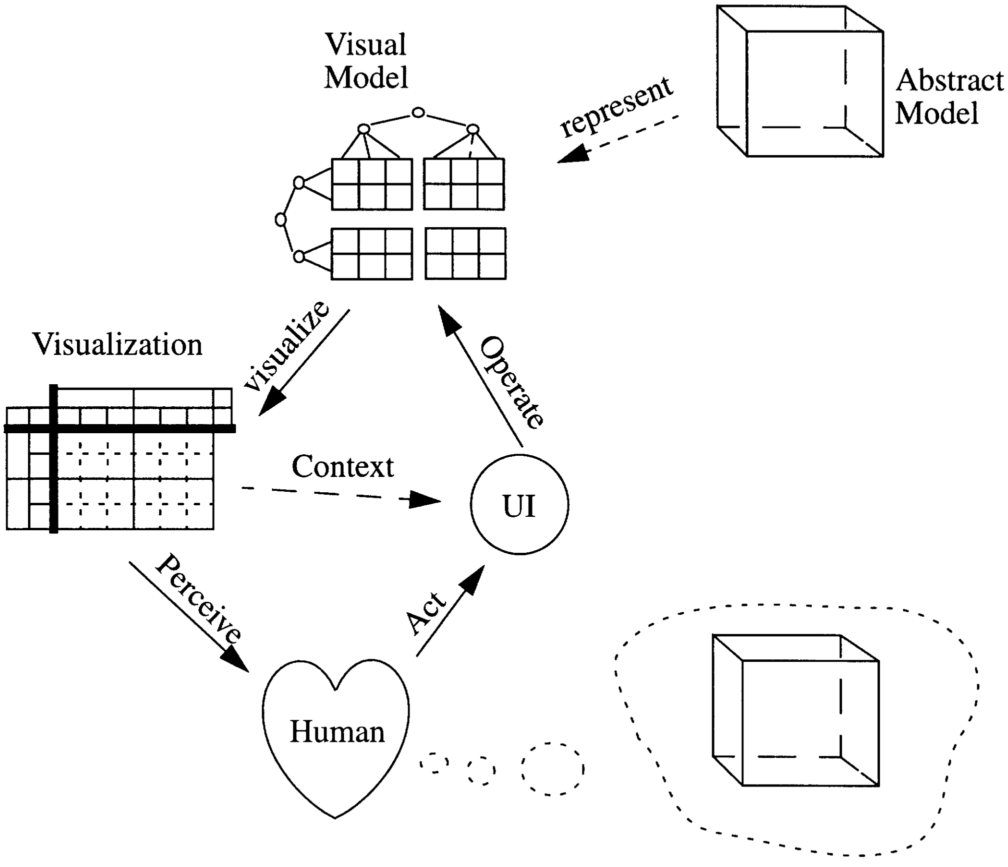
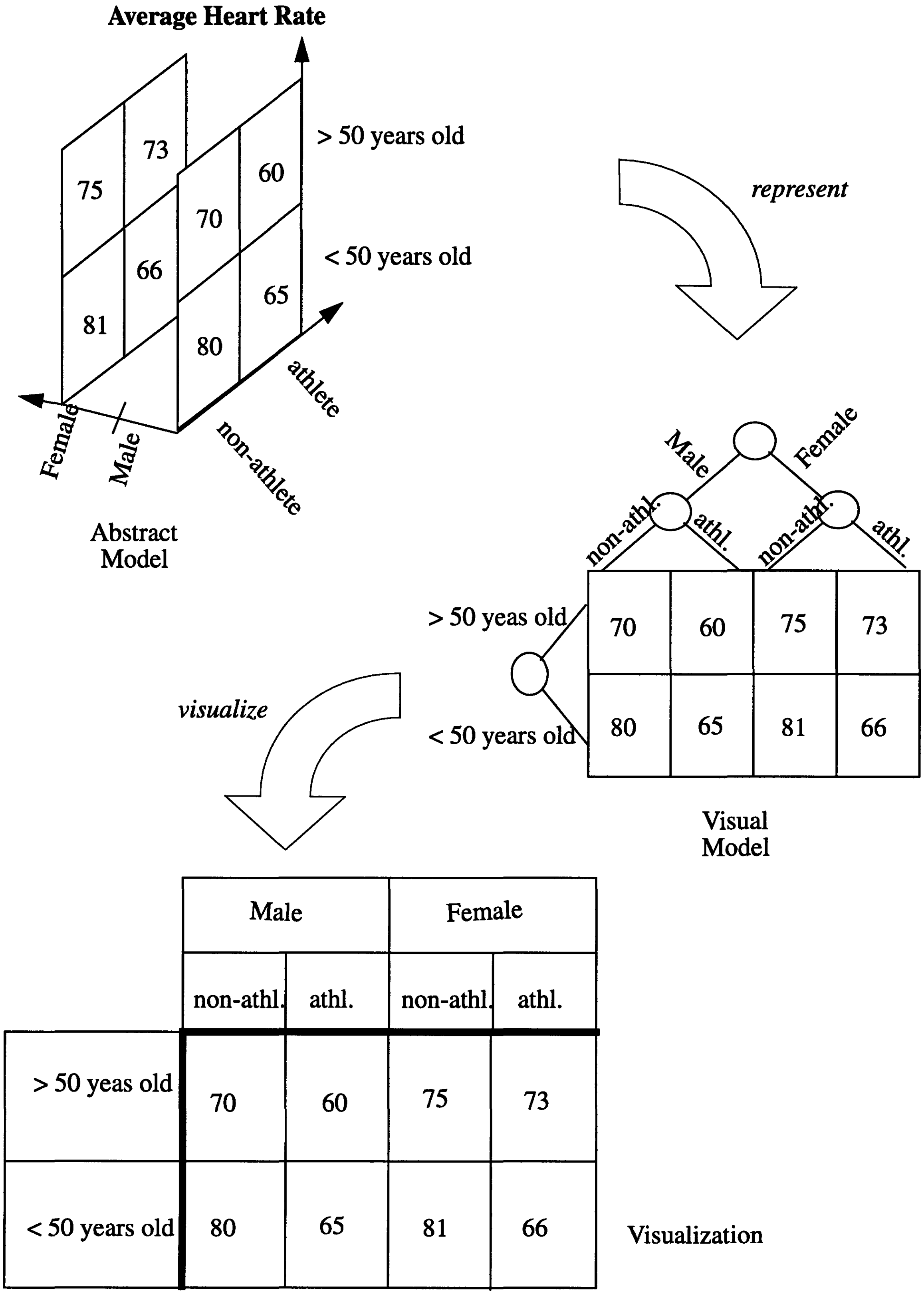


Figure 1.4: The Human-Computer Interaction according to SpreadCube

According to the visual model, the data is arranged into two dimensional slabs, which are indexed by a hierarchy of dimension keys. In the visual model each value continues to be addressed by N-tuple of dimension keys; however, the dimensional hierarchy conveys the false impression that dimensions may somehow be treated unequally.

The visual model is visualized on the screen as a two dimensional table whose cells contain data values. Each *axis* a of this table is a visualization of the dimensional hierarchy that is used for addressing each cell. That is how, ultimately the visualization conveys information to the human about the contents and the structure of the data set. A stylized example of this idea is illustrated on figure 1.5, where the data set consists of average heart rate values for different gender, age category and athletic abilities.



	Male		Female	
	non-athl.	athl.	non-athl.	athl.
> 50 yeas old	70	60	75	73
< 50 years old	80	65	81	66

Figure 1.5: Abstract Model, Visual Model and Visualization

In addition to conveying information about the structure and the content of the data set, the visualization also functions as a context for the user interface, which translates human actions into operations on the visual model. While the human participates in this interaction loop, he or she conceptualizes the abstract model and how it is being manipulated.

1.4.3 Illustrations

The best way to appreciate the contributions of this work, is by examining how the SpreadCube tool is used for analyzing actual data sets. Illustrations of the application of SpreadCube in three different scenarios are presented in appendix B. The three scenarios involve three different in nature data sets. The SpreadCube operations demonstrated in appendix B are described in the chapters to follow.

1.5 Contributions

- Integrating multidimensional data analysis with tabular and graphic presentation techniques in novel ways to produce a unique tool for exploratory data analysis and scientific visualization.

- Information presentation on several levels: graphics, text, color and sound at the same time.

- Bridging the gap between text and graphics representation of data A graph and a column of numbers are the same thing.

- Focus+context visualization with tri-level focus.

- Formalizing User Actions and their effect as operations on an abstract model.

Chapter 2

Related Work

2.1 The Field of Exploratory Data Analysis

Exploratory data analysis began to emerge as a separate field first in the works of John Tukey [20], who began to distill the methods for data analysis which scientists from various field were using to make sense of their data. He formulated the idea that *it is important to understand what you CAN DO before you learn to measure how WELL you seem to have DONE it.*[1]

The techniques which Tukey came up with combine methods for data presentation and preliminary analysis. For example, the method for writing down a column of numbers in a way which is most expressive evolved into a the so-called "stem-and-leaf" display where the column of numbers assumes the shape of its own histogram. Similarly a simple scatter plot graph evolved into a sequence of "box-and-whisker" plots each of which is a representation of the distribution at a plotted point.

All techniques were aimed at encoding as much information as possible using graphical and textual and positional representations. A symbol can encode information both within itself and within its position. In fact, Tukey thought that if one needs to place a mark on the sheet of paper, it might as well be a digit, which carries extra information and therefore is the most useful mark of all.

The techniques suggested by Tukey in his original works were not designed with the idea for a computer visualization. The premise was that all explorations are done by hand using a pencil and paper. Hence, the techniques were aimed at maximizing the efficiency of data representation on the flatland of a sheet of paper while at the same time minimizing

---

[1] John W. Tukey, Exploratory Data Analysis, 1977, Addison-Wesley Publishing Company., p. v

the effort on the part of the data explorer. Some of these, in particular the graphing techniques, can be readily implemented as screen visualizations. Others, like “scratching down numbers”, however, are not conveniently applicable to a computer application. The problem is that in the case of computer screen, the interaction between the data explorer and the medium is different.

With all this in mind, it appears that applying directly traditional EDA techniques from the graph paper to the electronic medium does not fully take advantage of what the electronic medium can do. This calls for the development of conceptually new techniques for computer EDA based on the principles of maximizing the efficiency of information presentation. Such techniques have been in constant development as the popularity of electronic medium has been increasing in recent years.

The SpreadCube tool for exploratory data analysis is a step in this direction. It utilizes multiple *presentation types* as a way to increase information content per given amount of screen space and to involve multiple perception channels at once. After a simple manipulation or two one can quickly assess general trends and distributions of the data displayed and formulate hypothesis.

2.2 The Table Lense

The ideas which SpreadCube visualization implements were inspired by the work of Ramana Rao and Stuart Card on *Table Lens* [2].

The Table Lens is a tool for visualizing and making sense of large tables. The Table Lens visualization introduces focus+context (or fisheye) techniques which work effectively on tabular information. The graphical mapping scheme which produces the fisheye effect can be described in terms of two degree of interest (DOI) transfer functions applied to the vertical and horizontal axis similar to the ones on figure 5.2. A DOI specifies how abstract layout is mapped onto the physical screen space according to the user's degree of interest associated with a particular part of the visualization. More specifically, there are two levels of focus, corresponding to two levels of details for data presentation, applied to the vertical and horizontal extends of the table. Thus, there are focused or unfocused rows and columns Depending on whether a cell belongs to a focused or unfocused row or column, or both, it is depicted differently. For example a cell which belongs to both focused row and column is depicted using greatest level of detail.

The Table Lens also introduces the concepts of presentation types. A presentation type is a particular way of representing information suitable for a data domain. For example a presentation type can be a textual representation, e.g. a number, or a graphical representation, e.g. a bar. The Table Lens allows for mixing several presentation types at the same time, producing an information rich visualization, which fuses symbolic and graphic representation. In later chapters, we will discuss the presentation types which SpreadCube uses and how they target different *channels* of human perception.

The data model employed by Table Lens is that of a relational table. The versatility of the relational model and the fact that a lot of data is already represented in this format makes the Table Lens a visualization tool suitable for a great spectrum of data sets. In addition, the table lense is equipped with a formula processor which allows for new variable derivation.

The Table Lens combines fairly simple data manipulation techniques like sorting and variable derivation, as well as visualization techniques, e.g. focus+context. As a result the user can have both a close-up and a panoramic view of the data, while at the same time using sorting operations to determine correlations between variables and distributions within variables. In this way the outcome of combining the simple techniques which Table Lens incorporates is more than just the sum of its parts.

The idea of capitalizing on the combination of relatively simple data manipulation and visualization techniques is taken even further in SpreadCube. The SpreadCube tool is based on structurally richer (OLAP) data model and explores a larger set of graphical techniques and presentation types which suit naturally the multidimensional OLAP model.

In addition the SpreadCube model explores the idea of increasing the number of degrees of interest for visualizing data, which leads to a tri-level focus scheme. The tri-level focus scheme allows for virtually infinite data sets to be represented on the screen because the lowest focus level permits mapping several data cells onto a single screen pixel.

2.3 The SpreadCube Model

The fundamentals of the SpreadCube data model where first outlined in a draft paper by Ramana Rao and Stuart Card [1]. The paper speculates on what the components of the SpreadCube data model should be and how the schema and the vault should be structured. It also discusses various possible data model operations and value domains. For most intends and purposes it views the SpreadCube data model as a generalization of the relational model to multiple dimensions. Hence, many of the operations which the data model proposed are extensions of relational operations to multiple dimensions.

The data model which SpreadCube currently implements follows closely the one suggested by Rao and Card with a few modifications. For example, the SpreadCube data model is coupled closely with a *visual model*, which specifies how the abstract structure should be laid out on a two-dimensional surface before it can be rendered on the screen. For many purposes the visual model can be thought of as another manifestation of the abstract data model, because it continues to embody all relationships within the data.

2.4 The OLAP Model

The data model which SpreadCube uses belongs to a class of data models known as OLAP (on-line analytical processing) models.

The need for data organization and maintenance has lead to the development of a range of data management systems: from simple files to relational data bases. While, these data management systems are well suited for transaction operations like storage and query, they do not provide convenient means for analytical data processing, such as in the case of exploratory data analysis.

To address these problem the field of on-line analytical processing was created. The OLAP model is not another data base management system, but rather an intermediary between a DBMS and a front end visualization and analysis tool, like a spreadsheet or a statistical package [7]. Typically, this configuration is implemented according to a client/server model. The implementation of the OLAP model runs on a server which services front end packages and presents them with a structurally rich data model. The OLAP server translates user requests coming from the front end packages into queries, e.g. SQL (structured query languange) statements, for the DBMS servers.

While the SpreadCube is not client/server based, it utilizes the very same structure, except that its components are integrated in a single application. The data, which is to be analyzed and visualized, is originally stored in a plain text file, using a generic format. A second, description file specifies how the raw data should be interpreted according to the abstract data model which SpreadCube uses. Once the data is loaded and structured according to this abstract data model, it is visualized in a way which is representative of the abstract structure.

The OLAP data model is a multidimensional data model. A data dimension represents a particular way of looking at the data, which allows data to be summarized, drilled and analyzed in a meaningful way. The power of the multidimensional model comes from its similarity with the natural model which people create in their minds when they think about analyzing and organizing data. For example, marketing data can be grouped and summarized according to time, product or distribution channel. Thus “time”, “product” and “channel” constitute three dimensions of that data set.

Currently, there are a number of commercial OLAP tools, such as: Lotus Improv, Comshare Commander and Brio’s DataPivot, that provide spreadsheet type of environment which supports multidimensional data sets and provide OLAP operations on them such as: slicing, aggregation and hierarchical drill-down. These tools, however, do not employ focus+context techniques or graphical representations, which greatly limits the amount of data that can be displayed on the screen.

The limits on the amount of data which can be displayed on the screen at the same time, can be a significant obstacle to understanding the data. If only a small portion of the data set is displayed at one time, the user is not able to observe at a glance long trends, or compare spatially distant sections of the data. When visualizing multidimensional data sets, which have more than two dimensions, the amount of data cells tends to grow very rapidly even if the data set is small. In such cases if the user’s view is limited to a small window onto the data which can be disorienting and one often looses track of where the current view is with respect to the dimensional hierarchy. Because of this, much of the interaction is based on the user navigating to a condensed report driven by their own model of the data as opposed to letting interesting effects in the data drive the pursuit.

This problem is solved in SpreadCube, by combining OLAP capabilities, with tri-level focus. The tri-level focus allows for virtually infinite data sets to be visualized as a whole on the screen. Focal cells are grouped into focal areas, which the user can resize or position at any place on the screen. Thus, depending on the point of view, a focus are can function as either a lens or a substitute for a scrollable window. In addition SpreadCube integrates graphical and text representation in the same view, which produces some interesting visualization effects and allows to have two- and three-dimensional plots of the data coexist with its textual representation in a tabular format.

Chapter 3

The Abstract Data Model

The goal of this chapter is to describe the SpreadCube data model and to introduce the basic terminology used in the this and later chapters.

3.1 What is an Abstract Data Model?

A data model is an archetype of data organization along with the operations for accessing and rearranging that data. The word "abstract" simply signifies the fact that the description of the model does not have to do with how data is represented in memory, or visualized on the screen. We will also use the term *dataset* in conjunction with "data model". A dataset is a specific collection of data organized according to a given data model.

Any problem which deals with data processing or data storage, must have defined, implicitly or explicitly, some kind of a data model. The requirements which a particular data model must meet depend on the operations which the data model should support and the goals of the particular problem which it addresses.

For example the relational data model was developed to provide a uniform way of storing and managing data in the form of tuples. Because of its generality, this model is well suited for data base building and maintenance. Indeed, virtually any piece of sentential information can be formatted as a tuple and stored in a relation. On the other hand, the relational model is not well suited for data analysis. Because of its simplicity, it fails to convey explicitly the multitude of ways in which the stored information can be interrelated. The relationships which exist within the data are important because they chart the paths along which the data can be aggregated, summarized, consolidated, summed, viewed and analyzed. Even though, such relationships can be inferred from the stored data itself, they are not represented within the data model.

Often it is useful to distinguish between the spacial and semantic organization of the data. In this sense, a data model has two aspects: a *vault*, which is the spatial component, and a *schema*, which is the semantic component. The vault is a container for the data; it consists of *cells* storing *data values*. The schema describes a way for accessing and interpreting the data values within the vault, both individually and collectively [1].

For example in a relational data model, the data is organized in a two dimensional table called *relation*, which is an instance of a vault. The schema, of a relational data model is a tuple of labels, describing what each column of the relation represents.

Further in this chapter, we will provide a working formal definition of the data model which SpreadCube uses. We will also use some formalism to describe the operations which the SpreadCube data model supports. However, we will not use formalism as a means of exposition, but rather as a means of resolving ambiguities. The main subject of this paper is exploratory data analysis, i.e. how we can gain insights on a data set from various perspectives, while forming different, mental pictures of it. It is therefore, important to gain a good intuition about the data model from looking at its multiple manifestations, and not simply its formal definition.

3.2 Multidimensional Data Model

The SpreadCube data model is a multidimensional model. The multidimensional model provides an organization of the data which is richer in structure compared to the relational model. That is how it can account for the various ways in which data can be interrelated.

3.2.1 Dimensions, Variables and Values

Ultimately data comes from observing a system in the physical world, which we wish to study. For example, the human body, the weather, the national economy or the marketing of a single company are all examples of such systems.

A *value* is the smallest unit of data which continues to have a meaning in the physical world. There can be even smaller units of data, like bits and bytes, but they only have to do with how a value is represented and not what it means.

A *variable* is an attribute to the physical system in consideration which gives rise to data values. For example hart rate, atmospheric pressure, gross domestic product, revenue, and etc. are all variables.

A *dimension* is an independent partitioning of the set of all values according to some aspect of the state of the system incidental with each value. Each partition, then is labeled by a dimension *key*.

For example the set of heart rate values belongs to a variable of the human body system. It can be partitioned into female and male heart rate values. Thus, “female” and “male” constitute the keys of a dimension describing an aspect of the human body system. We will call it: the “gender” dimension. A partitioning of the heart rate data into age of the human subjects, constitutes the “age” dimension. The two partitionings are independent because there can be subjects of the same age and different gender or of different age and same gender.

Likewise, the set of revenue values belong to a variable in the company marketing system. It can be partitioned according to the product which was responsible for the particular revenue value, and also according to the year when such revenue was generated. Therefore, “year” and “product” are two dimensions of the company marketing system.

A multidimensional data set comprises data from a single physical system. On, the other hand, a single physical system can be described in terms of multiple variables and multiple dimensions. For example, apart from the revenue variable, the description of the company marketing system may include other variables such as “profits” or “units sold”, and other dimensions, such as “sales channel” or “number of clients”.

To summarize: each value in a multidimensional data set belongs to a variable within a physical system that we wish to study. The context of each value observation corresponds to the state of the system incidental with that observation and is represented by a tuple of dimension keys.

3.2.2 Dimension Key Structure

A dimension key is a characterization of a relevant aspect of the system's state. A key can be *simple* or *compound.*

A simple key represents the finest interesting partitioning of the data set along its respective dimension. For example the "male" and "female" keys of the gender dimensions, are simple because they represent the finest possible partitioning according to gender.

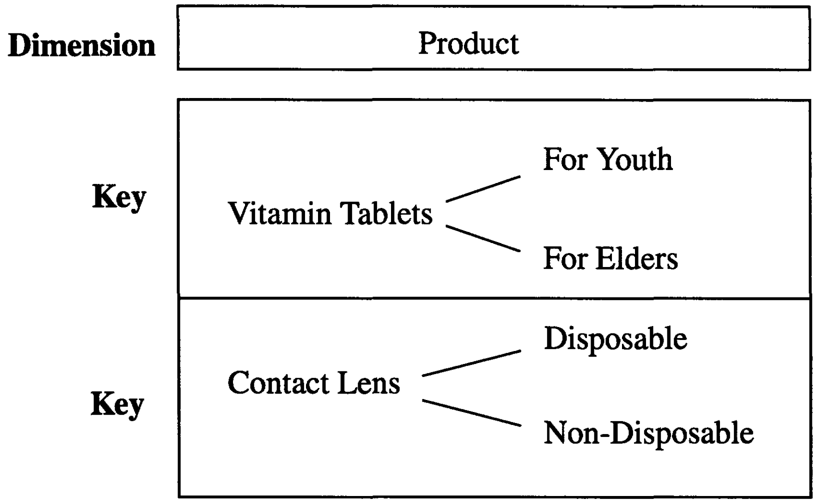


Figure 3.1: Compound Dimension Keys

A compound key is in fact a hierarchy of keys which label finer partitions. For example the partitions associated with the keys: "Contact Lens" and "Vitamin Tablets", of the product dimension, could in fact be partitioned further into: "Disposable Contact Lens" and "Non-Disposable", and "Vitamin Tablets for the young" and "Vitamin Tablets for the elder" (see figure 3.1). The attributes "Disposable", "Non-Disposable" and etc., cannot be taken out of the context of the key hierarchy, because they only apply to one dimension key, namely "Contact Lens".

There is an intermediate between simple and compound, which we will call a *pseudo-compound* key. Consider the "Year" dimension of the company marketing system. A "Year" can be further partitioned into four quarters (see figure 3.2).

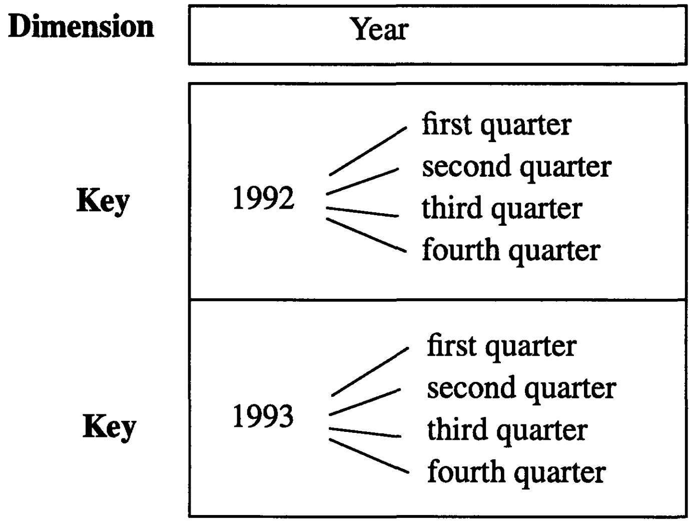


Figure 3.2: Pseudo-Compound Dimension Keys

The difference between pseudo-compound key and a compound key is that the sub-partitions of a pseudo-compound key repeat over the all keys of that dimension. For example, all years in the "year" dimension have first quarters. This situation occurs very often especially with regards to dimensions which quantify time and space. In such cases, even though it seems that a quarter is an inherent part of a particular year, it is reasonable to speak of a quarter outside the context of a that year. For example, if we study yearly fluctuations in the company's marketing, we may be interested to examine how variables, such as "profits" and "revenue" behave as functions of the quarter regardless of the actual year.

Thus, for all practical purposes, years and quarters behave as the keys of two independent dimensions (see figure 3.3)

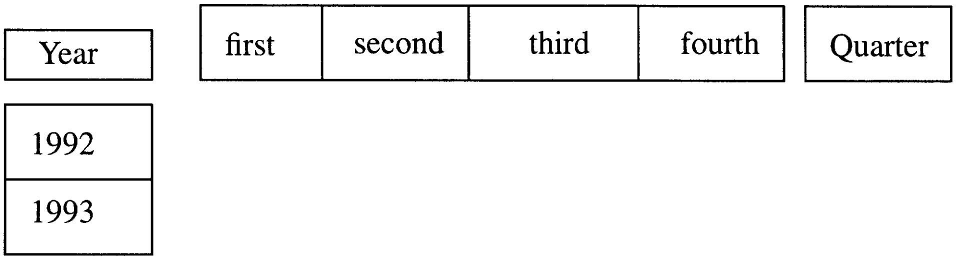


Figure 3.3: Pseudo-Compound Keys as belonging to Independent Dimensions

3.2.3 Dimension Variables

Often we are faced with analyzing the causal relationship between two or more variables, e.g. the causal relationship between the "units sold" and "profits" variable. In such case we can think of the values of one set off variables, e.g. "units", as characterizing the state of the system when the values of another set of variables were observed, e.g. "profits". This corresponds to using the former set of variables as an extra dimension along with all other dimensions of the system. That is why we will call variables used in such way *dimension variables.*

3.2.4 Record and Set Dimensions

Integrating values from different variables into a single data set can be handled uniformly in the spirit of the multidimensional data model by designating a special dimension whose keys correspond to the existing variables. We will call this the *record dimension* and all other dimensions will be called *set dimensions.* Thus, a set of values whose dimension keys are all equal, except for the record dimension key, are incidental with the same system state and represent a *record* of observations; hence the name "record dimension". On the other hand, a set of values with the same record dimension key, are a *set* of observations of the same variable; hence the name "set dimension".

The keys of a record dimension are *variable descriptors*. Each variable descriptor consists of a *variable label*, e.g. “profits”, and a *value domain*. The value domain characterizes the type of operations that are meaningful to the values of the variable referred to by the variable descriptor.

3.2.5 Value Domains

Depending on the variable to which a value belongs, it makes sense for it to participate in certain data operations and not others. For example, the values of the “revenue” variable can be added together or ordered, whereas the values “red”, “green” and “blue” of the “color” variable cannot.

Values which belong to the same variable should be able to participate in the same data operations. In fact, depending on the data operations of the current data model, it may be possible that the same set of operations is applicable to the values of several variables. Therefore, we can classify values into *value domains*, according to the set of operations which are meaningful for them. Clearly, the set of existing domains depends on the set of operations which the data model has.

Here is a list of example value domains [1]:

- **Nominal**. These values belong to a finite set. They can only be tested for equality but cannot e ordered. For example the three primary colors are nominal values.

- **Ordinal.** Ordinal values are similar to nominal, except in that they can be ordered. For example “first place”, “second place” and “third place” are such values.

- **Quantity.** These values represent an amount of something. They can be tested for equality and can be ordered. In addition they can be added together or multiplied by a fraction. For example height, pressure, profits, etc. are variable with quantity values.

- **Textual.** These are values which support typical string operations, like concatenation and search. For example the values of the variable "remarks".

- **Structured.** These are not simple values but rather structured entities of arbitrary complexity. The exact operations meaningful for them will depend on the particular structures which they implement. A structured data value can be as simple as a pair of numbers or it can be an entire data set.

3.3 The SpreadCube Data Model

In the previous section we have described the general notion of what the multidimensional data model is and what are its components. The SpreadCube data model is an instance of the multidimensional data model for which we have made specific choices of, representation, operations, value domains, and terminology. Even though the SpreadCube model is more specific than the general idea of a multidimensional data model described in the previous section, we will continue to refer to it as the "abstract" data model to distinguish it from the "visual" model described in the next chapter.

3.3.1 Dimensions as Spatial Entities

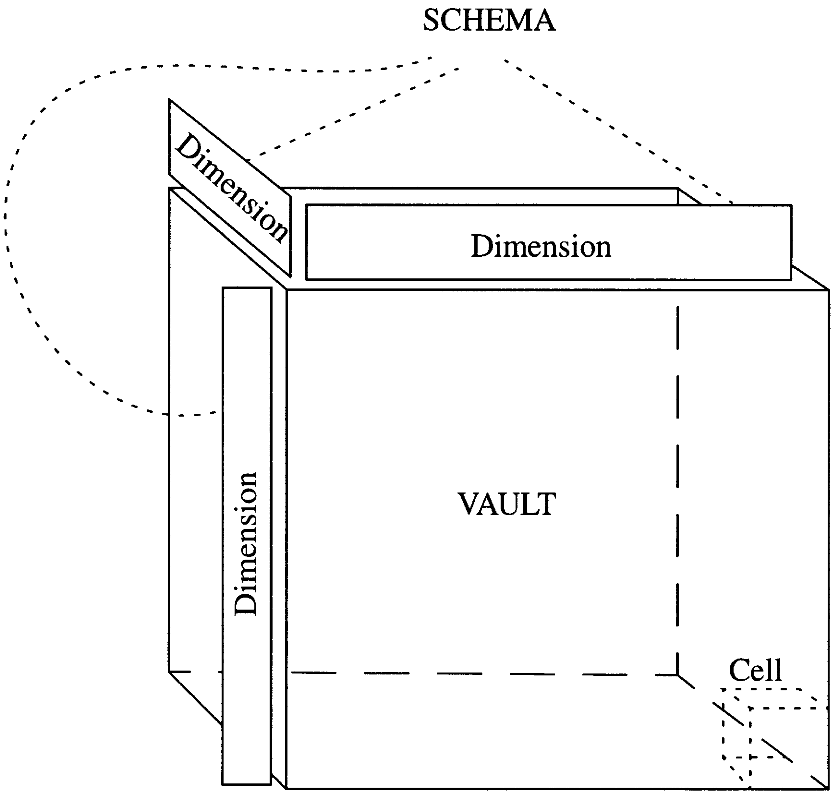


Figure 3.4: A Multidimensional Data Set

The structure of an N-dimensional data set can be conveniently expressed in terms of spatial relationships, if we map the N dimensions to N orthogonal spatial extends. In such case the data values are organized into a N-dimensional array as the one shown on figure 3.4.

According to such view of the multidimensional data model, the vault is a multidimensional array of *cells*. A cell is a placeholder for a single value; the context in which the value was observed is encoded by the spacial position of the cell.

The schema consists of N *dimension* objects. A dimension object has of a list of unique *keys* and a *label* (see figure 3.5). The label is the name of the dimension. The keys correspond to spatial coordinates. Therefore, since each cell in the vault is specified by a

N-tuple of spatial coordinates, it can be also specified by an *N*-tuple of dimension keys. The correspondence between spatial coordinates and dimension keys is what establishes the correspondence between the cell's spatial position and the interpretation of its value.

3.3.2 Sticks and Slabs

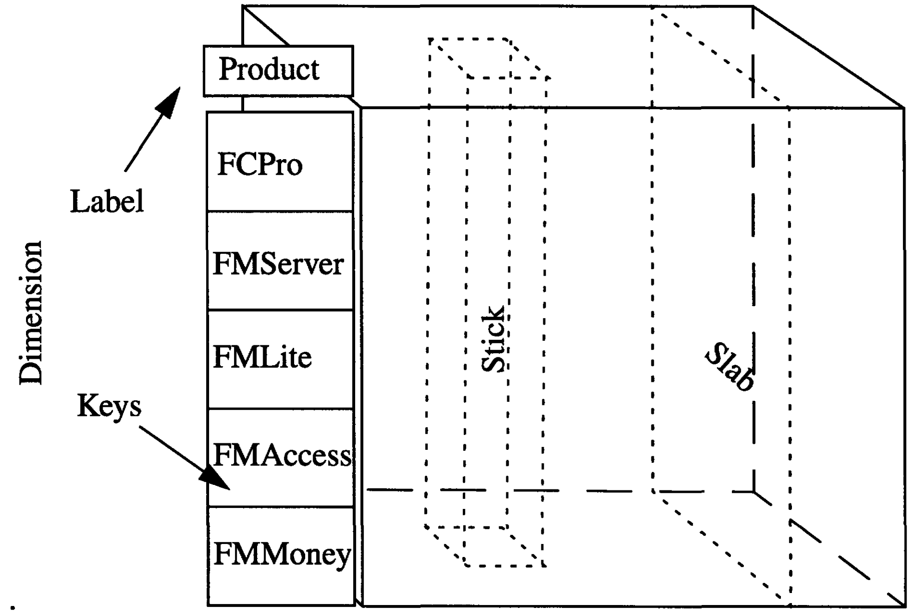


Figure 3.5: Dimensions, Sticks and Slabs

A unidimensional group of cells parallel to a dimension is called *stick* [1]. The cells within a stick share all but one spatial coordinates, i.e. they share all but one dimension keys.

A two dimensional group of cells parallel to two of the dimensions is called *slab*. The cells within a slab share all but two of the dimension keys.

3.3.3 Data Consolidation Paths

One can think of a slab as the result of grouping parallel sticks together. In fact, the grouping of sticks into slabs and slabs into 3-dimensional blocks and so on, correspond to the successive levels in a data consolidation path. This is demonstrated on figure 3.6

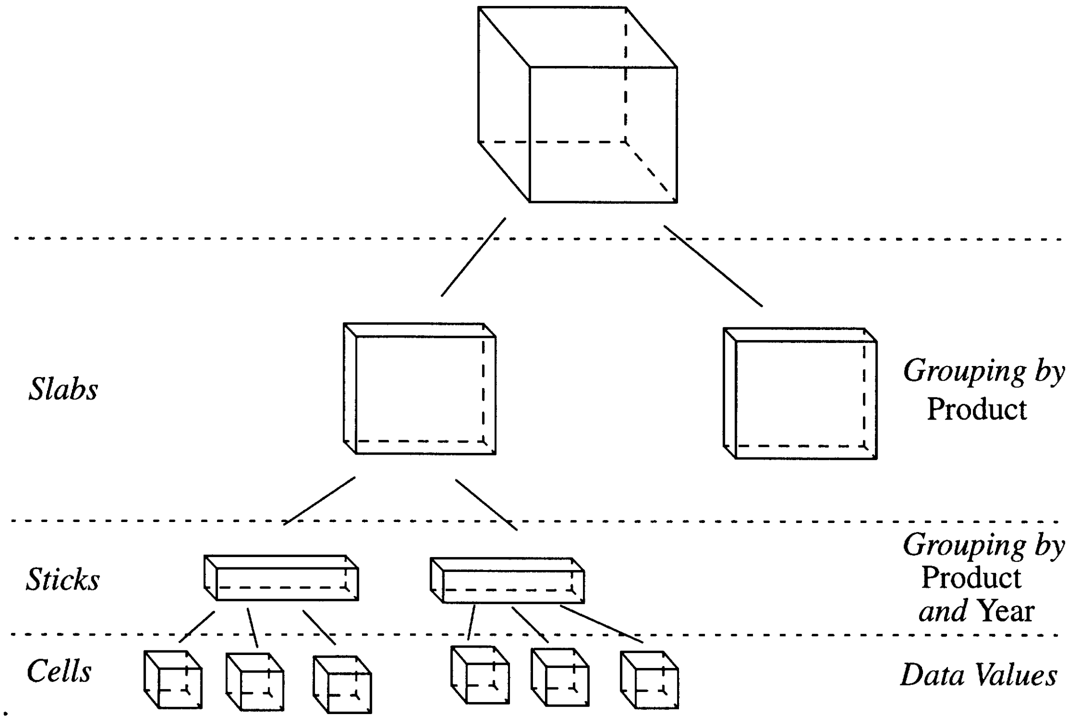


Figure 3.6: Levels of a Data Consolidation Path as Spatial Entities

The concept of a “data consolidation path” originates from the field of on-line analytical processing (OLAP) [7]. In the OLAP field this concept is used to describe what a dimension is.

Cells whose values belong to the same variable, form *(N-1)*-dimensional planes perpendicular to the record dimension (figure 3.7).

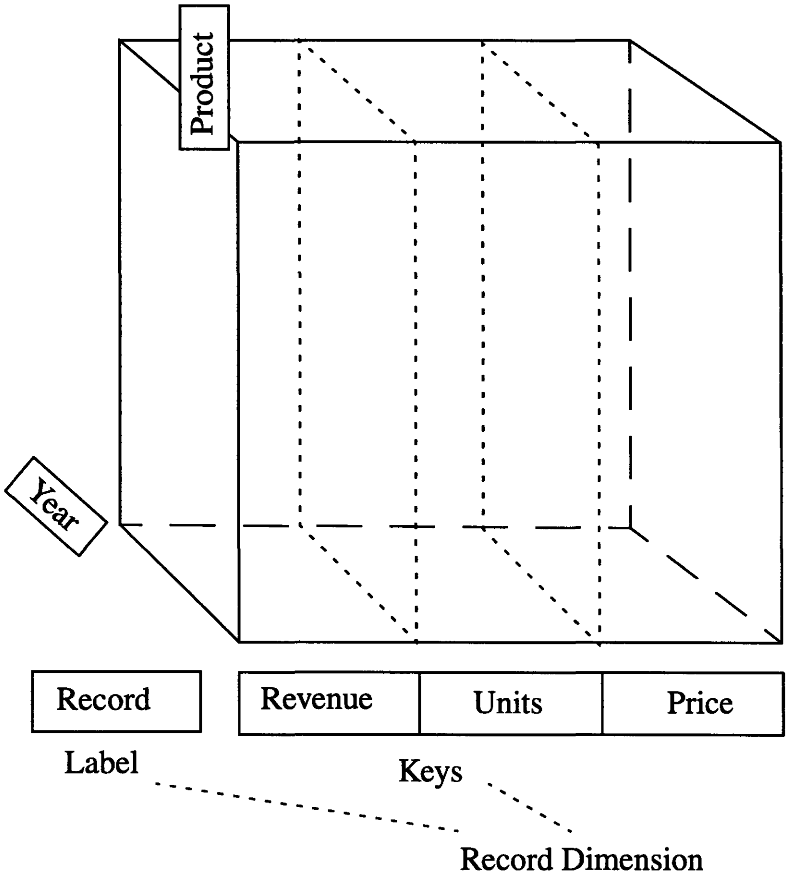


Figure 3.7: Record Dimension and Variables

In the case of two dimensional data set, these *(N-1)*-dimensional planes degenerate into unidimensional columns which resemble the columns of a relation in a relational model. In fact, for a lot of purposes, one can think of the SpreadCube data model as embedding the generalization of the relational model to multiple dimensions.

3.3.4 Value Domains

Of the example value domains listed in subsection 3.2.5 the SpreadCube data model uses *quantity*, *nominal* and *textual* value domains. The choice of these value domains was dictated by the set of data operations that the SpreadCube model has.

Values from other possible value domains, except for the structured domain can be cast into one of these three domains. For example ordinal values can be treated as quantities or nominals depending on the types of operations which one has in mind for them. Structured values are not handled because of the amount of complexity which they introduce which is beyond the scope of this paper. Furthermore, SpreadCube's philosophy is to express all of the data structure as part of the structure of the dataset and not within individual values.

3.3.5 Missing and Non-existent Values

Due to irregularities of the input data some cells in the vault may end up empty. In order to handle such irregularities uniformly, we will fill the empty cells with one of the two special values: *missing* or *nonexistent.* These values can belong to any domain. A missing value is placed in a cell for which an actual data value exists, but it is not available. A nonexistent value is placed in a cell which cannot possibly contain an actual data value because the existence of such value contradicts the semantics of the data. Such cases occur, for example, when a variable is used as a dimension.

3.4 Notation and Definition

In order to describe accurately the operation on the SpreadCube, we will introduce a sentential description of the data model:

$$\begin{aligned} C &\equiv \langle S, V \rangle \\ V &\equiv \| v[i_1, i_2, \ldots, i_N] \| \qquad 1 \le i_j \le M_j \qquad 1 \le j \le N \\ S &\equiv \langle D_1, D_2, \ldots, D_N \rangle \\ D_j &\equiv \langle K_j, L_j \rangle \\ K_j &\equiv (k_j[i_j]) \end{aligned} \tag{3.1}$$

We will use C, V, and S to denote a SpreadCube data model, a Vault and a Schema respectively. In equation (3.1) the delimiters $\langle\ \rangle$ denote a tuple, $\|\ \|$ denote an array and $(\ )$ denote a list and $[\ ]$ is used to denote indices, when it is not convenient to write the indices as subscripts. We will use the letters i and j to as running indices, and for fixed parameters we will use letters like d or p. Thus a schema S consists of N dimensions: $D_1,\ldots, D_N$. Each dimension is a pair of a key list K and a label L. The integers M_j denote the size of the dimensions D_j respectively.

Using the above notation, we can define a stick Q and a slab B as:

$$\begin{array}{lll} Q_1[d_2 \ldots d_N] \equiv \|v[i_1, d_2, \ldots, d_N]\| & 1 \le i_1 \le M_1 & \\ B_{1,2}[d_3 \ldots d_N] \equiv \|v[i_1, i_2, d_3, \ldots, d_N]\| & 1 \le i_1 \le M_1 & 1 \le i_2 \le M_2 \end{array} \tag{3.2}$$

For the sake of simplicity equation (3.2) assumes that the stick Q is parallel to the dimension D_1 and that the slab B is parallel to dimensions D_1 and D_2. This fact is denoted by the indices of Q and B.

3.5 Operations on the SpreadCube Data Model

The notation which was introduced in the previous section is used to provide a formulation for each of the operations below. Each formulation is laid out in a pre-post condition form:

$$\begin{array}{c} \{\text{Pre-Condition}\} \\ \text{Operation} \\ \{\text{Post-Condition}\} \end{array}$$

We will use the convention that the pre-post condition is applied iteratively over all indices which appear both on the left and right hand side of an equation. Also, all rules are applied iteratively over all indices i, wherever, i may occur.

The operations on the data model, described in this section, all have the effect of producing a new data set where either the contents or the structure is different. In the chapters to come we will discuss operations which merely affect the way the data is visualized but not its structure or content.

3.5.1 Schema Transformation Operations

The following operations apply only to the schema of the data set, i.e. they are independent of the vault contents.

Reorder

Arguments: **dimension**, **permutation**

Description: Permutes the *(N-1)* dimensional planes of cells perpendicular to **dimension**, according to the specified permutation.

Formulation:

$$\begin{gathered} \{\ \} \\ \mathrm{Reorder}(D_j, (p_1, \ldots, p_{M_j})) : \langle S, V \rangle \to \langle S', V' \rangle \\ \left\{ \begin{matrix} v'[i_1, \ldots, i_{j-1}, i_j, i_{j+1}, \ldots, i_N] = v[i_1, \ldots, i_{j-1}, p_{i_j}, i_{j+1}, \ldots, i_N] \\ k'_j[i_j] = k_j[p_{i_j}] \end{matrix} \right\} \end{gathered} \tag{3.3}$$

where $(p_1, \ldots, p_{M_j})$ denotes a permutation.

Transpose

Arguments: **permutation**

Description: Permutes the coordinates of the cells in the vault, according to **permutation**. This operation does not change the position of data values with respect to each other, but only the way data values are addressed. This operation is important for the purposes of visualizing the data set, since depending on the visualization method, each transposition results in a new view of the data.

Formulation:

$$\begin{gathered} \{\ \} \\ \mathrm{Transpose}(p_1, \ldots, p_N) : \langle S, V \rangle \to \langle S', V' \rangle \\ \left\{ \begin{matrix} v'[i_1, \ldots, i_N] = v[i_{p_1}, \ldots, i_{p_N}] \\ D'_j = D_{p_j} \end{matrix} \right\} \end{gathered} \tag{3.4}$$

Slice

Arguments: **`dimension`**, **`key`**

Description: Produces a *(N-1)* dimensional subset of the data set which contains only those cells addressed by the key **`key`** of the dimension **`dimension`**.

Formulation:

$$
\begin{gathered}
\left\{
\begin{array}{ll}
S \equiv \langle D_1, D_2, \ldots, D_N \rangle & \\
D_j \equiv \langle K_j, L_j \rangle & \\
k = k_j[d] \in K_j & \text{for some } \ 1 \le d \le M_j
\end{array}
\right\} \\
\mathrm{Slice}(D_j, k) : \langle S, V \rangle \rightarrow \langle S', V' \rangle \\
\left\{
\begin{array}{rl}
v'[i_1, \ldots, i_{j-1}, i_{j+1}, \ldots i_N] & = v[i_1, \ldots i_{j-1}, d, i_{j+1}, \ldots, i_N] \\
S' & = \langle D_1, \ldots, D_{j-1}, D_{j+1}, \ldots, D_N \rangle
\end{array}
\right\}
\end{gathered}
\tag{3.5}
$$

Extend

Arguments: **`dimension`**, **`key`**, **`spread_cube`**

Description: Attaches the data cells in the *(N-1)*-dimensional data set **`spread_cube`**, to the current data set. The dimension **`dimension`** of the current data set is extended with the key **`key`**, which is used to address the newly added data cells. It is assumed that the dimensions of **`spread_cube`** and the dimensions of the current data set other than **`dimension`** are the same.

Formulation:

$$
\begin{gathered}
\left\{\begin{array}{l}
C'' \equiv \langle S'', V'' \rangle \\
S'' \equiv \langle D_1, \ldots, D_{j-1}, D_{j+1}, \ldots, D_N \rangle \\
S \equiv \langle D_1, \ldots, D_N \rangle \\
D_j \equiv \langle K_j, L_j \rangle
\end{array}\right\} \\
\text{Extend}(D_j, k, C'') : \langle S, V \rangle \rightarrow \langle S', V' \rangle \\
\left\{\begin{array}{l}
v'[i_1, \ldots, i_N] = v[i_1, \ldots, i_N] \quad \text{for } 1 \le i_j \le M_j \\
k'_j[i_j] = k_j[i_j] \quad \text{for } 1 \le i_j \le M_j \\
v'[i_1, \ldots, i_N] = v''[i_1, \ldots, i_{j-1}, i_{j+1}, \ldots, i_N] \quad \text{for } i_j = M_j + 1 \\
k'_j[i_j] = k \quad \text{for } i_j = M_j + 1 \\
M'_j = M_j + 1
\end{array}\right\}
\end{gathered}
\tag{3.6}
$$

This operation can be used to add a new variable to the existing data set. In such case, **`dimension`** is the record dimension and **`key`** is the name of the new variable.

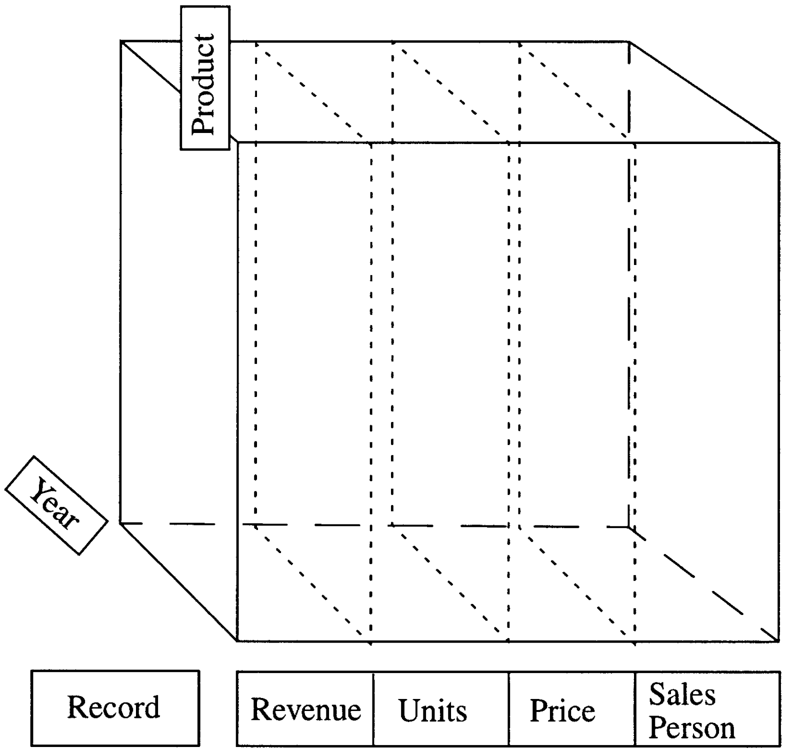


Figure 3.8: Extending the Record Dimension with “Sales Person”

The example on figure 3.8 is the result of extending the data set on figure 3.7 with the variable “Sales Person”

Remove

Arguments: **`dimension`**, **`key`**

Description: Removes the *(N-1)*-dimensional subset of cells addressed by **`key`** of the dimension **`dimension`**.

Formulation:

$$
\begin{gathered}
\left\{\begin{aligned}
& S \equiv \langle D_1, D_2, \ldots, D_N \rangle \\
& D_j \equiv \langle K_j, L_j \rangle \\
& k = k_j[d] \in K_j \qquad \text{for some } \; 1 \le d \le M_j
\end{aligned}\right\} \\
\text{Remove}(D_j, k) : \langle S, V \rangle \rightarrow \langle S', V' \rangle \\
\left\{\begin{aligned}
v'[i_1, \ldots, i_N] &= v[i_1, \ldots, i_N] \qquad \text{for } i_j < d \\
v'[i_1, \ldots, i_j - 1, \ldots, i_N] &= v[i_1, \ldots, i_j, \ldots, i_N] \qquad \text{for } i_j > d \\
k'_j[i_j] &= k_j[i_j] \qquad \text{for } i_j < d \\
k'_j[i_j - 1] &= k_j[i_j] \qquad \text{for } i_j > d
\end{aligned}\right\}
\end{gathered}
\tag{3.7}
$$

Merge

Arguments: **`dimension1`**, **`dimension2`**

Description: Produces a data set of one lower dimension by merging **`dimension1`** with **`dimension2`**. This operation can be conceptualized in the following way: The data set is cut into *(N-1)* dimensional slices perpendicular to dimension2. The slices are then arranged next to each other along **`dimension2`**. Each key of **`dimension2`** is replicated M_1 times, where M_1 is the size of **`dimension1`**, and is paired with a key of **`dimension2`**.

Formulation: Without loss of generality, we can assume that **`dimension1`** and

`dimension2` are the dimensions D_1 and D_2 of the schema. Then:

$$\begin{Bmatrix} D_1 \equiv \langle K_1, L_1 \rangle \\ D_2 \equiv \langle K_2, L_2 \rangle \end{Bmatrix}$$
$$\mathrm{Fold}(D_1, D_2) : \langle S, V \rangle \rightarrow \langle S', V' \rangle$$
$$\begin{Bmatrix} S' &=& \langle D'_2, D_3, \ldots, D_N \rangle \\ & D'_2 \equiv \langle K'_2, L'_2 \rangle & \\ k'_2[i_2 M_2 + i_1] &=& \langle k_2[i_2], k_1[i_1] \rangle \\ v'[i_2 M_2 + i_1, i_3, \ldots, i_N] &=& v[i_1, \ldots, i_N] \end{Bmatrix} \tag{3.8}$$

Unmerge

Arguments: **`dimension`**

Description: This operation reverses the effects of the operation “Merge”. It is assumed that **`dimension`** is the outcome of a previous merging operation. The result is a data set of one higher dimensionality.

3.5.2 Data Interpretation Operations

The following operations depend on the actual contents of the vault to modify the schema of the data set. The goal of each of these operations is to make the implicit semantic relationships within the data explicit in the schema of the data set.

Sort

Arguments: **`stick`**, **`direction`**

Description: Permutes the *(N-1)* dimensional planes of cells perpendicular to **`stick`**, so that the values of the input stick are sorted. The sorting is stable, in the sense that if **`stick`** contains two equal values, the mutual position of these remains unchanged. The **`direction`** argument, is a binary variable with possible values: `up` and `down`. The value `down` specifies that the values in the stick should increase in the direction of small to large array coordinates. Conversely, if **`direction`** is `down`, then the values of the stick should increase in the direction of large to small array coordinates.

Formulation:

$$\{\ \}$$

$$\mathrm{Sort}(Q_j[d_1\ldots, d_{j-1}, d_{j+1}, \ldots, d_N], \mathrm{direction}) : \langle S, V\rangle \rightarrow \langle S', V'\rangle$$

$$\left\{\begin{array}{ll} \langle S', V'\rangle = \mathrm{Reorder}(D_j, (p_1 \ldots p_N)) : \langle S, V\rangle & \text{for some } (p_1, \ldots, p_N) \text{ such that:} \\ v'_j[d_1\ldots, d_{j-1}, 1, d_{j+1}, \ldots, d_N] \geq \ldots \geq v'_j[d_1\ldots, d_{j-1}, M_j, d_{j+1}, \ldots, d_N] & \text{if direction = up or} \\ v'_j[d_1\ldots, d_{j-1}, 1, d_{j+1}, \ldots, d_N] \leq \ldots \leq v'_j[d_1\ldots, d_{j-1}, M_j, d_{j+1}, \ldots, d_N] & \text{if direction = down} \end{array}\right\} \quad (3.9)$$

Promote

Arguments: **`variable`**

Description: This operation increases the dimensionality of the data set, by promoting a variable into a dimension variable. It creates a new dimension whose keys are the unique values of **`variable`**. The operation can be conceptualized in the following way: first, **`variable`** is removed from the data set, then, the data set is broken into M_{N+1} data sets, each of which is of dimensionality N, where M_{N+1} is the number of unique values in **`variable`**. Each of the new data sets is addressed by a key in the newly formed dimension, thus producing a $(N+1)$ dimensional data set. The resulting data set contains the same amount of data, but has richer schema.

Formulation:

$$\left\{\begin{array}{l} S \equiv \langle D_1, \ldots, D_N \rangle \\ D_j \equiv \langle K_j, L_j \rangle \\ L_j = \text{Record} \\ var = k_j[d] \in K_j \qquad \text{for some } 1 \le d \le M_j \qquad \text{and some } 1 \le j \le N \\ \langle S'', V'' \rangle \text{ is such that } \langle S'', V'' \rangle = \text{Remove}(var) : \langle S, V \rangle \end{array}\right\}$$

$$\text{Promote}(var) : \langle S, V \rangle \rightarrow \langle S', V' \rangle \tag{3.10}$$

$$\left\{\begin{array}{l} S' = \langle D''_1, \ldots, D''_N, D'_{N+1} \rangle \\ D'_{N+1} = \langle K'_{N+1}, L'_{N+1} \rangle \\ K'_{N+1} = (k'_{N+1}[i_{N+1}]) \qquad \text{for } 1 \le i_{N+1} \le M_{N+1} \\ k'_{N+1}[i_{N+1}] \text{ are unique values of the set: } \{ v[i_1, \ldots, i_{j-1}, d, i_{j+1}, \ldots i_N] \} \\ v'[i_1, \ldots, i_N, i_{N+1}] = \begin{cases} v''[i_1, \ldots, i_N] & \text{if } v[i_1, \ldots, i_{j-1}, d, i_{j+1}, \ldots, i_N] = k'_{N+1}[i_{N+1}] \\ \textit{nonexistent} & \text{otherwise} \end{cases} \end{array}\right\}$$

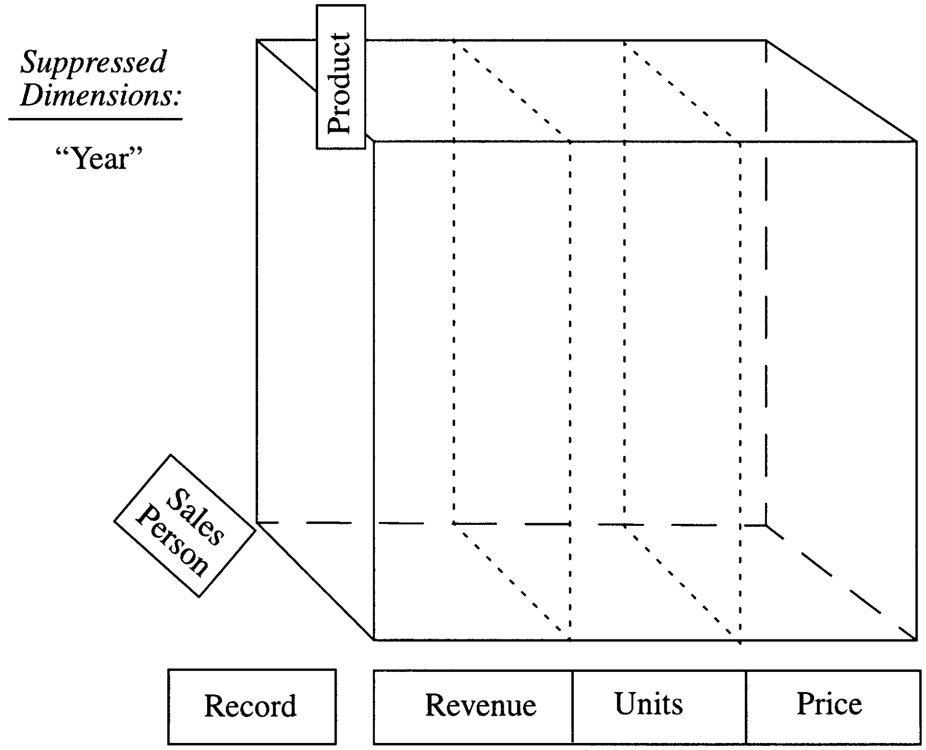


Figure 3.9: Promoting the variable "Sales Person" (compare to figure 3.8)

This operation is used to pull out more of the semantic structure into the schema. Also, it can be used in conjunction with a selection operation to split a N dimensional data sets into N dimensional subsets. The sketch on figure 3.9 represents the result after promoting the variable “Sales Person” to a dimension variable. For the sake of clarity, the “Year” dimension has been suppressed.

Reduce

Arguments: **`dimension`**

Description: This operation reverses the effect of the operation “Promote”. The argument **`dimension`**, must be a previously promoted variable.

Derive

Arguments: **`variable`**, **`function`**

Description: Extends the current data set with a new variable, whose values are the results of applying the unary function **`function`**, to the values of **`variable`**. The new variable is called *derived variable.*

Mark

Arguments: **`regions`**, **`labels`**

Description: The **`regions`** argument, is a list of regions of vault cells and **`labels`** is an equal length list of labels. This operations extends the current data set with a nominal variable whose values are members of **`labels`**, such that: Each cell from the region: r_m has the same set-dimension coordinates as the value l_m of the new nominal variable.

Formulation:

Let **`regions`** be the list: $\{r_1, \ldots, r_n\}$ and **`labels`** be the list: $\{l_1, \ldots, l_n\}$, where each region r_m is simply a collection of cells: $r_m = \{cell*\}$. In that case, we can formulate the mark operation in the following way:

$$\left\{\begin{array}{l} S \equiv \langle D_1, \dots, D_N \rangle \\ D_j \equiv \langle K_j, L_j \rangle \\ L_j = \text{Record} \end{array}\right\}$$

$$\text{Mark}(\{r_1, \dots, r_n\}, \{l_1, \dots, l_n\}) : \langle S, V \rangle \to \langle S', V' \rangle$$

$$\left\{\begin{array}{rcl} v'[i_1, \dots i_N] & = & v[i_1, \dots, i_N] \qquad \text{for } 1 \le i_j \le M_j \\ k'_j[i_j] & = & k_j[i_j] \qquad \text{for } 1 \le i_j \le M_j \\ v'[i_1, \dots, i_{j-1}, M_j + 1, i_{j+1}, \dots, i_N] & = & l_m \Rightarrow v[i_1, \dots, i_N] \in r_m \qquad \text{for } 1 \le i_j \le M_j \\ M'_j & = & M_j + 1 \\ k'_j[M'_j] & = & \textit{marking name} \end{array}\right\} \tag{3.11}$$

Where *marking name* stands for the name of the new nominal variable, which is generated in run time as an unique identifier.

Aggregate

Arguments: **`dimension`**, **`function`**

Description: Aggregates all sticks parallel to **`dimension`** according to some aggregation function specified by **`function`**. Each aggregated stick is substituted for a single value which is the result of its aggregation. As a consequence the dimensionality of the data set is reduced by one.

Formulation:

Let f denote the aggregation function. Then:

$$\{S \equiv \langle D_1, \dots, D_N \rangle\}$$

$$\text{Aggregate}(D_j, f) \; : \langle S, V \rangle \to \langle S', V' \rangle$$

$$\left\{\begin{array}{rcl} & S \equiv (D_1, \dots, D_{j-1}, D_{j+1}, \dots, D_N) & \\ v'[i_1, \dots i_{j-1}, i_{j+1}, \dots, i_N] & = & f(v[i_1, \dots, i_{j-1}, 1, i_{j+1}, \dots i_N], \dots, v[i_1, \dots, i_{j-1}, M_j, i_{j+1}, \dots i_N]) \end{array}\right\} \tag{3.12}$$

Chapter 4

The Visual Model

The goal of this chapter is to describe of the SpreadCube visual model and how it relates to the abstract data model, as well as how the data model operations are reflected in the visual model.

4.1 What is a Visual Model

The visual model is the projection of the abstract data model on a visual medium, like a two-dimensional flat computer screen. Every operation on the abstract data model has its visual counterpart in the visual model.

Therefore, the goals of the visual model are

- to provide an intuitive and compact representation of the abstract data model, given the constraints of the medium
- to provide a consistent representation of the operations on the abstract data model
- to function as a context for interface actions which specify operations on the data model or visualization methods.

To accomplish these goals, the visual model contains components similar to the ones in the abstract data model. Unlike the abstract data model case, however, the components of the visual model must have a straightforward representation on a two-dimensional screen. For example, the visual schema is a hierarchical tree-like structure and the visual vault can be either a collection of sticks or a collection of slabs. Both sticks and slabs have a fairly straight forward visualization on the screen, e.g. a slab is simply a table of numbers.

The visual model is only concerned with the layout of the visual objects on the screen and not with how they are rendered. That is, the rules of how a quantity or a nominal value should be depicted on the screen are below the visual model level of abstraction. Such issues are discussed in chapter 5.

Because of the constraint that every visual component must have a straight forward 2D representation, the visual model looses some of the simplicity of the abstract data model and introduces some extra structure. For example, in order to visualize multiple dimensions, a dimension hierarchy is introduced. The idea of dimension hierarchy, however, is by itself misleading, because according to the philosophy of the abstract data model all dimensions are created and treated as equal, at least as far as schema operations are concerned. That is why, it is important that when dealing with the visual model one understands what is the abstract model behind the visualization.

4.2 Description of the Visual Schema

The visual schema is implemented as a planar dimensional hierarchy. The concept of dimensional hierarchy is introduced in the following two sub-sections. We will use the same notation as the one introduced in chapter 3 to refer to the elements of the schema. Thus, a N-dimensional schema S contains a set of N dimensions: $D_1, D_2,\dots,D_N$. Each dimension D_j is a label L_j and a set of keys $k_j[i_j]$ for $1 \le i_j \le M_j$.

In order to avoid confusion between the dimensions of the schema and the dimensions of the screen, we will refer to the screen dimensions as *axes*. Thus, a screen has two axis *horizontal axis* and a *vertical axis.*

4.2.1 Linear Dimensional Hierarchy

Before introducing the idea of a planar dimensional hierarchy, it is useful to describe the simpler concept of a linear dimensional hierarchy.

In a linear dimensional hierarchy, the visual vault is a collection of sticks, aligned along a single screen axis. The dimensions are arranged in chosen order of seniority. For the purposes of illustration, and without loss of generality we can assume that the order is chosen to be: $D_1, D_2, \ldots, D_N$

Given the above order of dimensions, the top level of the hierarchical tree corresponds to D_1 and the branches correspond to the keys of D_1, the level of nodes below corresponds to D_2 and so on (see figure 4.1). The leaves of the tree store the actual data values. Furthermore, the leaves of any node from the lowest level of the tree, contain data values which share all but one dimension keys, and therefore those data values form a stick.

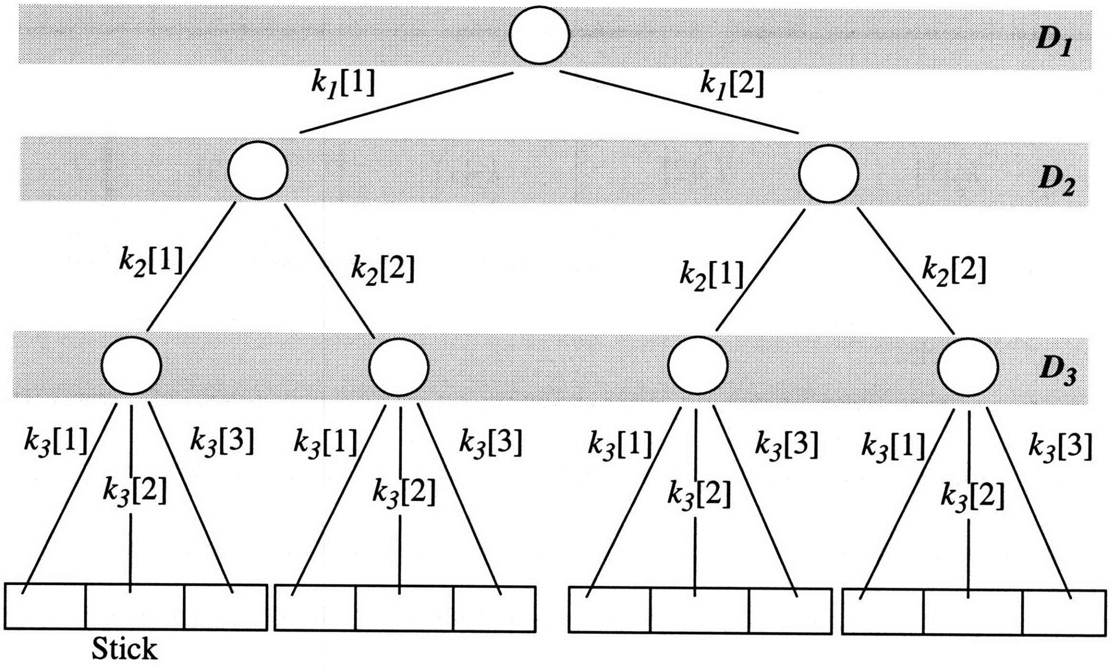


Figure 4.1: Linear Dimensional Hierarchy

The example on figure 4.1 illustrates the dimension hierarchy for a three dimensional data set, whose abstract vault size is: $3 \times 2 \times 2$. The hierarchical tree describes the arrangement of sticks along a particular screen axis. That is why we can speak of the hierarchical tree as being associated with a that screen axis, e.g. the horizontal screen axis on figure 4.1.

The similarity between figure 3.6 and figure 4.1 is not accidental. In fact each path from the leave of the hierarchical tree to its root represents a data consolidation path. This fact facilitates the interpretation of a hierarchical tree, because depending on which is more convenient the user can think about the visual model either in terms of the abstract data model or in terms of data consolidation paths.

A more compact way of representing the same dimensional hierarchy is shown on figure 4.2, where the branches are represented by boxes and the nodes are omitted. Each level of the hierarchy is labeled by the label of the corresponding dimension.

$k_1[2]$						$k_1[2]$						L_1
$k_2[1]$			$k_2[2]$			$k_2[1]$			$k_2[2]$			L_2
$k_3[1]$	$k_3[2]$	$k_3[3]$	$k_3[1]$	$k_3[2]$	$k_3[3]$	$k_3[1]$	$k_3[2]$	$k_3[3]$	$k_3[1]$	$k_3[2]$	$k_3[3]$	L_3
												Data

Stick

Figure 4.2: Tabular Representation of a Linear Dimensional Hierarchy

We will use the term *group* to refer to a node in the hierarchical tree, because it represents a stage in the data consolidation path through it. Also, we will call a level of nodes in the hierarchical tree a *grouping*. Thus a dimension in the visual schema is simply a grouping.

4.2.2 Planar Dimensional Hierarchy

In the case of a planar dimensional hierarchy, the visual vault is a collection of slabs tiled in the plane of the two screen axis. The visual schema consists of two hierarchical trees associated with the horizontal and vertical screen axes. These hierarchies determine the arrangement of the slabs on the screen (see figure 4.3).

The two hierarchical trees contain two complementary sets of dimensions. Within each tree the dimensions are arranged in a seniority order. However, there is no such ordering defined between dimensions from different hierarchical trees.

Just as the case with a linear dimensional hierarchy (figure 4.1) the leaves of the dimension trees contain data values. However, unlike the case on figure 4.1 each leave contains an entire row or column of data cells, depending on whether the hierarchical tree is associated with the horizontal or vertical axis. Since a slab data cell can belong to exactly one row-column pair, therefore a cell is completely specified by a pair of leaves from each tree.

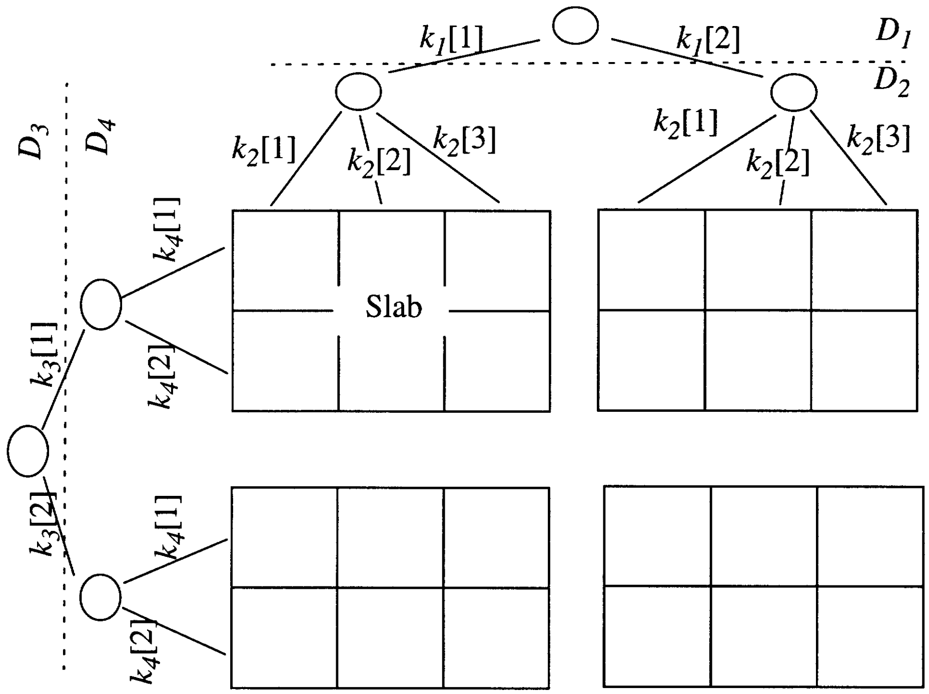


Figure 4.3: Planar Dimensional Hierarchy

The tabular representation of the dimensional hierarchy from figure 4.3 is shown on figure 4.4.

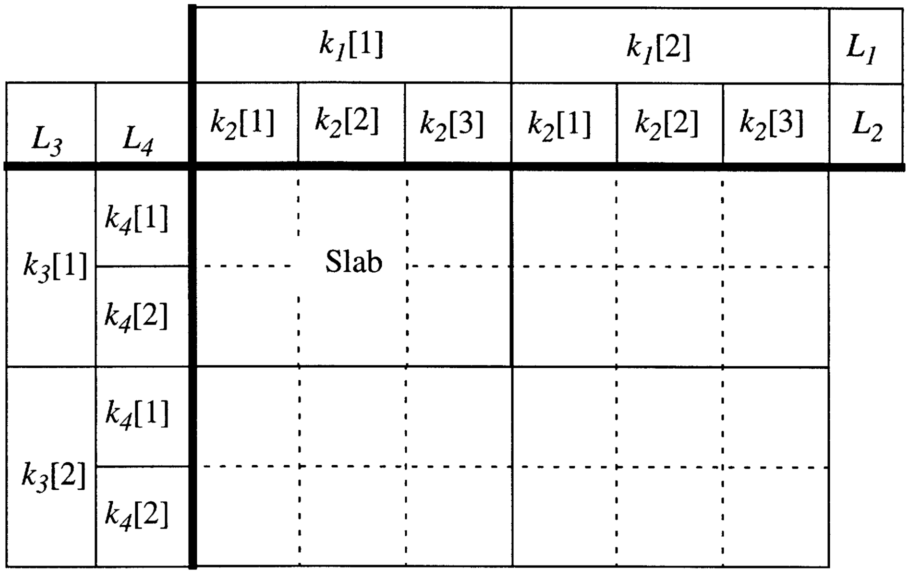


Figure 4.4: Tabular Representation of a Planar Dimensional Hierarchy

4.2.3 Nonexistent Rows and Columns

In order to provide a compact representation of the abstract data model, the visual model suppresses the representation of data cells with nonexistent values, whenever it is possible to do so without creating inconsistencies.

For example, if all cells in the row addressed by $k_4[2]$ on figure 4.4 contained nonexistent data values, then one can safely eliminate the entire row and “close the gap” by shifting the bottom of the table up, without altering the spacial relationships amongst the rest of the cells.

On the other hand, if only half of the cells of the $k_4[2]$ row had nonexistent values, e.g. the cells addressed by $k_4[2]$ and $k_1[1]$, then they will have to remain represented in the visual model. Indeed if we tried to eliminate these cells, and close the gap by shifting the bottom quarter of the table up, then we will have violated the spacial relationships between the left and the right halves of the table.

Therefore, the rules for suppressing the representation of cells in the visual model are the following:

- To preserve the spacial relationships between cells, only entire groups should be eliminate, given the current grouping of the visual axis.
- In order to represent all essential data, only empty groups can be eliminate, i.e. groups of cells with nonexistent data values.

A more rigorous implementation of the above two rules is demonstrated in the following section, where an empty group handled by simply setting the width of its associated hierarchical sub-tree to 0.

Given the rules for eliminating cells with nonexistent data values, we can conclude that the way and the extend to which non-existent data values are represented depends largely on the hierarchical structure of each axis.

4.3 Definition of the Visual Model

The goal of this chapter is to introduce a sentential notation and definition of the visual model and relate it to the abstract data model from chapter 3.

4.3.1 Notation and Definition

We will use the same notation as the one introduced in chapter 3 extended with a few new identifiers, such as: A which we will use to denote a screen axis and g, which will denote a group. We will use the same identifiers for visual and abstract objects. For most cases this should not cause confusion, because we will mainly discuss visual objects in this chapter. We will use the following notation: $\mathcal{A}(visual\ object)$ to refer to the abstract counter part of a visual object and $\mathcal{V}(abstrct\ objet)$ to refer to the visual counter part of an abstract object.

A visual model consists of a visual schema and a visual vault. The schema has two axes. Each axis is an ordered set of dimensions, such that the two sets are complimentary to each other. Each dimension is a grouping, which is a list of groups. Each group is a list of pairs consisting of a dimension key and a sub-group.

The structure of the visual model is formulated by equations (4.1). In this formulation we have assumed, for the sake of simplicity, that the order of dimensions for the two axis was chosen to be: $D_1, \ldots, D_{N_1}$ and $D_{N_1+1}, \ldots, D_{N_1+N_2}$ respectively, where N_1 and N_2 are the dimensionalities of the two axes.

$$
\begin{aligned}
C^N &\equiv \langle S^N, V^2 \rangle \\
S^N &\equiv \langle A_1^{N_1}, A_2^{N_2} \rangle \qquad \text{such that } N_1 + N_2 = N \\
A_1^{N_1} &\equiv \langle D_1, \ldots, D_{N_1} \rangle \\
A_2^{N_2} &\equiv \langle D_{N_1+1}, \ldots, D_N \rangle \\
A_1^{N_1} \equiv \langle D_1, \ldots, D_{N_1} \rangle \vdash D_j &\equiv (g[i_1, \ldots, i_j]) \qquad \text{for } 1 \le j \le N_1 \qquad \text{and } 1 \le i_j \le M_j \\
A_2^{N_2} \equiv \langle D_{N_1+1}, \ldots, D_N \rangle \vdash D_j &\equiv (g[i_{N_1+1}, \ldots, i_j]) \qquad \text{for } (N_1+1) \le j \le N \qquad \text{and } 1 \le i_j \le M_j \\
g[i_1, \ldots, i_j] &\equiv (\langle k_j[i_j], g[i_1, \ldots, i_{j+1}] \rangle) \qquad \text{for } 1 \le j \le N_1 \\
g[i_{N_1+1}, \ldots, i_j] &\equiv (\langle k_j[i_j], g[i_1, \ldots, i_{j+1}] \rangle) \qquad \text{for } (N_1+1) \le j \le N \\
g[i_1, \ldots, i_{N_1+1}] &\equiv (\,) \\
g[i_{N_1+1}, \ldots, i_{N+1}] &\equiv (\,) \\
V^2 &\equiv \| v[x, y] \| \qquad \text{for } 1 \le x \le |A_1^{N_1}| \qquad \text{and } 1 \le y \le |A_2^{N_2}|
\end{aligned}
\tag{4.1}
$$

In the above set of equations, the superscripts designate the dimensionality of each object. The notation $\alpha \vdash \beta$ means that proposition β is true in the environment where α is true. Therefore, if the dimensions on an axis were permuted, then the definition of each dimension will also change accordingly.

The dimensionalities N_1 and N_2 of the two axis are chosen arbitrarily as long as they satisfy the relationship $N_1 + N_2 = N$

The symbol | | denotes the size of an object. The size of each axis is defined in the following way:

$$
\begin{aligned}
\left|A_1^{N_1}\right| &= |D_1| \\
|D_1| &= \sum_{i_1} |g[i_1]| \\
|g[i_1, \ldots, i_j]| &= \sum_{i_{j+1}} |g[i_1, \ldots, i_{j+1}]| \qquad \text{for } 1 \le j \le N_1 \\
|g[i_1, \ldots, i_{N_1+1}]| &= \begin{cases} 0 & \text{if } g[i_1, \ldots, i_{N_1+1}] \text{ is empty} \\ 1 & \text{otherwise} \end{cases}
\end{aligned}
\tag{4.2}
$$

and similarly for $A_2^{N_2}$. A group $g[d_1, \ldots, d_m]$ belonging to the $A_1^{N_1}$ axis is said to be empty iff all vault cells which share the coordinates $d_1, \ldots, d_m$ contain non-existent data values, i.e.:

$$
\begin{aligned}
&(\forall j : m < j \le N \bullet \forall i_j : 1 \le i \le M_j \bullet v[d_1, \ldots, d_m, i_{m+1}, \ldots, i_N] = \texttt{nonexistent}) \Leftrightarrow \\
&\Leftrightarrow (g[d_1, \ldots, d_m] \text{ is empty})
\end{aligned}
\tag{4.3}
$$

and similarly for groups on the $A_2^{N_2}$ axis.

According to equations (4.2) the size of each axis is simply the width of the hierarchical tree associated with it. Also, the size of each group is the width of its sub-tree.

4.3.2 Establishing a Grid

In the beginning of this chapter, we pointed out that the visual model represents an abstraction level above the level of the rules which dictate how each object should be rendered. Nevertheless, however, the visual model does express the relative layout of objects on the screen. So far we have discussed the layout of visual objects implicitly by describing the hierarchical dependencies between them.

In order to make the layout of visual objects implicit, we will describe them in the environment of a *grid.* A grid is a discrete two dimensional coordinate system. We will call each point of the grid a *grid position*. To each visual object we can assign a grid position and a *grid width* and *grid height.*

The grid positions do not directly correspond to screen positions. In the relationship between grid and screen positions can be arbitrarily complex. Here is a metaphorical way to conceptualize this: Pretend that we draw a grid on sheet of rubber, and then arrange all visual objects on that grid. Once we are done, we stretch and shrink the rubber sheet in various ways so that it fits on the screen. This final fitting of the rubber grid on the screen corresponds to rendering the visual objects. Until then we will only be concerned with their general layout.

A grid is not part of the visual model. It only acts as an environment for making relationship between visual objects explicit and for that matter any grid would suffice. One important way in which we will use the concept of a grid, is to describe the relationship between the cells of the visual vault and those of the abstract data model.

The grid position, the grid width and the grid height are treated as attributes of each visual object. We will call these *spacial attributes* of a visual object. We will denote each of them as a suffix preceded by a **.** . For example:

$VisualObject.X$	denotes the horizontal grid position
$VisualObject.Y$	denotes the vertical grid position
$VisualObject.W$	denotes the object grid width
$VisualObject.H$	denotes the object grid height

The following set of equations defines the spatial attributes for some elements of the visual model:

$$\begin{aligned}
\langle C^N.X, C^N.Y, C^N.W, C^N.H\rangle &= \langle -N_2, -N_1, |A_1^{N_1}| + N_2, |A_2^{N_2}| + N_1\rangle \\
\langle S^N.X, S^N.Y, S^N.W, S^N.H\rangle &= \langle C^N.X, C^N.Y, C^N.W, C^N.H\rangle \\
\langle A_1^{N_1}.X, A_1^{N_1}.Y, A_1^{N_1}.W, A_1^{N_1}.H\rangle &= \langle 0, -N_1, |A_1^{N_1}|, N_1\rangle \\
\langle A_2^{N_2}.X, A_2^{N_2}.Y, A_2^{N_2}.W, A_2^{N_2}.H\rangle &= \langle -N_2, 0, N_2, |A_2^{N_2}|\rangle \\
\langle V^2.X, V^2.Y, V^2.W, V^2.H\rangle &= \langle 0, 0, |A_1^{N_1}|, |A_2^{N_2}|\rangle \\
\langle v[x,y].X, v[x,y].Y, v[x,y].W, v[x,y].H\rangle &= \langle x-1, y-1, 1, 1\rangle
\end{aligned} \tag{4.4}$$

In the above equations we have used a shorthand for defining simultaneously the position, width and height of an object, by making those attributes the elements of a 4-tuple and then equating them to another 4-tuple.

Equations (4.4) describe a tabular layout of the visual schema (see figure 4.4). Furthermore, according to these equations, the grid position of the vault was chosen to be the origin. The spatial attributes for the elements of an axis are described below. For the sake of simplicity, we have only shown the definitions of spatial attributes for the elements of the $A_1^{N_1}$ axis, which we have chosen to be the horizontal axis. The spatial attributes of the elements of $A_2^{N_2}$ are analogously defined.

$$\begin{aligned}
\langle D_j.X, D_j.Y, D_j.W, D_j.H\rangle &= \langle 0, N_1-j+1, |A_1^{N_1}|, 1\rangle \\
\langle g[\underbrace{1,\ldots,1}_{j}].X, g[\underbrace{1,\ldots,1}_{j}].Y\rangle &= \langle 0, N_1-j+1\rangle \\
\langle g[i_1,\ldots,i_j].W, g[i_1,\ldots,i_j].H\rangle &= \langle |g[i_1,\ldots,i_j]|, 1\rangle \qquad \text{for } 1 \le i_j \le M_j \\
g[i_1,\ldots,i_j].Y &= N_1-j+1 \qquad \text{for } 1 \le i_j \le M_j \\
g[i_1,\ldots,i_j].X &= \begin{cases} g[i_1,\ldots,i_j-1].X + g[i_1,\ldots,i_j-1].W & \text{for } i_j > 1 \\ g[i_1,\ldots,i_{j-1}].X + g[i_1,\ldots,i_{j-1}].W & \text{for } i_j = 1 \end{cases}
\end{aligned} \tag{4.5}$$

Using the concept of spatial attributes, we can now easily express the relationship between the cells $v[x,y]$ of the visual two dimensional vault and the cells $v[i_1,\ldots,i_N]$ of the abstract N dimensional vault:

$$\begin{aligned}
&(v[x,y].X = g[d_1,\ldots,d_{N_1+1}].X) \wedge (v[x,y].Y = g[d_{N_1+1},\ldots,d_{N+1}].Y) \Leftrightarrow \\
&\Leftrightarrow v[x,y] = v[d_1,\ldots,d_N]
\end{aligned} \tag{4.6}$$

4.4 Operations on the Visual Model

The user invokes operations on the abstract data model by operating on the visual model.

The result of those operations is conveyed to the user back through the visual model. The set of operations on the visual model act as an interface to the operations on the abstract data model.

The set of operations on the visual model is designed to mimics as close as possible the set of operations on the abstract data model. Thus, one of the objectives is to have one-to-one correspondence between visual and abstract operations.

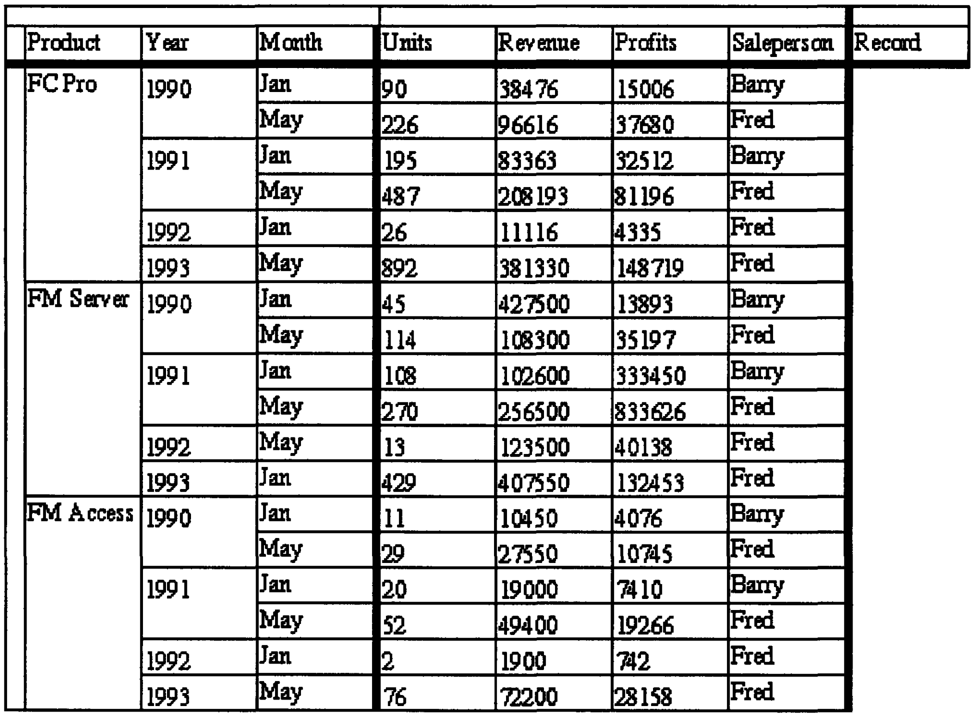
The visual operations, however must conform to the constraints of the media and the extend of the user's control. That is why they may sometimes differ in number and types of arguments from their abstract counterparts. Sometimes, an abstract operation may not be specified by a single visual operation but it will take several such operations to do so, e.g. a visual operation preceded by a sequence of visual transpositions.

This section describes the operations on the visual model and demonstrates how some of the operations on the abstract data model defined in chapter 3 affect the visual model

4.4.1 An Example Data Set

For the purposes of illustration, we will use as an example the data set shown on figure 3.8 extended with a "Month" dimension. Its visual model is shown on figure 4.5. In order to track what happens to individual data cells we have also depicted their values.

Initially, we have chosen to represent the record dimension along the horizontal axis, and the three other dimensions: "Product", "Year" and "Month" along the vertical axis. Therefore D_1, D_2, D_3, D_4 correspond to the Record, "Product", "Year" and "Month" dimensions, respectively. Note how this choice of arrangement of dimensions, causes the visual structure to resemble the structure of a relational table.



Product	Year	Month	Units	Revenue	Profits	Saleperson	Record
FC Pro	1990	Jan	90	38476	15006	Barry	
		May	226	96616	37680	Fred	
	1991	Jan	195	83363	32512	Barry	
		May	487	208193	81196	Fred	
	1992	Jan	26	11116	4335	Fred	
	1993	May	892	381330	148719	Fred	
FM Server	1990	Jan	45	427500	13893	Barry	
		May	114	108300	35197	Fred	
	1991	Jan	108	102600	333450	Barry	
		May	270	256500	833626	Fred	
	1992	May	13	123500	40138	Fred	
	1993	Jan	429	407550	132453	Fred	
FM Access	1990	Jan	11	10450	4076	Barry	
		May	29	27550	10745	Fred	
	1991	Jan	20	19000	7410	Barry	
		May	52	49400	19266	Fred	
	1992	Jan	2	1900	742	Fred	
	1993	May	76	72200	28158	Fred	

Figure 4.5: An Example Data Set

Also, note the effects of eliminating empty groups. Even though, the "Month" dimension has two keys: "Jan" and "May", the group of months under "FM Access", 1992 contains only one "Month" key.

In the following subsections we will exemplify some of the less obvious operations on the abstract data set, by applying them to the example data set from figure 4.5.

4.4.2 Reorder

Arguments: **`dimension`**, **`key1`**, **`key2`**

Description: Swaps the keys **`key1`** and **`key2`** in **`dimension`**. This is an example where the visual operation differs form the corresponding abstract operation in the number and types of its arguments. The abstract operation requires that a permutation is specified. For practical reasons, however, it is convenient to specify a permutation as a sequence of swaps of two elements rather than as an N-tuple.

Formulation:

$$\begin{gathered}\{m < n\} \\ \mathrm{Reorder}(D_j, k_j[m], k_j[n]) : C^N \to C'^N \\ \{\mathcal{A}(C'^N) = \mathrm{Reorder}(\mathcal{A}(D_j), (1, \ldots, m-1, n, m+1, \ldots, n-1, m, n+1, \ldots, N)) : \mathcal{A}(C^N)\}\end{gathered} \tag{4.7}$$

4.4.3 Transposition

Arguments: **`dimension1`**, **`dimension2`**

Description: Swaps the dimensions **`dimension1`** and **`dimension2`** in the schema. Just as in the case of the "Reorder" operation, the visual transposition is specified by a sequence of swaps of dimension objects.

Formulation:

$$\begin{gathered}\{m < n\} \\ \mathrm{Transpose}(D_m, D_n) : C^N \to C'^N \\ \{\mathcal{A}(C'^N) = \mathrm{Transpose}((1, \ldots, m-1, n, m+1, \ldots, n-1, m, n+1, \ldots, N)) : \mathcal{A}(C^N)\}\end{gathered} \tag{4.8}$$

When the dimension objects of the abstract schema are permuted the result is that the visual dimensions permute within an axis and between the axes. For example, permuting the four dimensions according to the permutation $(4, 1, 3, 2)$, produces the visual set shown on figure 4.6

		Jan				May				Month
Year	Product	Units	Revenue	Profits	Saleperson	Units	Revenue	Profits	Saleperson	Record
1990	FC Pro	90	38476	15006	Barry	226	96616	37680	Fred	
	FM Server	45	427500	13893	Barry	114	108300	35197	Fred	
	FM Access	11	10450	4076	Barry	29	27550	10745	Fred	
1991	FC Pro	195	83363	32512	Barry	487	208193	81196	Fred	
	FM Server	108	102600	333450	Barry	270	256500	833626	Fred	
	FM Access	20	19000	7410	Barry	52	49400	19266	Fred	
1992	FC Pro	26	11116	4335	Fred				.	
	FM Server				.	13	123500	40138	Fred	
	FM Access	2	1900	742	Fred				.	
1993	FC Pro				.	892	381330	148719	Fred	
	FM Server	429	407550	132453	Fred				.	
	FM Access				.	76	72200	28158	Fred	

Figure 4.6: The Example Data Set After a Transposition

4.4.4 Extend

Arguments: **dimension**, **key**, **spread_cube**

Description: Attaches the data cells in the *(N-1)*-dimensional data set **spread_cube**, to the current data set. The dimension **dimension** of the current data set is extended with the key **key**, which is used to address the newly added data cells. This visual model operation simply invokes the abstract data model operation *extend* with the same arguments (See "Extend" on page 50.)

4.4.5 Remove

Arguments: **dimension**, **key**

Description: Removes the *(N-1)*-dimensional subset of cells addressed by **key** of the dimension **dimension**. It reverses the effects of the *extend* operation. (See "Remove" on page 52.).

4.4.6 Merging Dimensions

Arguments: **dimension1**, **dimension2**

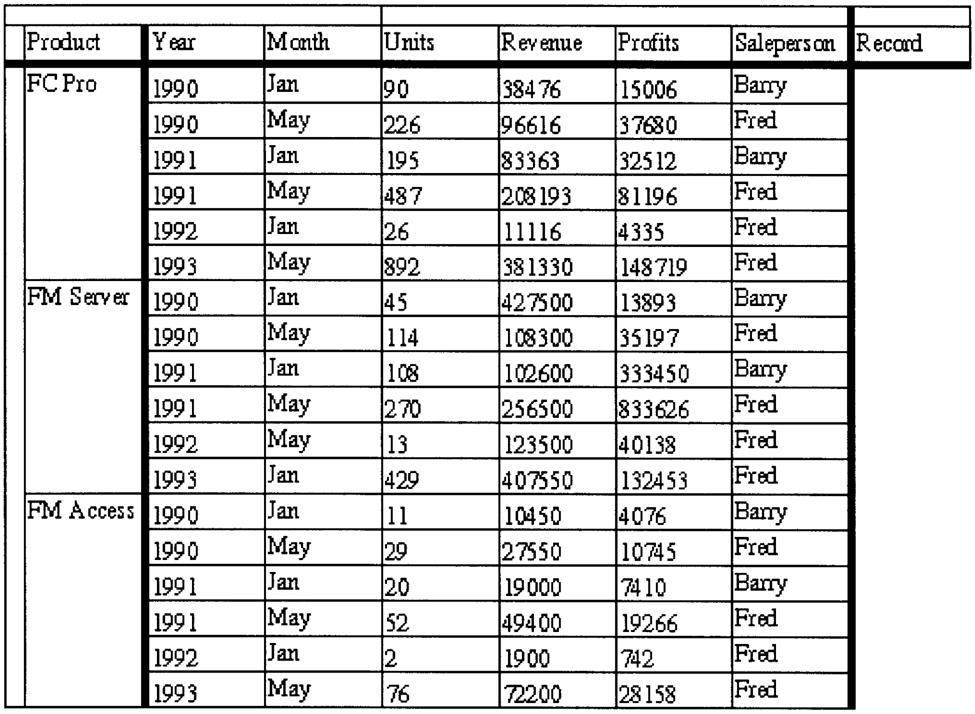
Description: When a dimension D_i is merged with dimension D_j the keys of D_i are paired with the keys of D_j and the dimensionality of the data set is reduced.

Formulation:

$$\left\{\begin{array}{c} C^N \equiv \langle S^N, V^2 \rangle \\ S^N \equiv \langle A_1^{N_1}, A_2^{N_2} \rangle \\ A_1^{N_1} \equiv \langle D_1, \ldots, D_{N_1} \rangle \\ A_2^{N_2} \equiv \langle D_{N_1+1}, \ldots, D_N \rangle \\ \langle m, n \rangle = \langle N_1, N_1 - 1 \rangle \quad \text{or} \quad \langle m, n \rangle = \langle N, N-1 \rangle \end{array}\right\} \tag{4.9}$$

$$\text{Fold}(D_m, D_n) : C^N \rightarrow C'^N$$

$$\{\text{Fold}(\mathcal{A}(D_m), \mathcal{A}(D_n)) : \mathcal{A}(C^N) \rightarrow \mathcal{A}(C'^N)\}$$



Product	Year	Month	Units	Revenue	Profits	Saleperson	Record
FC Pro	1990	Jan	90	38476	15006	Barry	
	1990	May	226	96616	37680	Fred	
	1991	Jan	195	83363	32512	Barry	
	1991	May	487	208193	81196	Fred	
	1992	Jan	26	11116	4335	Fred	
	1993	May	892	381330	148719	Fred	
FM Server	1990	Jan	45	427500	13893	Barry	
	1990	May	114	108300	35197	Fred	
	1991	Jan	108	102600	333450	Barry	
	1991	May	270	256500	833626	Fred	
	1992	May	13	123500	40138	Fred	
	1993	Jan	429	407550	132453	Fred	
FM Access	1990	Jan	11	10450	4076	Barry	
	1990	May	29	27550	10745	Fred	
	1991	Jan	20	19000	7410	Barry	
	1991	May	52	49400	19266	Fred	
	1992	Jan	2	1900	742	Fred	
	1993	May	76	72200	28158	Fred	

Figure 4.7: The Example Data Set After Merging the "Month" and "Year" dimensions

In the visual schema the merging operation is most conveniently visualized if D_i is at the bottom of a hierarchy and D_j is at the level above D_i in the hierarchy. If this is not the case, then one must first apply a transposition to the data set which brings D_i and D_j at the bottom two levels of a hierarchical tree on the same axis.

The example on figure 4.7 demonstrates merging the "Month" dimension with the "Year" dimension.

Another way to think about this operation is as merging of slabs. In the initial data set there were all together 12 slabs, arranged vertically, each corresponding to individual year and product. After merging the "Month" and "Year" dimensions, the slabs under each key of the "Product" dimension were merged to produce 3 larger slabs.

4.4.7 Slicing Through a Dimension

Arguments: **`dimension`**, **`key`**

Description: This operation extracts a *(N-1)*-dimensional slice from the data set in which all data cells are addressed by the same key.

Formulation:

$$\left\{\begin{array}{l} C^N \equiv \langle S^N, V^2 \rangle \\ S^N \equiv \langle A_1^{N_1}, A_2^{N_2} \rangle \\ A_1^{N_1} \equiv \langle D_1, \ldots, D_{N_1} \rangle \\ A_2^{N_2} \equiv \langle D_{N_1+1}, \ldots, D_N \rangle \\ j = 1 \quad \text{or} \quad j = N_1 + 1 \end{array}\right\} \tag{4.10}$$

$$\mathrm{Slice}(D_j, k_j[i_j]) : C^N \to C'^N$$

$$\{ \mathcal{A}(C'^N) = \mathrm{Slice}(\mathcal{A}(D_j), \mathcal{A}(k_j[i_j])) : C^N \}$$

	Product	Year	Month	Units	Revenue	Profits	Saleperson	Record
	FC Pro	1990	Jan	90	38476	15006	Barry	
			May	226	96616	37680	Fred	
		1991	Jan	195	83363	32512	Barry	
			May	487	208193	81196	Fred	
		1992	Jan	26	11116	4335	Fred	
		1993	May	892	381330	148719	Fred	

Figure 4.8: Slicing the Example Data Set through the "Product" dimension

As in the case of the merging operation, the most intuitive visual representation of the slicing operation requires that some restrictions are introduced. Namely, in order to slice through a dimension, its visual counterpart mast occupy the top of a dimensional hierarchy. If that is not the case, one must first apply an appropriate transposition.

The example on figure 4.8 illustrates the result of slicing the example data set through the "Product" dimension at the key "FC Pro".

It is easy to notice that the visual equivalent of slicing through a dimension, equivalent to following a path in the hierarchical tree down one branch from the root. But a path in the hierarchical tree is actually a data consolidation path, so another way to think about the slicing operation is as the opposite of data consolidation, i.e. data drilling. Once we have sliced through the "Product" dimension, we can continue slicing down through the "Year" and "Month" dimensions, producing finer and finer chunks of data.

4.4.8 Sorting

Arguments: **`group`**, **`direction`**

Description: Sorts all sticks within **`group`**, where **`group`** is a leaf of a hierarchical tree, i.e. either a column or row of sticks.

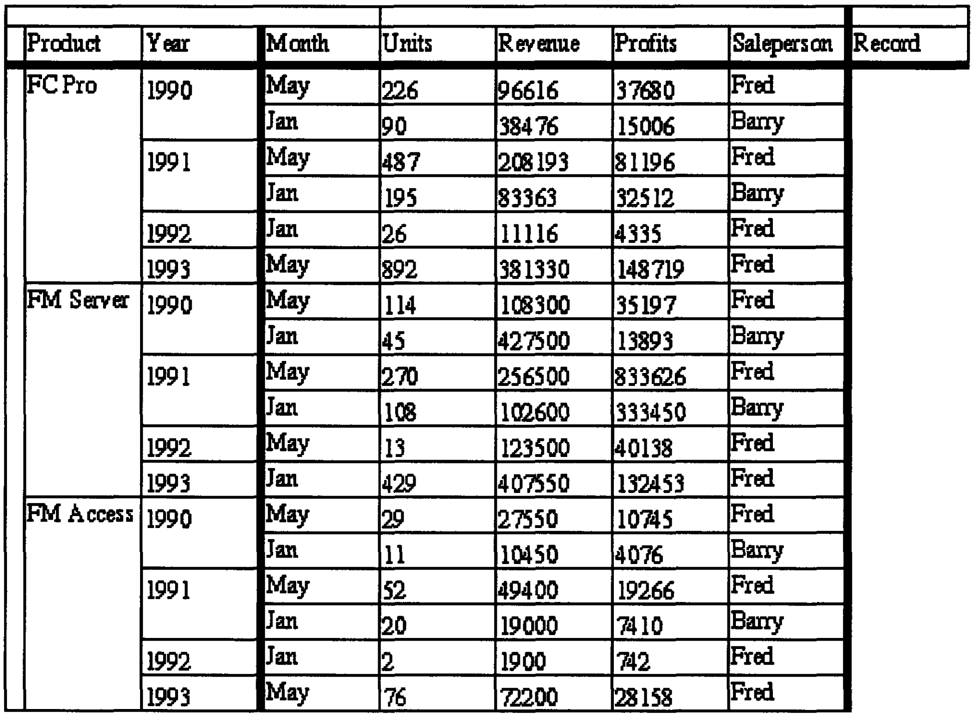
Formulation:

$$\left\{\begin{array}{c} C^N \equiv \langle S^N, V^2 \rangle \\ S^N \equiv \langle A_1^{N_1}, A_2^{N_2} \rangle \\ A_1^{N_1} \equiv \langle D_1, \ldots, D_{N_1} \rangle \\ A_2^{N_2} \equiv \langle D_{N_1+1}, \ldots, D_N \rangle \\ g = g[d_1, \ldots, d_{N_1+1}] \text{ or } g = g[d_{N_1+1}, \ldots, d_{N+1}] \end{array}\right\} \tag{4.11}$$

$$\text{Sort}(g, \text{direction}) : C^N \rightarrow C'^N$$

$$\mathcal{A}(C'^N) = \begin{cases} \text{Sort*}(Q_N[d_1, \ldots, d_{N_1}, i_{N_1}, \ldots, i_{N-1}], \text{direction}) : \mathcal{A}(C'^N) & \text{for } g = g[d_1, \ldots, d_{N_1+1}] \\ \text{Sort*}(Q_{N_1}[i_1, \ldots, i_{N_1-1}, d_{N_1+1}, \ldots, d_{N+1}], \text{direction}) : \mathcal{A}(C'^N) & \text{for } g = g[d_{N_1+1}, \ldots d_{N+1}] \end{cases}$$

The * in the above set of equations denotes that the respective function is applied multiple times for all indices *i*.



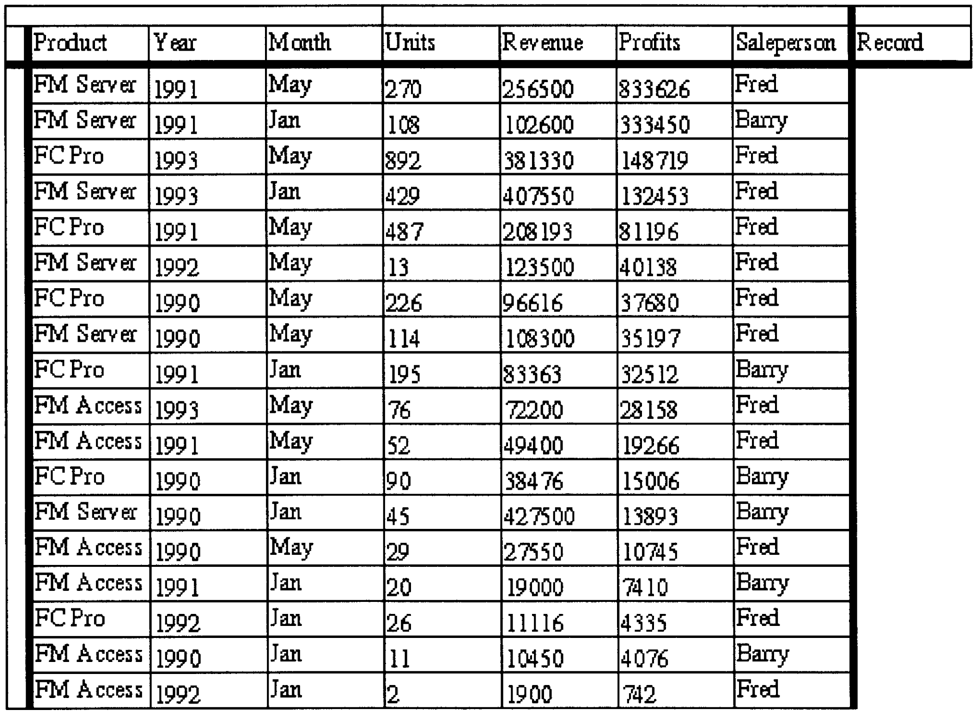
Product	Year	Month	Units	Revenue	Profits	Salepersom	Record
FC Pro	1990	May	226	96616	37680	Fred	
		Jan	90	38476	15006	Barry	
	1991	May	487	208193	81196	Fred	
		Jan	195	83363	32512	Barry	
	1992	Jan	26	11116	4335	Fred	
	1993	May	892	381330	148719	Fred	
FM Server	1990	May	114	108300	35197	Fred	
		Jan	45	427500	13893	Barry	
	1991	May	270	256500	833626	Fred	
		Jan	108	102600	333450	Barry	
	1992	May	13	123500	40138	Fred	
	1993	Jan	429	407550	132453	Fred	
FM Access	1990	May	29	27550	10745	Fred	
		Jan	11	10450	4076	Barry	
	1991	May	52	49400	19266	Fred	
		Jan	20	19000	7410	Barry	
	1992	Jan	2	1900	742	Fred	
	1993	May	76	72200	28158	Fred	

Figure 4.9: Sorting the Example Data Set by "Profits"

Since the visual vault consists of slabs, which can be viewed as either a collection of horizontal or vertical sticks, there are two possible orientations in which sorting can be applied: horizontally or vertically. In addition, even though sorting by a single stick is possible, it is far more common to apply the sorting operation to a whole collection of sticks, i.e. to the sticks of a row or column.

The example on figure 4.9 shows the result of sorting the example set by the "Profits" column. The direction of sort is "Up", which indicates that the large values of the "Profits" variable should appear towards the top of the table.

Since only the cells within a stick are permuted and each stick is only a 1 or 2 cells long, the overall result does not differ much from the picture on figure 4.5. After, merging the "Month" and the "Year" dimensions, however, the resulting set has fewer but longer sticks, because slabs were merged vertically. Then, sorting has a more noticeable effect. The effect will be even greater if we reapply the merging operation thus merging the "Year"+"Month" dimension with the "Product" dimension. The result after sort is shown on figure 4.10.



Product	Year	Month	Units	Revenue	Profits	Saleperson	Record
FM Server	1991	May	270	256500	833626	Fred	
FM Server	1991	Jan	108	102600	333450	Barry	
FC Pro	1993	May	892	381330	148719	Fred	
FM Server	1993	Jan	429	407550	132453	Fred	
FC Pro	1991	May	487	208193	81196	Fred	
FM Server	1992	May	13	123500	40138	Fred	
FC Pro	1990	May	226	96616	37680	Fred	
FM Server	1990	May	114	108300	35197	Fred	
FC Pro	1991	Jan	195	83363	32512	Barry	
FM Access	1993	May	76	72200	28158	Fred	
FM Access	1991	May	52	49400	19266	Fred	
FC Pro	1990	Jan	90	38476	15006	Barry	
FM Server	1990	Jan	45	427500	13893	Barry	
FM Access	1990	May	29	27550	10745	Fred	
FM Access	1991	Jan	20	19000	7410	Barry	
FC Pro	1992	Jan	26	11116	4335	Fred	
FM Access	1990	Jan	11	10450	4076	Barry	
FM Access	1992	Jan	2	1900	742	Fred	

Figure 4.10: Sorting after Merging the "Month", "Year" and "Product" dimensions

4.4.9 Promoting a Variable

Arguments: `variable`, `axis`, `level`

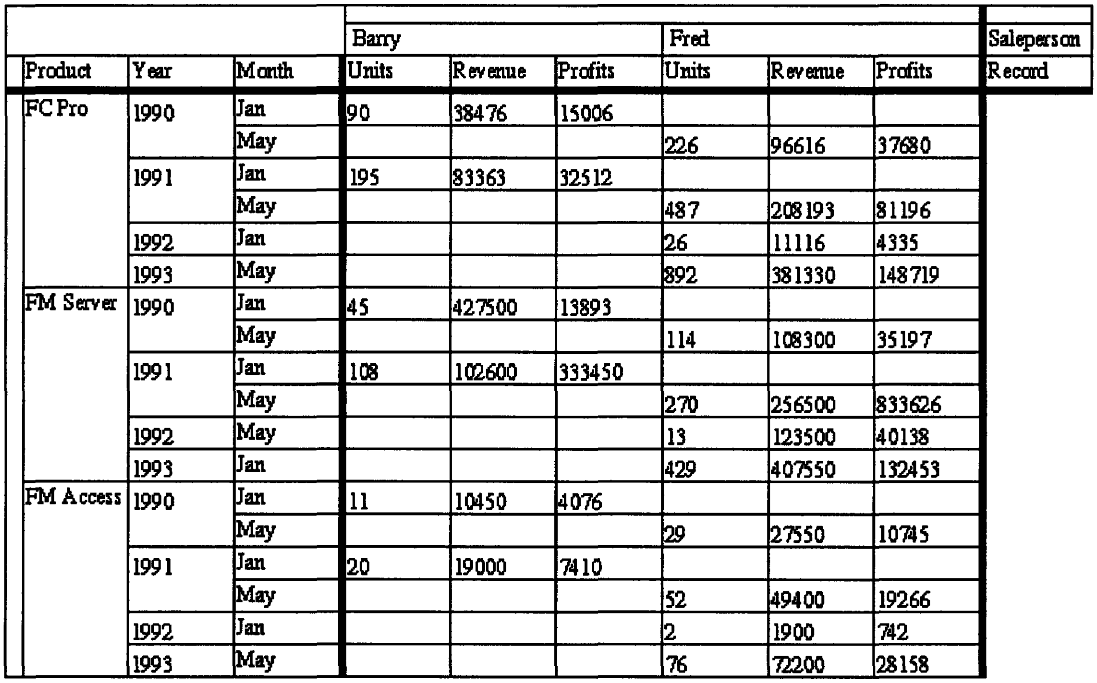
Description: Promotes `variable` to the level `level` of the hierarchical tree on `axis`.

Promoting a variable, is the operation through which a variable can be used as a dimension, and its values as dimension keys. When a variable is promoted, a new dimension is created whose visual counterpart, can be associated with either the horizontal or vertical axes. To indicate the hierarchical tree into which the new visual dimension is integrated, we will often describe the promotion operation as either *promoting a variable to the horizontal axis* or *promoting a variable to the vertical axis.*

Formulation:

$$
\begin{array}{l}
\left\{\begin{array}{ll} 1 \le l \le N_1 & \text{if } A = A_1^{N_1} \\ 1 \le l \le N_2 & \text{if } A = A_2^{N_2} \end{array}\right\} \\
\text{Promote}(var, A, l) : C^N \rightarrow C'^{N+1} \\
\left\{\begin{array}{ll} \mathcal{A}(C'^{N+1}) = \text{Promote}(\mathcal{A}(var)) : \mathcal{A}(C^N) & \\ \quad A'^{N_1+1}_1 = \langle D_1, \ldots, D_{l-1}, D_{N+1}, D_l, \ldots, D_{N_1} \rangle & \text{if } A = A_1^{N_1} \\ \quad A'^{N_2+1}_2 = \langle D_{N_1+1}, \ldots, D_{N_1+l-1}, D_{N+1}, D_{N_1+l}, \ldots, D_N \rangle & \text{if } A = A_2^{N_2} \end{array}\right\}
\end{array}
\qquad (4.12)
$$

The example on figure 4.11 shows the result of promoting the “Salesperson” variable to the horizontal axis.



			Barry			Fred			Salepersan
Product	Year	Month	Units	Revenue	Profits	Units	Revenue	Profits	Record
FC Pro	1990	Jan	90	38476	15006				
		May				226	96616	37680	
	1991	Jan	195	83363	32512				
		May				487	208193	81196	
	1992	Jan				26	11116	4335	
	1993	May				892	381330	148719	
FM Server	1990	Jan	45	427500	13893				
		May				114	108300	35197	
	1991	Jan	108	102600	333450				
		May				270	256500	833626	
	1992	May				13	123500	40138	
	1993	Jan				429	407550	132453	
FM Access	1990	Jan	11	10450	4076				
		May				29	27550	10745	
	1991	Jan	20	19000	7410				
		May				52	49400	19266	
	1992	Jan				2	1900	742	
	1993	May				76	72200	28158	

Figure 4.11: The Example Data Set after promoting “Salesperson”.

The empty cells represent nonexistent values.

4.4.10 Marking Region

Arguments: `regions`, `labels`

Description: This visual operation simply invokes the abstract data model operation *mark* with the same arguments. (See “Mark” on page 56.) In the current SpreadCube implementation the marking of a region operation uses default labels from the set: {“clsuter1”, “cluster2”,...}

4.4.11 Aggregation

Arguments: `dimension`, `function`

Description: This visual operation invokes the abstract data model operation *aggregate* with the same arguments. (See “Aggregate” on page 57.) In the current SpreadCube implementation, the aggregation operation uses a generic averaging function as the default function for aggregation. The averaging function depends on the data domain which is being aggregated. This dependence is summarized in table 5.3.

Chapter 5

Visualization Techniques

In chapter 4 we discussed the visual model of SpreadCube, which describes the layout of visual objects on a two dimensional medium. The goal of this chapter is to present the rules according to which visual objects are rendered on a physical medium and motivate them.

5.1 What is Visualization?

"A picture is worth a thousand words". In a simpler language, the proverb is stating that the bandwidth of our visual perception is far greater than the bandwidth of our auditory perception. Hence vision is the primary channel for presenting information to the user in a computer-human interaction loop. That is why we use the term *visualization* to refer to the act of information presentation, even though other senses like: auditory and even tactile perception may be involved.

5.2 Visualization Goals

Even though the bandwidth of our vision is quite remarkable, it is nevertheless limited. Limitations vary from purely physical, e.g. dots per inch, screen size, color differentiation, to mostly psychological, i.e. how quickly we can analyze and make sense of the information perceived. Whenever the amount of information perceived is close to the capacity of the channel, the observed object appears to be cluttered and our ability for effective analysis is greatly impeded.

The primary goal of information visualization, regardless of what is being visualized is to present most useful information with the least amount of clutter. To accomplish this goal, a good visualization must account for perception bottlenecks on all levels: from the physical to the mental.

5.2.1 Exploring Multiple Information Channels

A rather simple but useful model of perception is the following: Our minds receive information in parallel through various channels. For example each of our senses corresponds to a major information channel, e.g. vision, auditory, tactile and etc. channels.

A major channel, like vision, is actually a collection of smaller channels through which visual information is presented. Some such channels are: text, graphs, color, shading, highlighting, object's position, orientation and size, etc.

Similarly a channel like audition can be broken down into: pitch, timber, amplitude, amplitude variations, noises, e.g. a click and etc.

One way a perception bottleneck is created is if the bulk of the information is streamed through the same channel. Therefore, a simple way to alleviate such bottleneck is to spread the information across several channels.

Each channel has its own specific methods for analyzing information. This makes some types of information more suited for one channel than other. Thus, a picture may be worth a thousand words but a musical piece is worth a thousand pictures of scores. Therefore, it would seem that exploring multiple channels is not always appropriate. In general, however, the information which we are analyzing contains patterns that are not all readily picked out by a single information channel alone. In such cases exploring multiple channels is not just advantageous, but actually necessary, because it helps circumvent perception bottlenecks at the mental level.

Streaming information through multiple channels can prevent information bottlenecks at the physical level of perception too. This is possible because two different channels, e.g. text and color, do not interfere much with each other. For example, one can add more information to a page of text by coloring some words. By doing so, one would have increased the amount of information presented on the page without cluttering it.

5.2.2 Exploiting Redundancies

Yet another way of involving multiple channels is to have them analyze the same information. For example, representing a quantity value both graphically and textually at the same time. In this manner our mind's comprehension is enhanced because information processing is being done in parallel through virtually independent means.

5.2.3 Modeling Degrees of Interest

A feature which contributes to our efficiency of perception is the ability to maintain variable degrees of interest in different parts of our environment. In other words, when we focus our attention on a particular object, we decrease our interest in its surroundings, yet we still remain aware of them.

A well designed visualization should reflect the user's partitioning of the environment into regions of different degrees of interest. For example regions which represent a higher degree of interest to the user should be rendered in greater detail and vice versa. This way the efficiency of the visualization is improved, i.e. greater information content with fewer resources.

The techniques of employing several degrees of interest involves designating certain areas of the visualization to be the *focus* of attention and the rest to be the *context*. Such techniques are known as *focus+context* techniques, or as *fisheye* techniques, because they often cause the visualized objects to be distorted as viewed through the eyes of a fish. [2][3][4][5][13].

5.2.4 Maintaining User's Interest and Attention

In order to convey any kind of information, a visualization must first capture and maintain user's attention. A large contribution to that effect has the artistic design and simply how pleasant the application outlook is to the user.

Another important aspect of maintaining user's attention is avoiding distractions or confusions. Extraneous gadgets, like pop-up windows and such, which could divert user's attention should be avoided as much as possible. Any abrupt change on the screen will have the effect of disorienting the user, who will have to spend time trying to understand what had happened. Therefore to avoid confusion of such sort, any less than trivial change in the current visualization should be animated. [2][3][13]

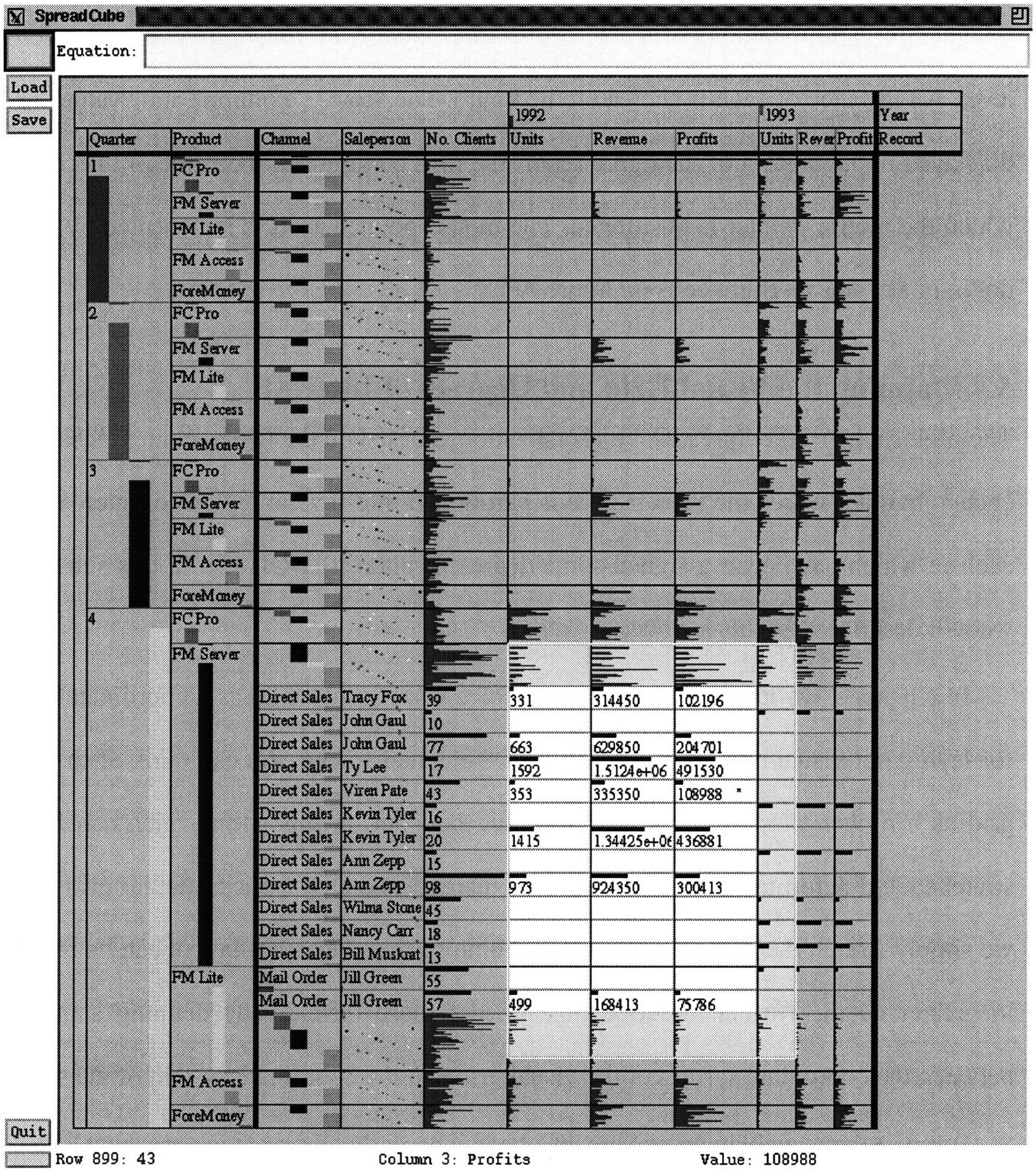
5.2.5 Familiarity

Regardless of how well one visualization is designed, there is always a learning threshold which one must overcome before he or she can understand the language of the visualization. Therefore, a visualization must explore people's preconceived notions of how information should be represented. On the other hand, conforming to such notions may be too restricting in terms of introducing novel features and potentially better, but unfamiliar representations. The best solution is to have the option of defaulting to a familiar view, or demonstrating how the current visualization contains one or several familiar views.

5.3 The SpreadCube Visualization

In the rest of this chapter we will describe the SpreadCube visualization and each of its components and operations in detail. However, since most of these components are intertwined with one another, it is best to begin by first showing some overviews of the whole picture and then drill into details.

The example data set which is used here, is an extension of the example data set introduced in chapter 3. The set dimensions are: "Year", "Quarter", "Product", "Channel", "Saleperson" and "No. Clients" (number of clients). The key values for the "No. Client" dimension are quantities, and all other key values are nominals.



Channel	Salesperson	No. Clients	Units	Revenue	Profits
Direct Sales	Tracy Fox	39	331	314450	102196
Direct Sales	John Gaul	10			
Direct Sales	John Gaul	77	663	629850	204701
Direct Sales	Ty Lee	17	1592	1.5124e+06	491530
Direct Sales	Viren Pate	43	353	335350	108988
Direct Sales	Kevin Tyler	16			
Direct Sales	Kevin Tyler	20	1415	1.34425e+0(	436881
Direct Sales	Ann Zepp	15			
Direct Sales	Ann Zepp	98	973	924350	300413
Direct Sales	Wilma Stone	45			
Direct Sales	Nancy Carr	18			
Direct Sales	Bill Muskrat	13			
Mail Order	Jill Green	55			
Mail Order	Jill Green	57	499	168413	75786

Row 899: 43 Column 3: Profits Value: 108988

Figure 5.1: The SpreadCube Visualization

The SpreadCube visualization (figure 5.1 and figure A.1) is exemplification of incarnating the visualization goals listed above. It uses multiple information channels by employing several *presentation types* [2] for visualizing information: text, graphs, color, position and sound. There are three degrees of interest, or focus levels, that can be assigned to various regions of the screen: *background* level, *limelight* level and *spotlight* level. An instance of each is shown on the figure. The "row", "column" and "value" fields displayed in the bottom of the figure report the row, column and the value of the cell over which the mouse pointer is positioned. The mouse pointer itself is represented by the red dot in one of the spotlight cells on figure A.1.

5.4 Mapping the Visual Grid and Degrees of Interest

The philosophy of the SpreadCube visualization is based on the focus+context idea (references). In other words the visual space is partitioned into several regions of interest (ROI) each of which have been assigned some degree of interest (DOI) [2]. In this sub-section we will describe how this is done.

In chapter 4 we introduced the concept of a visual grid which is a coordinate system for laying out visual objects independent of the actual screen. We can think of the visual grid as a rubber sheet; the visual objects are laid out on this sheet which can be then stretched and deformed to fit the screen. To increase the DOI for a particular grid region, we simply stretch the rubber sheet at that region so that the visual layout can be rendered in greater detail. We can describe this process more formally by defining a mapping between the coordinate system of the visual grid and the coordinate system of the screen.

Because the visual layout of SpreadCube is inherently tabular, it is a reasonable design to choose the ROI as rectangular regions containing cells of the visual vault. One advantage of this approach is that the mapping of an area can be described independently in terms of the mapping of the two screen axis.

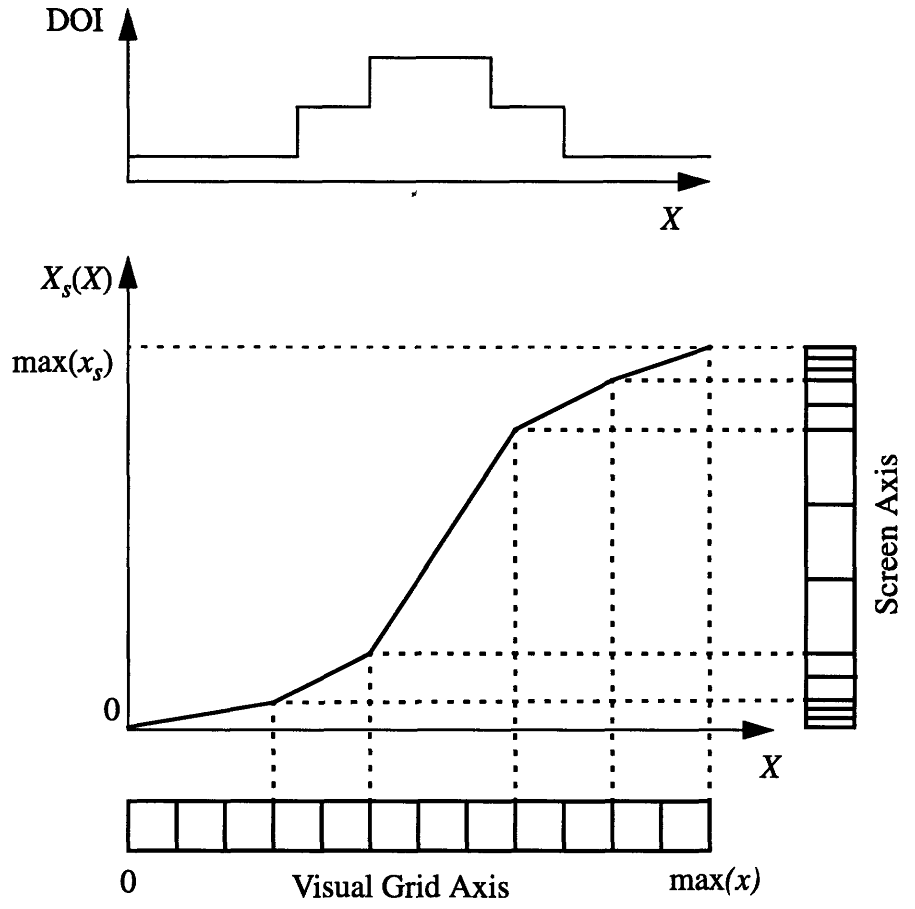


Figure 5.2: Mapping Visual to Screen Axes

An example of a mapping between visual and screen coordinates, analogous to figure 2 in Rao & Card's "Table Lense" [2], is shown on figure 5.2. We use (X_s, Y_s) to denote screen coordinates and (X, Y) to denote visual layout coordinates.

For the sake of clarity the above figure only demonstrates the mapping between the horizontal visual and screen axes. The mapping between the two corresponding vertical axes is analogous.

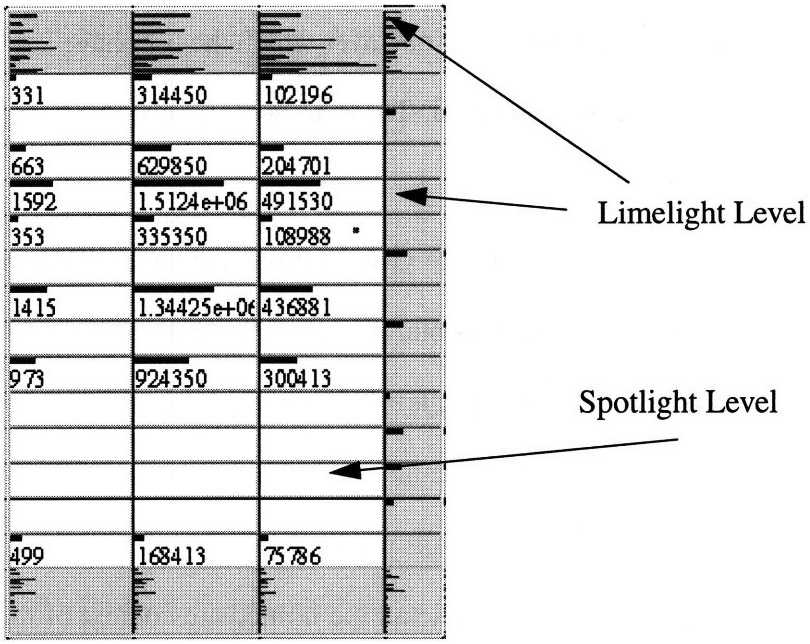
The point $\max(x_s)$ represents the largest horizontal screen coordinate. One specific goal of the SpreadCube visualization is to map the entire visual layout on the screen and to utilize as much as possible of the screen space. Therefore, the boundary conditions on the grid to screen mapping are that the origin and the point $(\max(x), \max(y))$ of the grid must map to the points $(0, 0)$ and $(\max(x_s), \max(y_s))$ of the screen.

5.5 Three Levels of Focus and Focal Area

The term "degree of interest" refers to our subjective partitioning of the visual space into more and less interesting regions.The objective analogue of this term, i.e. what is actually seen on the screen, is *focus level.* Ideally a high degree of interest should correspond to high focus level. However it is only up to the human operator to control the focus levels through the user interface in response to his or her changing degrees of interest.

The SpreadCube visualization is designed to handle three focus levels: *background*, *limelight* and *spotlight* levels, listed in order of increasing detail. The spotlight level displays vault cells large enough to include a legible textual representation of their contents. A group of spotlight cells appear similar to a group of cells in a spreadsheet. The limelight level does not guarantee the high detail of the spotlight level, however it guarantees a level of detail sufficient to represent a value individually. In other words, a limelight cell is guaranteed to be represented by at least one dedicated screen pixel. The background level does not impose any constraints on the number of cells mapped to a single screen pixel. That is why, when the data set is large enough, groups of values on the background level may be averaged together, according to some averaging method, before they are mapped onto the screen.

The DOI function applies to the screen axes and not to individual cells. Therefore, the focus level of an individual cell depends on the focus level of its row and column. Any background cells belongs to a background row and column. Any spotlight cell belongs to a spotlight row and column. A limelight cell could belong to a limelight row and column or a spotlight row and a limelight column or vice-versa.



331	314450	102196
663	629850	204701
1592	1.5124e+06	491530
353	335350	108988
1415	1.34425e+0	436881
973	924350	300413
499	168413	75786

Figure 5.3: A Focal Area

A continuous group of cells whose level of focus is higher than background, is called a *focal area*. The structure of a SpreadCube focal area is such that the central cells of the area are spotlighted while the peripheral cells are limelighted (see figure 5.3 or figure A.2). The limelight cells provide a context for the spotlight cells which is of greater resolution than the one provided by the background level. In this way the user is given a fine resolution control to point at the cells in the immediate vicinity of the spotlighted area.

That is how the focal area implements the focus+context principle in itself. The relative arrangement of spotlight and limelight areas is representative of a typical DOI partitioning. Indeed, the degree of interest is likely to fall off gradually with the distance from the spotlight area. Metaphorically speaking, the pack of focused cells functions as a magnifying glass, which can be positioned over any place of the visual vault. And just like some real magnifying glasses, it includes an additional lens for extra magnification.

SpreadCube can support multiple focal areas at the same time, but only one of them can be *active*. The active focal area is the one over which the user have active control as far as positioning and resizing is concerned. These issues are discussed in greater detail in chapter 6 on user interface.

The three levels of focus allow for the visualization of data sets with virtually no limitations of the size. For this to be possible, however, the existence of all three levels of focus is necessary. Without the spotlight level, one will not be able to view any region of the data set legibly. Without the background level, the largest possible size of the visual vault will be limited to the number of pixels on the screen. Without the limelight level, one will not be able to observe in sufficient detail the immediate context of the spotlight area, and to have fine control over its positioning.

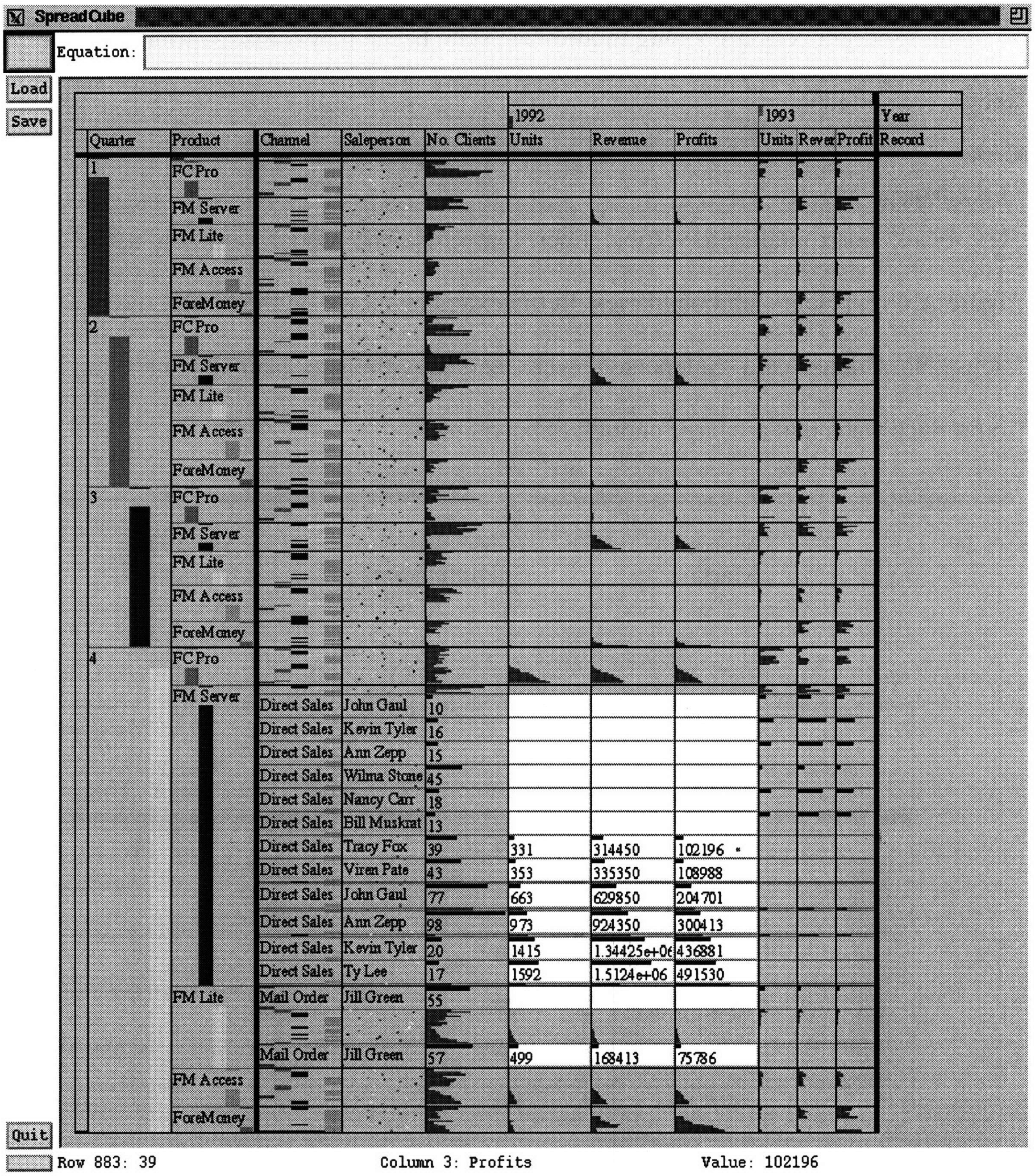
The example shown on figure 5.1 and figure A.1 contains 1972 rows, while the actual screen size of the vertical axis is about 800 pixels. Therefore, on average, one pixel represents more than one value along the vertical orientation. That is why the use of the limelight cells is essential in order to be able to observe the immediate surroundings of the spotlight area.

5.6 Sorting

Sorting is a simple but important operation from both visualization point of view and EDA point of view. It is also a basis for a number of visualization techniques in SpreadCube, and that is why it is worth paying a special attention to it.

5.6.1 Effects on the Focal Area

After permuting the rows or columns of the visual vault, the spotlight cells are not anymore packed together and the focal area is broken. When the focal area is broken, the limelight cells which used to form the halo of the focal area become obsolete and are reverted to background cells. The motivation is that the limelight cells do not really represent what is interesting to the user or else they would have been spotlighted.



					1992			1993			Year
Quarter	Product	Channel	Saleperson	No. Clients	Units	Revenue	Profits	Units	Rever	Profit	Record
1	FC Pro										
	FM Server										
	FM Lite										
	FM Access										
	ForeMoney										
2	FC Pro										
	FM Server										
	FM Lite										
	FM Access										
	ForeMoney										
3	FC Pro										
	FM Server										
	FM Lite										
	FM Access										
	ForeMoney										
4	FC Pro										
	FM Server	Direct Sales	John Gaul	10							
		Direct Sales	Kevin Tyler	16							
		Direct Sales	Ann Zepp	15							
		Direct Sales	Wilma Stone	45							
		Direct Sales	Nancy Carr	18							
		Direct Sales	Bill Muskrat	13							
		Direct Sales	Tracy Fox	39	331	314450	102196				
		Direct Sales	Viren Pate	43	353	335350	108988				
		Direct Sales	John Gaul	77	663	629850	204701				
		Direct Sales	Ann Zepp	98	973	924350	300413				
		Direct Sales	Kevin Tyler	20	1415	1.34425e+0	436881				
		Direct Sales	Ty Lee	17	1592	1.5124e+06	491530				
	FM Lite	Mail Order	Jill Green	55							
		Mail Order	Jill Green	57	499	168413	75786				
	FM Access										
	ForeMoney										

Row 883: 39 Column 3: Profits Value: 102196

Figure 5.4: SpreadCube After Sorting by “Profits”

The spotlight cells, however, remain in focus as they are permuted. This way, the user can track their motion with respect to the rest. This is illustrated on figure 5.4 and figure A.4 where the column of sticks corresponding to "Profits" in 1992 was sorted. [3]

The spotlight cells all belong to the same slab, hence they remain within that slab after they were sorted.

5.6.2 Merge Marks

The black horizontal and vertical lines that cross the visual vault on figure 5.4 and figure A.4 represent slab boundaries. In the example shown on the figure, the two dimensions "No. Clients" and "Saleperson" were merged with the "Channel" dimension in order to produce slabs that are large enough to be visible.

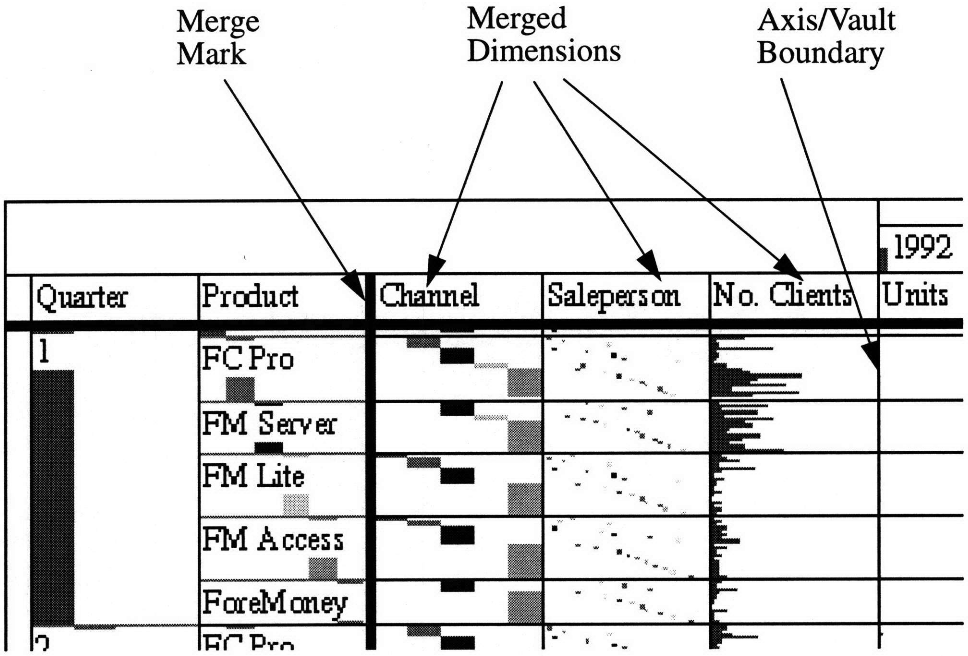


Figure 5.5: Vertical Merge Mark

The visual equivalent of merging two dimensions, as explained in chapter 4 is merging of slabs. In the visual model, only the dimensions which constitute the bottom of the dimensional hierarchy can be merged. That is why the merging of slabs can be specified conveniently by simply specifying a level in the hierarchical tree under which all dimensions are merged into one.

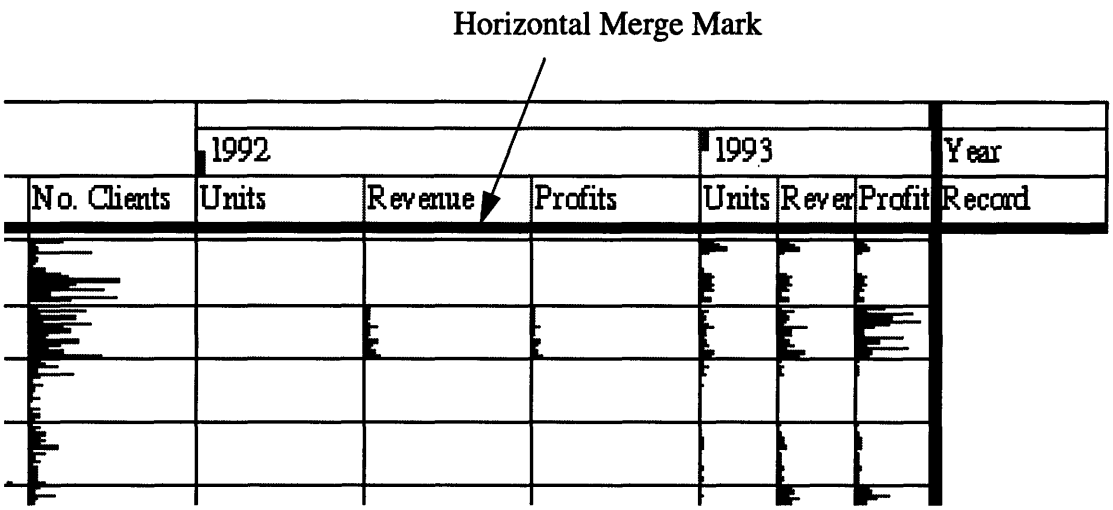


Figure 5.6: Horizontal Merge Mark

Such level is defined by a vertical or horizontal merge marks (see 5.5 and 5.6). The merge mark can slide left-right or up-down thus merging or un-merging dimensions. The visual effect of sliding a merge mark is a change in the slab boundaries. For example, the slab boundaries on the right of the vertical merge mark, i.e. on the side of the merged dimensions, correspond to the visual key boundaries of the dimension immediately to the left of the merge mark.

Slabs of values which belong to different variables cannot be merged together. Indeed, if merging slabs from different variables were possible then a sorting operation in the resulting larger slab would require that values of different variable types be compared to each other. In order to prevent this form happening we create add a restriction according to which a merge mark cannot move above the level of the record dimension. For example on figure 5.6 the merge mark is set immediately under the level of the record dimension which causes the width of the slabs to be 1.

5.6.3 Sorting Non-Quantity Values

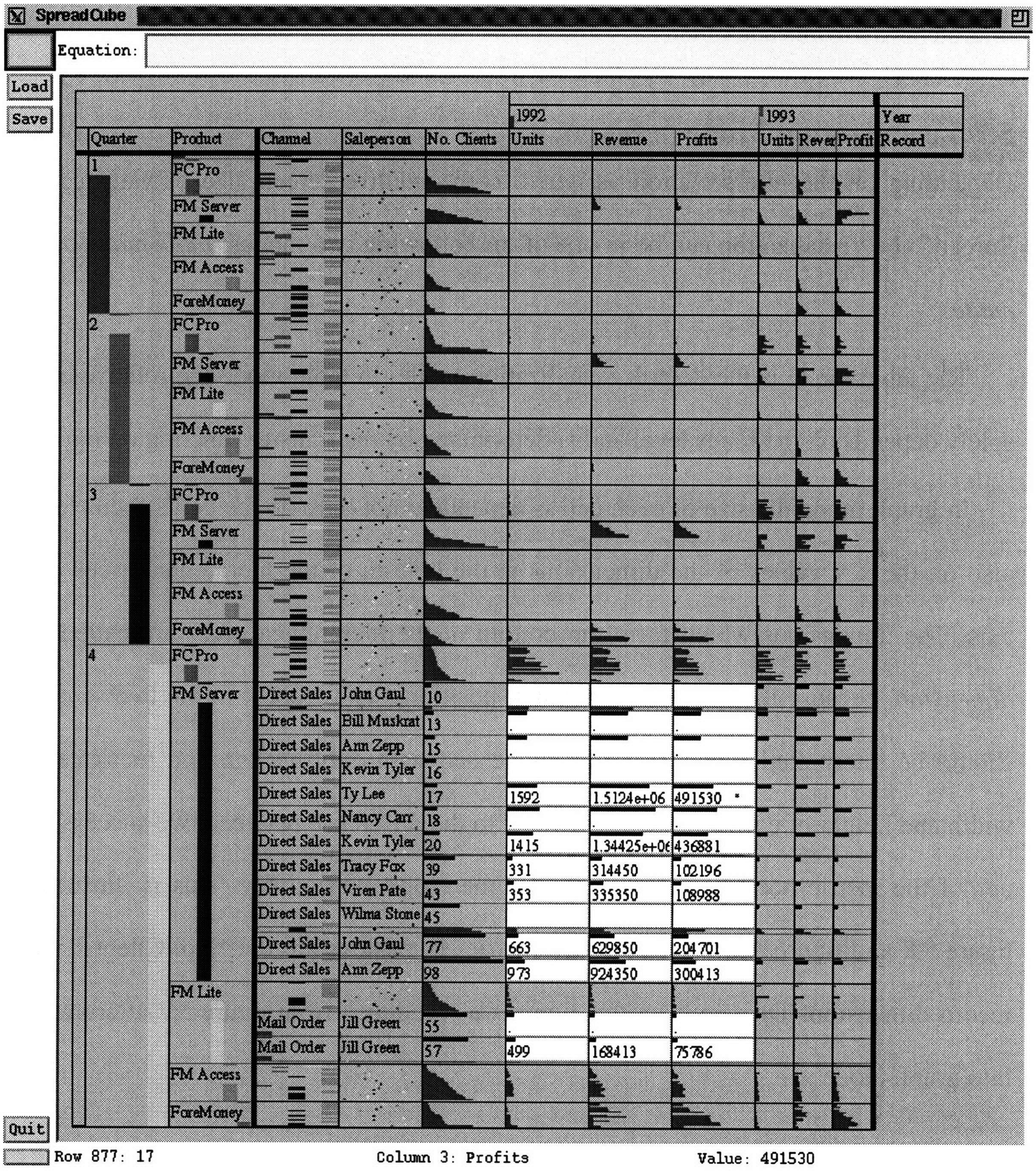
Sorting, by definition, is rearranging a sequence of values in a increasing or decreasing order and that is why it seems that sorting can only apply to sets of values for there is an ordering relation defined, e.g. quantity values. Often, however, sorting is used as a grouping operation which groups equal values together, and in such cases it should apply to any set of values which can be compared to each other.

Rather than providing a special grouping operation that will play the role of sorting in the case of non-quantity values, we can simply introduce an arbitrary ordering relation amongst those values and still use sorting on them for grouping. For example, nominal values, e.g. the rgb colors, are not intrinsically ordered. We can, however, assigned an arbitrary order according to the sequence in which each nominal value was first encountered and then used that order for sorting. Thus, in the case of rgb colors, we would assign the following order: "red" < "green" < "blue".

5.6.4 Sorting Dimension Keys

From a visualization point of view the dimension key values can be sorted just like any variable row or column. Unlike the row/column sort, however, sorting the dimension keys does not result in a stick sort of operation on the abstract data model, as described in chapter 3 but is in fact a special case of the "reorder" operation.

When several dimension are merged together as is the case on figure 5.1, their keys are replicated and grouped into tuples. It would seem that a sorting operation on a merged dimension should be sorting the tuples, but such operation does not produce useful visualizations. Instead, the sorting of a merged dimension is defined to apply only to the keys of one dimension and not to the tuple as a whole. As far as the abstract data model is concerned, both ways of defining the sort operation are legal, because they are both special cases of an abstract data model operation, namely the "reorder" operation.



Quarter	Product	Channel	Saleperson	No. Clients	1992 Units	Revenue	Profits
4	FM Server	Direct Sales	John Gaul	10			
		Direct Sales	Bill Muskrat	13			
		Direct Sales	Ann Zepp	15			
		Direct Sales	Kevin Tyler	16			
		Direct Sales	Ty Lee	17	1592	1.5124e+06	491530
		Direct Sales	Nancy Carr	18			
		Direct Sales	Kevin Tyler	20	1415	1.34425e+06	436881
		Direct Sales	Tracy Fox	39	331	314450	102196
		Direct Sales	Viren Pate	43	353	335350	108988
		Direct Sales	Wilma Stone	45			
		Direct Sales	John Gaul	77	663	629850	204701
		Direct Sales	Ann Zepp	98	973	924350	300413
	FM Lite	Mail Order	Jill Green	55			
		Mail Order	Jill Green	57	499	168413	75786

Row 877: 17 Column 3: Profits Value: 491530

Figure 5.7: SpreadCube Sorted by "No. Clients"

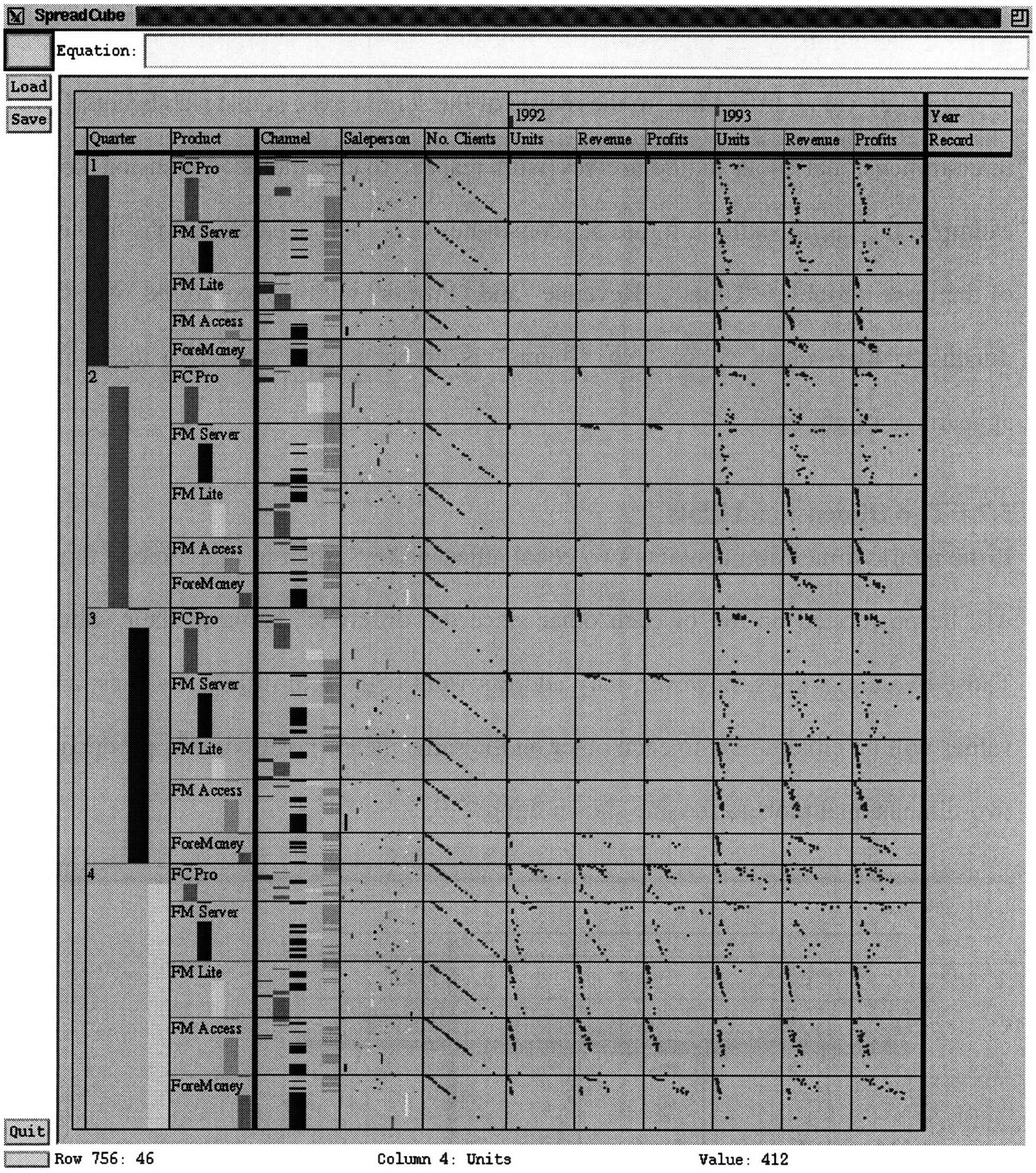
The example on figure 5.7 and figure A.5 illustrates sorting the "No. Client" dimension which has been previously merged together with the "Channel" and "Saleperson" dimension. The sort causes equal keys from the "Channel" and "Saleperson" dimension to migrate away from each other.

5.7 Visualization Mode

Depending on the interpretation assigned to the relative screen size of vault cells, the SpreadCube visualization can be in one of the following two modes: *table mode* or *graph mode.*

The table mode is the default visualization mode. In table mode the relative cell size solely depends on the focus level and is determined by the transfer function on figure 5.2.

In graph mode the size of each cell is dependent not only on the cell's focus level but also on the key values of the dimensions at the bottom of the hierarchical trees of each axis. The dimensions which form the bottom of the hierarchical trees are called *metric dimension*, because they define how spatial positions with respect to the two screen axis should be interpreted. Thus, in graph mode and with only background focus cells, the width and height of each cell is proportional to the difference between two successive values of the metric horizontal and vertical dimensions respectively. This is illustrated on figure 5.8 and figure A.6 where the two metric dimensions are the "No. Clients" and the record dimension. The result on this figure was produced from figure 5.7 after switching into graph mode.



Row 756: 46 Column 4: Units Value: 412

Figure 5.8: SpreadCube in Graph Mode

If the keys of a metric dimension are not quantity values, then in both graph and table mode, they are evenly spaced, which is the case for the record dimension on figure 5.8.

Graph mode explores the fact that a column (or a row) of numbers can be its own graph if the numbers are represented graphically and spaced appropriately. The effect of applying key value differences to the metric of the screen axis, is that slab become two- or three-dimensional plots of themselves with respect to the metric dimension keys. For example, the visual vault on figure 5.8 constitutes of 21×6 separate graphs of the values of the three variables "Units", "Revenue" and "Profits" with respect to the "No. Clients" dimension key values. Since "No. Clients" is graphed with respect to itself, its plots appear as straight lines.

5.7.1 Two Dimensional Plots

If the metric dimension contains two equal adjacent keys, then in graph mode those keys will be positioned on top of each other since the difference between their values is 0. Those two equal keys, however, may address vault cells with different values, and such values will be plotted next to each other on the same metric position, thus producing truly two-dimensional plots as the one shown figure 5.9.

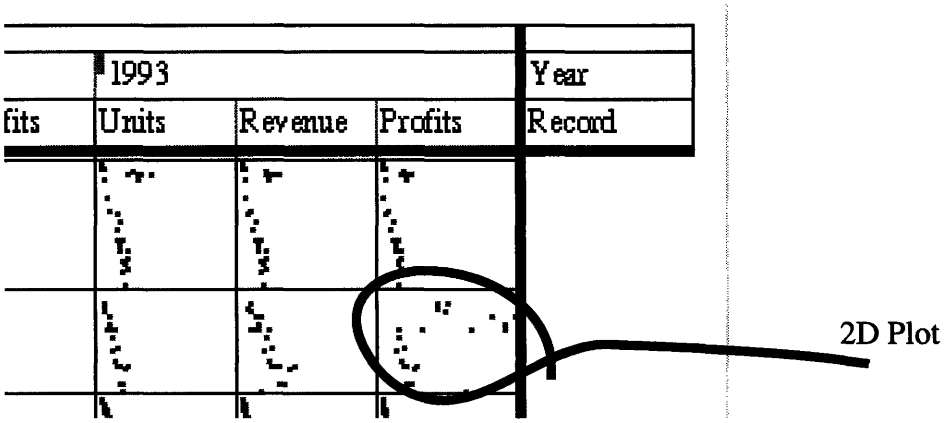


Figure 5.9: Two Dimensional Plots

The marked plot on the above figure is shown enlarged on figure 5.10 to illustrate in a more conventional way what is being plotted against what.

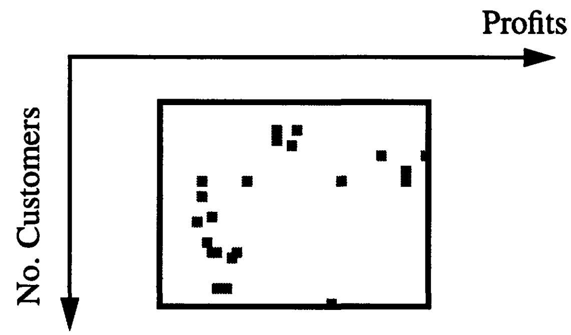


Figure 5.10: The 2D Plot on figure 5.9

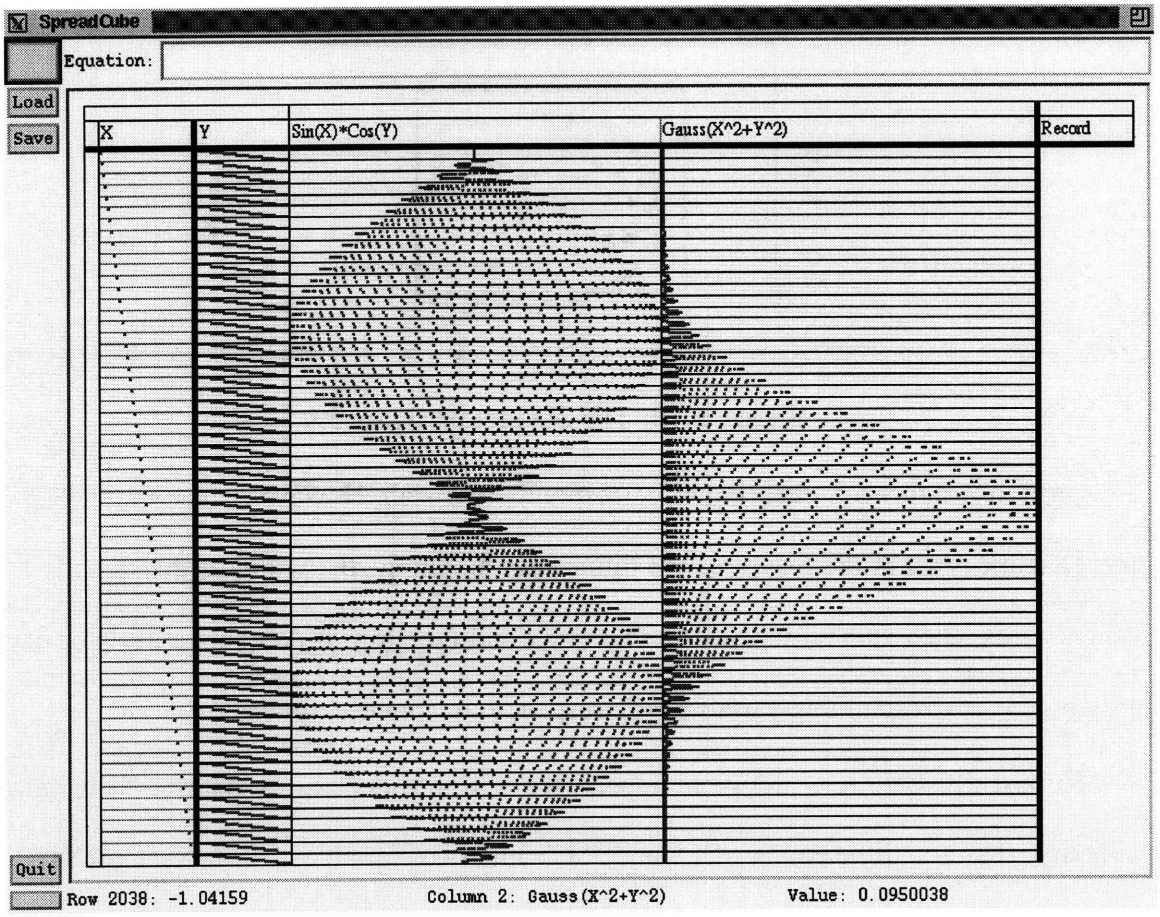
On figure figure 5.8 each graph corresponds to a slab. However, since the horizontal merge mark is set at the leaves of the dimension hierarchy, the width of the slabs is 1. In other words, each slab is degenerate to a stick and therefore it contains a 2D plot of the values of its corresponding variable with respect to *one* metric dimension.

To best illustrate how 2D plots appear, we will introduce a toy dataset. This dataset contains two set dimensions: "X" and "Y" and two variables: "Sin(X)*Cos(Y)" and "Gauss(X^2+Y^2)". The two variables are functions of the keys of "X" and "Y":

$$\text{Sin(X)*Cos(Y)} = \sin(x)\cos(y) \tag{5.1}$$

$$\text{Gauss(X^2+Y^2)} = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{(x^2+y^2)}{2}\right) \tag{5.2}$$

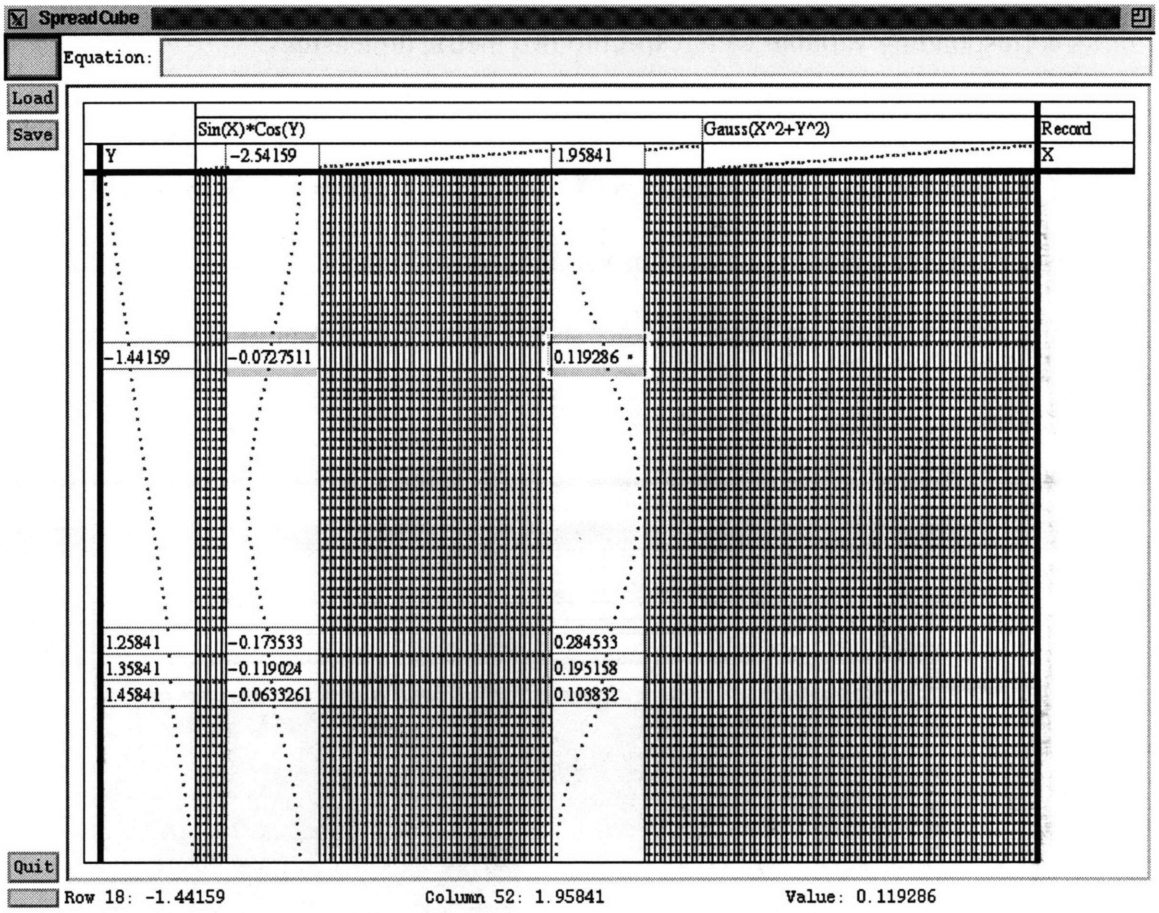
The values for the two variables have been generated for all combinations of keys of "X" and "Y" in the range $[-\pi, \pi]$ in steps of 0.1.



```
Row 2038: -1.04159        Column 2: Gauss(X^2+Y^2)        Value: 0.0950038
```

Figure 5.11: A Toy Data Set

After loading the data into SpreadCube it is first visualized in a relational layout (see figure 5.11, figure A.7 and figure A.8). The visualization on figure A.8 includes a focal area, whose limelight cells show the fine structure of the plot. Each of the two record columns consists of multiple 2D plots stringed in a sequence.The example on figure A.8 also demonstrates the existence of a focal area in graph mode.



	Sin(X)*Cos(Y)	
Y	-2.54159	1.95841
-1.44159	-0.0727511	0.119286
1.25841	-0.173533	0.284533
1.35841	-0.119024	0.195158
1.45841	-0.0633261	0.103832

```
Row 18: -1.44159          Column 52: 1.95841          Value: 0.119286
```

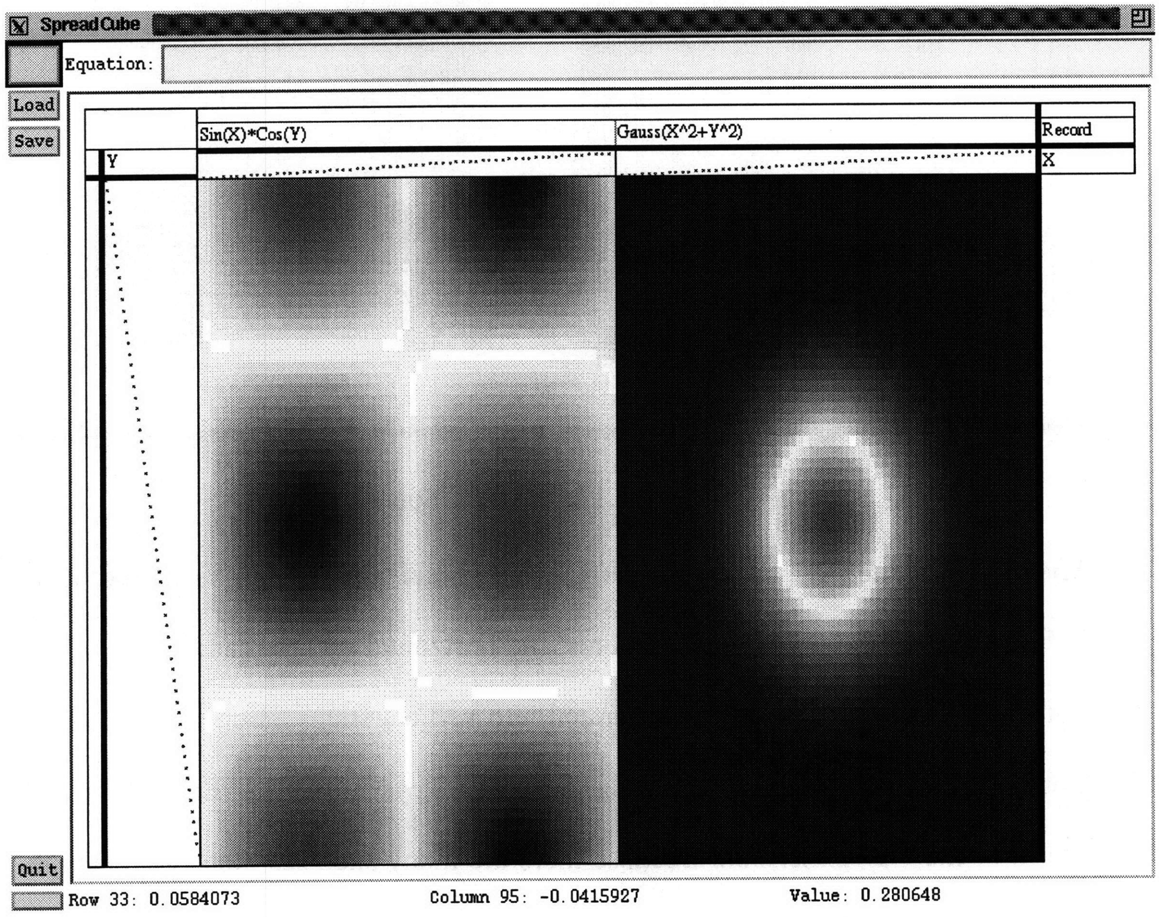
Figure 5.12: Toy 2D Plots

After moving the “X” dimension to the horizontal axis we arrive at the view on figure 5.12 and figure A.9. On these figures, the horizontal merge mark is set below the level of the leaves, so slabs have degenerated into vertical sticks. Each stick contains the 2D plot of its respective variable with respect to the keys of the “Y” dimension and for fixed keys of the “X” dimension. Because there are many such 2D plots, the example on figure 5.12 and figure A.9 focuses on two of them. This example is also a demonstration of the co-existence of multiple focal areas.

5.7.2 Three Dimensional Plots

If the merge marks on both axis are set at least a level higher than the level of the leaves, then the slabs are not degenerate. In that case each slab will contain a 3D plot of the values of its corresponding variable with respect to *two* metric dimensions.

The result of sliding the horizontal merge mark up and consequently producing a 3D plot of the two variables with respect to "X" and "Y" is shown on figure A.10. The third dimension is represented by saturation variation of red and blue colors. The red shades represent values above the average, whereas the blue shades represent values below the average.



Row 33: 0.0584073 Column 95: -0.0415927 Value: 0.280648

Figure 5.13: Toy 3D Plots

The focal areas which existed on figure 5.12 continue to exist on figure A.10 as well. They were left on purpose to show the correspondence between the two figures. In fact figure A.10 represents what will happen to figure 5.12 if the 2D plots were flipped 90^o out of the plane of the page. The same 3D plots without the focal area are shown on figure 5.13 and figure A.11.

5.8 Visualizing Values

Data values can be visualized individually or collectively. For a given value, the choice between individual or collective representation depends on the degree of interest assigned to that value and the context in which it appears. As a rule, both limelight and spotlight values are represented individually.

When we speak of the visualization of values, we mean the visualization of both dimension keys and vault values. Even though, we may choose to visualize a dimension key in a way different from the way we visualize a vault value, both dimension keys and vault values pose the same visualization problem. Therefore, as far as finding a solution to this problem is concerned, we will treat the two equally.

5.8.1 Representing Individual Values

A value can be represented by a number of presentation types at the same time. The presentation types used in SpreadCube to represent individual values include: **text**, **color**, **color saturation**, **bar length**, **bar positioning**, **point positioning** and **sound pitch**. Examples of using these presentation types are shown on figure A.3.

The following factors affect which presentation types are used and how they are used to represent individual values:

1. **Value**. The cell value is depicted using to the appropriate presentation type accord-

ing to table 5.1:

Presentation Type	Use
text	The value is unparsed and printed as text
color	The value is represented as a color from a set of distinct colors
color saturation	The value is represented as a color saturation of either red or blue colors. Saturated red corresponds to the highest values, whereas saturated blue corresponds to the lowest values of a variable. The average values are represented as white.
bar length	The value is represented as the length of a bar from the origin of the cell
bar positioning	The value is represented as the position of a bar from the origin of the cell.
point positioning	The value is represented as the position of a point from the origin of the cell
sound pitch	The value is played as a pure tone with pitch in the range of 100-900 kHz, scaled proportionally to the value

Table 5.1: Use of Presentation Types According to Value

2. **Value domain**. The value domain is the main factor that determines which presentation type is used. The correspondence between a value domain and a presentation type is summarized in table 5.2. Since we treat dimension keys as values for visualization purposes, we consider "variable descriptor" as a possible value domain.:

Value Domain	Presentation Type
quantity	bar length, or point positioning, or color saturation, and/or sound pitch
nominal	color and/or bar positioning
textual	text
variable descriptors	text (the label is represented)

Table 5.2: Presentation Types for Each Value Domain

3. **Focus level**. Cells and the values which they contain are depicted differently depending on their focus level. Spotlight cells are always unshaded and use the text presentation type along with any other appropriate presentation type. Limelight cells are shaded but not as dark as the background cells.

4. **Visualization mode**. In table mode, stick quantity values are depicted as bar lengths, whereas in graph mode they are depicted as point positions within a cell. The visualization mode also determines whether the cell boundaries are visible or not. Thus in graph mode all cell boundaries are invisible except for the spotlight cell boundaries.

5. **Cell size**. Apart from the focus level, the cell size depends also on the amount of available space and, in graph mode, on the values of the metric dimensions. According to user preferences, the cell value may be represented as text as long as there is enough space, or the text representation type may be suppressed unless the value is a spotlight value. The default is that only spotlight values are represented as text. The cell size, in addition to visualization mode, determines whether cell borders are visible. So if the width or the height of the cell is less than 2 pixels, then the borders are not shown.

6. **Modulating variables**. In the SpreadCube visualization we can use the values of one variable A to specify how the values of another variable B should be rendered. In particular, if the values a and b of the two variables share all set dimension keys then we use a to specify an additional representation type for b and how it should be used. In such cases we will say that A modulates B, because the effect is that after the modulation, the representation of B carries information about A along with the information about B.

For example if *A* is a nominal variable then modulating *B* adds the color presentation type to the presentation of *B*, so that the values of *B* are plotted with colors specified according to the values in *A*. If *A* is a quantity variable then its values will be used to determine the color saturation with which the values of *B* are plotted. If *A* is a text variable, then textual labels will be added to the representation of *B*'s values.

7. **Stick versus Slab**. The presentation type depends on whether the value is a part of a degenerate slab, i.e. a stick or a regular slab. In the former case, quantity values are represented as bar lengths, or point positions, and in the latter case, they are represented as saturation variations.

8. **Vault value versus Dimension Key**. A vault value and a dimension key of the same domain may be represented slightly differently. For example the boundaries of a dimension key of non-merged dimension are always plotted as long as the width and the height of the key is at least two pixels. In addition, the user may choose to highlight focused dimension keys, or leave them shaded.

9. **User Preferences**. The user can specify the set of colors that should be used for the color presentation type and whether sound pitch should be used in parallel with other presentation types.

5.8.2 Representing Sticks and Slabs

When the width or the height of a cell is less than one screen pixel, then the cell cannot be represented individually. Such is the case for background cells in very large datasets. The solution is to average values close together, until the averaged area is at least a pixel tall and a pixel wide, and then depict the value of the average using a set of presentation types, as if it were a regular value.

Each value domain has its own methods for averaging. Those are summarized in the following table:

Value Domain	Averaging Method
quantity	Weighted average. The weight is the cell size
nominal	The average value is the most frequent value
textual	The average value is the most frequent value

Table 5.3: Averaging Methods according to Value Domains

When the dataset is so large that the width or height of a stick (or slab) is less than one screen pixel, then the stick data is not represented in any way. In such cases, the user will have to change the merge marks in order to merge adjacent sticks together so that the resulting sticks are large enough to be displayed.

5.9 Visualization Operations

Operations on the dataset are invoked through operations on the visual model which was discussed in chapter 4. As far as such operations are concerned, the SpreadCube visualization is simply a context which fills them with input arguments, and presents the results to the human.

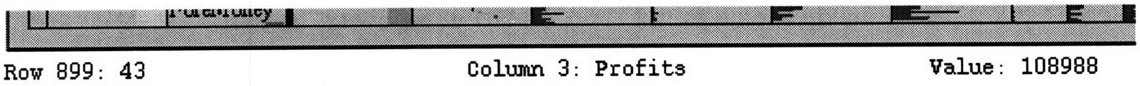
The SpreadCube visualization, however, defines its own set of methods which allow the user to control how objects are represented. Some of these methods simply let the user specify the color palette, the size of the spotlight and limelight cells, and etc. A few of these operations, however, are not purely cosmetic, but have a great impact on how visual objects are interpreted. One such example is switching between graph and table modes.

In the remaining part of this section we list these operations and their effects. The actual user actions which lead to invoking these operations are discussed in the next chapter on user interface. Instead, we will describe these operations here as procedures which take a set of arguments. The arguments are elements of the visual model.

5.9.1 Describe an Object

Arguments: **`visual_object`**

Description: Displays a short description of **`visual_object`** in the bottom row of the SpreadCube window. For example, the description of a vault cell consists of the cell value, the row and column number and the keys of the metric dimensions (see figure 5.14).



```
Row 899: 43                       Column 3: Profits                   Value: 108988
```

Figure 5.14: Description of a Vault Cell

Formulation:

$$\text{Describe}(obj) \tag{5.3}$$

5.9.2 Setting the Visualization Mode

Arguments: **`spread_cube`**

Description: These operation toggles between table mode and graph mode, which were discussed in section 5.7. The major effect of each mode is on the interpretation of the spatial extends of the two screen axes.

Formulation:

$$\text{ToggleMode}(C) \tag{5.4}$$

5.9.3 Variable Modulation

Arguments: **`variable`**

Description: Modulates all variables that are currently visualized according to **`variable`**. The effect is that an extra presentation type is added to the presentation types of all variables. In other words, for any variable X and any of its values x, the application of the presentation type added to x is determined by the value v of **`variable`** iff x and v have equal set dimension keys. The presentation type itself is determined by the domain of **`variable`**, according to the following table:

Value Domain of Modulating Variable	Added Presentation Type
quantity	color saturation
nominal	color
textual	text

Table 5.4: Added Presentation Type according to Modulating Variable

Formulation:

$$\text{ModulateVariable}(var) \tag{5.5}$$

An example of variable modulation is shown on figure A.12, where the "Profits" variable is used to modulate the "Units" and "Revenue" variables. A straightforward interpretation of the picture is that the red colors which appear in the rendering of the "Units" and "Revenue" variables mark those values which correspond to high profits.

5.9.4 Space Modulation

Arguments: **`row`** (or **`column`**)

Description: This operation is only applicable in graph mode and for quantity row or columns. The values of a row or column are used to modulate the spatial metric in the orientation parallel to the row or column. The spatial metric is warped in such a way that positions farther from the origin correspond proportionally to larger values **`row`** and vice-versa. The visual effect is that all rows (or columns) are graphed with respect to **`row`**. An example is shown on figure A.13 where the "Units" row is used for spatial modulation, while at the same time the "Profits" row is used for variable modulation. The effect is quite similar to promoting the "Units" variable and then switching into graph mode, but spatial modulation provides more flexibility. Furthermore, spatial modulation is more efficient than promoting a variable, because it does not introduce non-existent values.

Formulation:

$$\text{ModulateSpace}(var) \tag{5.6}$$

5.9.5 Range Selection

Arguments: **`range`**

Description: Produces a new nominal variable, called *selection* variable, which "marks" the **`range`** according to the *mark* operation as described in section 3.5.2 of the data model chapter. This operation affects not only the visualization but also the schema of the data set. The reason for including it here, is because it is coupled with some interesting visualizations. Namely, immediately after a selection variable is created or updated, it is automatically used for variable modulation. The effect is that selected ranges appear in a different color from the rest, and the action of making a selection "feels" like coloring a range.

An illustration of range selection is shown on figure A.14. The range which is being selected is indicated on figure 5.15. The selected range is colored in green. One selection variable can be used to mark several complimentary selections at the same time. In that case each new selection will be colored differently from the previous one.

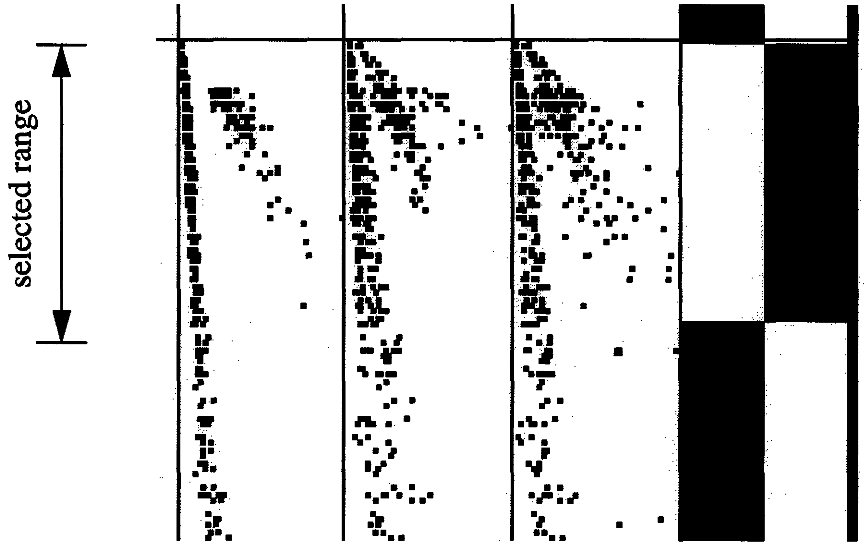


Figure 5.15: Range selected on figure A.14

5.9.6 Region Selection

Arguments: `region`

Description: This operation has the same effect on the dataset as the range selection operation, except that the input is not a unidimensional range, but a two-dimensional region. The nominal variable produced as a result of region selection is used in variable modulation to color the selected region. This is illustrated on figure A.15, and the selected region is shown on figure 5.16.

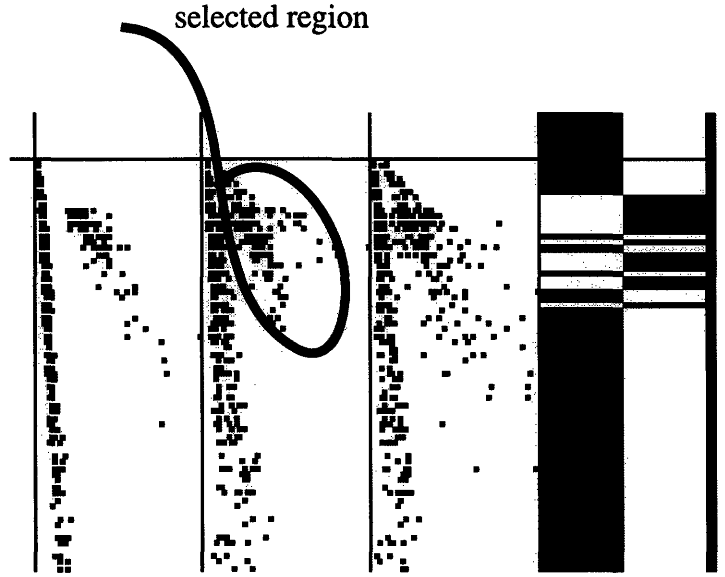


Figure 5.16: Region selected on figure A.15

Both region selection operations and range selection operations ultimately lead to the *mark* operation on the data set. The difference between the two is purely visual and is caused by the fact that in graph mode some rows are rendered on top of each other, which produces two-dimensional plots.

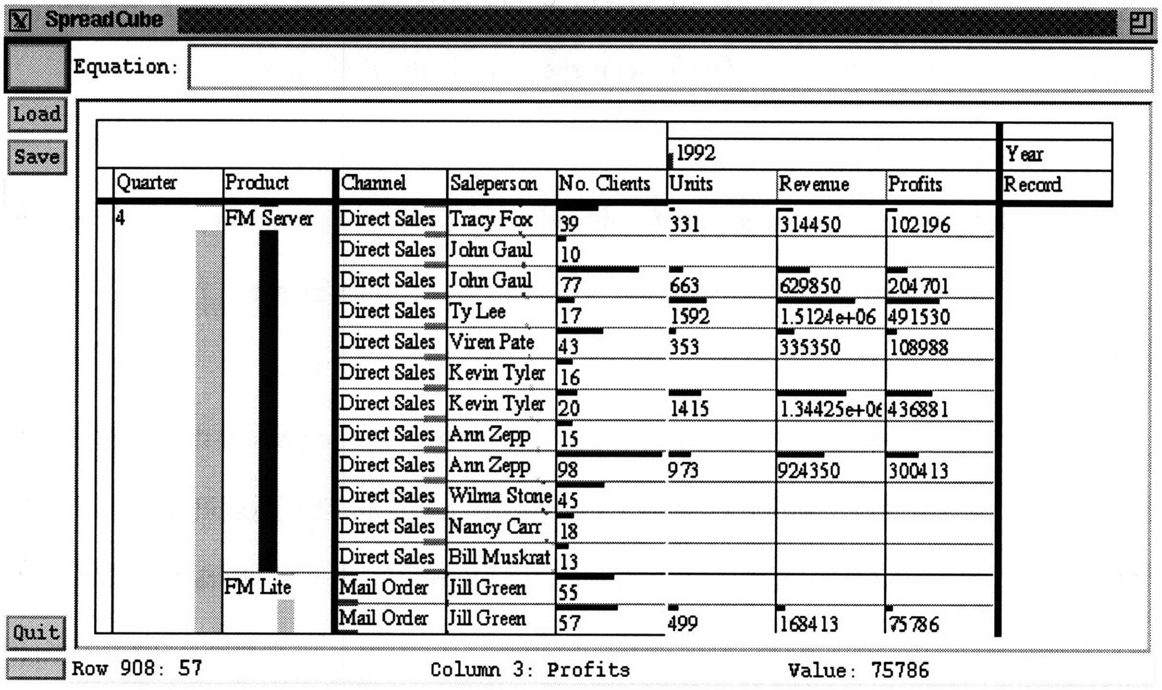
5.9.7 Hiding

Visual objects can be hidden to leave more room for the rest of the visual layout to be rendered. In SpreadCube one can hide an entire dimension group associated with a dimension key.

There is more to hiding of an object, than simply making room for the remaining visual objects. It is a way of tidying up both the visual and the mental picture of the dataset. It allows the human to concentrate on what is important by eliminating what is not.

5.9.8 Zooming-In

Zooming-In is an operation which hides all but the spotlight cells and the associated with them dimension keys. Thus, after applying this operation on the visualization on figure 5.1, the result is (figure 5.17 and figure A.16):



					1992			Year
Quarter	Product	Channel	Saleperson	No. Clients	Units	Revenue	Profits	Record
4	FM Server	Direct Sales	Tracy Fox	39	331	314450	102196	
		Direct Sales	John Gaul	10				
		Direct Sales	John Gaul	77	663	629850	204701	
		Direct Sales	Ty Lee	17	1592	1.5124e+06	491530	
		Direct Sales	Viren Pate	43	353	335350	108988	
		Direct Sales	Kevin Tyler	16				
		Direct Sales	Kevin Tyler	20	1415	1.34425e+06	436881	
		Direct Sales	Ann Zepp	15				
		Direct Sales	Ann Zepp	98	973	924350	300413	
		Direct Sales	Wilma Stone	45				
		Direct Sales	Nancy Carr	18				
		Direct Sales	Bill Muskrat	13				
	FM Lite	Mail Order	Jill Green	55				
		Mail Order	Jill Green	57	499	168413	75786	

Row 908: 57 Column 3: Profits Value: 75786

Figure 5.17: After Applying the Zoom-In Operation to the set on figure 5.1

The zooming-in operation converts the SpreadCube's focus+context visual environment into a spreadsheet-looking visualization. In the original focus+context environment, the user controls the focal area by sliding it over the visual layout, like a lens. After applying the zooming-in operation, the controls of the focal area are still the same but the visual effect is different, because now the focal area stays still while the visual layout slides underneath. In other words, it functions like a scrollable window!

5.10 Discussion

The design of the SpreadCube visualization integrates a number of techniques for presenting information. Each of these techniques by itself, like laying out information in a tabular format or plotting data points, is not new. However their integration has given rise to an unconventional presentation environment, and has been the source for a many design decisions.

This section discusses some of the thoughts aroused from reflecting on what the SpreadCube visualization *could* have been and what it *should* have been like.

5.10.1 New but Familiar

Innovations come at a cost. The cost for introducing a new visualization and user interface is the amount of time and concentration which the user must invest in order to best take advantage of the idea.

The SpreadCube visualization uses two main avenues to approach this problem. The first one is to base itself on views which are already familiar. The second one is to always have a natural way of transforming itself into something that is already familiar.

For instance the SpreadCube visualization presents dimensional hierarchies in a tabular layout which has been in use long before computers had screens. In addition it uses a more or less traditional graphical language for encoding data values, e.g. quantities are represented by length.

A potential source of confusion is the coexistence of multiple presentation types in a multi-level focus environment, where the user controls the focus levels through a focal area. In such cases the zooming-in operation is especially helpful because it puts the novice in a spread-sheet like environment, in which the focal area is conveniently transformed into a scrollable window.

Alternatively, one could choose to get rid of all limelight and spotlight regions and set the visual mode to graph mode. The result will be a collection of plots presented in a near-traditional way.

5.10.2 Modulating a Variable or Spatial Extends

The idea of variable modulation can be extended to using more than one modulating variable. For example, the presentation type for an extra modulating variable of the quantity domain can be an error bar, leading to a error bar plot in graph mode. The presentation type for two extra modulating variables of the quantity domain can be a vector whose horizontal and vertical projections represent a value from each variable. The resulting visualization will be a vector plot (or field plot).

Similarly the idea of spatial modulation can be extended to different ways of warping the space according to the values of a quantity column. For example, rather than warping space such that quantities increase proportional to the distance from the origin, it can be warped such that the quantities of the modulating column increase exponentially or logarithmically with their distance from the origin. The resulting visual effects will be that all non-modulating columns will be plotted with respect to the modulating column according to a logarithmic or exponential scales, respectively.

5.10.3 Using Audio

The use of audio as an addition all other presentation types, is quite helpful making the user aware of data variations or pinpointing a value.

This is especially true in cases when the depicted values are packed too close together on the screen, as is the case with background values. In such cases, one can quickly scan with the mouse pointer a column or a row of data while looking for irregularities and interesting patterns using both visual and auditory perception. The two channels: auditory and visual back each other up, thus reducing the chances for missing a detail which is worth focusing on.

The audio representation is also very helpful when color saturation is used to represent quantity values as is the case on figure A.11, where it is some times difficult to make visual distinction between the various saturation levels.

5.10.4 Serendipity

The use of multiple presentation types, like text and graphics, and the integration of multiple presentation techniques, like tabular layout of data or plots, shows that the SpreadCube visualization can be at least as powerful as most visualization tools.

The essential power of the tool, however, lies in its ability to present the user with revealing views which were not anticipated. In most conventional tools which deal with information visualization, like Matlab, Excel, SPlus and etc., the constructed visualizations are solely the results of user's premeditated actions. In other words, the user knows what he or she wants to observe, and simply wishes to see what it looks like.

In SpreadCube, however, each action causes an entire layout of plots or tabulated data to be rearranged in a meaningful way. For example, spatial modulation produces multiple graphs at the same time. Consequently, even though the user's actions may have been driven by some original intend, it is very likely that some of these actions will give rise to a view which was never intended.

Chapter 6

User Interface

The user interface (UI) translates user actions into operations on the components of the visual model. These operations on the visual model correspond to operations on the abstract data model. The visual layout, dictated by the visual model and its visualization, is used as a context which supplies the arguments to the invoked visual model operations.

6.1 Goals and Philosophy

The primary goal of an user interface is to be invisible, while at the same time allowing the human to utilize the full power of the application.

For an user interface to be invisible it must keep the attention of the human focused on the work being done, and not on looking for control gadgets. In general the SpreadCube's user interface tries to avoid most conventional UI gadgets like pull-down menus, pop-up windows and scroll bars completely. These gadgets have the disadvantages of taking up valuable space, obscuring details of the work window and often taking up several mouse clicks in order to launch an action. Most importantly, however, these gadgets act as a distraction to the user, diverting his or her attention from the work window.

Instead we have attempted to build SpreadCube's user interface according to the philosophy of context sensitiveness and direct manipulation. Context sensitiveness means that invoked actions depend on the place in the visualization where they have been invoked and on the current visualization state. For example a mouse click inside a focal area is interpreted differently from a mouse click on a background cell. Direct manipulation means, that visual objects are their own controls.

6.1.1 User Interface Grammar

In order to be invisible, an user interface must be intuitive. This means that the user can *guess* how to invoke an action based on his or her experience so far with the application and with the physical world.

One way to make an user interface intuitive is to structure its actions according to a predictable grammar (references). For example, almost without exception traditional user interfaces group "Save", "Open" and "Quit" actions as "file actions", and consequently they are accessible through a "File" pull-down menu. In this sense, pull down menus are useful because they directly correspond to the grammar of the user interface.

6.1.2 UI Story and UI Language

As far as SpreadCube's user interface is concerned, we prefer not to use pull-down menus, for invoking operations on the visual model So, instead of using an explicit grammar specified by pull-down menus, we have chosen to make the user interface more intuitive by building a story around it, in which the characters are visual objects.

A story explores both the user's imagination and and the user's concrete visual and motor memory. For example, moving a dimension label tile from one axis to another rewards the user with an interesting visual performance as it causes the dimension hierarchies on the two axis to be rearranged. The effects are simultaneously interpreted on several level: on the perception level, on the visual model level and on the abstract model level, where the operation of dataset transposition is conceptualized. This redundant multiple level association of an user interface action with its effects helps commit this action to the implicit memory. That is why, next time when such action is needed, the user will be able to automatically retrieve it from his or her memory without intentional recall.

The concept of UI language is useful for describing denotationally what the UI story is. Just like a normal story is told in a normal human language, the UI story can also be represented symbolically if the "words" are well chosen. Later sections of this chapter describe what these words are and how they can be put together to form a sentence.

6.2 Techniques for Context Sensitiveness and Direct Manipulation

6.2.1 Use of Gestures

In the context of user interfaces, a gesture is a scribble, or a doodle which invokes an action. A gesture is made with a mouse or a stylus depending on the pointing device[1] which is being used. When the gesture is made with a mouse it is an action similar to drawing with the mouse, i.e. moving the mouse pointer while keeping the first mouse button depressed.

Usually, the gestures used in an user interface are simple stylized strokes, which are mnemonic of the actions that they invoke. For example, in SpreadCube, a flick gesture over a variable column or row indicates that the column (or the row) should be sorted. The direction of the flick indicates the direction of the sort.

The meaning of each gesture is different depending on the context in which the gesture is made. For example a flick gesture perpendicular to a dimension is mnemonic of slicing that dimension and hence invokes that operation. The position of the gesture with respect to the dimension, indicates the particular dimension key at which the slice is performed.

Using gestures integrates consistently with the idea of a user interface story. When wielding a gesture the mouse pointer becomes an implement for affecting visual objects.

In many ways gestures are superior to conventional context sensitive techniques, because they allow a pointing device, like a mouse, to be used as a means of invoking actions on objects, as opposed to simply selecting those objects. In addition a gesture can supply some of the arguments to the action that it invokes, as is the case with sorting where the direction of the gesture indicates the direction of the sort.

---

[1] The SpreadCube user interface assumes the use of a pointing device. Depending on the hardware involved, the pointing device can be a mouse or a stylus. For the sake of concreteness, however, in the rest of this chapter, we will assume that the pointing device is a mouse.

6.2.2 Drag and Drop

Drag and drop is a technique where a visual object is picked up and dragged, using the pointing device, and subsequently dropped onto another object. The invoked action depends on the context of the two objects involved and the objects themselves.

For example, in SpreadCube dropping a dimension tile onto another dimension results into a transposition in which the two dimensions are permuted.

6.2.3 Slide and Resize

In SpreadCube visual objects can be resized by grabbing and pulling their edges. In the case of the focal area, which is treated as a visual object, the resizing action changes the number of spotlighted or limelighted cells.

In addition the focal area can be slid to any position on the visual vault, which causes the cells at the new position to be "magnified", i.e. spotlighted or limelighted appropriately, and the cells at the original position of the focal area to be reverted to background cells.

6.2.4 Mutating the Pointer

A mouse action has different effects in different areas of the screen depending on the visual context. Some of these differences are prompted by mutating the mouse pointer appropriately as it passes through special *control points* in the visualization.

For example, the default mouse pointer is a hand with an extended forefinger: ☞ indicating that the mouse is used for gesturing. As the pointer slides over a merge mark, it is mutated into a double arrow: ↔ indicating that the merge mark can be slid left-right, for a vertical merge mark, and up-down for a horizontal merge mark. If the middle mouse button is depressed, the mouse pointer becomes a tool for grabbing; this is prompted by changing the pointer to a grabbing hand: ✊.

6.2.5 Pop-up Menus

A pop-up menu is invoked by depressing the right mouse button. The menu which pops up depends on the visual context where the pointer is. For example, if the right mouse button is depressed while the pointer is inside a variable column, the pop-up menu lists actions which apply to variables or use variables as arguments. An example is shown on figure A.17 in the appendix. The upper-left corner of the menu corresponds to the position where the menu was invoked.

6.3 The SpreadCube User Interface

In this and the following sections we introduce the SpreadCube user interface in a manual-like format. The objective is to provide the reader with the available user interface controls controls. In the next several sections we will propose a way of formalizing user actions, and we will use some the actions described in this section as an example.

6.3.1 Data Input

```
1993    May     FM Access   Fred     76       72200     28158
1990    Jan     FM Access   Barry    11       10450     4076
1990    May     FM Access   Fred     29       27550     10745
1992    Jan     FM Access   Fred     2        1900      742
1991    Jan     FM Access   Barry    20       19000     7410
1991    May     FM Access   Fred     52       49400     19266
1990    May     FC Pro      Fred     226      96616     37680
1990    Jan     FM Server   Barry    45       427500    13893
1990    May     FM Server   Fred     114      108300    35197
1991    Jan     FC Pro      Barry    195      83363     32512
1991    May     FC Pro      Fred     487      208193    81196
1991    Jan     FM Server   Barry    108      102600    333450
1991    May     FM Server   Fred     270      256500    833626
1992    Jan     FC Pro      Fred     26       11116     4335
1992    May     FM Server   Fred     13       123500    40138
1993    May     FC Pro      Fred     892      381330    148719
1993    Jan     FM Server   Fred     429      407550    132453
1990    Jan     FC Pro      Barry    90       38476     15006
```

Figure 6.1: An Example Data File (`example.data`)

The input data consists of two parts corresponding to a vault and a schema. In SpreadCube the two parts are two files: a `.data` file and a `.schema` file. The former contains the actual data and the latter is a description of what the data file is (see figure 6.1 and figure 6.2). The example data set defined by figure 6.1 and figure 6.2 is the dataset whose visual layout is shown on figure 4.5.

```
file: "example.data";

columns:
        dimension: "Year" quantity;
        dimension: "Month" nominal;
        dimension: "Product" nominal;
        variable: "Saleperson" nominal;
        variable: "Units" quantity;
        variable: "Revenue" quantity;
        variable: "Profits" quantity;

record:
        "Units";
        "Revenue";
        "Profits";
        "Saleperson";

xaxis:
        record;

yaxis:
        "Product";
        "Year";
        "Month";
```

Figure 6.2: An Example Schema File (`example.schema`)

The `.schema` file uses a simple declarative syntax. For example, the `columns:` declaration on figure 6.2 defines what the columns of the data file are, what their labels and domains are, and also whether they contain vault data (`variable:`) or they contain dimension key values (`dimension:`). The `record:`, `xaxis:` and `yaxis:` declarations define which dimensions are visualized on which axis and what variables constitute the record.

The input-output operations as well as some other global operations launched from a pop-up menu, which is activated by pressing the right mouse button while the mouse pointer is outside the visual vault and the axes (see figure A.18).

6.3.2 Manipulating the Visual Layout

Visual objects are manipulated using drag-and-drop or slide manipulations.

During a drag-and-drop a tile with the label of the grabbed visual object is dragged from one place to another. For example, to permute two dimensions in the dimension hierarchy, one drags one dimension tile on top of another (see figure A.19). The same manipulation is used for promoting a variable to an axis, or reducing a variable to the record dimension, in which case the tile with the variable name is being dragged to the destination axis, or to the destination place in the record dimension. A drag-and-drop manipulation is also used to reorder the keys of a dimension. The manipulation is executed by first pressing the middle mouse button over an object to grab it, and then, while keeping the middle mouse button pressed, drag the object to its destination and release the button to drop the object. During a drag-and-drop manipulation the mouse pointer shape changes into a grabbing hand: ✋ (see figure A.19).

The slide manipulation is applied to merge marks. An example is shown on figure A.20. As the mouse pointer is moved across a merge mark, it changes into a double arrow: ↔ to indicate that sliding is possible. To initiate sliding, one must depress the middle mouse button, which grabs the object and then slide it to its destination where the middle mouse button is released.

6.3.3 Data Manipulations

Sorting

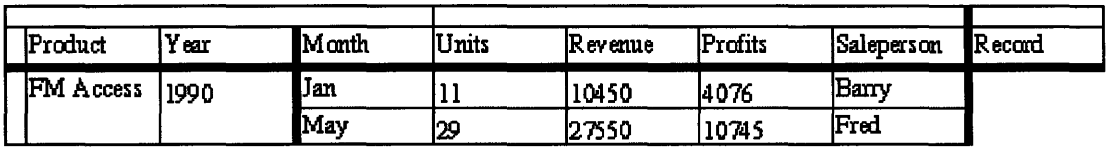
Sorting is done by making a flick gesture over the column or the row which is to be sorted. The direction of the gesture indicates the direction of the sort. When sorting a raw, a flick gesture west indicates up sort, and a flick gesture east indicates down.

The same manipulation is used to sort dimension keys of dimensions which have been merged.

Slicing

Slicing is done by a flick gesture in outward direction over the dimension key at which one wishes to slice. The orientation of the gesture must be perpendicular to the axis containing the dimension. For example, to slice over a dimension key of the vertical axis, the slicing gesture must be horizontal and westward.

If the slicing operation is not over the dimension at the top of hierarchy, then it is equivalent to iteratively slicing on at all levels above the current level. For example, slicing over the "Year" dimension of the data set show on figure 4.5 leads to the layout shown on figure 6.3.



Product	Year	Month	Units	Revenue	Profits	Saleperson	Record
FM Access	1990	Jan	11	10450	4076	Barry	
		May	29	27550	10745	Fred	

Figure 6.3: Slicing the Example Data Set through the "Year" Dimension.

If the direction of the gesture is inward, then the effect of slicing is reversed for the current level in the dimension hierarchy.

Range and Region Selection

A region of cells is selected by drawing a loop around it while depressing the middle mouse button (grabbing). Immediately after releasing the middle mouse button, a new selection variable is created as explained in section 5.9.6 and demonstrated on figure A.15.

A range of cells is selected by drawing a straight segment while depressing the middle mouse button. The ends of the segment mark the ends of the region. After releasing the middle mouse button a selection variable is created as explained in section 5.9.5 and demonstrated on figure A.14.

Aggregation

Aggregation is done on per-group basis. When a group is aggregated all sticks under this group are replaced with their averages according to some averaging function. The averaging function depends on the value domains of the data. For quantity values it is the arithmetic mean, for nominals and text it is the most dominant value.

And example is shown on figure A.21. The aggregation operation is launched from a pop-up menu over the group which is being aggregated. Since aggregation averages the values within a stick, the coarseness of the aggregation can be controlled by sliding the merge marks, which change the stick sizes.

The operation opposite to aggregation is "drill", which is also part of the group menu shown on figure A.21.

6.3.4 Formula Processor

The formula processor enables the user to create new variables as a function of already existing variables. This is done by typing an equation inside the "Equation" box (see figure 6.4).

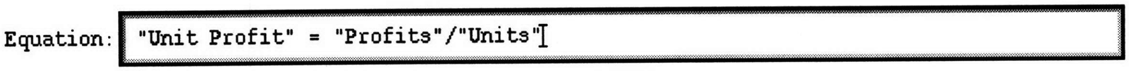

Equation: "Unit Profit" = "Profits"/"Units"

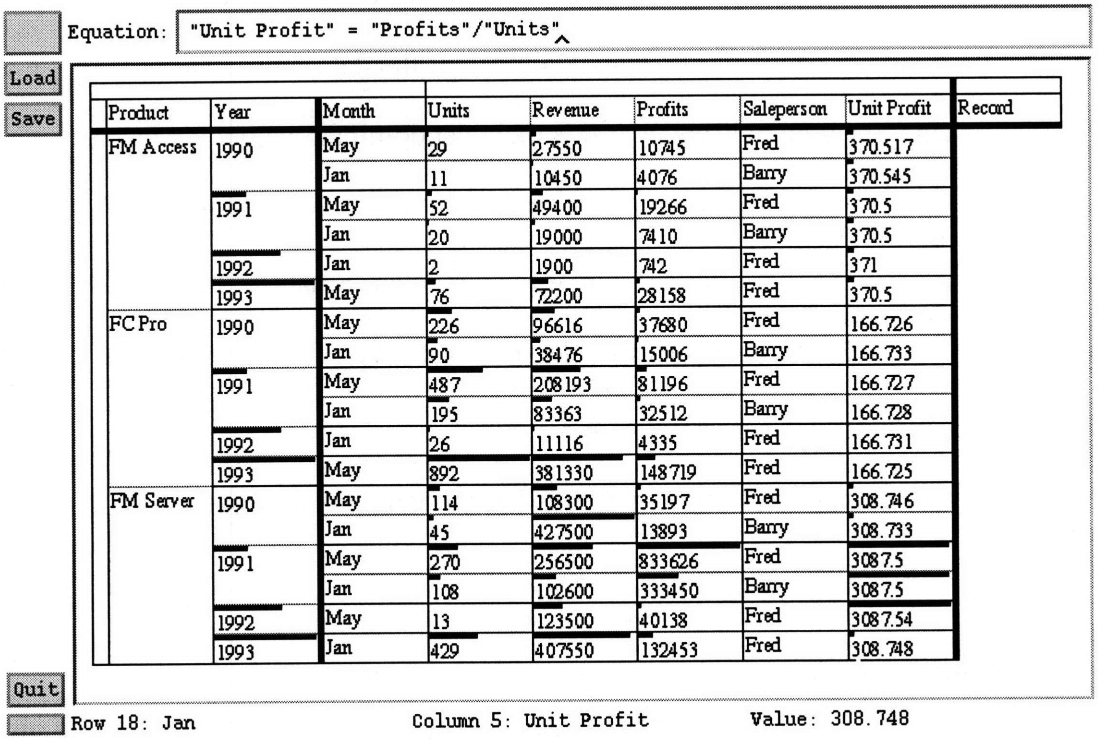
Figure 6.4: "Equation" Box.

The left hand side of the equation is the name of the new output variable. The right hand side is an arithmetic expression which may contain other variable names, constants, and functions, as in the example on the figure.

For example, after typing the expression:

```
"Unit Profit" = "Profits"/"Units"
```

The example data set on figure 4.5 becomes: (figure 6.5)



Equation: "Unit Profit" = "Profits"/"Units"

Product	Year	Month	Units	Revenue	Profits	Saleperson	Unit Profit	Record
FM Access	1990	May	29	27550	10745	Fred	370.517	
		Jan	11	10450	4076	Barry	370.545	
	1991	May	52	49400	19266	Fred	370.5	
		Jan	20	19000	7410	Barry	370.5	
	1992	Jan	2	1900	742	Fred	371	
	1993	May	76	72200	28158	Fred	370.5	
FC Pro	1990	May	226	96616	37680	Fred	166.726	
		Jan	90	38476	15006	Barry	166.733	
	1991	May	487	208193	81196	Fred	166.727	
		Jan	195	83363	32512	Barry	166.728	
	1992	Jan	26	11116	4335	Fred	166.731	
	1993	May	892	381330	148719	Fred	166.725	
FM Server	1990	May	114	108300	35197	Fred	308.746	
		Jan	45	427500	13893	Barry	308.733	
	1991	May	270	256500	833626	Fred	3087.5	
		Jan	108	102600	333450	Barry	3087.5	
	1992	May	13	123500	40138	Fred	3087.54	
	1993	Jan	429	407550	132453	Fred	308.748	

Row 18: Jan Column 5: Unit Profit Value: 308.748

Figure 6.5: Computing a New Variable

The formula processor applies the *extend* operation of the data model to add the new variable to the data set.

6.4 Focal Area

This section is a continuation of the previous one, in which we present the SpreadCube UI controls in a user manual-like fashion. Since, however, the focal area is one of the novelties presented in this paper, we have separated the description of its controls in a section by itself.

6.4.1 Creation

A Focal Area is created by making a loop gesture indicating the region of vault cells which should be spotlighted. If the region specified for spotligthing is too big, then the focal area automatically selects a smaller, co-centric region which is possible to be spotlighted given the constraints of the screen.

If there was a previously existing focal area, it is destroyed after the selection of a new one, unless the user specifies otherwise, by depressing a modifier key, e.g. the shift key, while making the loop gesture.

With three levels of focus, there are a quite a few degrees of freedom for the control of the focal area: the size of the spotlight region, the size of the limelight region and the centers of the two regions. To simplify the specification of a new focal area, the SpreadCube user interface only requires that the spotlighted region be specified. Initially, the horizontal and vertical extends of the halo of limelight cells around the spotlight region is automatically calculated, and once displayed the user can resize it. When the limelight region is created it focus coincides with the focus of the spotlight region, however, by moving the edges of each of the two regions the user can change that subsequently.

6.4.2 Resizing

Both the limelight and the spotlight regions can be resized by grabbing their edges and dragging them. As the mouse pointer moves over the edge of a region its shape changes into a resizing arrow: ⇤, which can be either left, right, up or down arrow depending on the edge. The new pointer shape indicates that if one were to depress the middle mouse button (grab), then one could pull the edge under the mouse pointer, and resize the focal area.

Resizing the focal area is an indication of a new *desired* size. However, the *actual* size, may be smaller if the desired size does not conform to the constraints of the screen.

6.4.3 Motion

The focal area can be moved from one place to another by simply clicking on the new destination with the first mouse button. The motion of the focal area from its original position to its destination is animated, so that the user can trace how the distortion of the visual grid changes as a result of repositioning the focal area.

Moving the focal area in the manner described above is suggestive of its lens-like nature. A different way to control the motion of the focal are, which is more suggestive of its scrollable window-like nature, is the following:

If the mouse pointer is positioned about a quarter distance from an edge of the spotlighted region, its shape changes into an arrow. For example, if the mouse pointer is near the top edge of the spotlight area, the pointer shape becomes an up arrow: ↑ (see figure A.22). This is an indication that if the user clicks the first mouse button, the focal area is going to move a single row up, and similarly for the other three directions.

If the mouse pointer is positioned near an edge inside the limelight area, then its shape is also changed into an arrow. This time however, a click results in scrolling the focal area in the direction of the arrow, as opposed to only moving it by single row or column.

The arrow shapes of the mouse pointer are reminiscent of the arrows in a scroll bar, of a scrollable window. One significant difference, however, is that scroll bar arrows occupy screen space regardless of whether the user needs them or not, whereas mouse pointer arrows appear only if they could be useful and do not require extra screen space.

The effect of the focal area functioning as a scrollable window is amplified after zooming-in, as described in section 5.9.8. In such case the apparent position of the focal area on the screen remains fixed, while the visual grid slides underneath.

6.4.4 Deciding the Limelight Extend

The horizontal and vertical limelight extends are automatically calculated according to the following rules:

1. There is a minimum limelight extend of 10 pixels, because the motion controls of the focal area (see previous subsection) assume that limelight region exists.

2. If the width and height of the background cells is at least a pixel, then the limelight extend is set to be the minimum of 10 pixels. Otherwise, it is decided as a result of maximizing its size, while at the same time minimizing the cost of shrinking the background cells.

6.5 Formalizing User Actions

In the following sections we will present a formal way of describing operations on the visual model in terms of user actions. This will complete the formalization of the cause and effect sequence which leads from user actions to manipulations on the abstract data model.

To accomplish this, we will provide a vocabulary and a syntax for describing what an user interface does. Then, we will present some examples of formalizing user actions which lead to certain operations on the visual model.

6.6 The UI Vocabulary

The UI vocabulary has both nouns and verbs. The nouns are called *UI objects* and the verbs are called *UI Actions*.

6.6.1 UI Objects

Each UI object is denoted by a name and a collection of attributes as follows:

$$\text{UI\_Object}\begin{Bmatrix} Attribute_1 \\ \vdots \\ Attribute_n \end{Bmatrix}$$

The set of all UI objects is a superset of all visual objects, which are treated as having no attributes. In addition, we define the following UI objects:

Environment

Attributes: Other UI Objects ordered from specific to general.

Description: Represents the context for an action. The context is based on the in the screen position where the action has been invoked. The screen position translates to a position on the visual grid (see section 4.3.2). The attributes of the environment are the objects on the visual grid to which this position is internal. For example:

$$\text{Env}\begin{Bmatrix} g[i_1, \ldots, i_j] \\ D_j \\ A_1 \\ S \\ C \end{Bmatrix}$$

is the environment of a group on the dimension D_j on the axis A_1 which is part of the schema S which is part of the SpreadCube C. Typically, however, we will only list those attributes which are relevant for the current expression.

Menu

Attributes: Visual model operations.

Description: A collection of pointers to visual model operations which the user can choose to launch. The arguments to those operations are provided by the current environment.

Formulation:

$$\text{Menu}\left\{\begin{matrix} Operation_1 \\ \vdots \\ Operation_n \end{matrix}\right\}$$

Region

Attributes: A set of cells

Description: A collection of cells from the visual vault.

Formulation:

$$\text{Region}\{\, Cell^* \,\}$$

Gesture

Attributes: A **`direction`** and a **`region`**.

Description: A description of a unistroke gesture. If the gesture is a flick gesture, then **`direction`** indicates the direction of the flick and **`region`** is the collection of cells whose vertical or horizontal coordinates, depending on whether the gesture's orientation is vertical or horizontal, are within the two ends of the gesture. If the gesture is not a flick, then **`direction`** is meaningless and **`region`** is the collection of vault cells whose coordinates are *inside* the area surrounded by the gesture.

In SpreadCube there are really only two major kinds of gestures: a flick gesture, which can be a flick north, south, east or west, and a circle gesture. Therefore, we have chosen to formalize the description of a gesture in this particular way. For a richer gesture set, we would have needed a more complex description a gesture's attributes, in order to capture the possible information that it may contain.

Formulation:

$$\text{Gesture}\left\{\begin{matrix}\text{Direction:} & dir \\ \text{Region:} & region\end{matrix}\right\}$$

6.6.2 UI Actions

In this subsection we describe the set of user actions which SpreadCube uses, and provide a denotation for each of them. Each action is denoted in the following form:

$$E \vdash \text{Action}(Arg)$$

Where E is the environment in which the action is executed and Arg is an additional information supplied by the user at the time when the action was invoked. For example Arg can be a modifier key, which was pressed, or a menu selection. The following is a list of the primitive UI Actions that SpreadCube uses:

Point!

Arguments: <none>

Invoked by: Positioning the mouse pointer over a visual object

Gesture!

Arguments: a **`modifier`** or <none>

Invoked by: Making a gesture, with while keeping the first mouse button depressed. The **`modifier`** argument is a single bit of information which indicates whether a modifier key was pressed while the gesture is made. In SpreadCube the Shift key is used as a modifier.

Grab!

Arguments: a `region` or <none>

Invoked by: Pressing the middle mouse button over a visual object, or selecting a `region` by holding down the middle mouse button and making a loop gesture.

Drop!

Arguments: <none>

Invoked by: Releasing the middle mouse button.

Popup!

Arguments: <none>

Invoked by: Pressing the right mouse button. This action causes a pop-up menu to appear.

Launch!

Arguments: `menu_item`

Invoked by: User selecting an item from a pop-up menu. This action only makes sense in the context of a menu invoked by the *Popup!* action.

All complex user manipulations can be described in terms of the above simple six fundamental actions and five types of UI objects. The simplicity of the fundamental actions is a prerequisite for an intuitive user interface. Indeed, when there are only a few such actions the user quickly grasps what they are and how to combine them. The following chapter describes how one can formally relate these actions to operations on the visual layout and operations on the visualization.

6.7 From User Actions to Visual Operations

The number of ways in which primitive UI actions can be combined is too large to list all possible combinations here in a manner similar to listing out all operations on the abstract data model or on the visual data model. Instead we will only show a few examples to demonstrate how each action is involved into forming a phrase of the UI language.

6.7.1 Notation

Operations on the visual model take place when a combination of appropriate environment and user action is present, e.g. the mouse pointer being inside a cell and user clicking the left mouse button. Similarly UI Objects can be "conjured" as a result of a user action and appropriate environment. This is reflected in the notation that we present here. Thus: $\alpha \vdash \beta$ denotes that β takes place if the conditions expressed by α are present, if β is an UI action or an operation on the visual model. Alternatively if β is an UI object, then the same notation means that β is conjured if the conditions expressed by α are present.

6.7.2 Examples

The UI language which is proposed here and described by the examples below, consists of a sequence of statements like: $\alpha \vdash \beta$. The right-most expression in this sequence is the operation on the visual model a visualization operation which is being invoked. All other expressions are UI objects and UI actions.

Describe an Object

$$\mathrm{Env}\{obj\} \vdash \mathrm{Point!} \vdash \mathrm{Describe}(obj) \tag{6.1}$$

The above should be read in the following way: The visualization operation "Describe(*obj*)" is invoked if a "Point!" action is present in the environment where *obj* is the most concrete object.

Variable Modulation

$$\text{Env}\left\{\begin{matrix} cell \\ Q \\ B \\ var \end{matrix}\right\} \vdash \text{Popup!} \vdash \text{Menu}\left\{\begin{matrix}\text{Modulat\_Variable}\\ \text{Modulate\_Space}\\ \text{Cancel\_Modulation}\\ \text{Remove}\end{matrix}\right\} \vdash \text{Launch!(Modulate\_Variable)} \tag{6.2}$$

$$\vdash \text{ModulateVariable}(var)$$

where Q and B denote a stick and a slab respectively.

The above should be interpreted in the following way: The "ModulateVariable(*var*)" operation is invoked if the "Launch!(Modulate_Variable)" operation is present in the environment where the menu "Menu{...}" has been created as a result of the "Popup!" action in the environment of "Env{...}".

Transposition

$$\text{Env}\left\{\begin{matrix} \text{Env}\left\{\begin{matrix} label_m \\ D_m \\ \vdots \\ C\end{matrix}\right\} \vdash \text{Grab!} \vdash label_m \\ label_n \\ D_n \\ \vdots \\ C \end{matrix}\right\} \vdash \text{Drop!} \vdash \text{Transpose}(D_m, D_n) : C \tag{6.3}$$

Where C denotes a data set and D_m and D_n are the two dimensions which are exchanged in the transposition operation.

Slicing a Dimension

Below is an example of slicing a dimension on the horizontal axis, i.e. the $A_1^{N_1}$ axis.

$$\text{Env}\left\{\begin{matrix} g[i_1, \ldots, i_j] \\ k_j[i_j] \\ D_j \\ A_1^{N_1} \\ C \end{matrix}\right\} \vdash \text{Gesture!} \vdash \text{Gesture}\left\{\begin{matrix} \text{Direction: } \textbf{north} \\ \text{Region: <none>} \end{matrix}\right\} \vdash \text{Slice}(D_j, k_j[i_j]) \qquad (6.4)$$

and similarly for the vertical axis $A_2^{N_2}$.

Sorting

Below is an example of sorting up a column:

$$\text{Env}\left\{\begin{matrix} Q_N[d_1, \ldots, d_{N_1}, i_{N_1}, \ldots, i_{N-1}] \\ L[d_1, \ldots, d_{N_1}] \\ \vdots \\ C \end{matrix}\right\} \vdash \text{Gesture!} \vdash \text{Gesture}\left\{\begin{matrix} \text{Direction: } \textbf{north} \\ \text{Region: <none>} \end{matrix}\right\} \qquad (6.5)$$

6.8 Discussion

6.8.1 Use of Gestures

Typically, gestures are associated with pen-based user interfaces, e.g. an user interfaces for a personal digital assistant, where the only communication between the user and the computer is through a stylus. More traditional user interfaces prefer to use the mouse pointer to point at things rather than gesture. The SpreadCube user interface relies on gestures as a means for context sensitive user input. We already described several reasons which make gestures very suited for such tasks. Why are not gestures more widely used in modern user interfaces then?

One drawback of using gestures is that it is not immediately apparent to the user what is the set off available gestures and what are their effects. A way to solve this problem is to keep small the number of possible gestures per given context. Then while in a particular context, have the application display the possible gestures and their corresponding actions in a prompter window. The prompter window can be turned off when the user becomes familiar with the existing gesture set.

Another drawback of using gestures is that one must provide a recognition engine. If the recognition rates not at least 99.99% then the user will experience a lot of frustration while trying to make a gesture but invoking the wrong action. A solution is to design the gesture set in such a way that two different gestures are not easily confused and that a particular gesture does not allow for many variations in how it can be drawn.

6.8.2 What Makes a User Interface Successful

The success of a user interface depends on both objective and subjective factors, with the subjective factors being the majority. There has been some research on what makes an user interface successful (references), but the ultimate test is the market: if it sells then its good. On the other hand, while a theoretically good design does not guarantee the market success of an user interfaces, it is virtually always the case that a popular user interface has a theoretical ground for its success.

Chapter 7

Implementation

The goal of this chapter is to describe some of the highlights of the SpreadCube implementation, without going into specific details. The current implementation of SpreadCube is simply a vehicle for illustration of the ideas presented in the previous chapters, and therefore it is only peripheral to the subject matter of this thesis.

7.1 Overall Structure

The components of the implementation were already suggested by the chapters of this paper. Indeed the major two components are the *data model* and the *visual model.* There is also a *file layout* component, a *user interface* component and a *formula processor* (see figure 7.1)

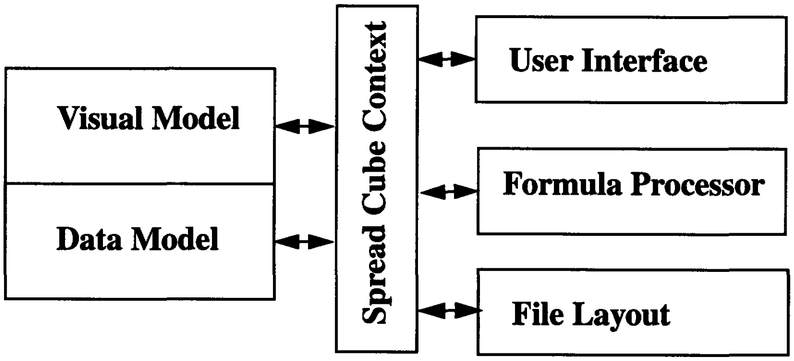


Figure 7.1: Layout of the SpreadCube Implementation

The file layout parses the input data and feeds it to the data model module.The user interface receives events from the user. It is responsible for managing the basic window widgets, like the “Load” and “Save” button as well as mouse and keyboard input directed to a particular object of the visual model. The formula processor parses arithmetic expressions input by the user and creates new variables which are subsequently added to the data model.

The *SpreadCube Context* object, which is also represented on figure 7.1, is used as an intermediary between components of the application. This arrangement allows for simplification from both technical and practical point of view. Thus, for example, the formula processor does not need to know what the interface to the data model is even though, its results must be eventually integrated into the data set. Instead it calls a method of the SpreadCube context, which later dispatches onto a method in the data model. Similarly the user interface component communicates with the visual model through the SpreadCube context.

The visual model and data model on the other hand, are built on top of each other because their relationship is not that of communicating parties, but that of ancestors and descendents. In other words the data structures within the data model are referenced by the visual model and wrapped with another layer of information which describes how objects are represented on the screen.

7.2 Data Model

The abstract data model which was introduced in chapter 3 represents the conceptual view of how the data is structured. The actual implementation differs from what we could call the "conceptual implementation" described in chapter 3. One difference, for example, is that the vault is not implemented as a N-dimensional array, but as a dimensional hierarchy. Implementing the vault as a N-dimensional array would require the dynamic allocation of an array whose dimensionality is also specified dynamically. Despite the difference between the conceptual and actual implementation, the latter is designed to exhibit the same behavior as the former.

The choice of representation for the implementation of the data model is driven by the needs of the visual model, or by the answer to the following problem: How to find a the contents of a visual vault cell given its position on the visual metric, and how to find the value of a node from the dimensional hierarchy, given its position in the hierarchy. The solution to this problem is required in order to correctly assign data values and data objects to visual objects.

7.2.1 Data Structures

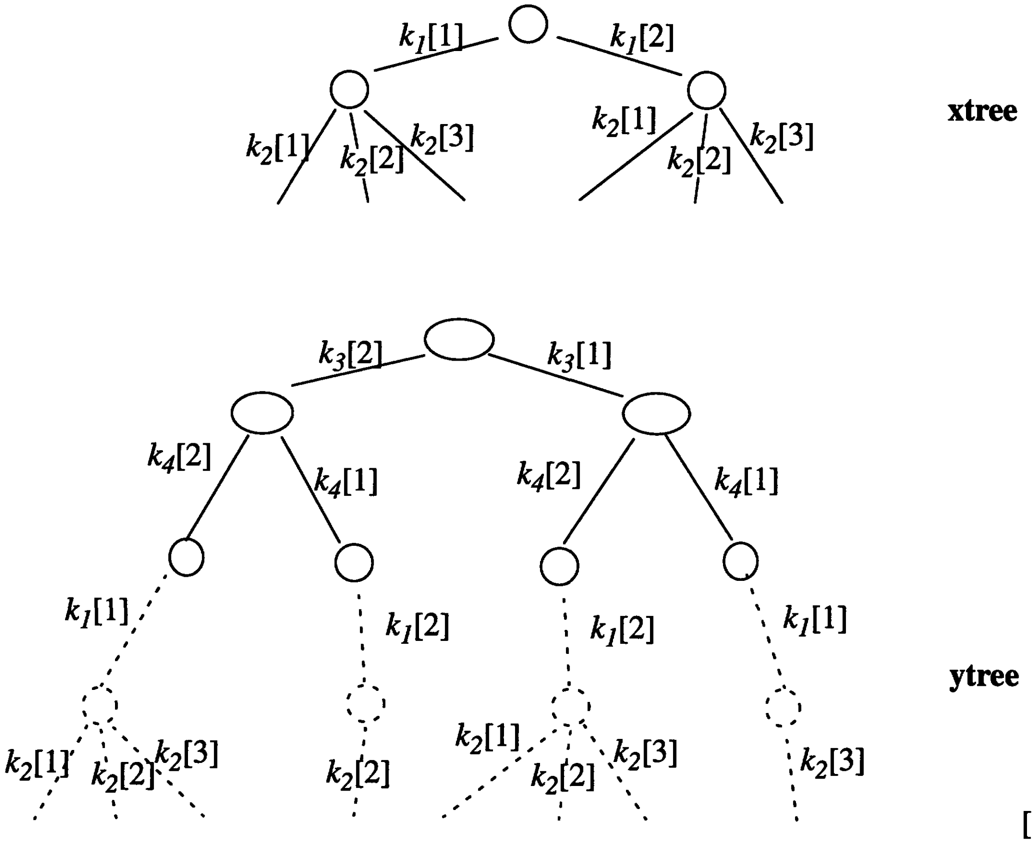


Figure 7.2: *xtree* and *ytree*.

The data model consists of two trees: *xtree* and *ytree* associated with the horizontal and vertical axes respectively. The two trees implement a planar dimensional hierarchy similar to the one shown on figure 4.3 Unlike the example figure 4.3 the two trees are not treated equally. In fact only one of these trees is used to store the data and the other is only used as template. The choice of which tree is used for what is arbitrary. In the case of SpreadCube it is the xtree which is used as a template and the ytree is used to store the data. In fact the ytree by itself is similar to a linear hierarchy like the one on figure 4.1.

The example of an xtree and ytree shown on figure 7.2 is based on the visual model from figure 4.3. The solid lines correspond to the branches that are actually represented by the visual model and consequently visualized. The data values are stored at the leaves of the xtree.

The xtree is the union of all dotted sub-trees of the ytree. From figure 7.2 one can see that a dotted sub-tree need not be complete with all branches of the xtree. In fact missing branches correspond to missing or non-existent data.

7.2.2 Accessing the Vault

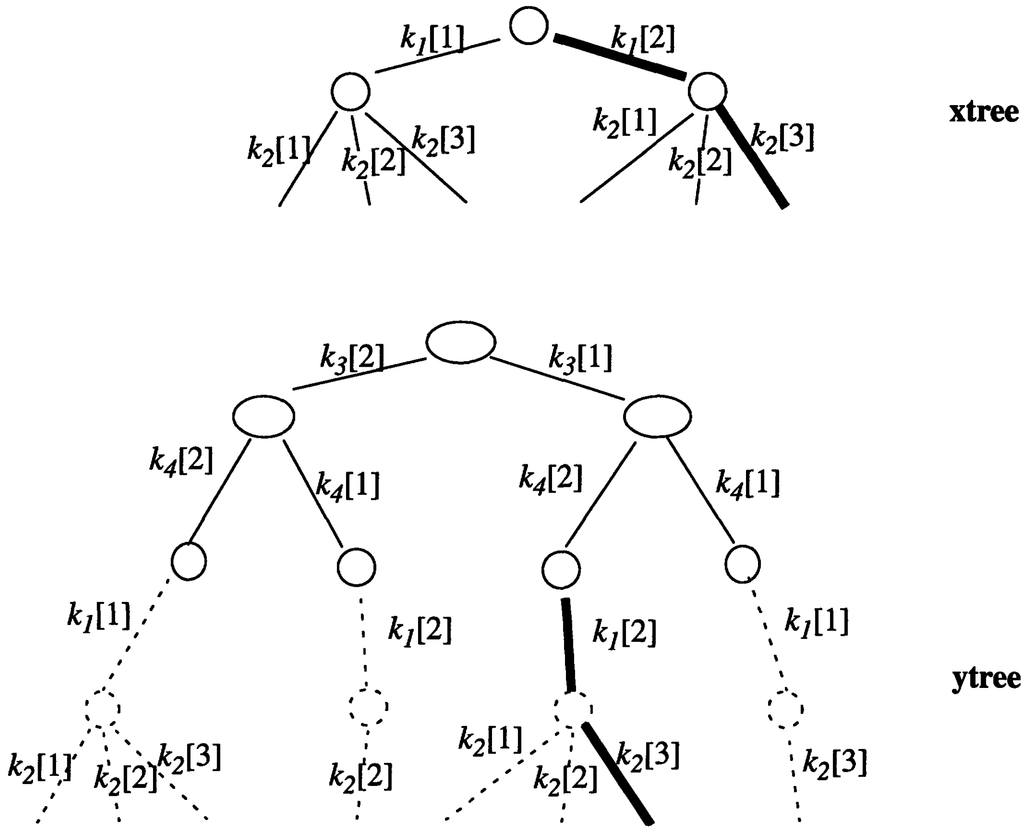


Figure 7.3: Accessing an existing cell

The values of the vault are rendered on the screen by iterating over all visible coordinates. Here are two examples of how the value of the visual vault can be retrieved from the data model given a pair of coordinates on the visual grid:

Case 1: $x = 6$, $y = 3$. (Illustrated by figure 7.3)

First the xtree is walked up starting from the $6^{\text{-th}}$ leave and pushing the labels of the branches on a stack: $k_2[3]$, $k_1[2]$ (The path is represented by a thick line on the illustration). Then the ytree is walked down starting from the $3^{\text{-rd}}$ node of the solid part of the ytree, and following those branches whose names are popped from the stack: $k_1[2]$, $k_2[3]$ (The path is represented by a thick line). Walking down the ytree in this manner ends in a leave which contains the desired data value.

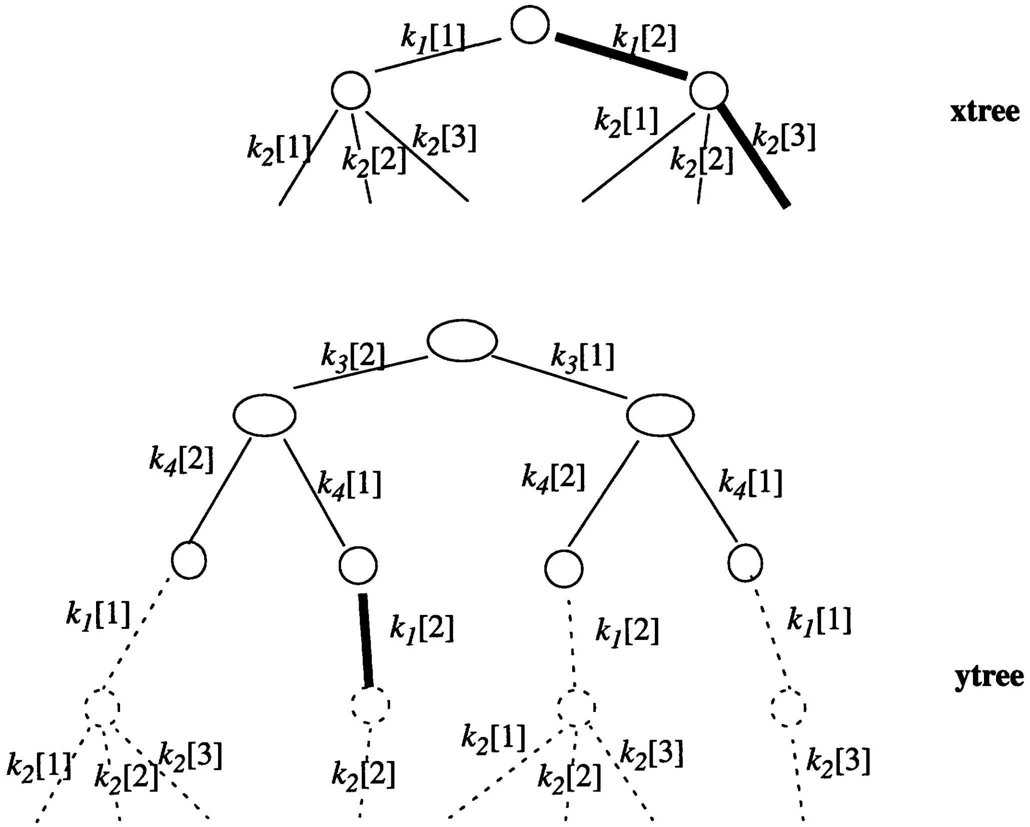


Figure 7.4: Accessing a non-existent data value

Case 2: x = 6, y = 2. (Illustrated by figure 7.4)

The process begins as in the previous case: The xtree is walked up and the following branch labels are pushed on the stack: $k_2[3], k_1[2]$. The y tree is walked down starting from the $2^{\text{-nd}}$ solid node. Unlike the previous case, however, after popping the label $k_1[2]$ from the stack and descending the branch with that label, one arrives at a node which does not have a branch with label $k_2[3]$. This indicates that the data value which is the content of the visual cell with coordinates (6,2) does not exist.

7.3 Visual Model

The visual model wraps another layer of information about the objects of the data model described in the previous section. The extra layer of information specifies where those objects are located on the visual grid and how they should be rendered. In addition the visual model defines methods for rendering of visual objects.

7.3.1 Visual Objects

For implementation purposes one can think of visual objects as rectangular areas on the visual grid. One visual object can have several other visual objects as its children. For example, the children of a screen axis are dimensions. The children of each dimension are the dimension keys.

7.3.2 Constraint Description of the Layout

The physical layout of how visual objects are rendered on the screen is determined by first calculating the layout of the two. Once the length and position of each axis has been determined, then the positions of horizontal and vertical ticks of the visual grid are calculated based on information about where the focal areas are. Finally, the rest of the visual objects, namely the cells of the visual vault are positioned with respect to the visual grid.

The layout of the two axes is determined based on a set of constraint specification. The idea is borrowed from the constraint definition of the Motif toolkit [26]. There are *positional* and *dimensional* constraints.

The positional constraints are defined with respect to the object's left, right, top and bottom neighbors, if those exist. Thus, a visual object can be either attached to another object, or neighboring with another object or none. For example if *B* is specified as the right neighbor of *A*, then the right edge of *A* must be to the left of the left edge of *B*. If *A*, however is attached on the right to *B*, then the right edge of *A* must have the same horizontal coordinate as the left edge of *B*.

The dimensional constraints arise from the requirements of minimum and maximum size of each visual object. If the visual object has children, then its dimensional constraints are determined by the dimensional and positional constraints of its children.

The final size and position of each visual object is determined by a fixed point iteration over the set of all constraints. Each iteration step includes determining the required dimensional constraints of an object, and based on that information updating the positional constraints of its right and bottom neighbors.

7.4 User Interface

The user interface was built using a custom widget library on top of the Motif toolkit. It supplies the visual model with a drawing board object where the visual model renders its objects.

7.4.1 Gesture Recognition

An interesting feature of the user interface is the gesture recognizer. Each gesture input by the user is simply a sequence of points. The gesture recognizer uses a dynamic warping to compare that sequence of points to a set of predefined templates. Each template consists of a sequence of points which represent a stylized version of the particular gesture class that the template represents. For each template the recognizer calculates a score of how well it matches the input gesture. The result of the recognition is the template which matches best, unless the match score is below a certain threshold in which case the gesture is deemed unrecognized.

Dynamic warping is a technique which compares segments of the gesture to best matching segments of the template. One can think of this procedure as stretching and shrinking the individual segments of the gesture without changing their orientation so as to maximize the matching score. That is how a gesture and a template can have different number of points, and segments of different lengths. In fact each template which specifies a gesture is a very simple set of segments For example the template for a flick-east gesture is: {(0,0), (1, 0)}

7.4.2 Tone Generator

The tone generator is used for representing data values as acoustic pitches. Currently, it is implemented to work with SunOS and SOLARIS platforms. The tone generator consists of a client and a server component, which run as two separate processes. The client component is called by the rendering methods of the visual model as a way of specifying the duration and the pitch of the generated tone. The server component generates raw audio data at the sampling rate of 8kHz which can be piped to the `/dev/audio` of the SunOS and SOLARIS platforms. The client and the server communicate through a non-blocking pipe whose output is checked by the server at regular intervals.

The client/server interaction of the tone generator components is necessary in order to accomplish the following tasks without audible noise interference in the generated tone:

- specify the parameters of a tone asynchronously as a response to the user pointing at a data value.
- Asynchronously cancel the generation of a tone if the generation of a new one is requested.

Because of the above two features, when the tone generator is activated, the user can slide the mouse pointer over a column of data and listen to the “chirping” that the computer produces in pure tones and without noises.

Chapter 8

Conclusion

8.1 Parts and Whole

The main objective of this thesis was to demonstrate how the appropriate combination of several data manipulation and visualization techniques can produce a powerful tool for data analysis.

Indeed if any of the SpreadCube operations were to be taken away, the usefulness of the remaining operation set diminishes greatly. For example, the selection and the variable promotion operation are tightly coupled together when used to isolate a piece of the data from the rest. If either of these operations is taken away, one would not be able to isolate ad-hoc specified portions of the data.

Similarly the three operations: aggregation, selection and promotion of variable are also coupled together to provide the user with the ability of aggregating ad-hoc specified portions of the data. This can be done, but first selecting the desired region, promoting the selection variable and then aggregating on the nodes of the selection variable dimension.

The next section describes similar examples of stringing two or three primitive operations into useful sequences.

8.2 Useful Combinations of Operations

8.2.1 Queries

One can think of the selection variable as a result from a query. The query can be issued by entering an appropriate predicate expression into the formula processor. For example, the expression:

```
"Result" = A > B
```

Will produces a selection variable called `Result` whose values will be either `true` or `false` depending on whether the values of `A` and `B` with the same set dimension coordinates satisfy the predicate `A > B` or not. When this selection variable is promoted, the visual vault will be split into two parts: One which satisfies the predicate, and one which does not. After slicing the selection variable dimension over the node with value 1, one will be visualizing the results of the query.

8.2.2 Marking and Extracting Outliers

Outliers are values which do not seem to conform to the common trends of the context in which they appear. Such values may have originated because of unusual circumstances of the system which is being studied, or simply because of noise in the input data. In both cases it is often desirable to isolate them from the rest of the data.

After a column is sorted, outlier values “stick out” visually, perhaps because they are too big or too small. The user can then select those values, thus producing a selection variable. After promoting that variable the entire data set will be split into a section which is responsible for the outlier values and a section which contains the more “typical” values. Once the outliers have been isolated in such manner, one can use a slice operation to view the section of the data which has caused them and in order to understand the reasons behind their occurrence.

8.2.3 Producing Standard Looking Graphs

Typically, when in graph mode, SpreadCube produces multiple graphs at the same time. Most of the times this is advantageous because it provides the user with a number of thumbnail plots, which can be a way of quickly previewing the data.

If one wishes, however, to concentrate on a particular plot, one can do so by two slice operations through a dimension on each axis. This is illustrated by the transition between figure B.28 and figure B.29 of appendix B.

8.3 Future Experimentation and Research

The EDA tool presented in this thesis is a demonstration of what the combination of several technologies can achieve. This section puts the idea in perspective and outlines possible ways in which additional data exploration and data mining techniques can be integrated into a tool like SpreadCube

8.3.1 Front End for Pattern Recognition

The data exploration environment which SpreadCube provides is well suited as a front end for pattern recognition methods.

For example, a typical pattern recognition problem is classifying samples characterized by a feature vector. In the context of the SpreadCube tool, the role of a feature vector is played by the sticks parallel to the record dimension. The result of the classification is a nominal variable whose values are the labels specifying the classes assigned to the samples. The resulting nominal variable can then be promoted to split the visual vault into data belonging to different classes. Alternatively, one can apply a different classification algorithm and produce a second nominal variable as the result. In this way the results of two separate classification algorithms could be compared side by side, by looking at their representation as two nominal variables.

8.3.2 Various 2D and 3D Plot Routines

So far, we have demonstrated how switching into graph mode produces scatter plots out of a column of data or a color intensity plot out of a matrix of data.

The visualization concepts provided by SpreadCube have the potential for naturally extending the variety of the plots that can be generated. For example, by changing the presentation type, a scatter plot can become a line plot. By adding an extra operation for overlaying parallel sticks on top of each other, one will be able to produce plots with multiple line graphs. The reason why this is possible is because parallel sticks share one metric dimension.

Similarly, by changing the presentation type, a color intensity plot can be turned into a 2D surface plot or a contour plot.

8.3.3 Graphical Data Input

The idea of fusing graphics and text together, which SpreadCube explores, can be approached not only from the visualization point of view but also from the data input point of view.

Imaging the scenario where we wish to investigate the dependence of a model on some input parameter. In that case we can sketch the values of the input parameter by simply *drawing* them into a data column, in a manner similar to using a free-hand drawing tool from any graphics program. Thus we will be creating a new variable whose values are input graphically. We can use this new variable in arithmetic expressions to produce other derived variables. Later we can *reshape* the graphically input variable and observe how the derived variables which depend on it change.

8.3.4 User Interface Analysis

It is beyond the scope of this work, to foresee how well potential users can adapt to the existing user interface. In fact, because of its novelty, one could expect that SpreadCube's UI is far from perfect. The only sensible way in which to improve the UI and to evaluate it is by gradually evolving it based on the responses of users over a long period of time.

On the other hand, we have provided the conceptual framework within which this new UI paradigm will evolve. By outlining this framework, demonstrating its consistency and simplicity, we have provided the theoretical justification for the success of SpreadCube's UI, which even though not a guarantee, is a prerequisite.

References

[1] Ramana Rao and Stuart Card, *The SpreadCube Data Model*, *draft* May 10, 1994

[2] R.Rao, S.K. Card *The Table Lens: Merging Graphical and Symbolic Representations in an Interactive Focs+Context Visualization for Tabular Information,* Proceedings of teh ACM SIGCHI Boston, MA, April 1994, ACM

[3] R. Rao, C. Stuart, *Exploring large tables with the table lens*, Human Factors in computing systems (CHI) - Conference Proceedings v. 2.

[4] Peter Pirolli and Ramana Rao, *Table Lens as a Tool for Making Sense of Data, Advanced Visual Intefaces 96, Gubbio Italy, ACM Press*

[5] Michael Spenke, Christain Beilken, and Thomas Berlage, *FOCUS: The Interactive Table for Product Comparison and Selection*, To appear in UIST96, Seattle, Nov 96, ACM Press

[6] E.F. Codd, S.B. Codd, C.T. Salley, *Beyond Decision Support*, Computerworld, July 26, 1993

[7] E.F. Codd, S.B. Codd, C.T. Stalley, *Providing OLAP to User-Analysts:* An IT Mandate, Codd & Date, Inc.

[8] E.F. Codd and S.B. Codd, *OLAP (On-line Analytical Processing) with TM/1*, E.F. Codd & Associates

[9] D. King, *Execu-View: Visualizing the Multidimensional Enterprise*, Comshare Inc. 1992

[10] Robert Spence and Mark Apperley, Database Navigation: *An Office Environment for the Professional*, Behavior and Information Technology, 1982, 1(1), pp. 43-54

[11] M.C. Chuah, S.F. Roth, J. Mattis, and J. Kolojejchick, SDM: *Malleable Informatio-Graphics*, Information Visualization 95, IEEE Computer Society Press.

[12] Jade Goldstein, Steven F. Roth, John Kolojejchick and Joe Mattis: *A Framework for Knowledge-based Interactive Data Exploration*, Journal of Visual Languages and Computing, 1994, vol 5, pp 339-363

[13] John Lamping, Ramana Rao, and Peter Pirolli, *A Focus+Context Technique based on Hyperbolic geometry for Visualizing Large Hierarchies,* Human Factors in computing systems (CHI) - Conference Proceedings v. 1.

[14] George W. Furnas Benjamin B. Bederson, Space-Scale Diagrams: *Understanding Multiscale Interfaces*, Human Factors in computing systems (CHI) - Conference Proceedings v 1.

[15] Ishantha Lokuge and Suguru Ishizaki, GeoSpace: *An Interactive Visualization System for Exploring Complex Information Spaces*, Human Factors in computing systems (CHI) - Conference Proceedings v 1.

[16] G.Alvin Mead, The Sorted Binary Plot: *A New Technique for Exploratory Data Analysis*, Technometrics, Feb. 1989, vol 31 No.

[17] W.S.Hodgkiss, *A Modular Approach to Exploratory Data Analysis*, Ocean 89, Part 4. Published by IEEE, IEEE Service Center.

[18] J.Angstenberger, R.Weber, W.Meier, Data Engine: *A Software Tool for Intelligent Data Analysis*, Wescon Conference Record 1994. Wescon, Los Angeles, CA

[19] Tufte, Edward R., *Envisioning Information*, 1990, Cheshire, Conn. (P.O. Box 430, Cheshire 06410): Graphics Press.

[20] John W. Tukey, *Exploratory Data Analysis*, 1977, Addison-Wesley Publishing Company.

[21] Frederick Mosteller, John W. Tukey, 1977, *Data Analysis and Regression - a second course in statistics*, Addison-Wesley Publishing Company.

[22] David C. Hoaglin, Frederick Mosteller, John W. Tukey, *Exploring Data Tables, Trends, and Shapes*, Wiley Series in Probability and Mathematical Statistics, 1985, John Wiley & Sons, Inc.

[23] David C. Hoaglin, Frederick Mosteller, John W. Tukey, *Understanding Robust and Exploratory Data Analysis*, Wiley Series in Probability and Mathematical Statistics, 1983, John Wiley & Sons, Inc.

[24] Paul F. Velleman, David C. Hoaglin, *Applications, Basics, and Computing of Exploratory Data Analysis*, 1981, Boston, MA: Duxbury Press.

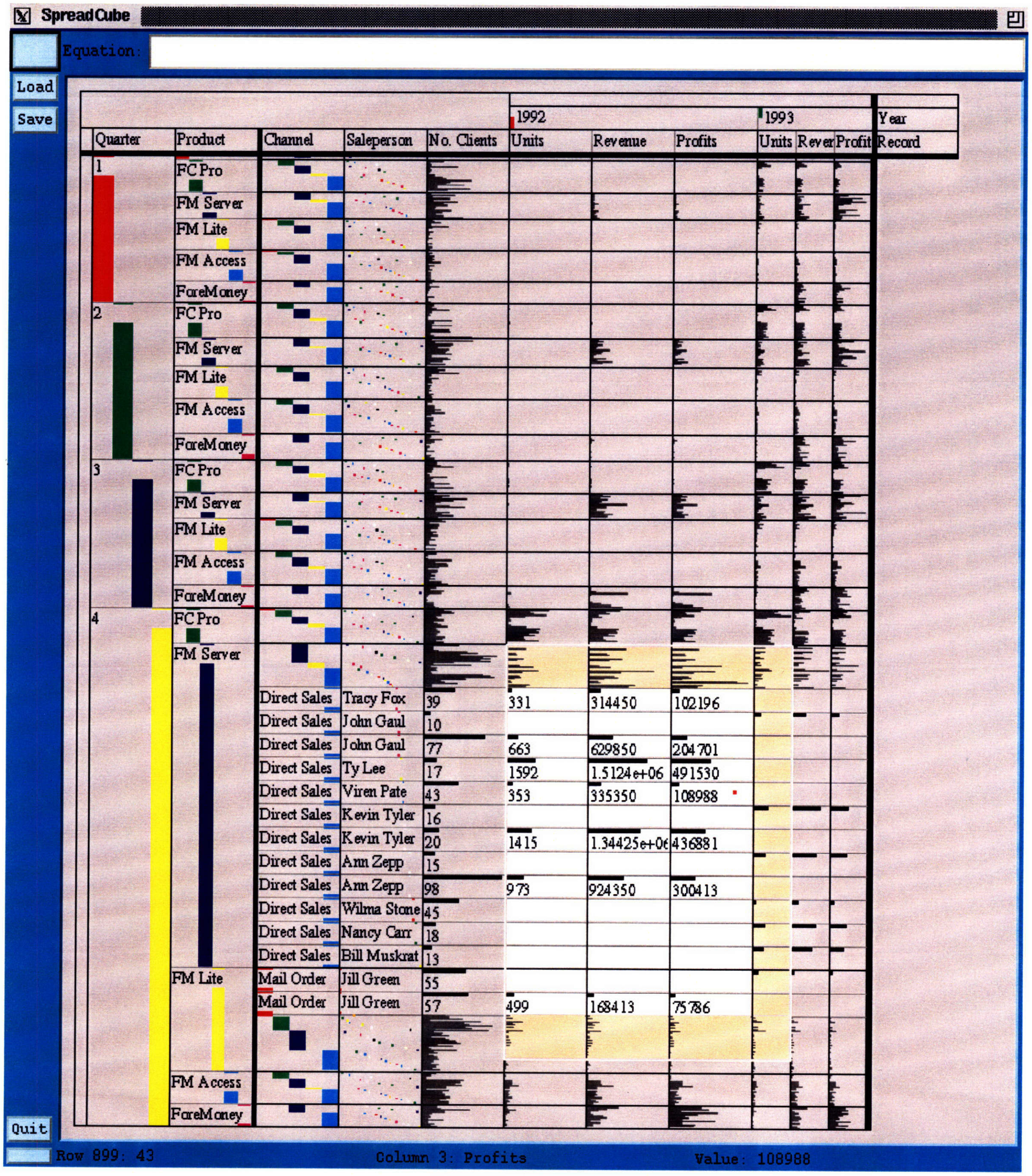
[25] Bertrand Meyer, *Introduction to the Theory of Programming Languages*, Prentice Hall International Series in Computer Science, 1990, Prentice Hall International Ltd.

[26] Dan Heller, *Motif Programming Manual for OSF/Motif Version 1.1*, O'Reilly & Associates, Inc. 1991

Appendix A

Color Plates

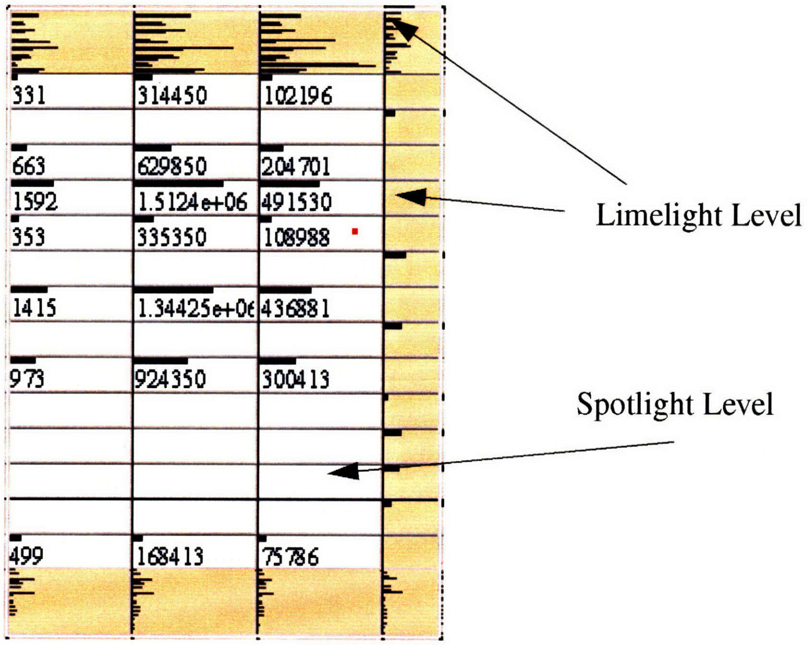
A.1 Color Plates for Chapter 5



Channel	Saleperson	No. Clients	Units	Revenue	Profits
Direct Sales	Tracy Fox	39	331	314450	102196
Direct Sales	John Gaul	10			
Direct Sales	John Gaul	77	663	629850	204701
Direct Sales	Ty Lee	17	1592	1.5124e+06	491530
Direct Sales	Viren Pate	43	353	335350	108988
Direct Sales	Kevin Tyler	16			
Direct Sales	Kevin Tyler	20	1415	1.34425e+0(	436881
Direct Sales	Ann Zepp	15			
Direct Sales	Ann Zepp	98	973	924350	300413
Direct Sales	Wilma Stone	45			
Direct Sales	Nancy Carr	18			
Direct Sales	Bill Muskrat	13			
Mail Order	Jill Green	55			
Mail Order	Jill Green	57	499	168413	75786

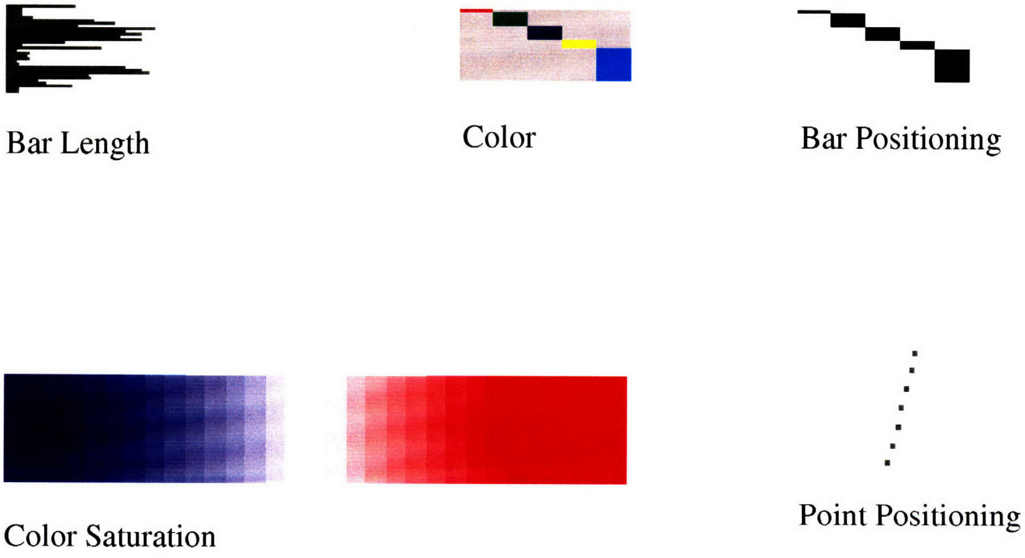
Row 899: 43   Column 3: Profits   Value: 108988

Figure A.1: The SpreadCube Visualizatoin



331	314450	102196	
663	629850	204701	
1592	1.5124e+06	491530	
353	335350	108988	
1415	1.34425e+0	436881	
973	924350	300413	
499	168413	75786	

Figure A.2: A Focal Area



Bar Length

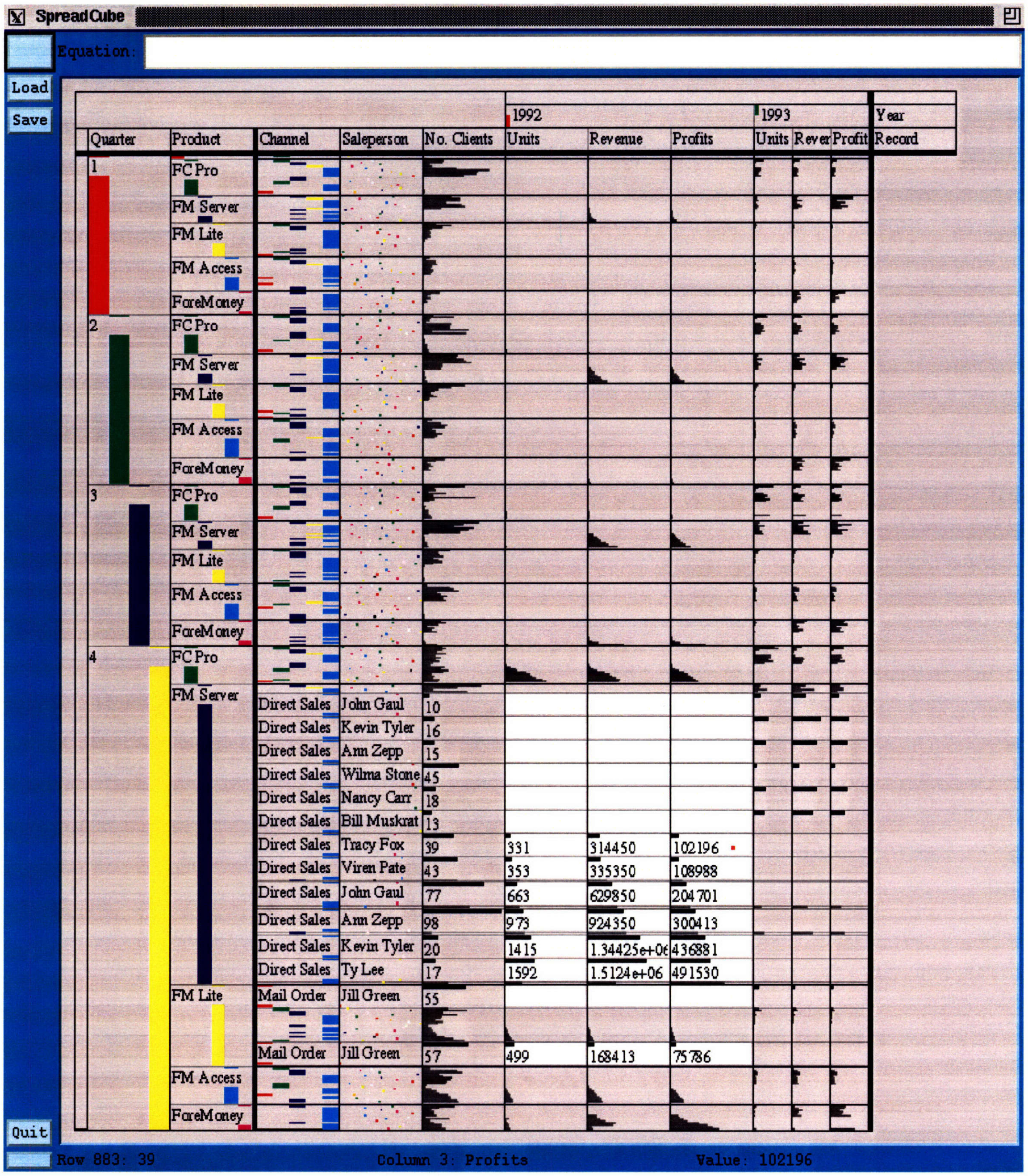
Color

Bar Positioning

Color Saturation

Point Positioning

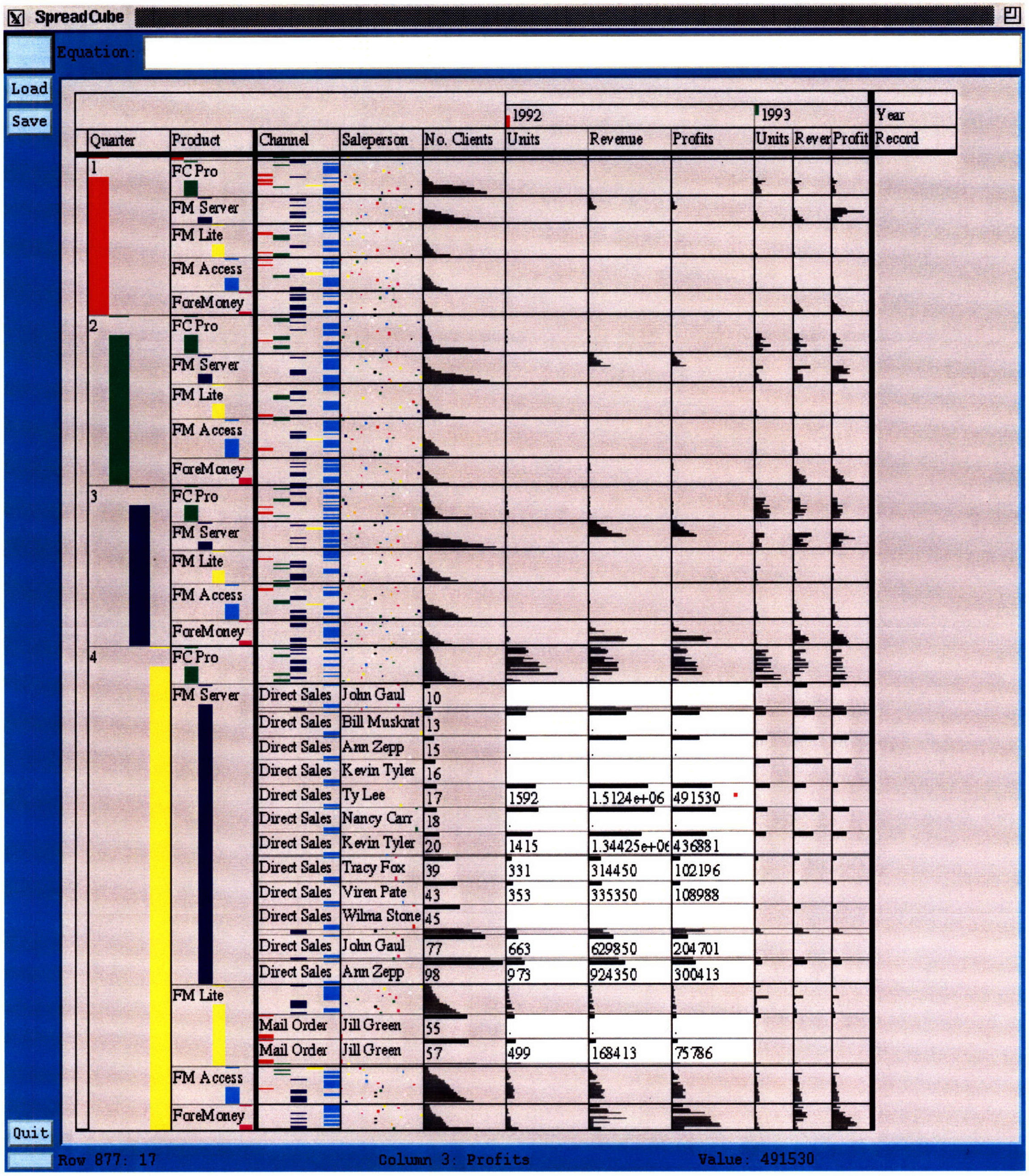
Figure A.3: Presentatoin Types



Channel	Saleperson	No. Clients	Units	Revenue	Profits
Direct Sales	John Gaul	10			
Direct Sales	Kevin Tyler	16			
Direct Sales	Ann Zepp	15			
Direct Sales	Wilma Stone	45			
Direct Sales	Nancy Carr	18			
Direct Sales	Bill Muskrat	13			
Direct Sales	Tracy Fox	39	331	314450	102196
Direct Sales	Viren Pate	43	353	335350	108988
Direct Sales	John Gaul	77	663	629850	204701
Direct Sales	Ann Zepp	98	973	924350	300413
Direct Sales	Kevin Tyler	20	1415	1.34425e+0	436881
Direct Sales	Ty Lee	17	1592	1.5124e+06	491530
Mail Order	Jill Green	55			
Mail Order	Jill Green	57	499	168413	75786

Row 883: 39 Column 3: Profits Value: 102196

Figure A.4: SpreadCube After Sorting by “Profits”



Quarter	Product	Channel	Saleperson	No. Clients	1992 Units	Revenue	Profits
4	FM Server	Direct Sales	John Gaul	10			
		Direct Sales	Bill Muskrat	13	.	.	.
		Direct Sales	Ann Zepp	15	.	.	.
		Direct Sales	Kevin Tyler	16			
		Direct Sales	Ty Lee	17	1592	1.5124e+06	491530
		Direct Sales	Nancy Carr	18	.	.	.
		Direct Sales	Kevin Tyler	20	1415	1.34425e+0	436881
		Direct Sales	Tracy Fox	39	331	314450	102196
		Direct Sales	Viren Pate	43	353	335350	108988
		Direct Sales	Wilma Stone	45			
		Direct Sales	John Gaul	77	663	629850	204701
		Direct Sales	Ann Zepp	98	973	924350	300413
	FM Lite	Mail Order	Jill Green	55	.	.	.
		Mail Order	Jill Green	57	499	168413	75786

Row 877: 17 Column 3: Profits Value: 491530

Figure A.5: SpreadCube Sorted by "No. Clients"

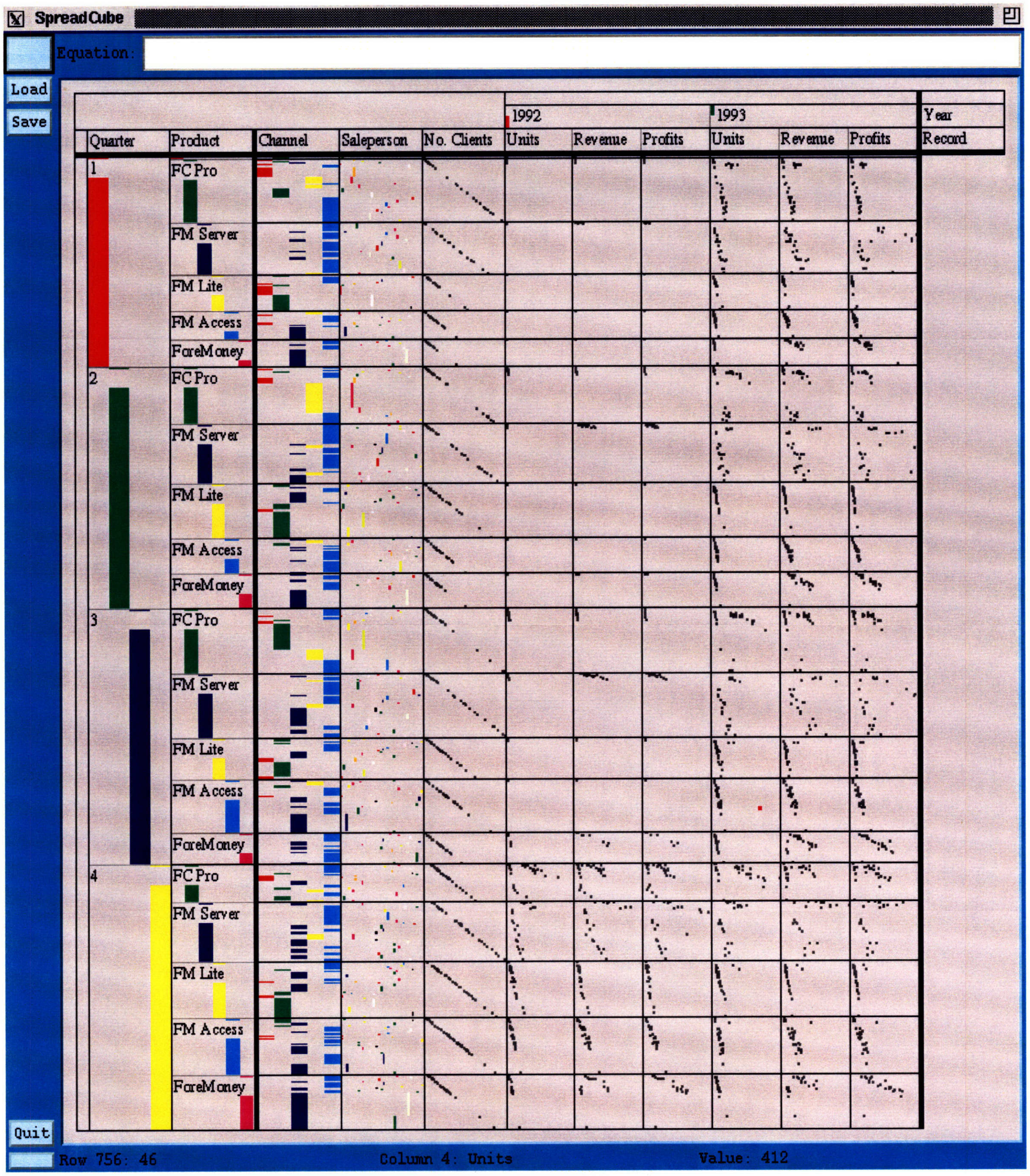
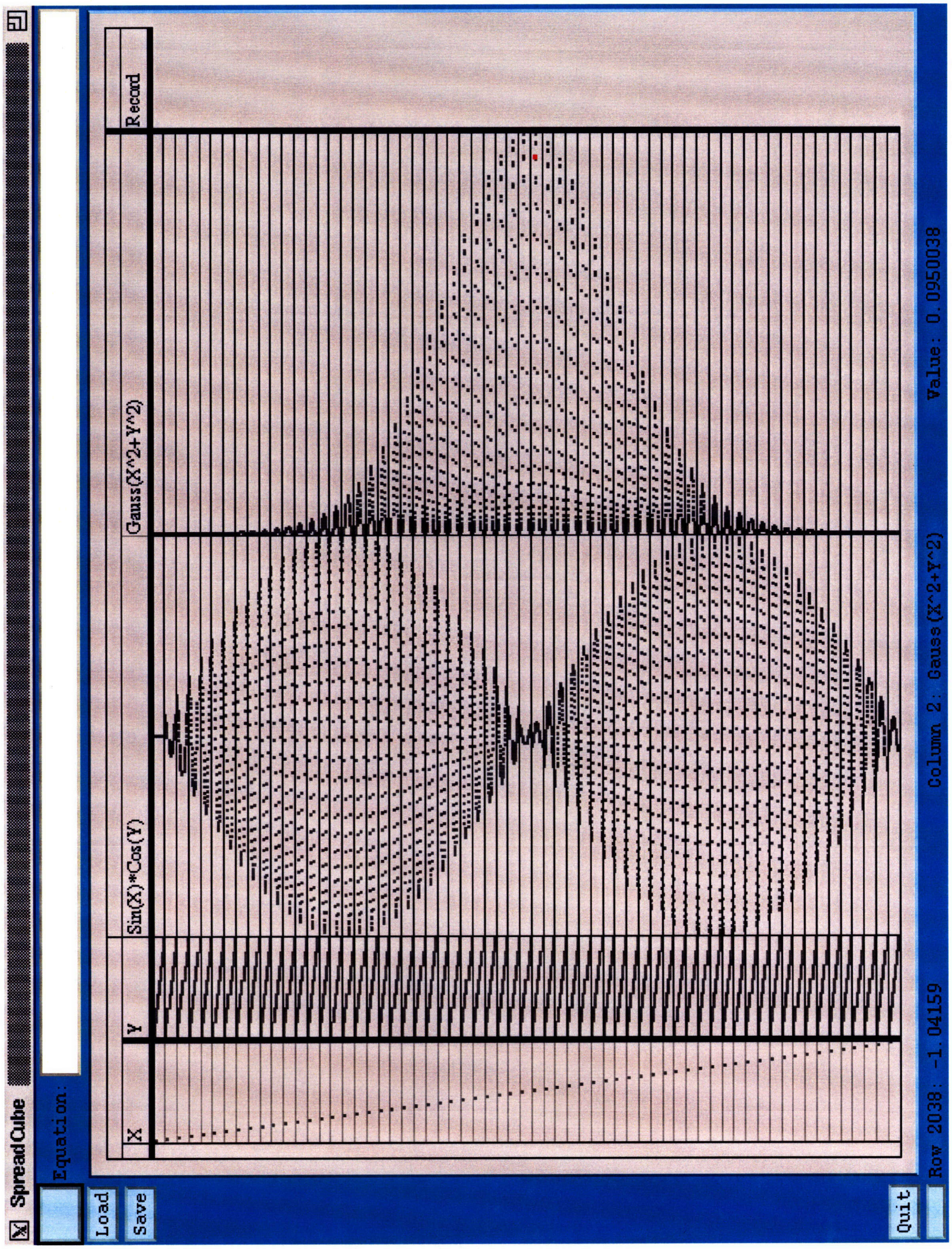
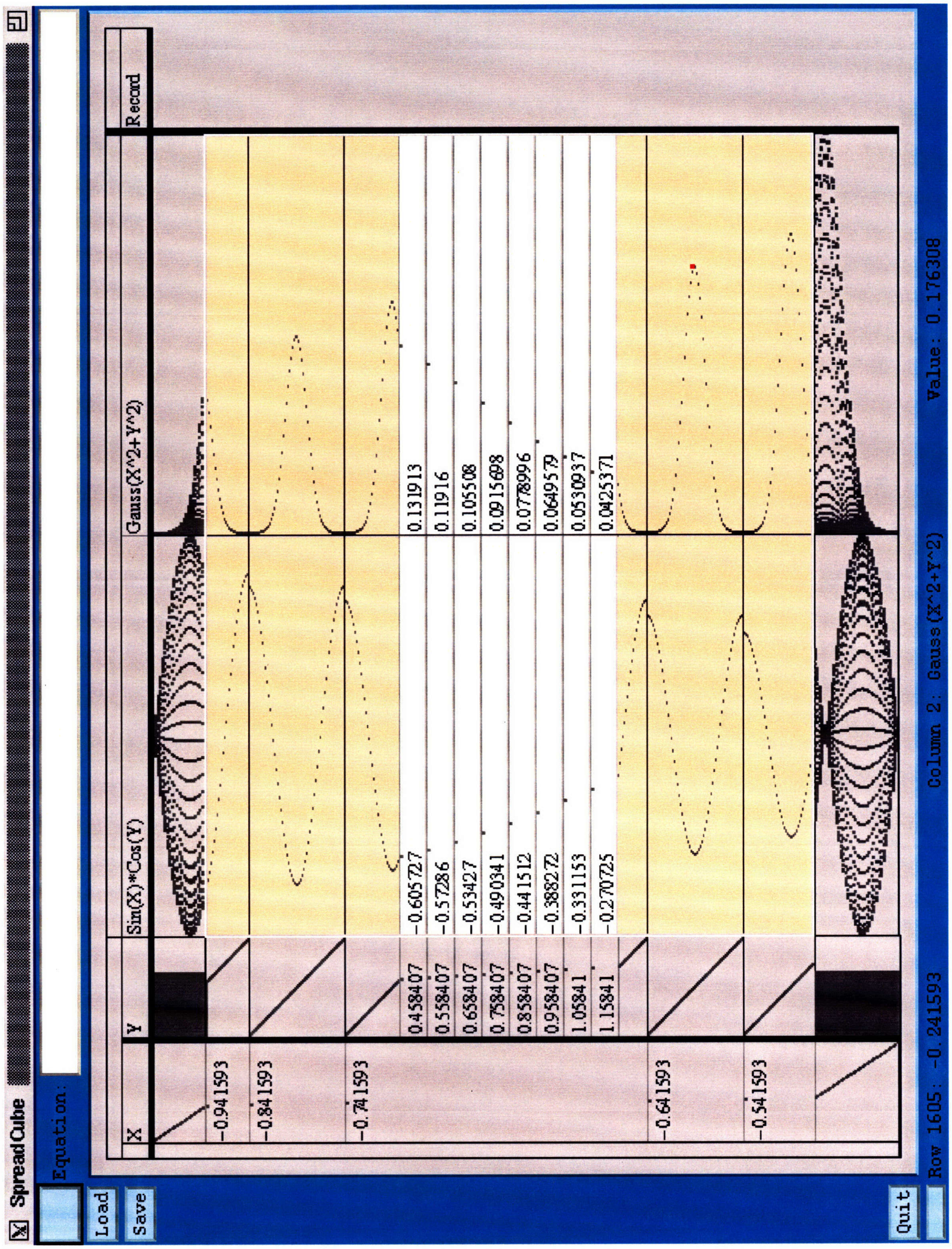


Figure A.6: SpreadCube in Graph Mode



```
Row 2038: -1.04159          Column 2: Gauss(X^2+Y^2)          Value: 0.0950038
```

Figure A.7: A Toy Data Set



X	Y	Sin(X)*Cos(Y)	Gauss(X^2+Y^2)
-0.941593			
-0.841593			
-0.741593			
	0.458407	-0.605727	0.131913
	0.558407	-0.57286	0.11916
	0.658407	-0.53427	0.105508
	0.758407	-0.490341	0.0915698
	0.858407	-0.441512	0.0778996
	0.958407	-0.388272	0.0649579
	1.05841	-0.331153	0.0530937
	1.15841	-0.270725	0.0425371
-0.641593			
-0.541593			

Row 1605: -0.241593 Column 2: Gauss(X^2+Y^2) Value: 0.176308

Figure A.8: The Toy Data Set with a Focal Area

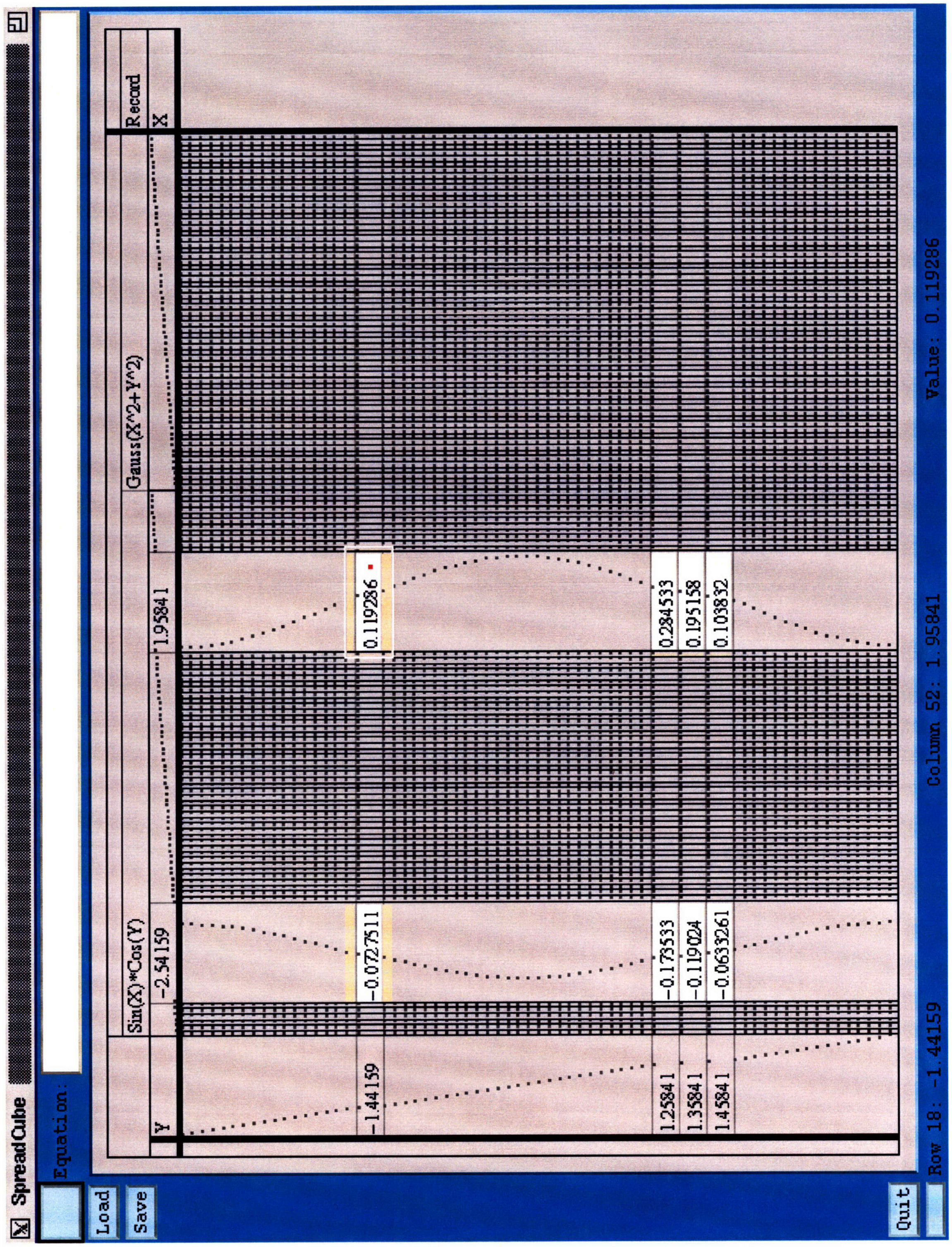
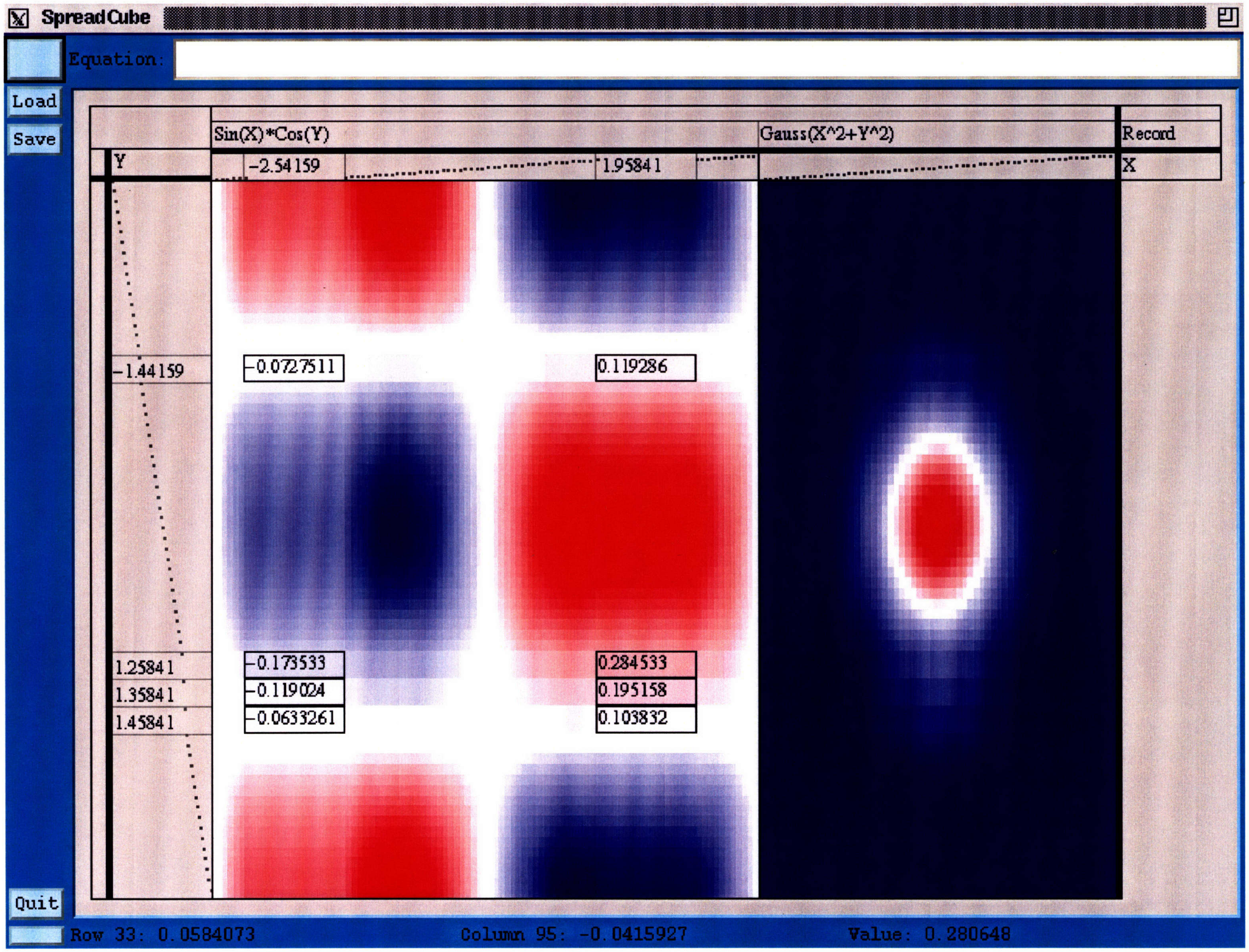
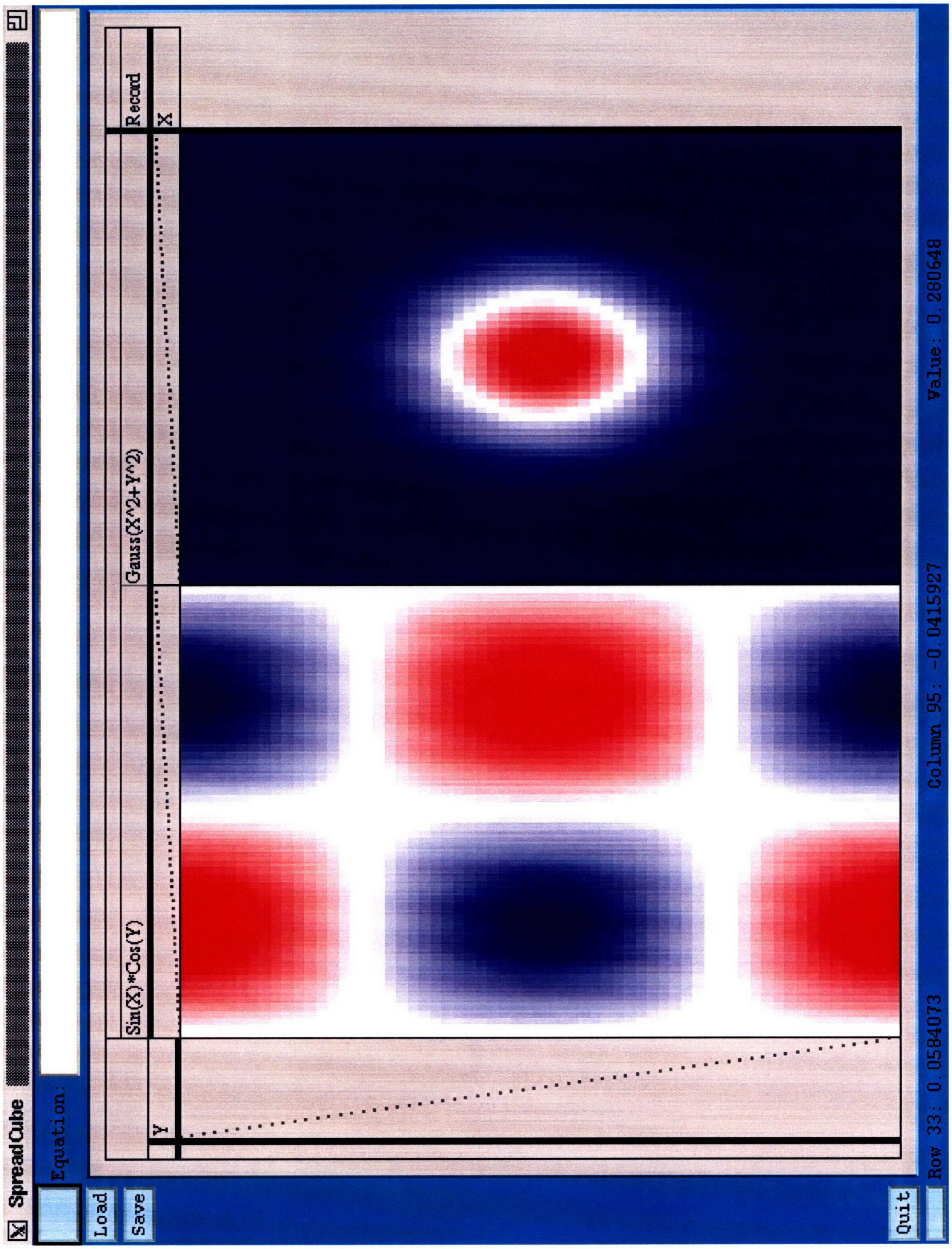


Figure A.9: Toy 2D Plots



```
Row 33: 0.0584073          Column 95: -0.0415927          Value: 0.280648
```

Figure A.10: Toy 3D Plots (with focal areas)



```
Row 33: 0.0584073    Column 95: -0.0415927    Value: 0.280648
```

Figure A.11: Toy 3D Plots

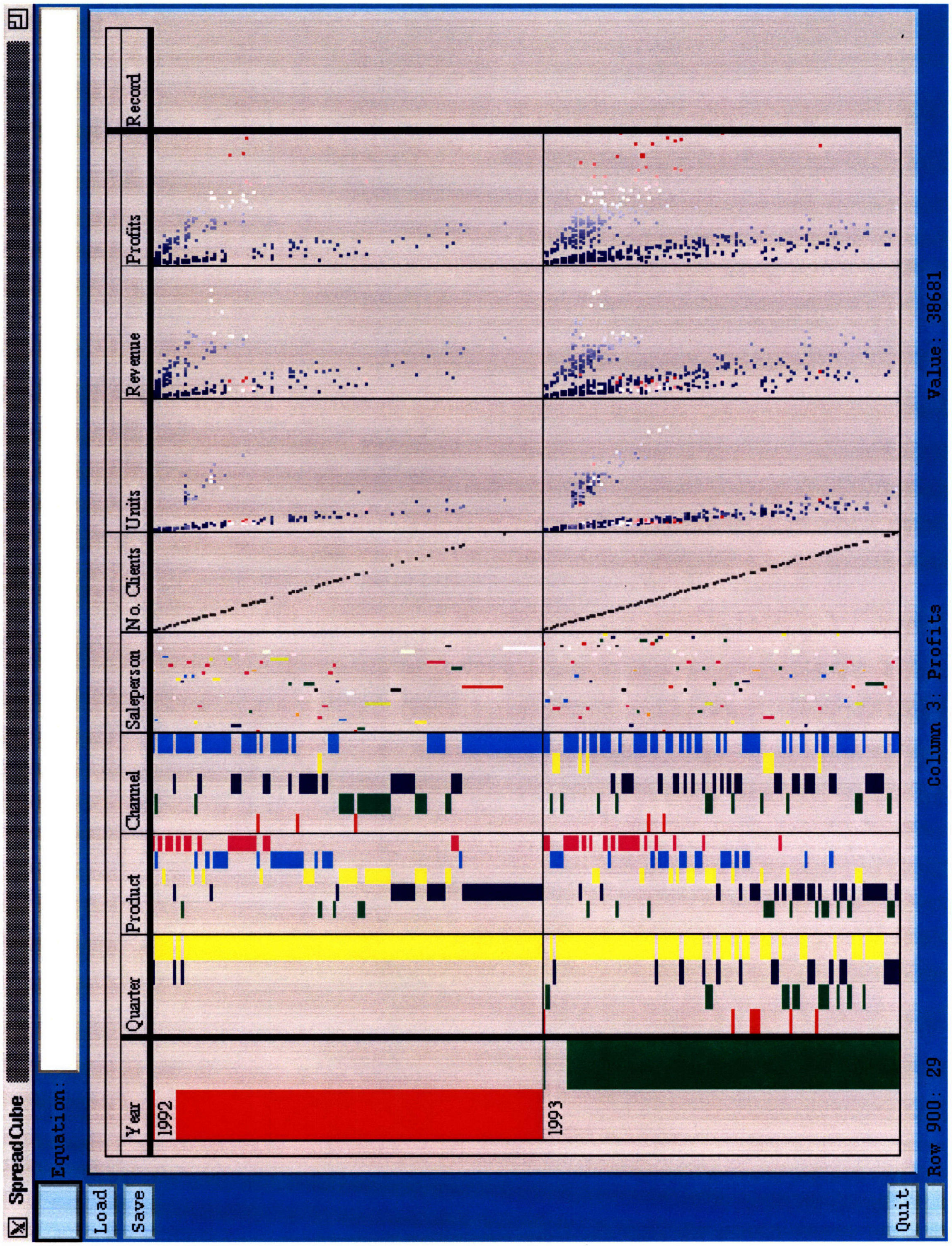


Figure A.12: Using the "Profits" variable to Modulate "Units" and "Revenue"

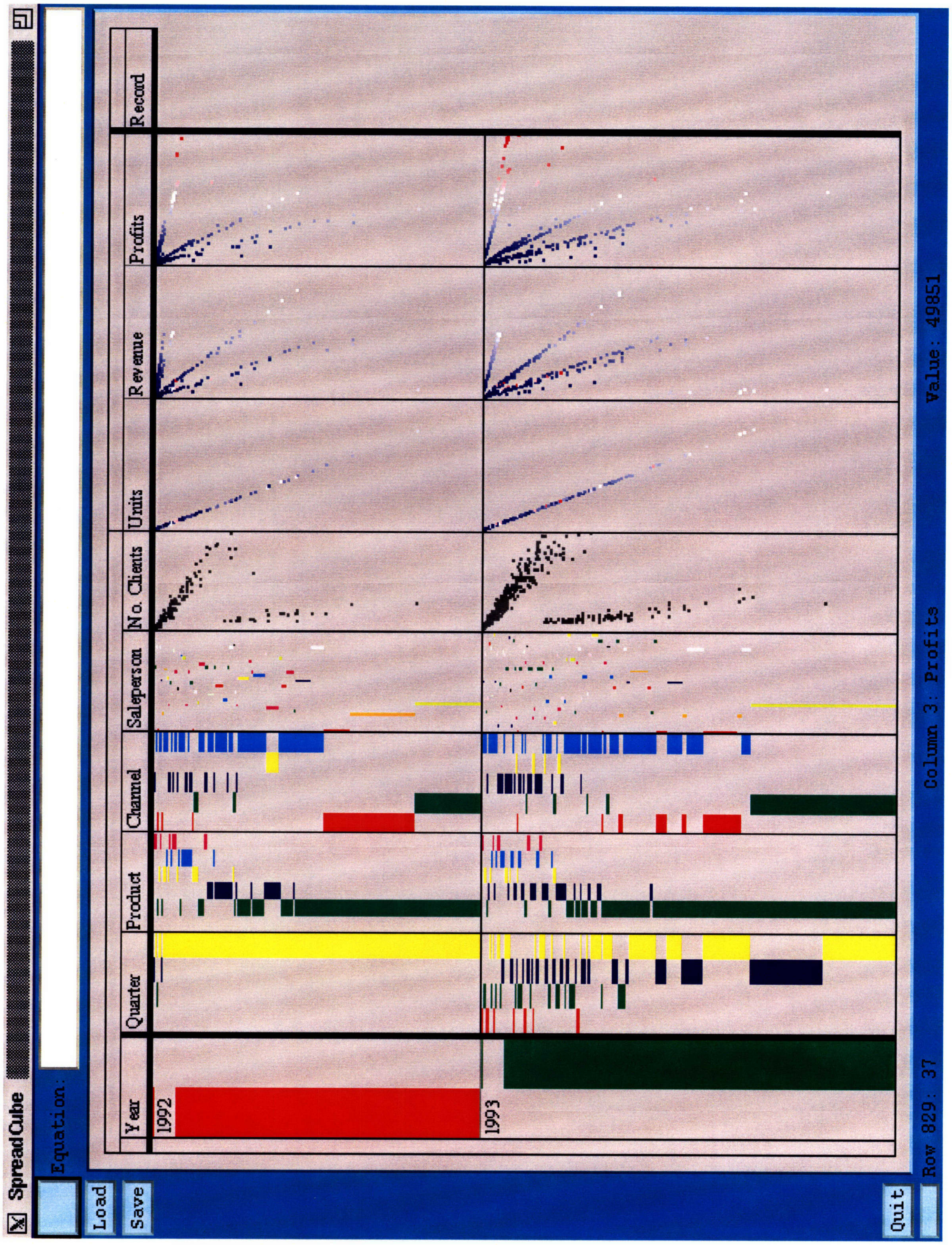


Figure A.13: Space Modulation According to the “Units” column

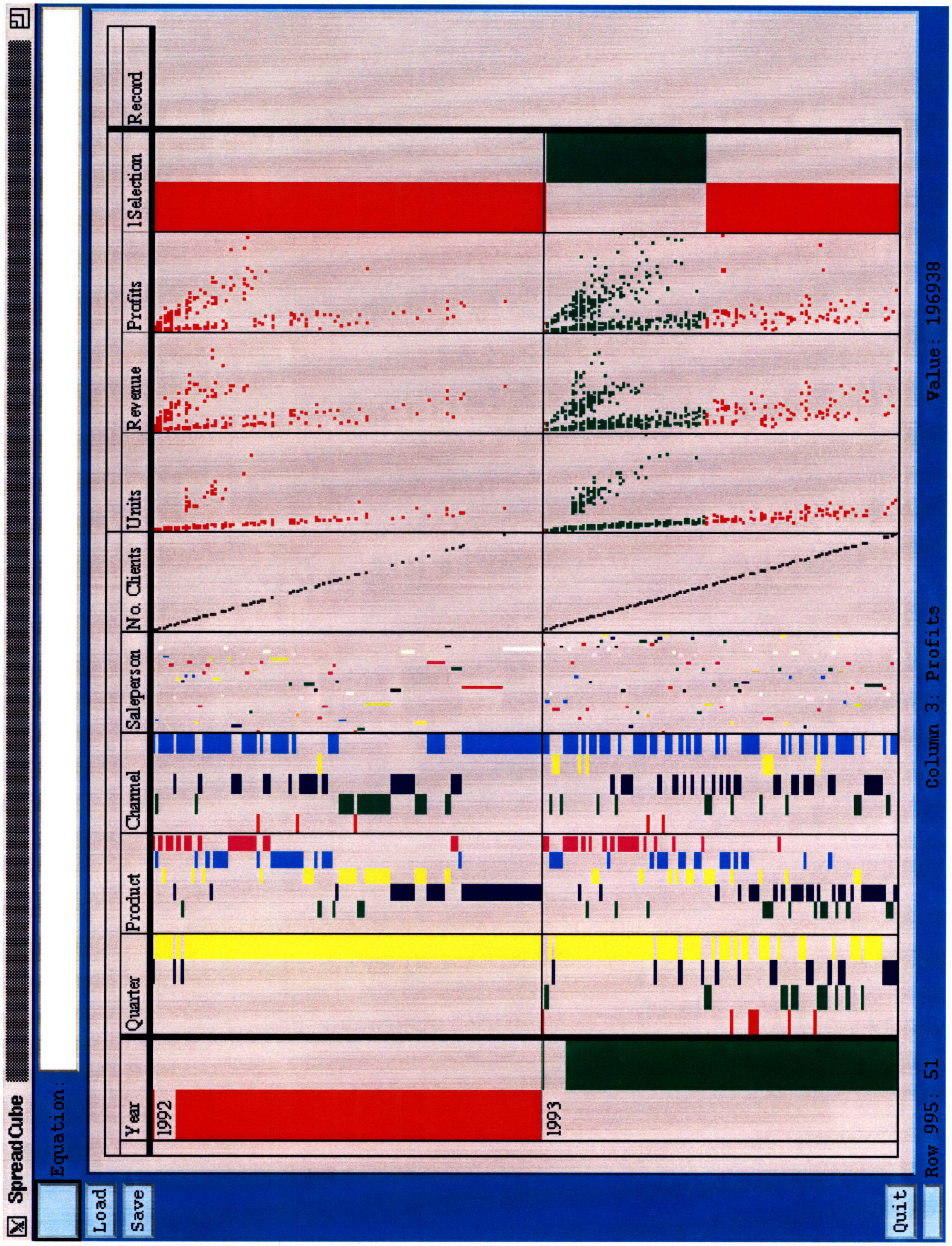


Figure A.14: Range Selection

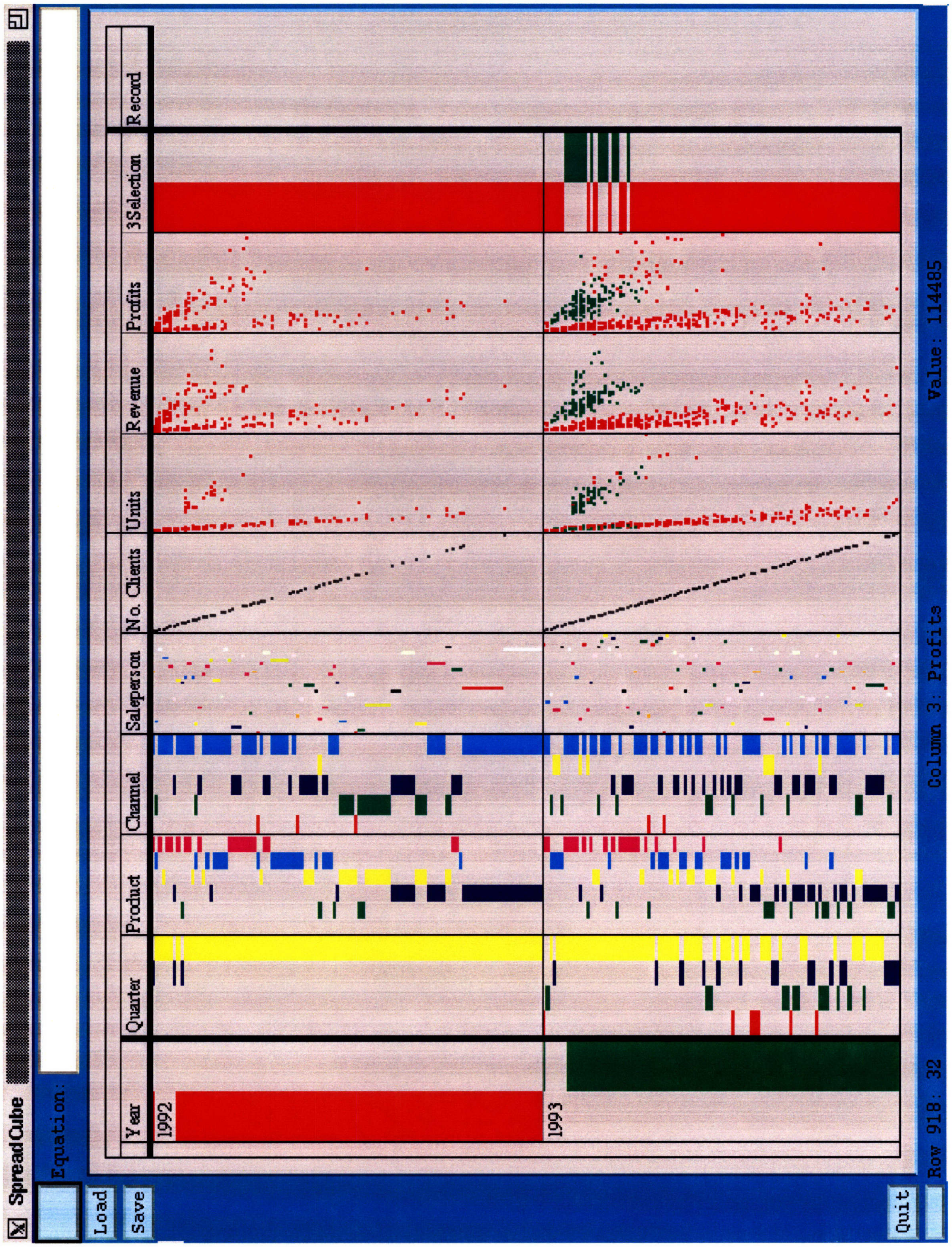
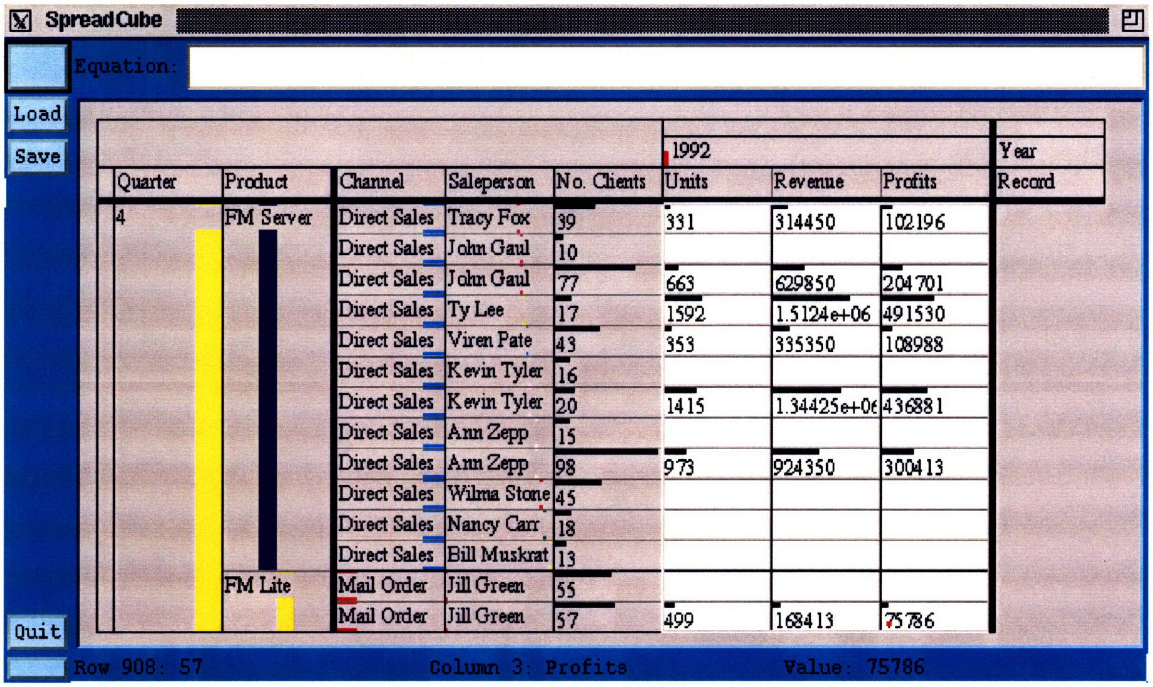


Figure A.15: Region Selection

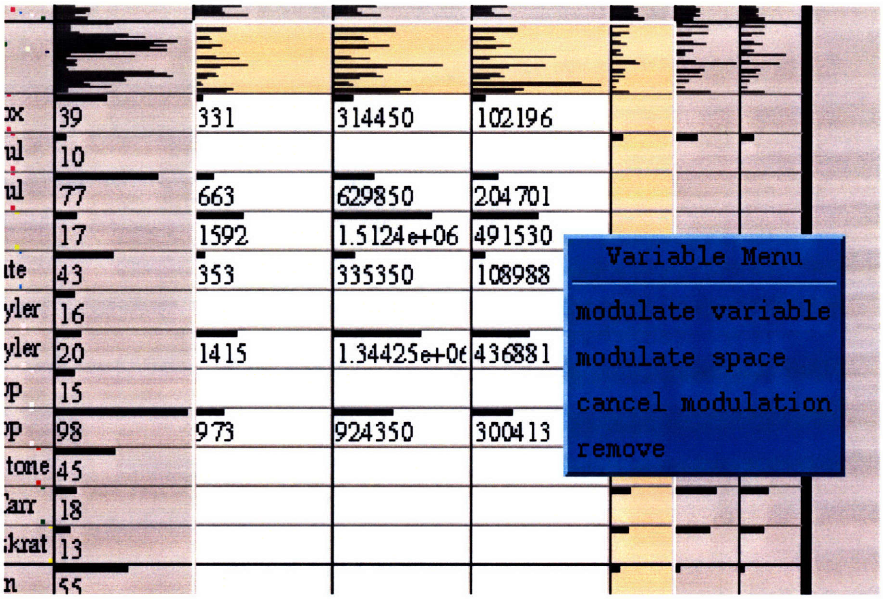


					1992			Year
Quarter	Product	Channel	Saleperson	No. Clients	Units	Revenue	Profits	Record
4	FM Server	Direct Sales	Tracy Fox	39	331	314450	102196	
		Direct Sales	John Gaul	10				
		Direct Sales	John Gaul	77	663	629850	204701	
		Direct Sales	Ty Lee	17	1592	1.5124e+06	491530	
		Direct Sales	Viren Pate	43	353	335350	108988	
		Direct Sales	Kevin Tyler	16				
		Direct Sales	Kevin Tyler	20	1415	1.34425e+0	436881	
		Direct Sales	Ann Zepp	15				
		Direct Sales	Ann Zepp	98	973	924350	300413	
		Direct Sales	Wilma Stone	45				
		Direct Sales	Nancy Carr	18				
		Direct Sales	Bill Muskrat	13				
	FM Lite	Mail Order	Jill Green	55				
		Mail Order	Jill Green	57	499	168413	75786	

Row 908: 57 Column 3: Profits Value: 75786

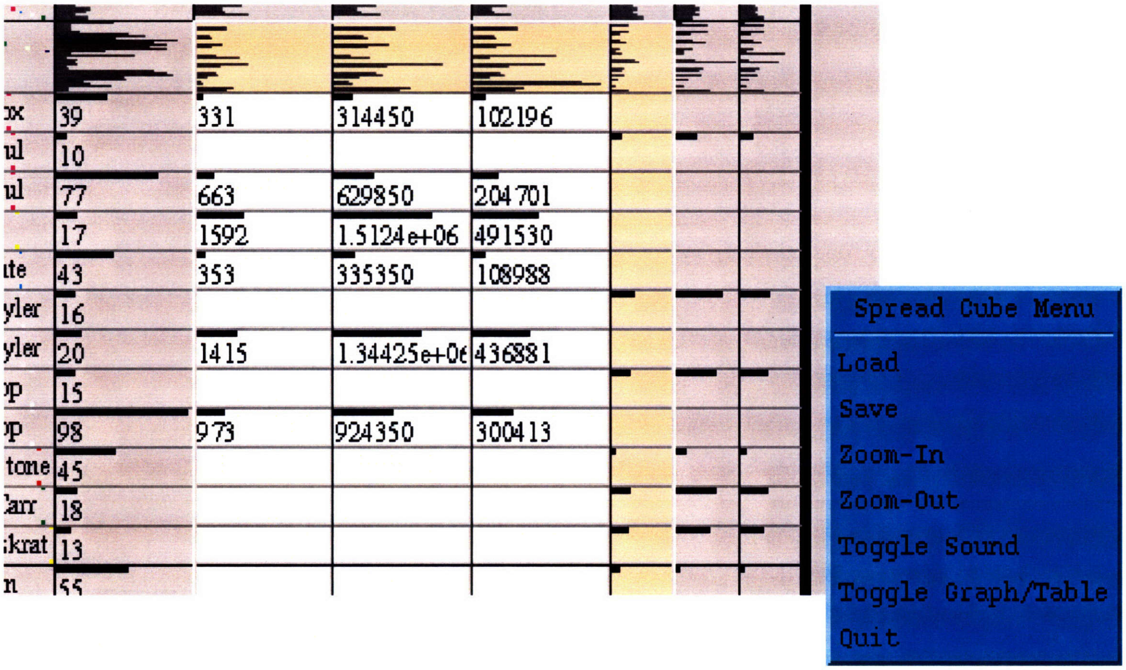
Figure A.16: After Applying the Zoom-In Operation to the Set on figure A.1

A.2 Color Plates for Chapter 6



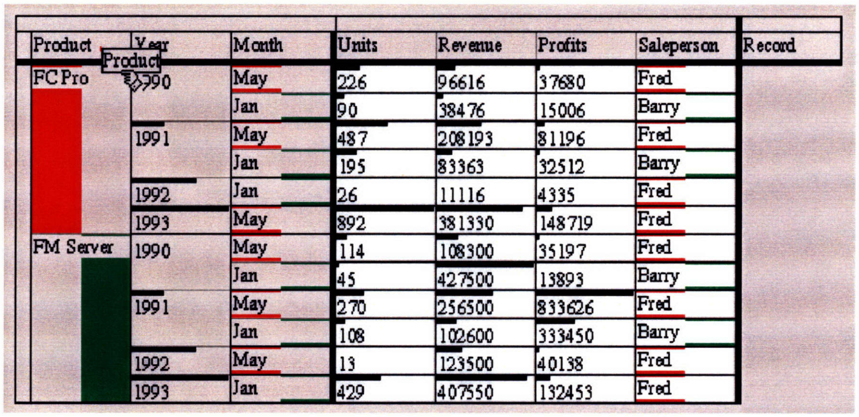
ıx	39	331	314450	102196
ul	10			
ul	77	663	629850	204701
	17	1592	1.5124e+06	491530
ıte	43	353	335350	108988
yler	16			
yler	20	1415	1.34425e+0	436881
ɔp	15			
ɔp	98	973	924350	300413
tone	45			
arr	18			
krat	13			
n	55			

Figure A.17: A Pop-up Menu



ıx	39	331	314450	102196
ul	10			
ul	77	663	629850	204701
	17	1592	1.5124e+06	491530
ıte	43	353	335350	108988
yler	16			
yler	20	1415	1.34425e+0	436881
ɔp	15			
ɔp	98	973	924350	300413
tone	45			
arr	18			
krat	13			
n	55			

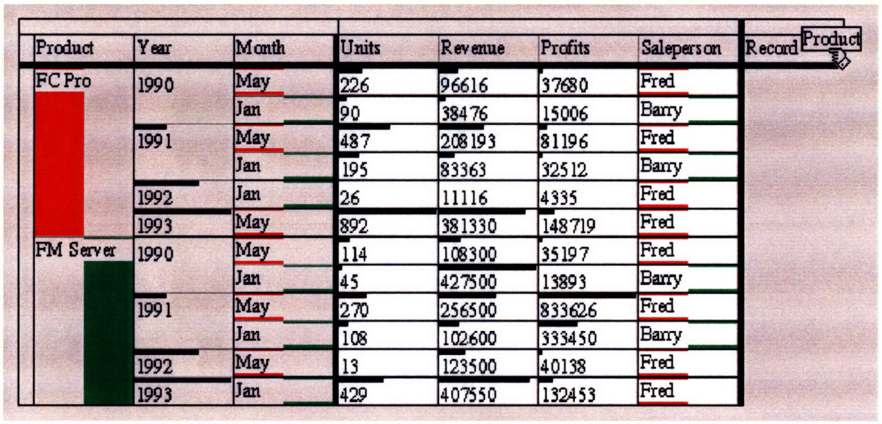
Figure A.18: Main and I/O Pop-up Menu



Product	Year	Month	Units	Revenue	Profits	Saleperson	Record
FC Pro	1990	May	226	96616	37680	Fred	
		Jan	90	38476	15006	Barry	
	1991	May	487	208193	81196	Fred	
		Jan	195	83363	32512	Barry	
	1992	Jan	26	11116	4335	Fred	
	1993	May	892	381330	148719	Fred	
FM Server	1990	May	114	108300	35197	Fred	
		Jan	45	427500	13893	Barry	
	1991	May	270	256500	833626	Fred	
		Jan	108	102600	333450	Barry	
	1992	May	13	123500	40138	Fred	
	1993	Jan	429	407550	132453	Fred	

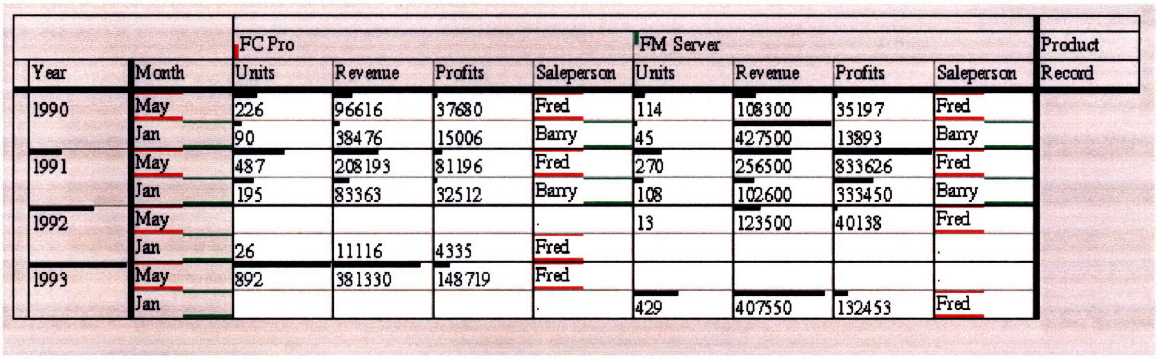
Drag

+



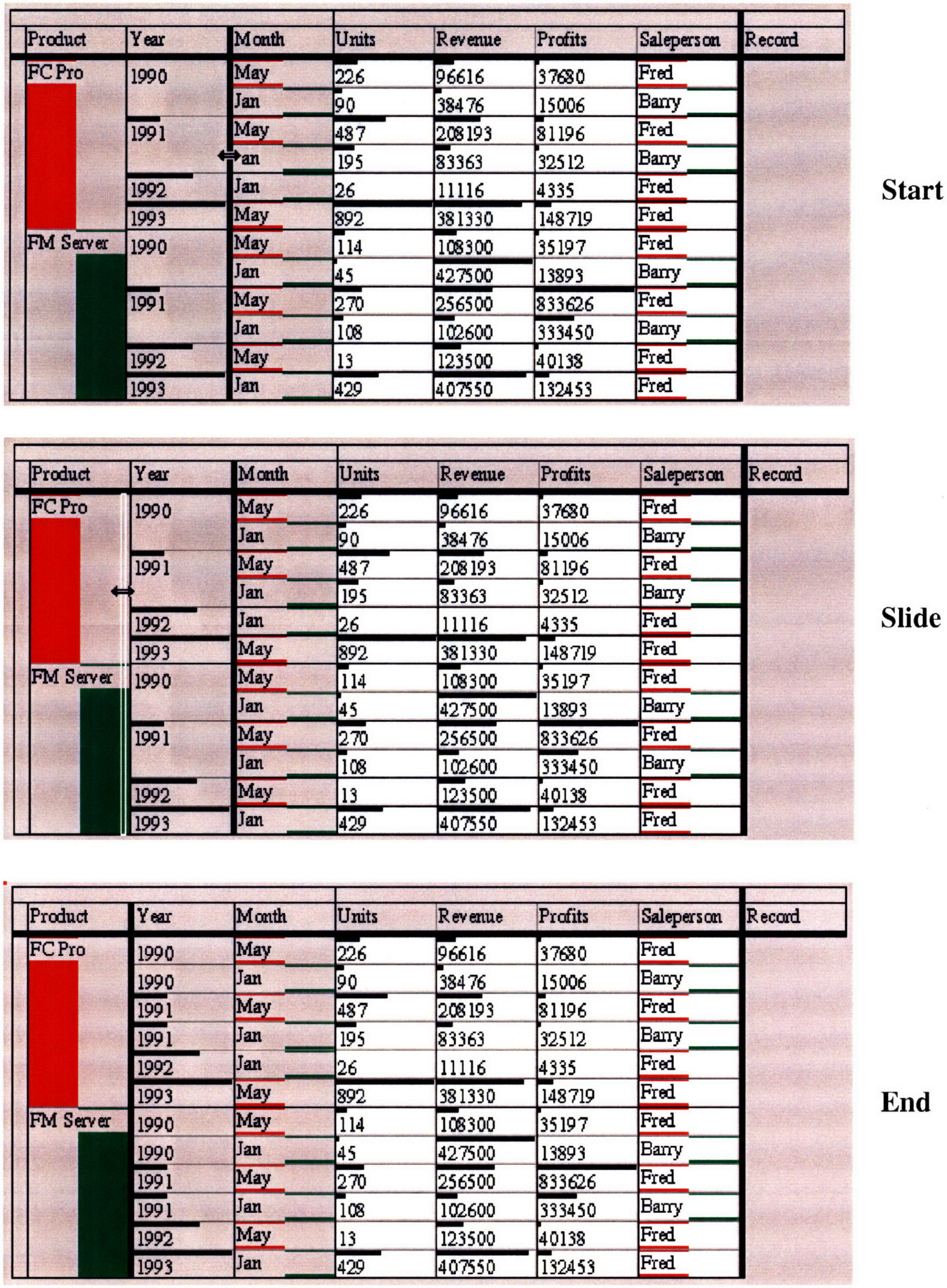
Product	Year	Month	Units	Revenue	Profits	Saleperson	Record
FC Pro	1990	May	226	96616	37680	Fred	
		Jan	90	38476	15006	Barry	
	1991	May	487	208193	81196	Fred	
		Jan	195	83363	32512	Barry	
	1992	Jan	26	11116	4335	Fred	
	1993	May	892	381330	148719	Fred	
FM Server	1990	May	114	108300	35197	Fred	
		Jan	45	427500	13893	Barry	
	1991	May	270	256500	833626	Fred	
		Jan	108	102600	333450	Barry	
	1992	May	13	123500	40138	Fred	
	1993	Jan	429	407550	132453	Fred	

Drop



		FC Pro				FM Server				Product
Year	Month	Units	Revenue	Profits	Saleperson	Units	Revenue	Profits	Saleperson	Record
1990	May	226	96616	37680	Fred	114	108300	35197	Fred	
	Jan	90	38476	15006	Barry	45	427500	13893	Barry	
1991	May	487	208193	81196	Fred	270	256500	833626	Fred	
	Jan	195	83363	32512	Barry	108	102600	333450	Barry	
1992	May				.	13	123500	40138	Fred	
	Jan	26	11116	4335	Fred				.	
1993	May	892	381330	148719	Fred				.	
	Jan				.	429	407550	132453	Fred	

Figure A.19: Transposition Done by "Drag-and-Drop"



Start

Product	Year	Month	Units	Revenue	Profits	Saleperson	Record
FC Pro	1990	May	226	96616	37680	Fred	
		Jan	90	38476	15006	Barry	
	1991	May	487	208193	81196	Fred	
		an	195	83363	32512	Barry	
	1992	Jan	26	11116	4335	Fred	
	1993	May	892	381330	148719	Fred	
FM Server	1990	May	114	108300	35197	Fred	
		Jan	45	427500	13893	Barry	
	1991	May	270	256500	833626	Fred	
		Jan	108	102600	333450	Barry	
	1992	May	13	123500	40138	Fred	
	1993	Jan	429	407550	132453	Fred	

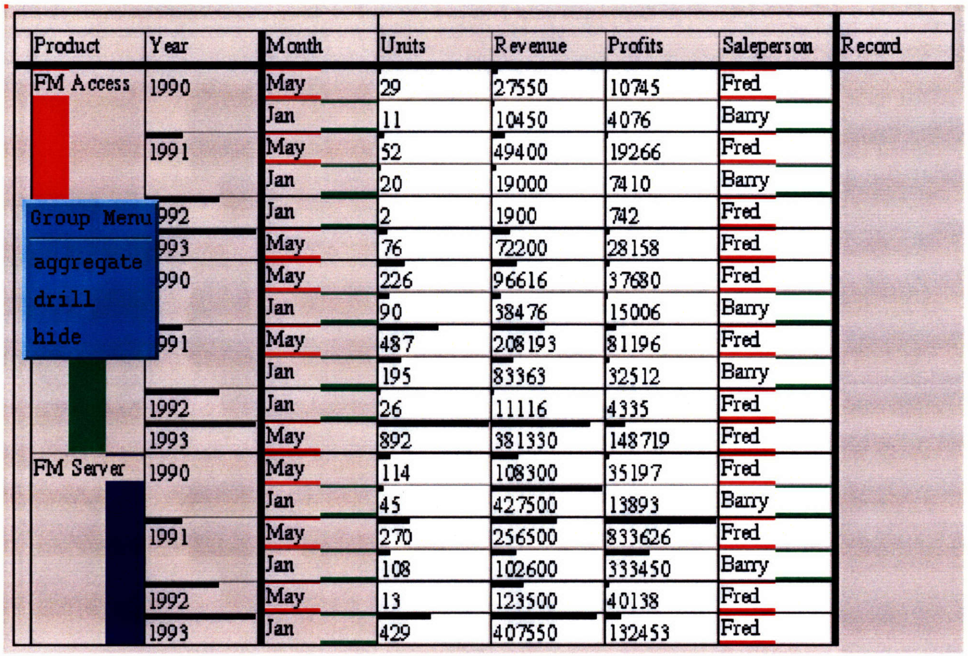
Slide

Product	Year	Month	Units	Revenue	Profits	Saleperson	Record
FC Pro	1990	May	226	96616	37680	Fred	
		Jan	90	38476	15006	Barry	
	1991	May	487	208193	81196	Fred	
		Jan	195	83363	32512	Barry	
	1992	Jan	26	11116	4335	Fred	
	1993	May	892	381330	148719	Fred	
FM Server	1990	May	114	108300	35197	Fred	
		Jan	45	427500	13893	Barry	
	1991	May	270	256500	833626	Fred	
		Jan	108	102600	333450	Barry	
	1992	May	13	123500	40138	Fred	
	1993	Jan	429	407550	132453	Fred	

End

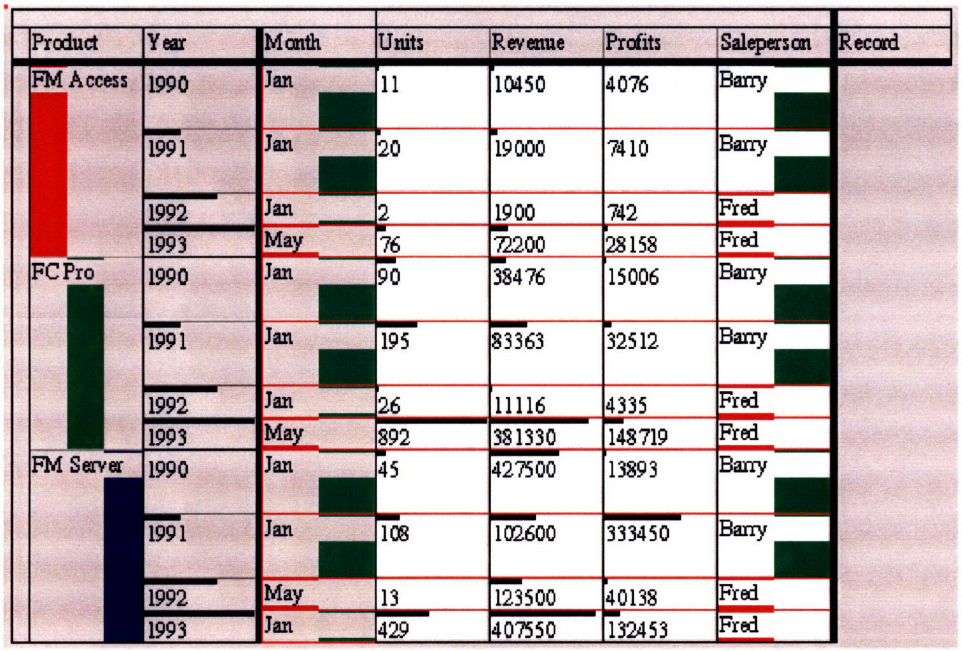
Product	Year	Month	Units	Revenue	Profits	Saleperson	Record
FC Pro	1990	May	226	96616	37680	Fred	
	1990	Jan	90	38476	15006	Barry	
	1991	May	487	208193	81196	Fred	
	1991	Jan	195	83363	32512	Barry	
	1992	Jan	26	11116	4335	Fred	
	1993	May	892	381330	148719	Fred	
FM Server	1990	May	114	108300	35197	Fred	
	1990	Jan	45	427500	13893	Barry	
	1991	May	270	256500	833626	Fred	
	1991	Jan	108	102600	333450	Barry	
	1992	May	13	123500	40138	Fred	
	1993	Jan	429	407550	132453	Fred	

Figure A.20: Sliding a Merge Mark



Product	Year	Month	Units	Revenue	Profits	Saleperson	Record
FM Access	1990	May	29	27550	10745	Fred	
		Jan	11	10450	4076	Barry	
	1991	May	52	49400	19266	Fred	
		Jan	20	19000	7410	Barry	
	992	Jan	2	1900	742	Fred	
	993	May	76	72200	28158	Fred	
	990	May	226	96616	37680	Fred	
		Jan	90	38476	15006	Barry	
	991	May	487	208193	81196	Fred	
		Jan	195	83363	32512	Barry	
	1992	Jan	26	11116	4335	Fred	
	1993	May	892	381330	148719	Fred	
FM Server	1990	May	114	108300	35197	Fred	
		Jan	45	427500	13893	Barry	
	1991	May	270	256500	833626	Fred	
		Jan	108	102600	333450	Barry	
	1992	May	13	123500	40138	Fred	
	1993	Jan	429	407550	132453	Fred	

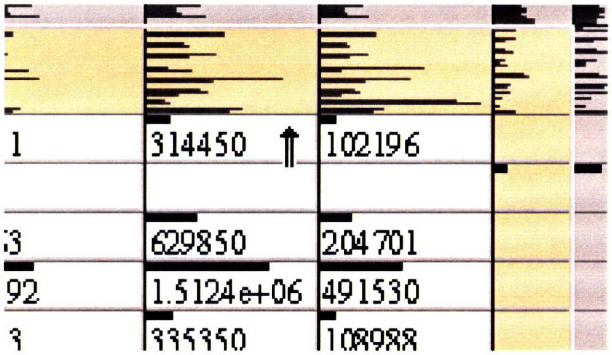
Aggregate



Product	Year	Month	Units	Revenue	Profits	Saleperson	Record
FM Access	1990	Jan	11	10450	4076	Barry	
	1991	Jan	20	19000	7410	Barry	
	1992	Jan	2	1900	742	Fred	
	1993	May	76	72200	28158	Fred	
FC Pro	1990	Jan	90	38476	15006	Barry	
	1991	Jan	195	83363	32512	Barry	
	1992	Jan	26	11116	4335	Fred	
	1993	May	892	381330	148719	Fred	
FM Server	1990	Jan	45	427500	13893	Barry	
	1991	Jan	108	102600	333450	Barry	
	1992	May	13	123500	40138	Fred	
	1993	Jan	429	407550	132453	Fred	

Aggregated

Figure A.21: Aggregating A Group



1	314450	102196
3	629850	204701
92	1.5124e+06	491530
3	335350	108988

Figure A.22: Clicking the First Mouse Button Moves the Focal Area Up.

Appendix B

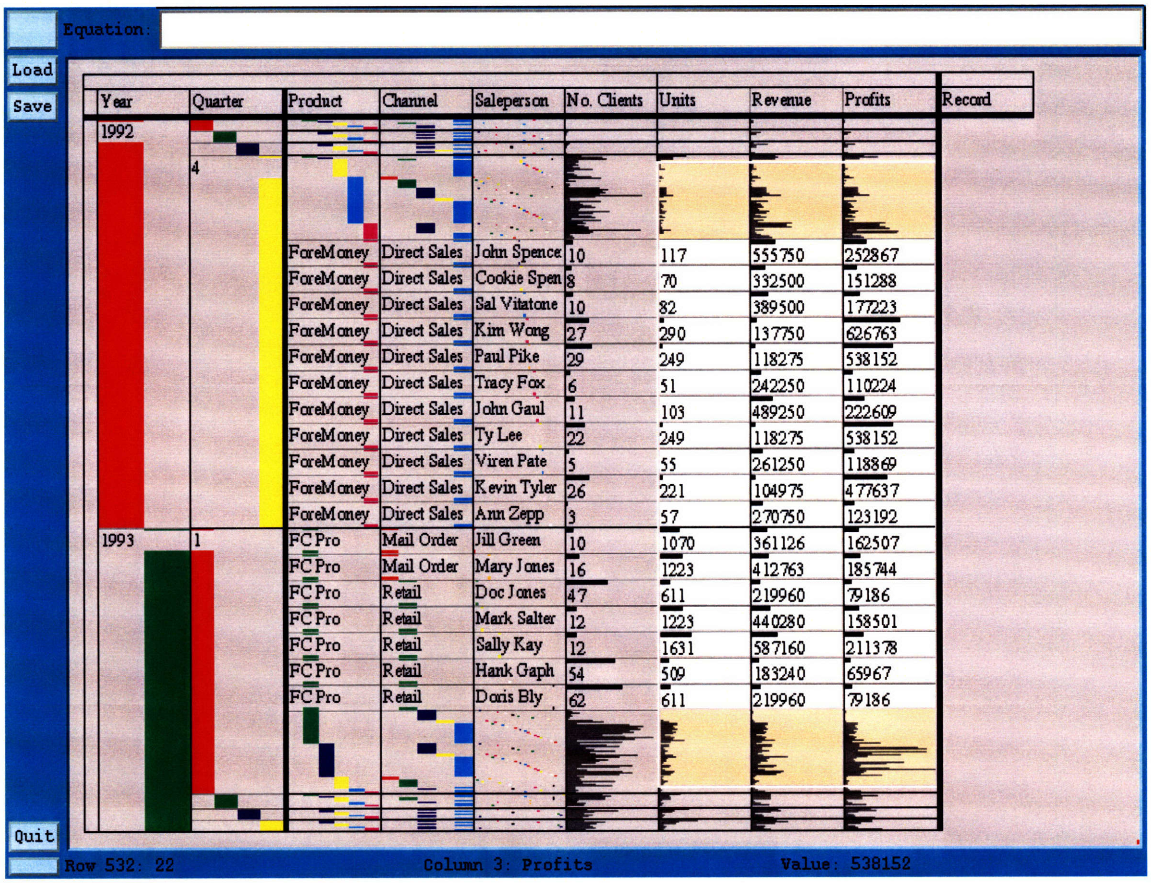
Case Studies

In this appendix we will provide three extended examples of using the SpreadCube data analysis tool on three essentially different data sets and discovering facts about them.

B.1 Marketing Data

This example illustrates handling highly dimensional data.

For this example, we will use the marketing dynamics data set which was already introduced in previous chapters (see figure B.1).

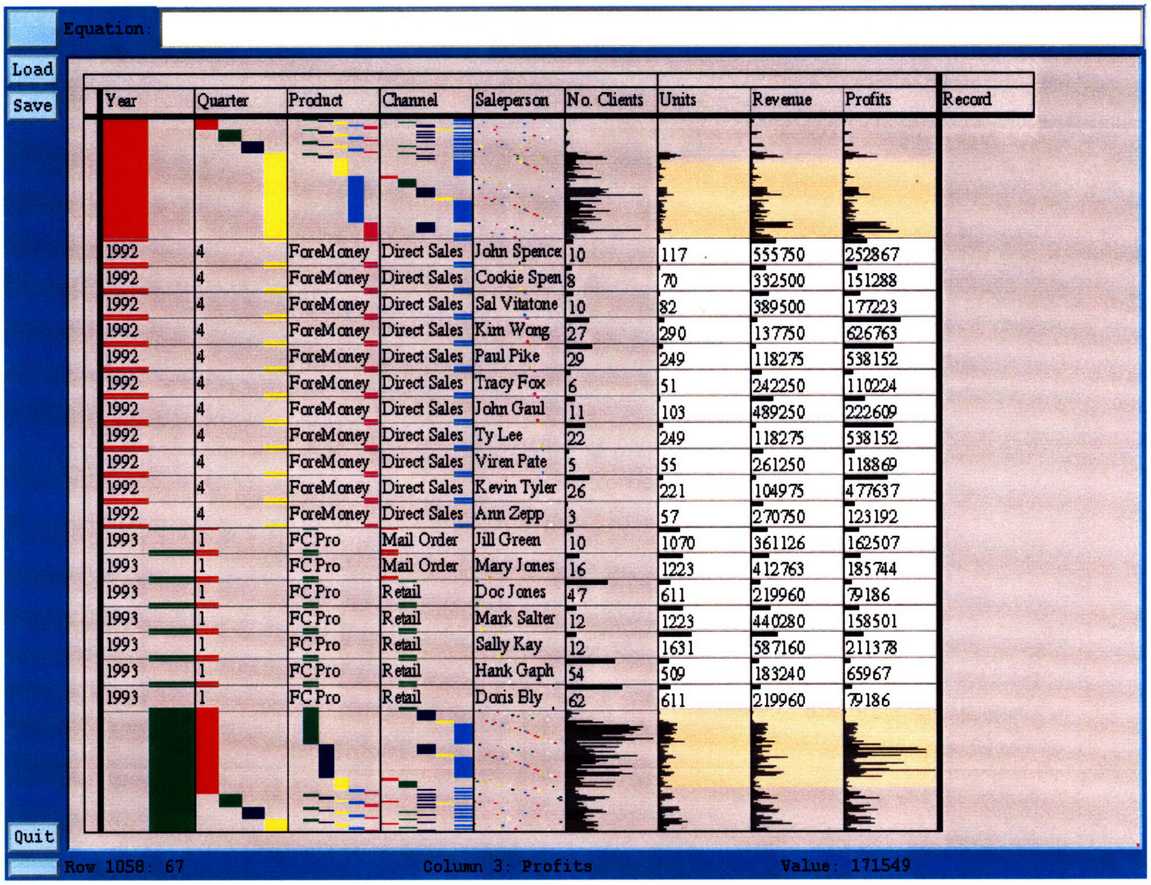


Year	Quarter	Product	Channel	Saleperson	No. Clients	Units	Revenue	Profits	Record
1992	4	ForeMoney	Direct Sales	John Spence	10	117	555750	252867	
		ForeMoney	Direct Sales	Cookie Spen	8	70	332500	151288	
		ForeMoney	Direct Sales	Sal Vitatone	10	82	389500	177223	
		ForeMoney	Direct Sales	Kim Wong	27	290	137750	626763	
		ForeMoney	Direct Sales	Paul Pike	29	249	118275	538152	
		ForeMoney	Direct Sales	Tracy Fox	6	51	242250	110224	
		ForeMoney	Direct Sales	John Gaul	11	103	489250	222609	
		ForeMoney	Direct Sales	Ty Lee	22	249	118275	538152	
		ForeMoney	Direct Sales	Viren Pate	5	55	261250	118869	
		ForeMoney	Direct Sales	Kevin Tyler	26	221	104975	477637	
		ForeMoney	Direct Sales	Ann Zepp	3	57	270750	123192	
1993	1	FC Pro	Mail Order	Jill Green	10	1070	361126	162507	
		FC Pro	Mail Order	Mary Jones	16	1223	412763	185744	
		FC Pro	Retail	Doc Jones	47	611	219960	79186	
		FC Pro	Retail	Mark Salter	12	1223	440280	158501	
		FC Pro	Retail	Sally Kay	12	1631	587160	211378	
		FC Pro	Retail	Hank Gaph	54	509	183240	65967	
		FC Pro	Retail	Doris Bly	62	611	219960	79186	

Figure B.1: Initial data set.

This data set is a modified version of the one created by Ramana Rao as a test case for the TableLens [2]. It consists of six set dimensions: "Year", "Quarter", "Product", "Channel", "Saleperson" and "No. Clients" and three quantity variables: "Units", "Revenue" and "Profits".

Suppose we are interested in investigating what the relationship between "Units" and "Profits" is and what factors affect it. We are not sure what exactly we should be looking for, but our intuition is that such relationship should not depend on the particular year or quarter, so we merge those two dimensions along with the others: (figure B.2)



Year	Quarter	Product	Channel	Saleperson	No. Clients	Units	Revenue	Profits	Record
1992	4	ForeMoney	Direct Sales	John Spence	10	117	555750	252867	
1992	4	ForeMoney	Direct Sales	Cookie Spen	8	70	332500	151288	
1992	4	ForeMoney	Direct Sales	Sal Vitatone	10	82	389500	177223	
1992	4	ForeMoney	Direct Sales	Kim Wong	27	290	137750	626763	
1992	4	ForeMoney	Direct Sales	Paul Pike	29	249	118275	538152	
1992	4	ForeMoney	Direct Sales	Tracy Fox	6	51	242250	110224	
1992	4	ForeMoney	Direct Sales	John Gaul	11	103	489250	222609	
1992	4	ForeMoney	Direct Sales	Ty Lee	22	249	118275	538152	
1992	4	ForeMoney	Direct Sales	Viren Pate	5	55	261250	118869	
1992	4	ForeMoney	Direct Sales	Kevin Tyler	26	221	104975	477637	
1992	4	ForeMoney	Direct Sales	Ann Zepp	3	57	270750	123192	
1993	1	FC Pro	Mail Order	Jill Green	10	1070	361126	162507	
1993	1	FC Pro	Mail Order	Mary Jones	16	1223	412763	185744	
1993	1	FC Pro	Retail	Doc Jones	47	611	219960	79186	
1993	1	FC Pro	Retail	Mark Salter	12	1223	440280	158501	
1993	1	FC Pro	Retail	Sally Kay	12	1631	587160	211378	
1993	1	FC Pro	Retail	Hank Gaph	54	509	183240	65967	
1993	1	FC Pro	Retail	Doris Bly	62	611	219960	79186	

Row 1058: 67   Column 3: Profits   Value: 171549

Figure B.2: Merging the "Year" and "Quarter" dimensions.

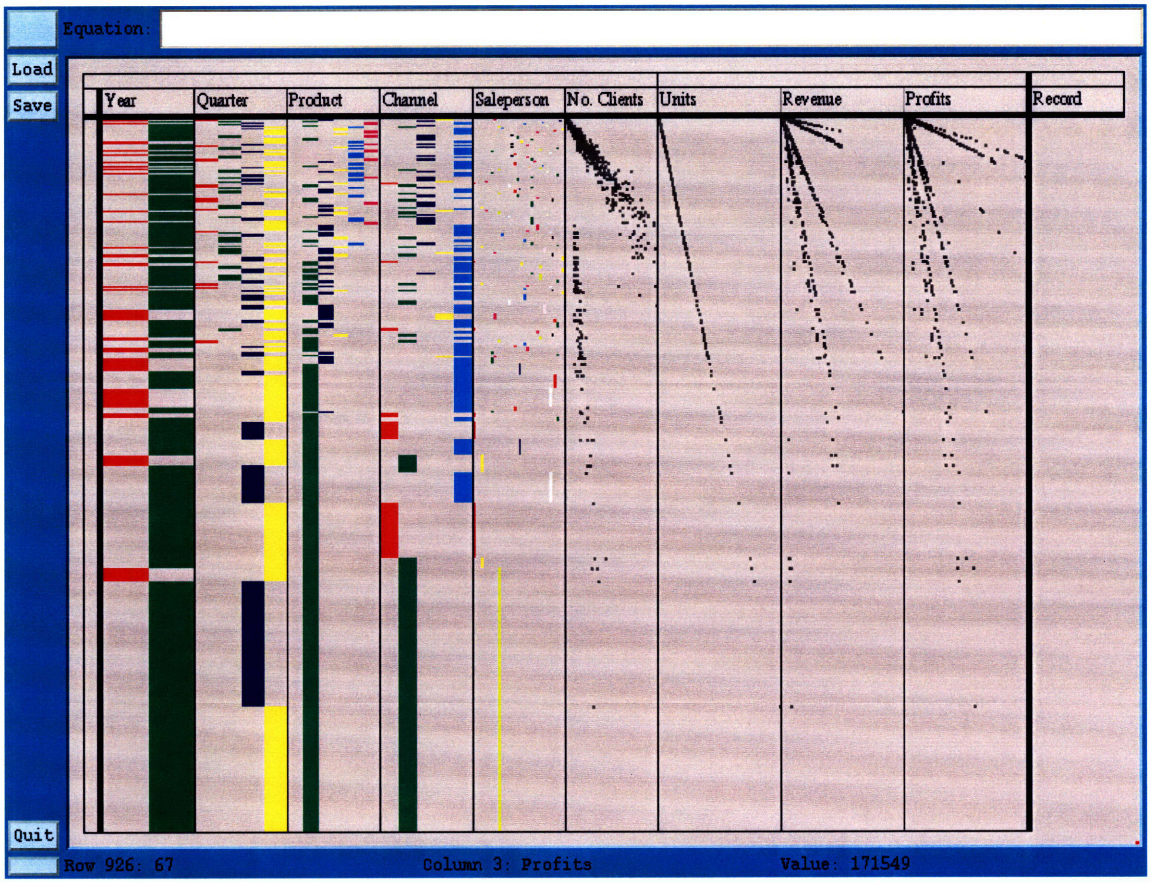
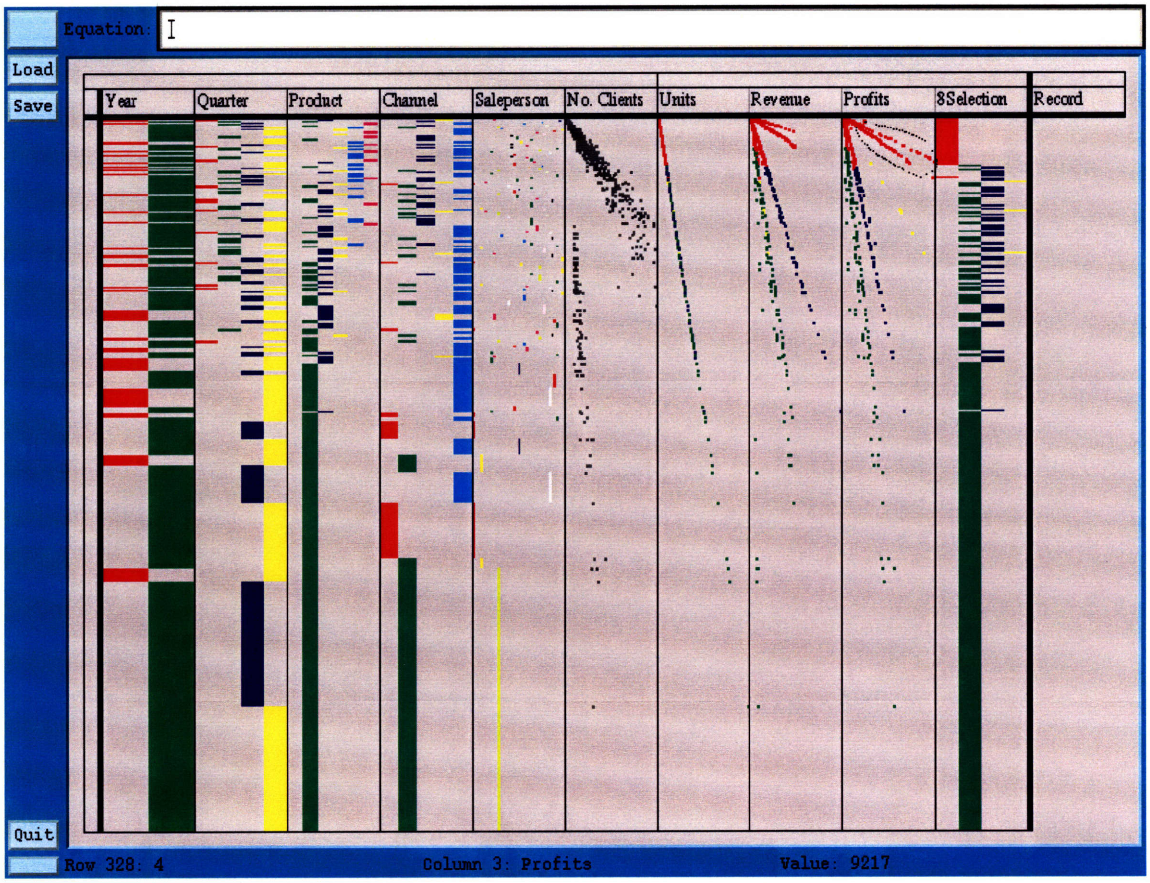


Figure B.3: Spatial modulation using the "Units" variable

Then we switch to graph mode and use the "Units" variable for spatial modulation. As result we obtain the graphs of the other two variables with respect to the "Units" variable. (figure B.3).

From these graphs, we notice that "Profits" grow proportionally with "Units", but there are several constants of proportionality involved. This suggest that we may break up the "Profits" graph into approximately four regions where each region corresponds to different rate profit growth.

After making the first selection, a new selection variable "8Selection" was created and used for variable modulation (figure B.4).



Row 328: 4

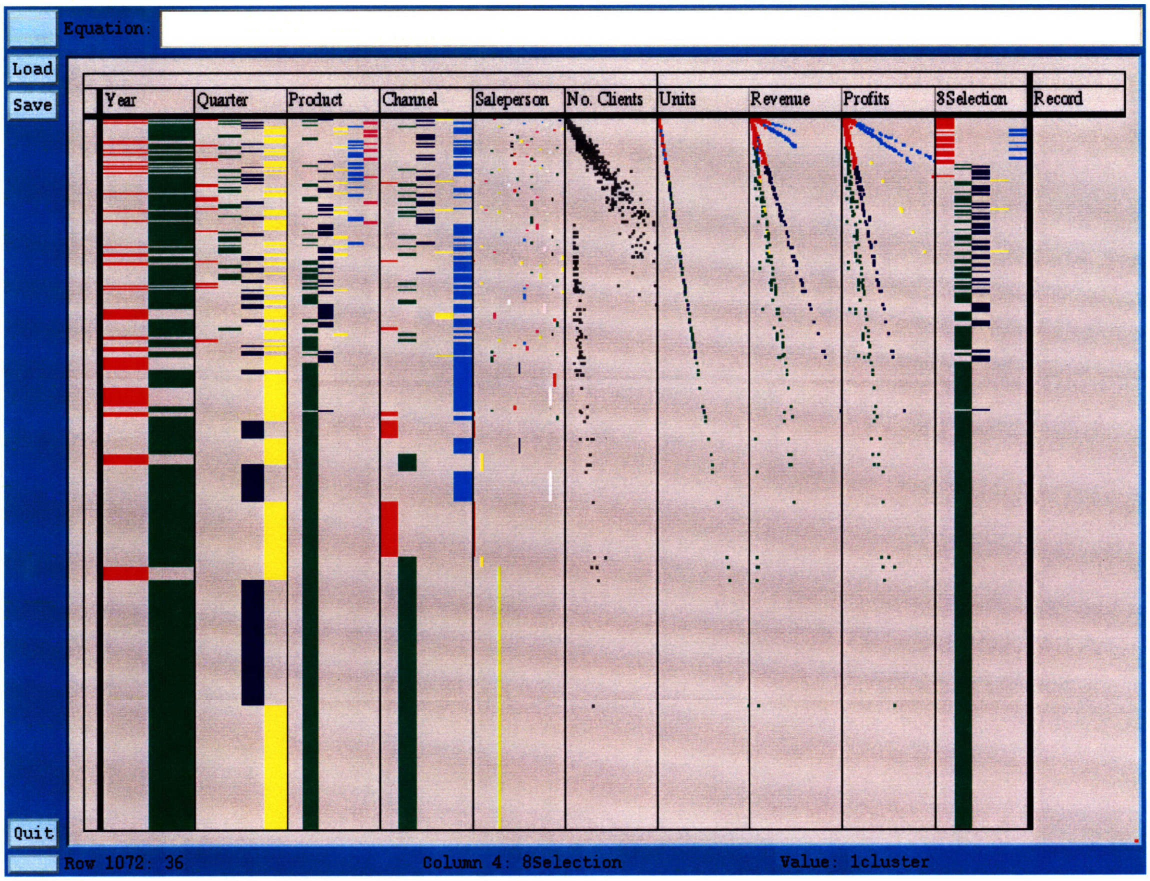
Column 3: Profits

Value: 9217

Figure B.4: Making the fourth selection on the "Profits" versus "Units" graph.

The situation displayed on figure B.4 is after making three consecutive selections and we are about to make the fourth selection using the same selection variable as the output.

Each selection was made by drawing a loop while pressing the middle mouse button. The loop for the fourth selection is displayed on figure B.4. The result after completing the selection is shown on figure B.5.



Row 1072: 36   Column 4: 8Selection   Value: 1cluster

Figure B.5: After selecting four regions of the "Profits" versus "Units" graph.

At this point, we decide to break up the table according to the selected regions. We do that by promoting the selection variable. The result is shown on figure B.6.

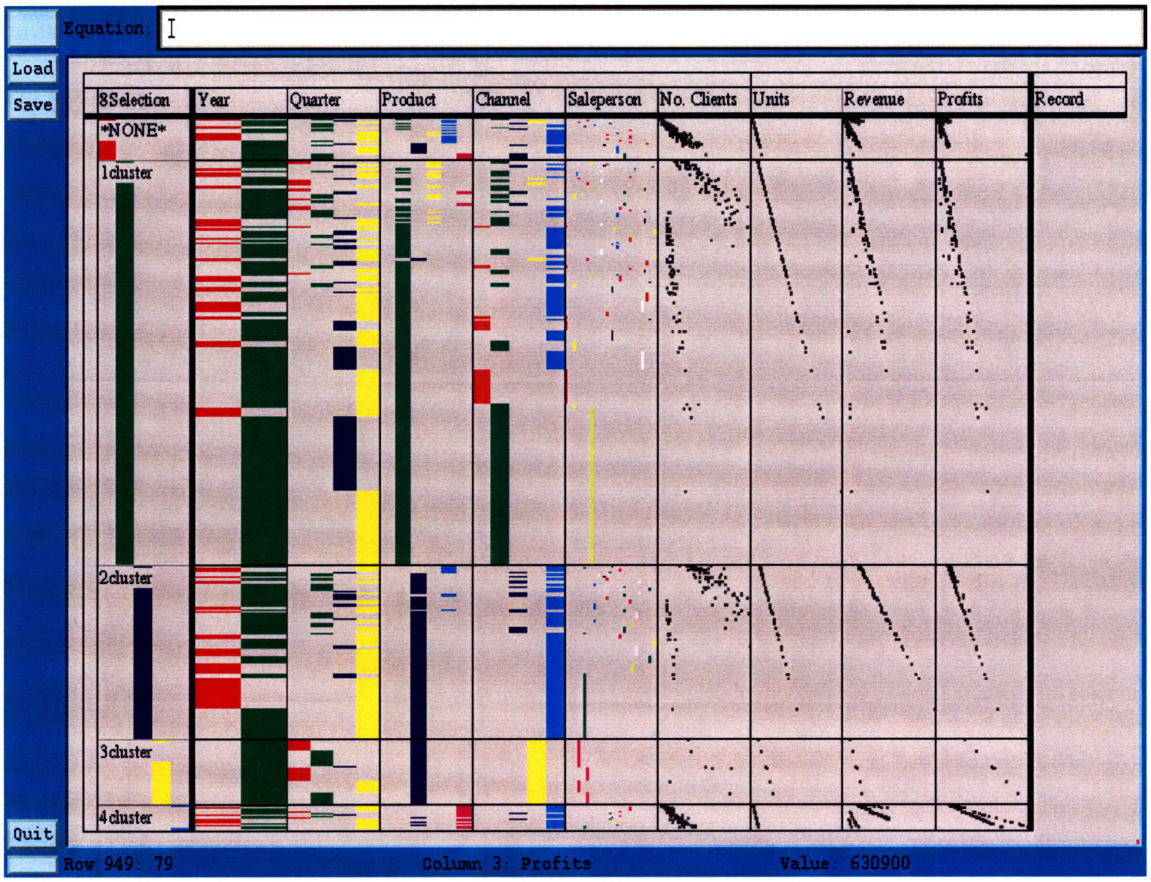


Figure B.6: Promoting the selection variable.

The four selections that we have made correspond to dimension keys: "cluster1" through "cluster4". The points which we did not select correspond to the "*None*" key. Compare figure B.5 with figure B.6 and notice how in figure B.6 it appears that we have taken pieces of the graphs on figure B.5 and, without disturbing those pieces, we have spread them out on the screen surface, so that each of them is a separate graph by itself.

We are now interested in looking closer at what has caused the high profits/units ratio in the case of "cluster4", so we slice through the key "cluster4" of the selection variable dimension. The result is shown on figure B.7.

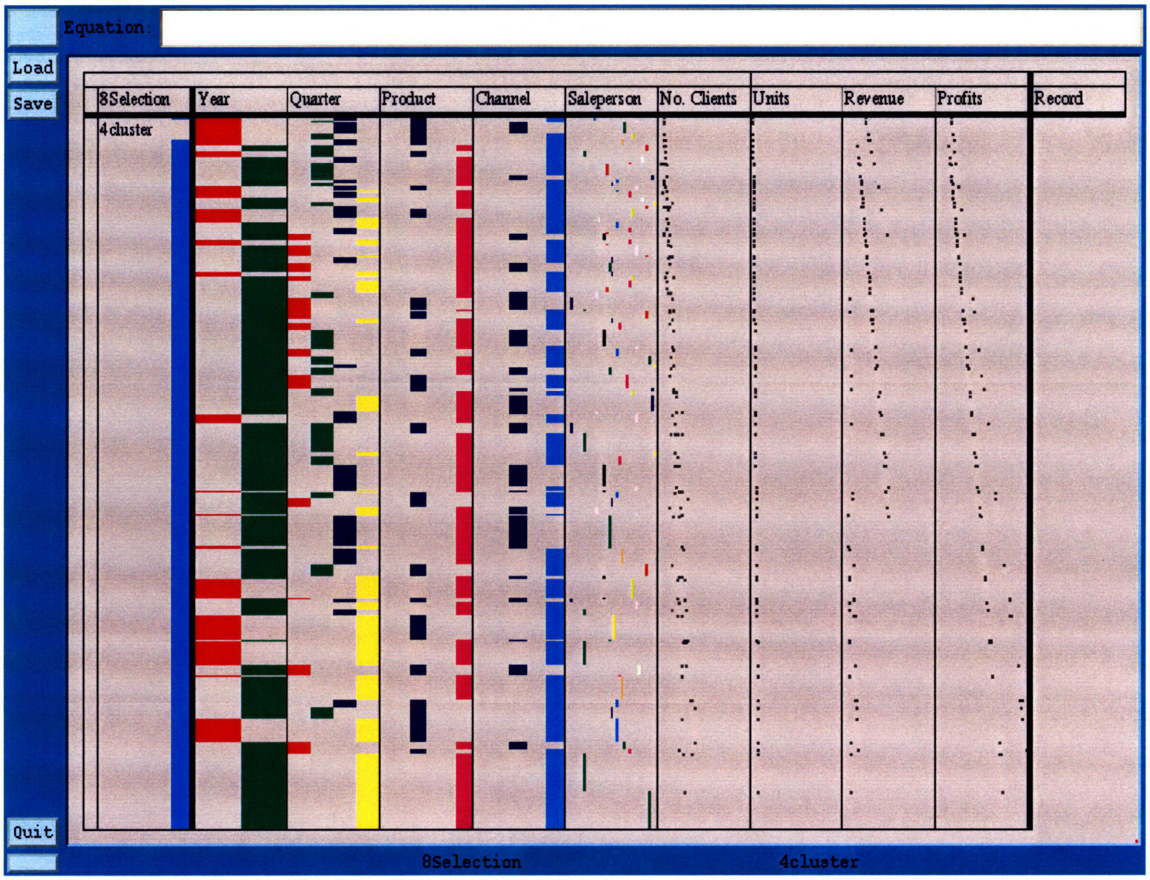
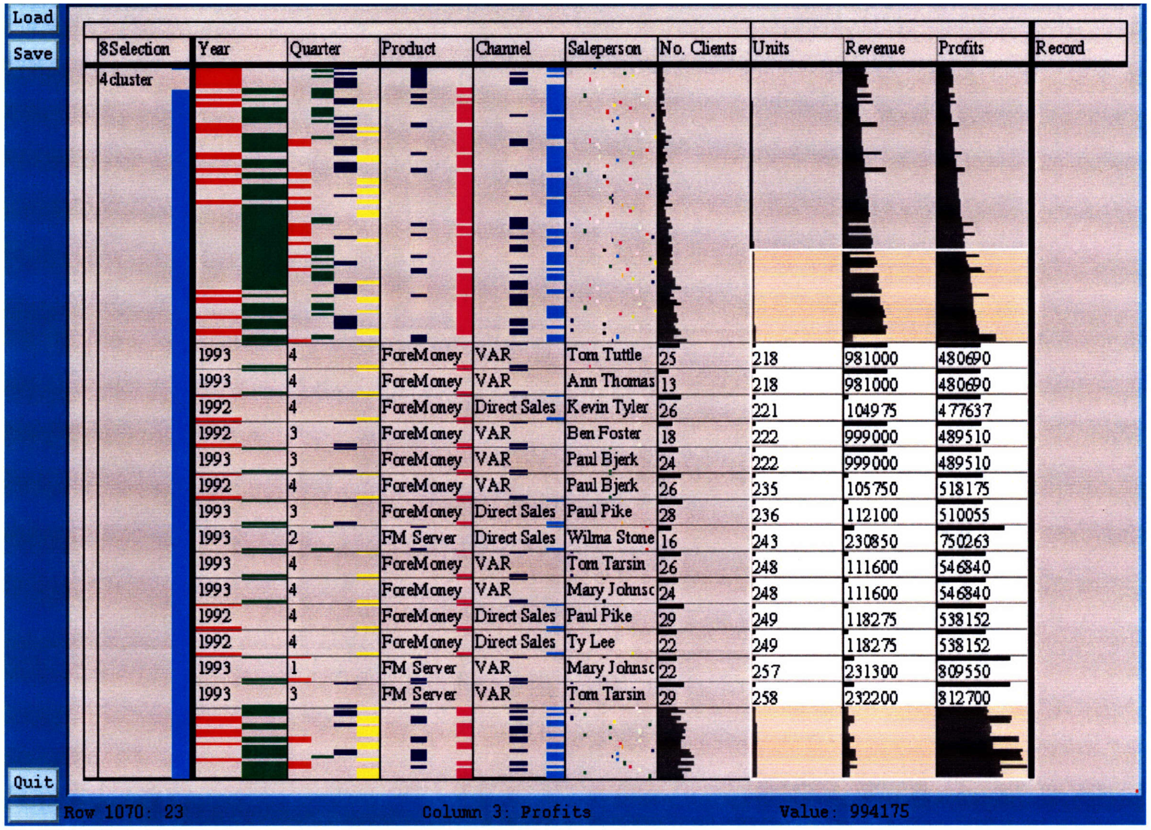


Figure B.7: Slicing through the fourth selection.

From figure B.7 we can immediately conclude that the high profits to units ratio was due to two out of the four products and the fact that they were sold on two out of the four possible channels.

We now switch back to table mode and open a focal area to see in detail what those products and channels are. The result is shown on figure B.8.

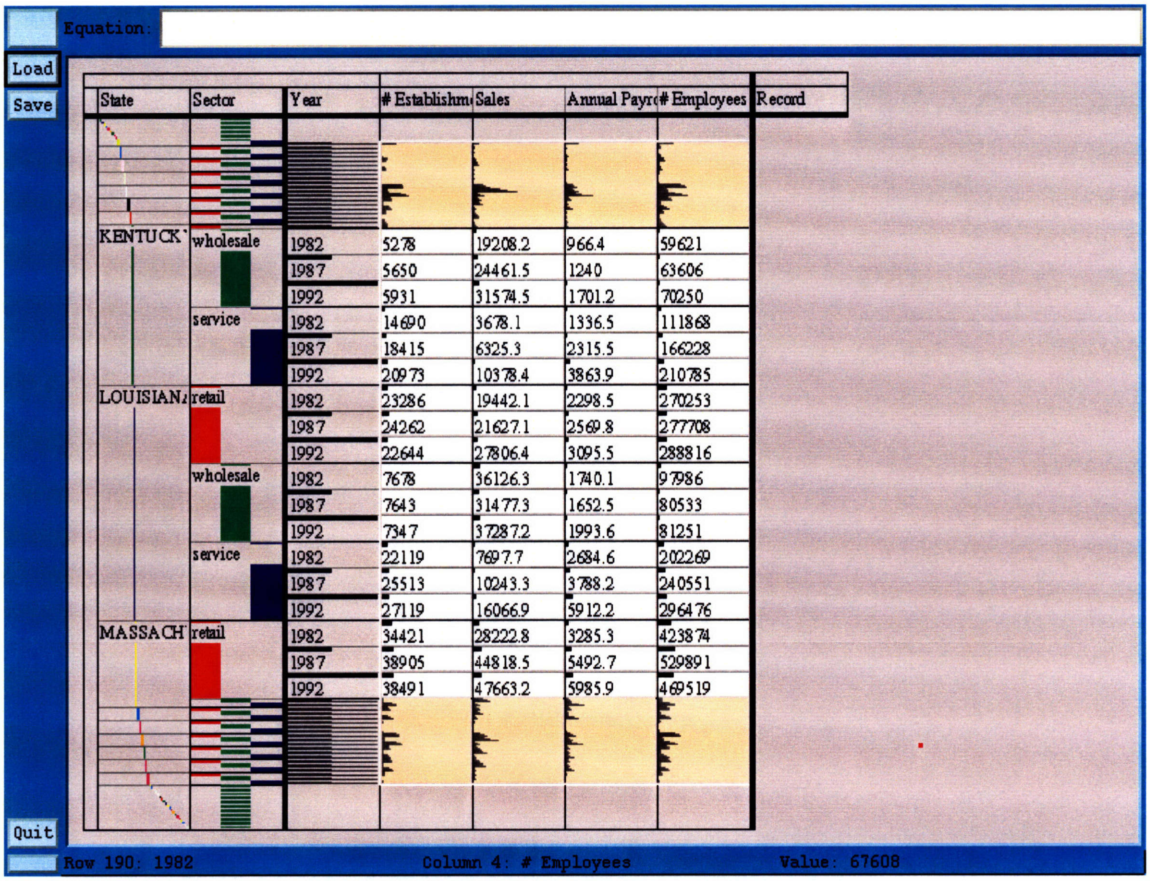


8Selection	Year	Quarter	Product	Channel	Saleperson	No. Clients	Units	Revenue	Profits	Record
	1993	4	ForeMoney	VAR	Tom Tuttle	25	218	981000	480690	
	1993	4	ForeMoney	VAR	Ann Thomas	13	218	981000	480690	
	1992	4	ForeMoney	Direct Sales	Kevin Tyler	26	221	104975	477637	
	1992	3	ForeMoney	VAR	Ben Foster	18	222	999000	489510	
	1993	3	ForeMoney	VAR	Paul Bjerk	24	222	999000	489510	
	1992	4	ForeMoney	VAR	Paul Bjerk	26	235	105750	518175	
	1993	3	ForeMoney	Direct Sales	Paul Pike	28	236	112100	510055	
	1993	2	FM Server	Direct Sales	Wilma Stone	16	243	230850	750263	
	1993	4	ForeMoney	VAR	Tom Tarsin	26	248	111600	546840	
	1993	4	ForeMoney	VAR	Mary Johnsc	24	248	111600	546840	
	1992	4	ForeMoney	Direct Sales	Paul Pike	29	249	118275	538152	
	1992	4	ForeMoney	Direct Sales	Ty Lee	22	249	118275	538152	
	1993	1	FM Server	VAR	Mary Johnsc	22	257	231300	809550	
	1993	3	FM Server	VAR	Tom Tarsin	29	258	232200	812700	

Row 1070: 23 Column 3: Profits Value: 994175

Figure B.8: Back to table mode.

B.2 Census Data



State	Sector	Year	# Establishm	Sales	Annual Payr	# Employees	Record
KENTUCK	wholesale	1982	5278	19208.2	966.4	59621	
		1987	5650	24461.5	1240	63606	
		1992	5931	31574.5	1701.2	70250	
	service	1982	14690	3678.1	1336.5	111868	
		1987	18415	6325.3	2315.5	166228	
		1992	20973	10378.4	3863.9	210785	
LOUISIAN	retail	1982	23286	19442.1	2298.5	270253	
		1987	24262	21627.1	2569.8	277708	
		1992	22644	27806.4	3095.5	288816	
	wholesale	1982	7678	36126.3	1740.1	97986	
		1987	7643	31477.3	1652.5	80533	
		1992	7347	37287.2	1993.6	81251	
	service	1982	22119	7697.7	2684.6	202269	
		1987	25513	10243.3	3788.2	240551	
		1992	27119	16066.9	5912.2	296476	
MASSACH	retail	1982	34421	28222.8	3285.3	423874	
		1987	38905	44818.5	5492.7	529891	
		1992	38491	47663.2	5985.9	469519	

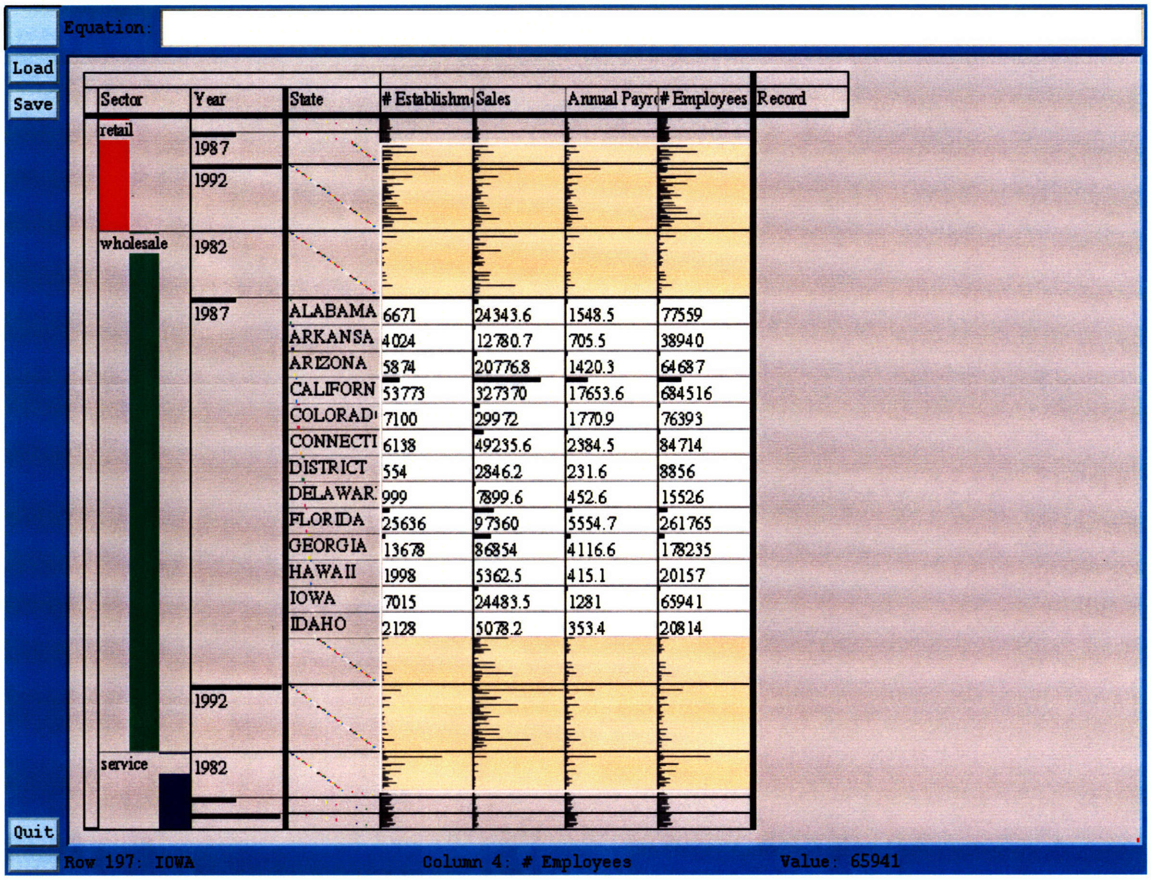
Row 190: 1982 Column 4: # Employees Value: 67608

Figure B.9: Initial data set

In this example, we illustrate the use of SpreadCube as a viewing tool. The data set which is used here is from the U.S.Census Bureau on State Economic Profiles. It can be found at the following URL:

`http://www.census.gov/epcd/www/prof92.ec.html`

There are three set dimensions: "State", "Sector" and "Year" and four variables: "# Establishments", "Sales" in millions of dollars, "Annual Payroll" in millions of dollars, and "# Employees" (figure B.9).



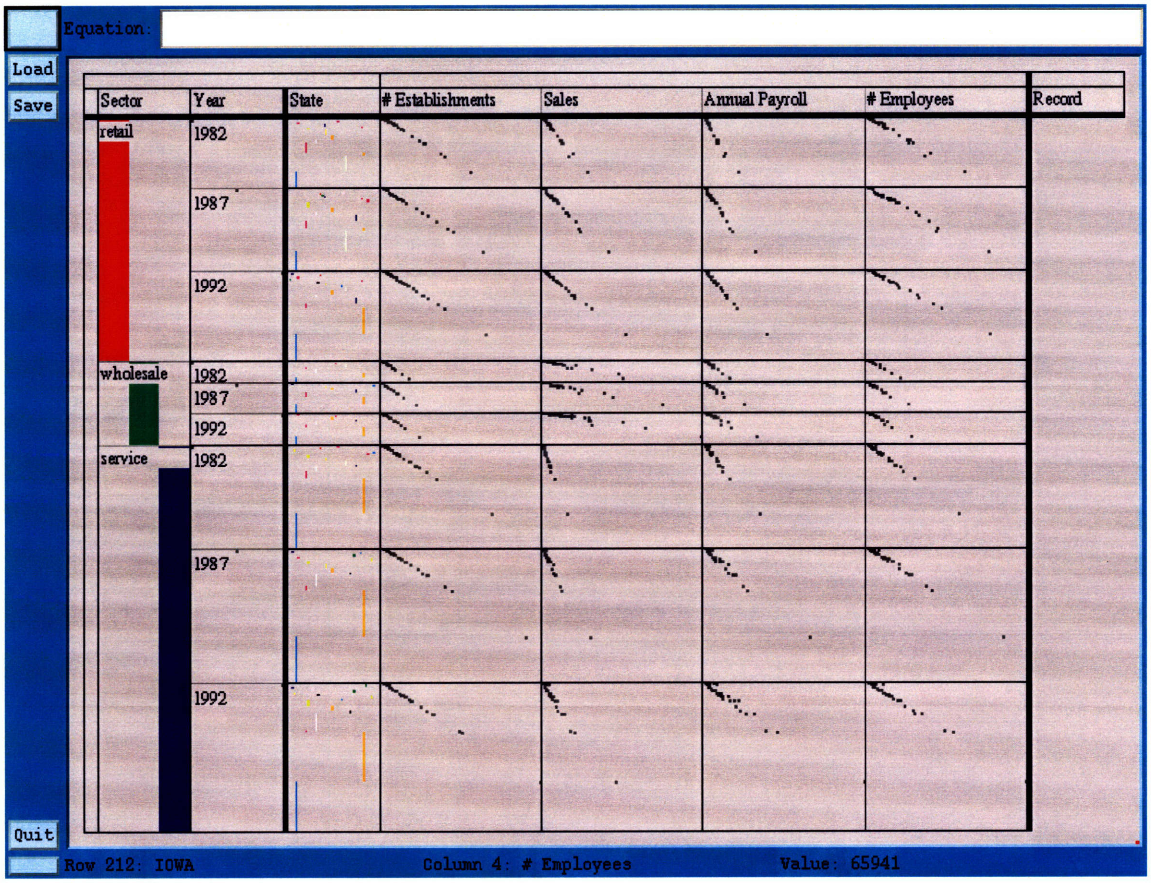
Sector	Year	State	# Establishm	Sales	Annual Payr	# Employees	Record
wholesale	1987	ALABAMA	6671	24343.6	1548.5	77559	
		ARKANSA	4024	12780.7	705.5	38940	
		ARIZONA	5874	20776.8	1420.3	64687	
		CALIFORN	53773	327370	17653.6	684516	
		COLORAD	7100	29972	1770.9	76393	
		CONNECTI	6138	49235.6	2384.5	84714	
		DISTRICT	554	2846.2	231.6	8856	
		DELAWAR	999	7899.6	452.6	15526	
		FLORIDA	25636	97360	5554.7	261765	
		GEORGIA	13678	86854	4116.6	178235	
		HAWAII	1998	5362.5	415.1	20157	
		IOWA	7015	24483.5	1281	65941	
		IDAHO	2128	5078.2	353.4	20814	

Row 197: IOWA Column 4: # Employees Value: 65941

Figure B.10: Transposition placing the "Sector" dimension at the top of the hierarchy.

First we would like to group the data by sectors so, we move the "Sector" dimension to the top of the dimensional hierarchy figure B.10.

Driven out of curiosity, we decide to graph all variables with respect to "# Establishments", so we switch into graph mode and use the "# Establishments" variable for spatial modulation. Perhaps one motivation was that "# Establishments" roughly corresponds to the size of the market so it is a nice variable to graph with respect to. The result is shown on figure B.11.



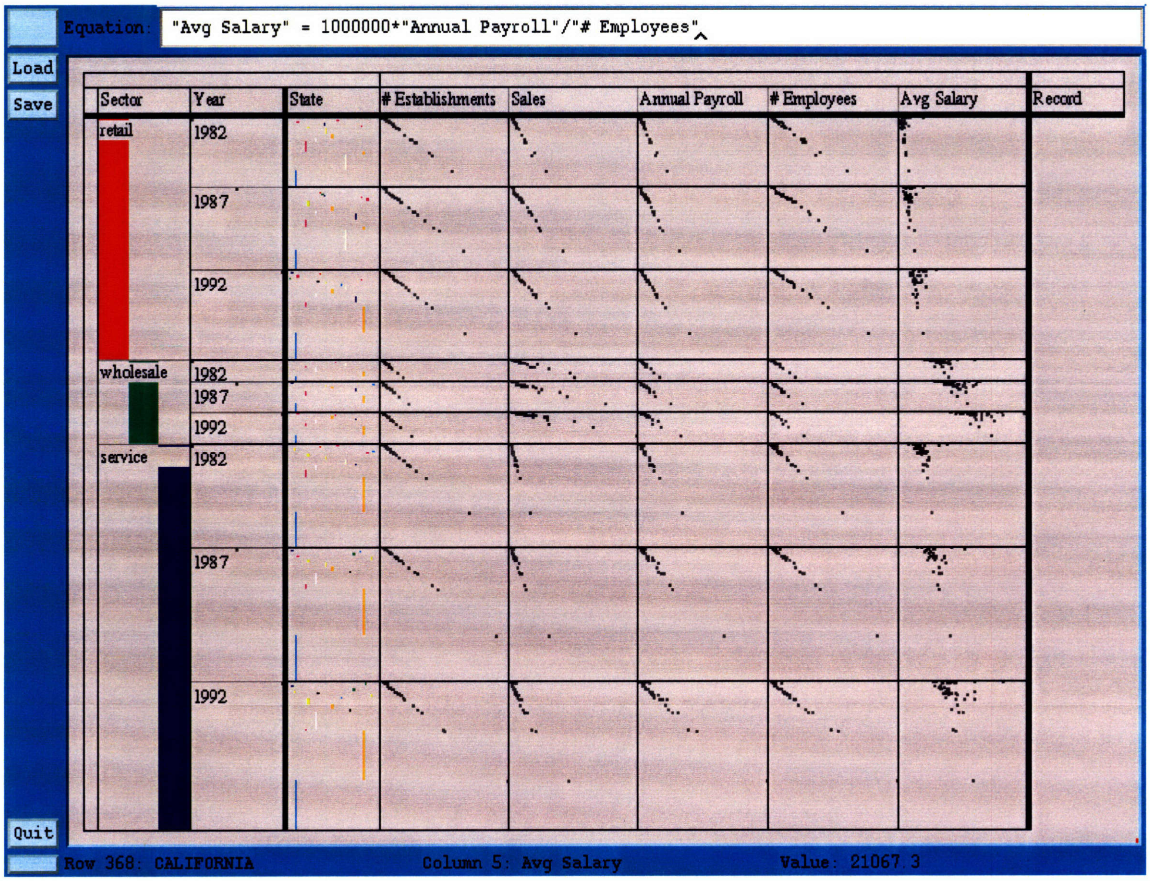
Row 212: IOWA   Column 4: # Employees   Value: 65941

Figure B.11: Spatial modulation by "# Establishments".

From figure B.11 we notice that variables appear to be roughly proportional to one another. We decide to test this hypothesis. In particular we decide to test the relationship between "# Employees" and "Annual Payroll", so we calculate a new variable corresponding to average salary:

```
"Avg Salary" = 1000000*"Annual Payroll"/"# Employees".
```

The reason for multiplying by 1000000 is that the data for "Annual Payroll" is given in units of millions. The result is shown on figure B.12.



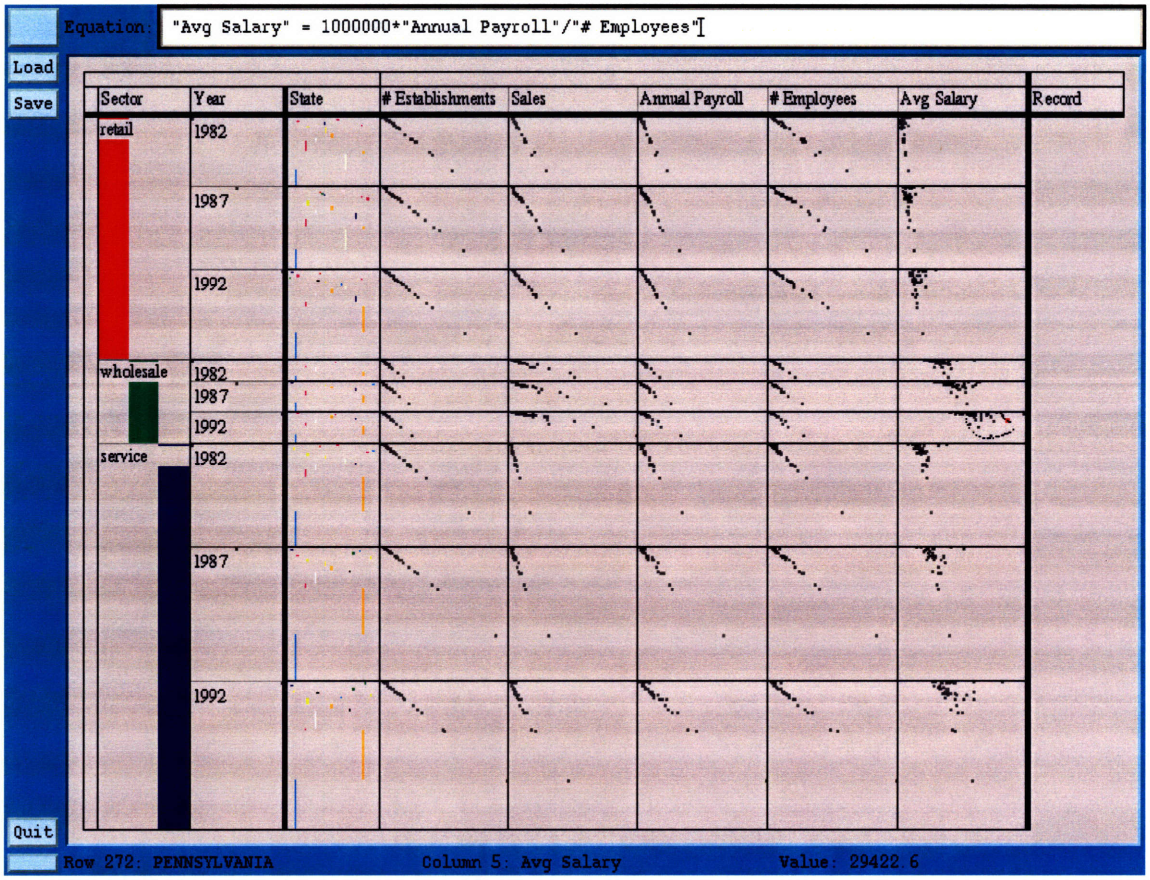
Equation: "Avg Salary" = 1000000*"Annual Payroll"/"# Employees"

Row 368: CALIFORNIA Column 5: Avg Salary Value: 21067.3

Figure B.12: Computing the "Avg. Salary" variable.

From figure B.12 we conclude that indeed average salary is roughly constant with respect to, or at least independent of the number of establishments.

On the other hand, we can see a definite linear trend of average salary increase over time - a piece of information which we did not seek intentionally. Furthermore, we also notice that the bulk of highest average salaries is in the wholesale sector, so focus on that cluster.

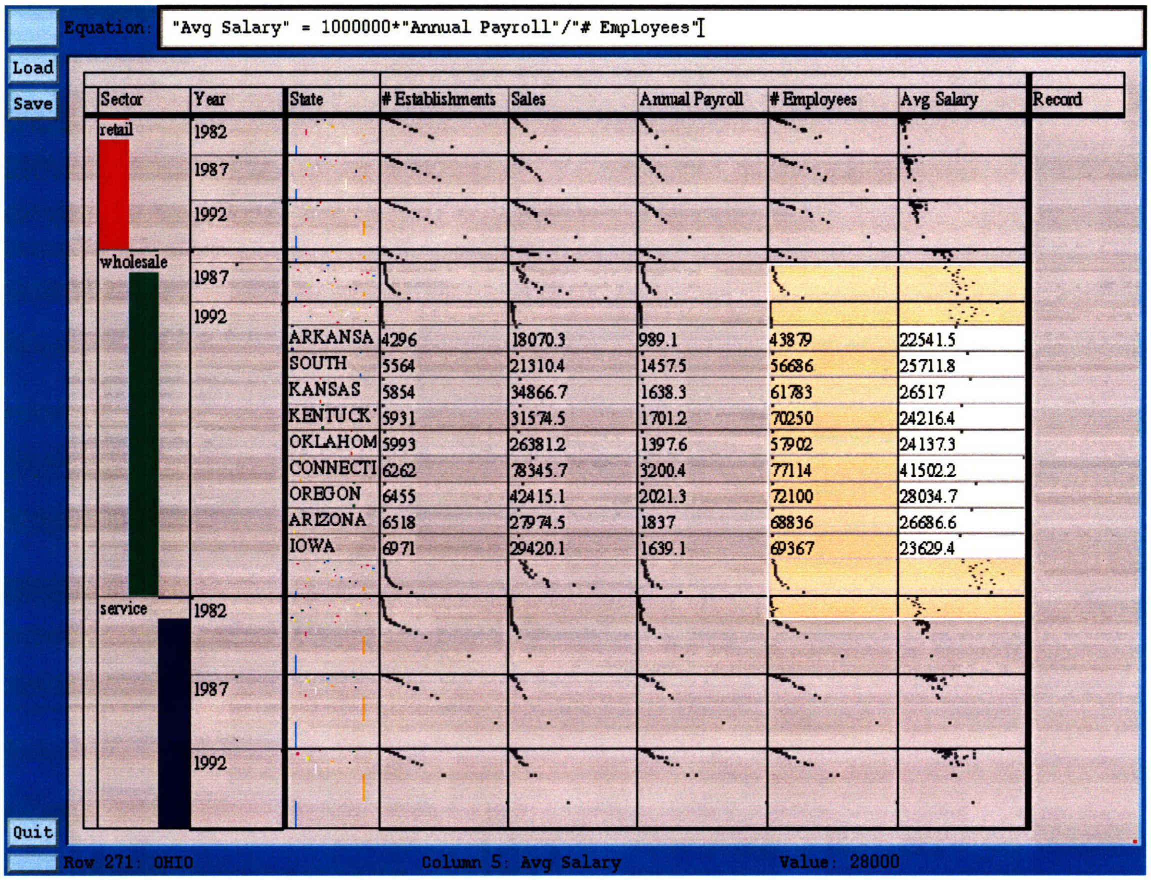


Equation: "Avg Salary" = 1000000*"Annual Payroll"/"# Employees"

Row 272: PENNSYLVANIA   Column 5: Avg Salary   Value: 29422.6

Figure B.13: Focusing on the highest average salary cluster.

Figure B.13 shows the act of making a looping gesture over a cluster of average salaries in the wholesale sector in order to create a focal area at that cluster. The result is shown on figure B.14.



Equation: "Avg Salary" = 1000000*"Annual Payroll"/"# Employees"

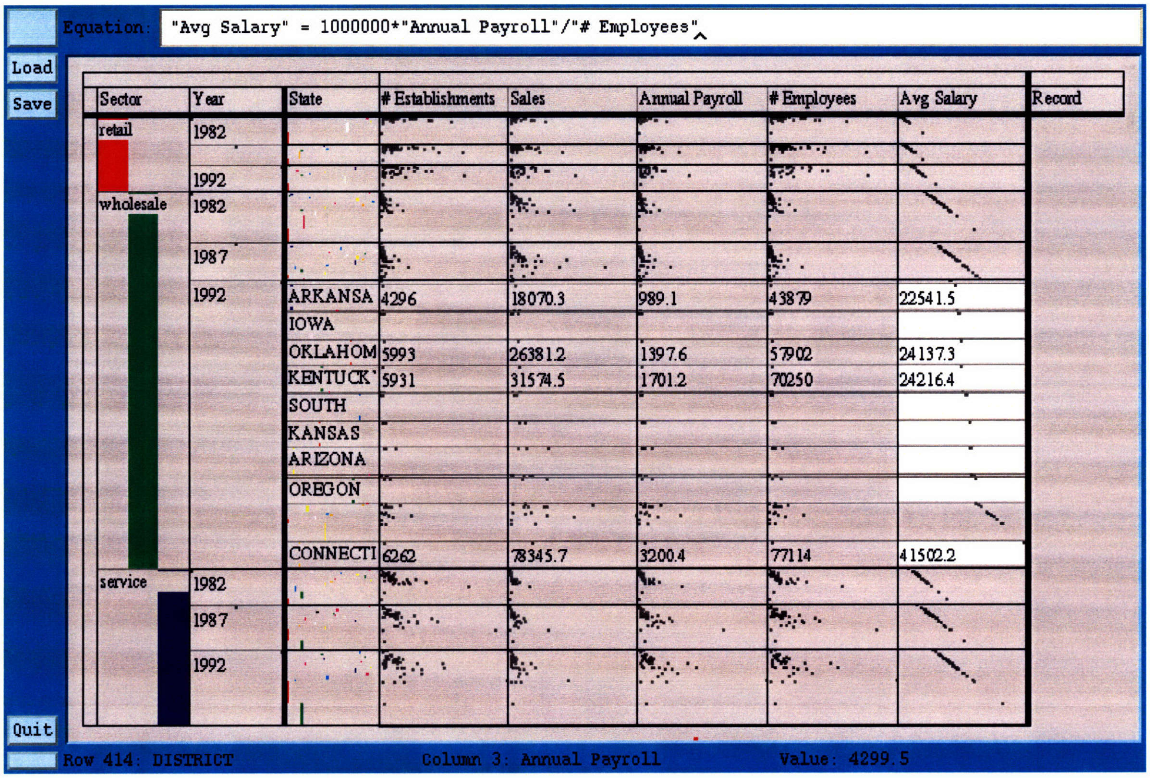
Sector	Year	State	# Establishments	Sales	Annual Payroll	# Employees	Avg Salary	Record
retail	1982							
	1987							
	1992							
wholesale	1987							
	1992	ARKANSA	4296	18070.3	989.1	43879	22541.5	
		SOUTH	5564	21310.4	1457.5	56686	25711.8	
		KANSAS	5854	34866.7	1638.3	61783	26517	
		KENTUCK'	5931	31574.5	1701.2	70250	24216.4	
		OKLAHOM	5993	26381.2	1397.6	57902	24137.3	
		CONNECTI	6262	78345.7	3200.4	77114	41502.2	
		OREGON	6455	42415.1	2021.3	72100	28034.7	
		ARIZONA	6518	27974.5	1837	68836	26686.6	
		IOWA	6971	29420.1	1639.1	69367	23629.4	
service	1982							
	1987							
	1992							

Row 271: OHIO Column 5: Avg Salary Value: 28000

Figure B.14: The focal area resulting from the manipulation on figure B.13.

Figure B.14 also illustrates the situation of a focal area while in graph mode: a combination of graphical and textual representation at the same time.

We notice that Connecticut seems to have a very high average salary in the wholesale sector and we decide to check whether that is indeed the case. To do that we sort by salaries. The result is shown on figure B.15, which also illustrates how the focal area breaks up after sorting.



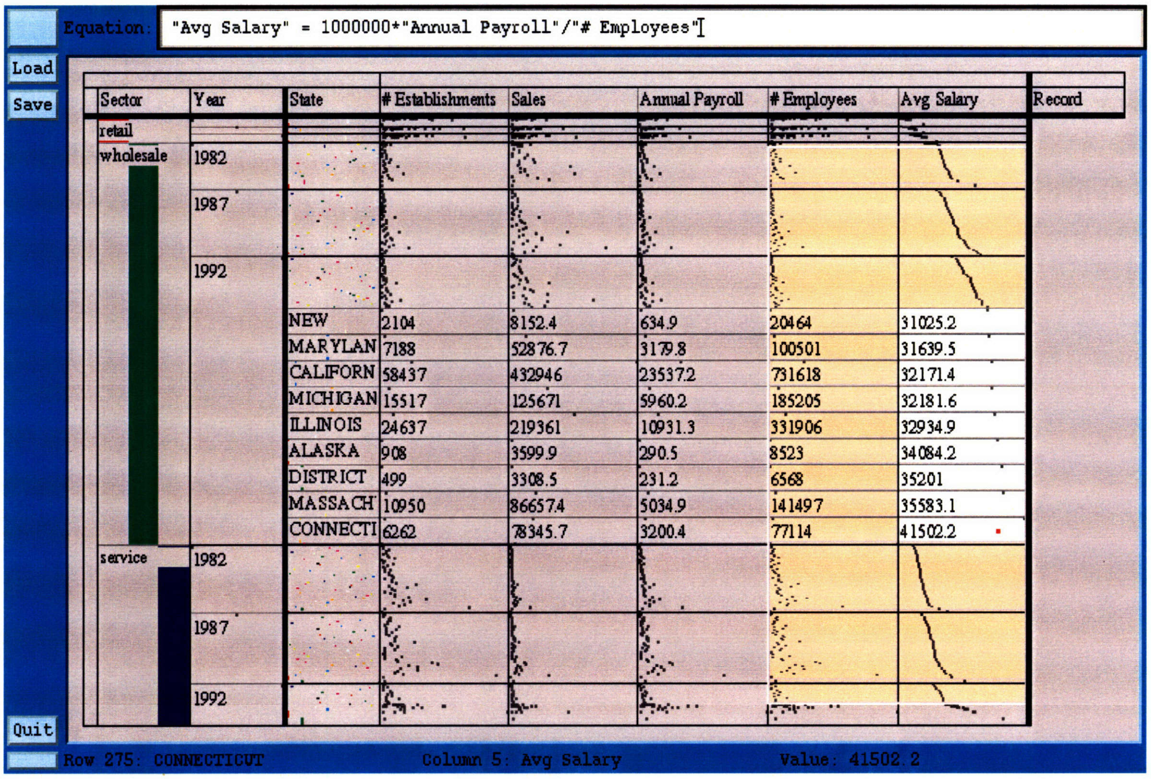
Equation: "Avg Salary" = 1000000*"Annual Payroll"/"# Employees"

Sector	Year	State	# Establishments	Sales	Annual Payroll	# Employees	Avg Salary	Record
retail	1982							
	1992							
wholesale	1982							
	1987							
	1992	ARKANSA	4296	18070.3	989.1	43879	22541.5	
		IOWA						
		OKLAHOM	5993	26381.2	1397.6	57902	24137.3	
		KENTUCK	5931	31574.5	1701.2	70250	24216.4	
		SOUTH						
		KANSAS						
		ARIZONA						
		OREGON						
		CONNECTI	6262	78345.7	3200.4	77114	41502.2	
service	1982							
	1987							
	1992							

Row 414: DISTRICT Column 3: Annual Payroll Value: 4299.5

Figure B.15: Sorting by "Avg Salary"

After refocusing on the largest values of the sorted column we arrive at figure B.16, from which we conclude that indeed Connecticut had the highest average salary in the wholesale sector in 1992 followed by Massachusetts and DC.



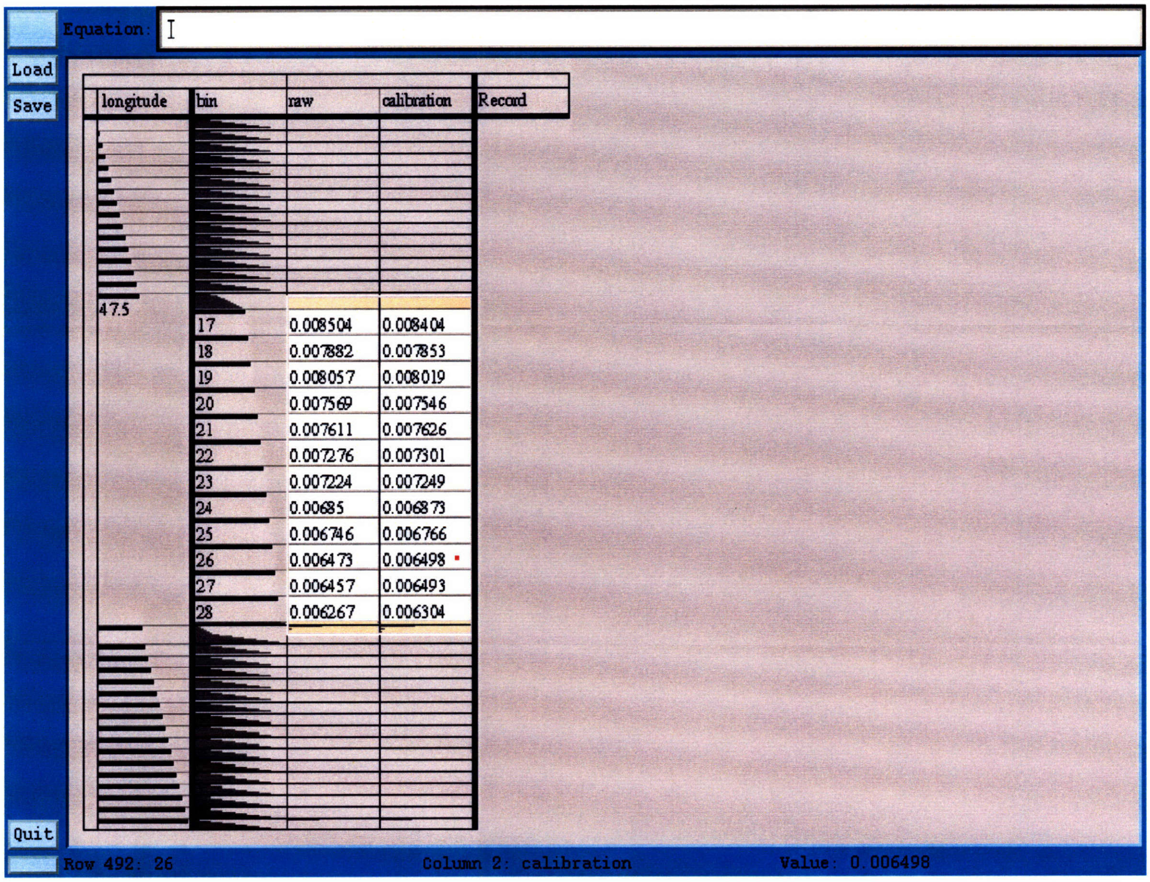
Equation: "Avg Salary" = 1000000*"Annual Payroll"/"# Employees"

Sector	Year	State	# Establishments	Sales	Annual Payroll	# Employees	Avg Salary	Record
retail								
wholesale	1982							
	1987							
	1992							
		NEW	2104	8152.4	634.9	20464	31025.2	
		MARYLAN	7188	52876.7	3179.8	100501	31639.5	
		CALIFORN	58437	432946	23537.2	731618	32171.4	
		MICHIGAN	15517	125671	5960.2	185205	32181.6	
		ILLINOIS	24637	219361	10931.3	331906	32934.9	
		ALASKA	908	3599.9	290.5	8523	34084.2	
		DISTRICT	499	3308.5	231.2	6568	35201	
		MASSACH	10950	86657.4	5034.9	141497	35583.1	
		CONNECTI	6262	78345.7	3200.4	77114	41502.2	
service	1982							
	1987							
	1992							

Row 275: CONNECTICUT Column 5: Avg Salary Value: 41502.2

Figure B.16: Refocusing on the highest average salaries in the "Wholesale" sector.

B.3 Astrophysics Data



longitude	bin	raw	calibration	Record
47.5	17	0.008504	0.008404	
	18	0.007882	0.007853	
	19	0.008057	0.008019	
	20	0.007569	0.007546	
	21	0.007611	0.007626	
	22	0.007276	0.007301	
	23	0.007224	0.007249	
	24	0.00685	0.006873	
	25	0.006746	0.006766	
	26	0.006473	0.006498	
	27	0.006457	0.006493	
	28	0.006267	0.006304	

Row 492: 26 Column 2: calibration Value: 0.006498

Figure B.17: Initial data set

This example illustrates the use of SpreadCube for analyzing scientific data. The data set contains actual data from an astrophysics experiment which the author had previously done.

B.3.1 Background

The goal of this experiment is to study the structure of the Galaxy by observing Doppler shifts of the 21-cm line. Radio emissions at the 21-cm wavelength originate from thermally excited hydrogen atoms as a result of a spin-flip transitions. Since most of the matter in the Galaxy is made up of atomic hydrogen, we can study the distribution of matter by studying the intensity of the 21-cm emissions from different directions on the Galaxy.

Because distant hydrogen clouds move with respect to us, we observe the 21-cm line shifted due to a Doppler shift. The reason for this is illustrated on figure B.18.

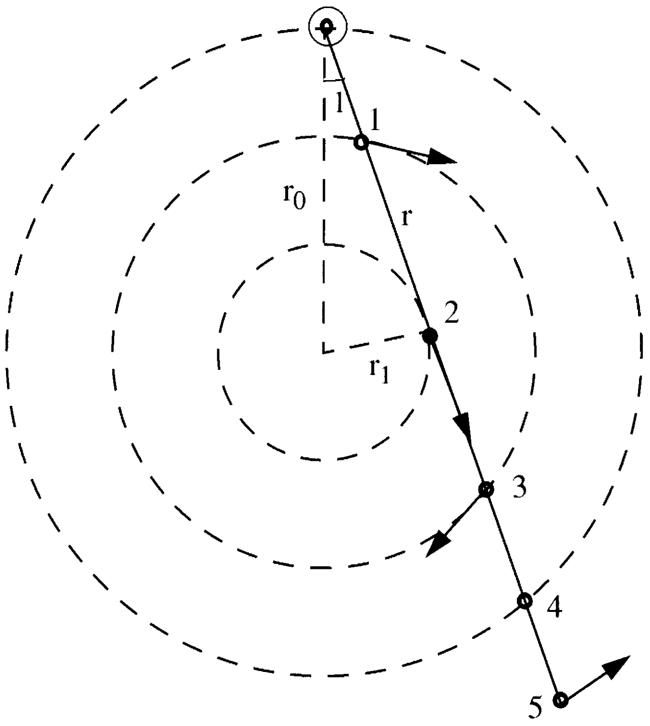


Figure B.18: Apparent radial velocities of hydrogen clouds

The illustration on figure B.18 is a sketch of a top view on the disc of our Galaxy. The small circle on the top marks the Solar system, the diagonal straight line is a direction of observation.

The Galactic disk spins clockwise and as a result distant hydrogen clouds, marked by the points 1, 2, 3, 4 and 5, move with respect to us. The apparent motion, and therefore, the observed Doppler shift is greatest for points which move tangetially to the line of sight, e.g. point 2. This is illustrated on figure B.19, which shows a sample profile of the 21-cm line when pointing the radio telescope at the direction indicated on figure B.18.

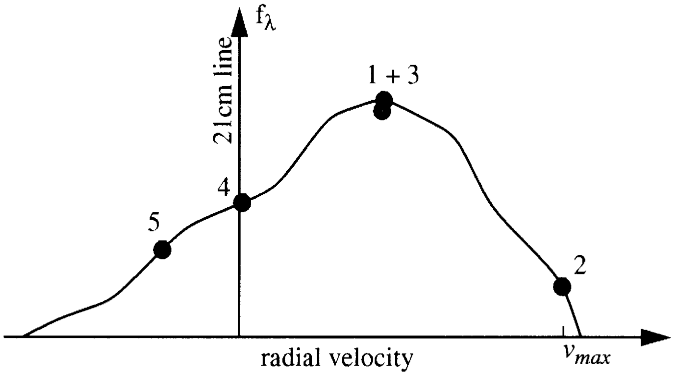


Figure B.19: A Profile of the 21-cm line for the observations on figure B.18
The horizontal axis represents wavelength, while the vertical axis represents intensity. The points 1, 2, 3, 4 and 5 on the graph correspond to those points on figure B.18.

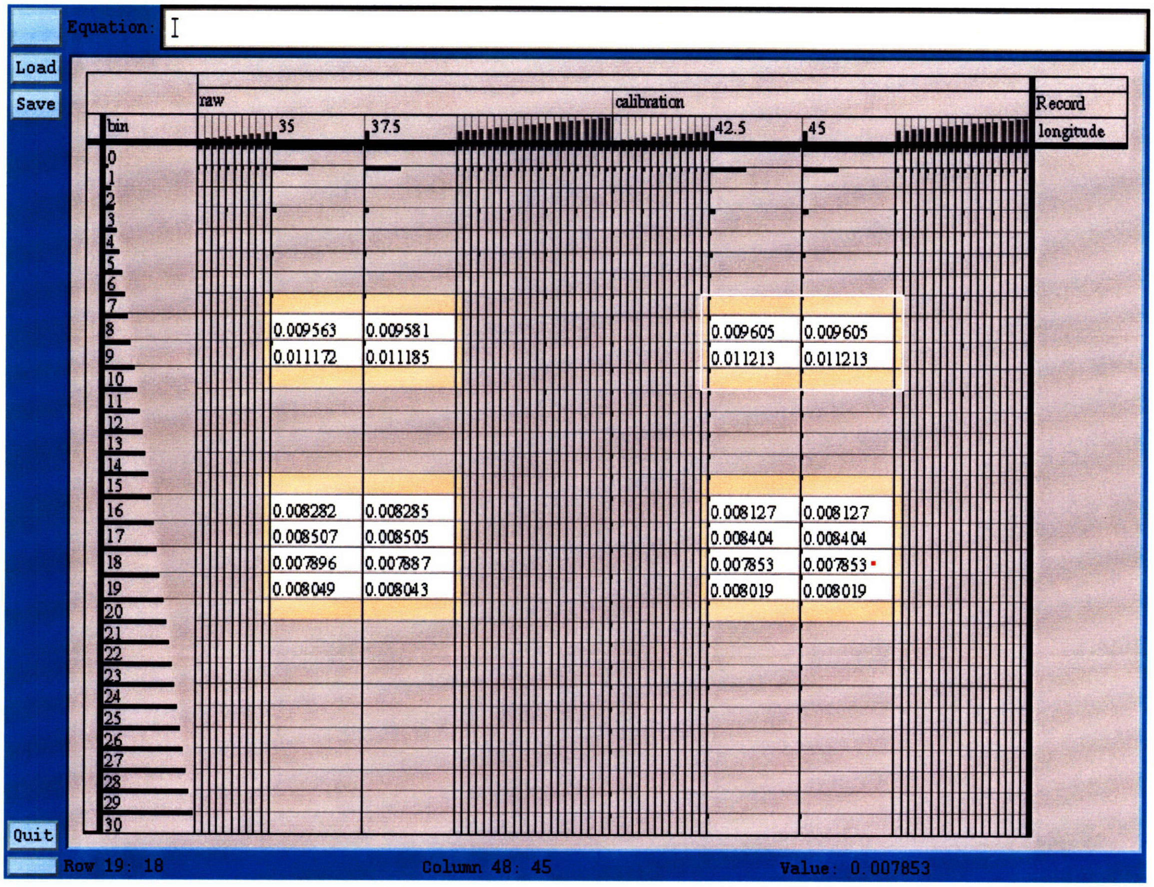
B.3.2 The Experiment

The experiment involves pointing the radio telescope at different galactic longitudes *l* in the range of 10° to 90° and collecting 21-cm profiles, like the one shown on figure B.19. First the telescope is pointed at a direction which is poor in 21-cm emissions and a calibration spectrum is collected. Then a raw spectrum is collected by pointing the telescope in the desired direction. The difference of the two is the actual profile.

Based on those profiles, the velocity curve of the Galaxy was calculated, i.e. the radial velocity as a function of the Galactic radius. This curve was needed to calculate hydrogen distribution in the galaxy for galactic latitudes from 10° to 90°.

It is beyond the scope of this demonstration however, to calculate the velocity curve. Without the velocity curve, we can still deduce the hydrogen density distribution but only for points on the galactic disc, which are tangential to the line of site, like point 2 on figure B.18.

B.3.3 The Analysis



	raw		calibration	
bin	35	37.5	42.5	45
8	0.009563	0.009581	0.009605	0.009605
9	0.011172	0.011185	0.011213	0.011213
16	0.008282	0.008285	0.008127	0.008127
17	0.008507	0.008505	0.008404	0.008404
18	0.007896	0.007887	0.007853	0.007853
19	0.008049	0.008043	0.008019	0.008019

Row 19: 18 Column 48: 45 Value: 0.007853

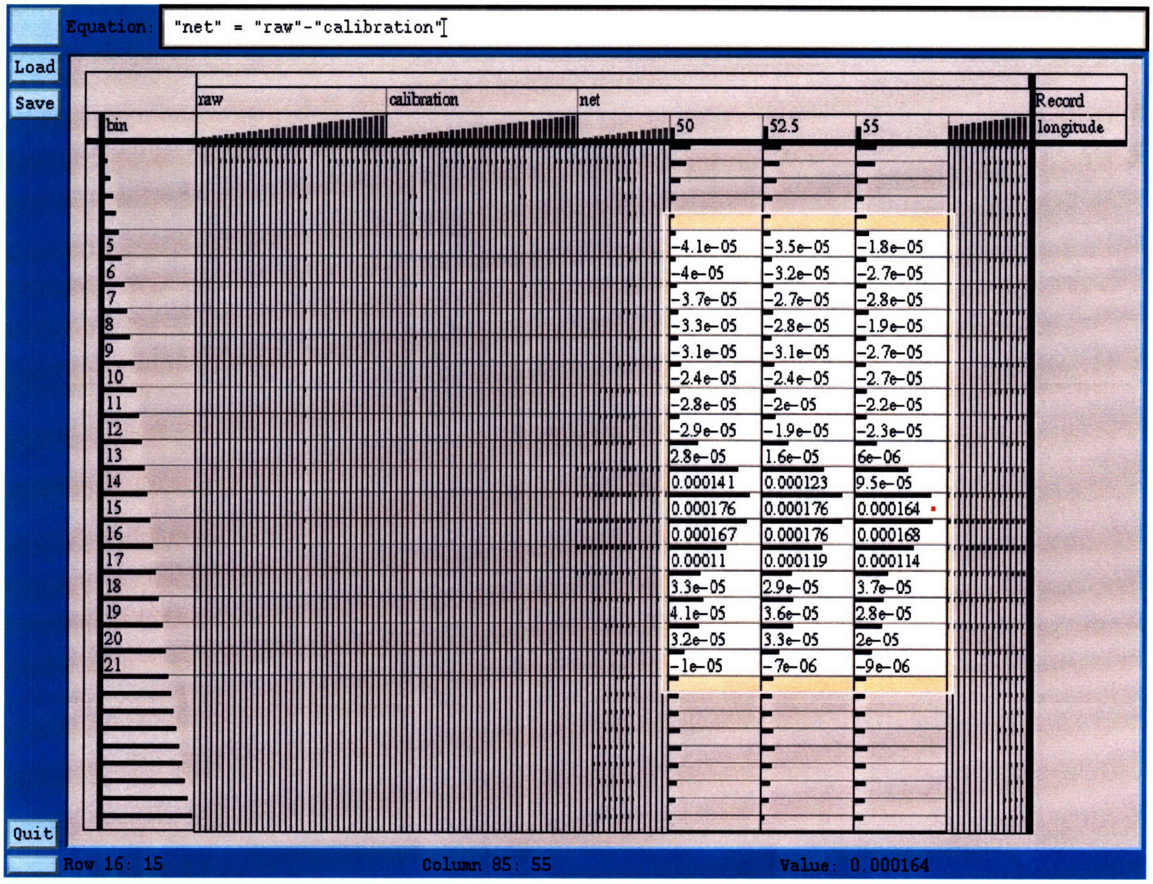
Figure B.20: Moving the “longitude” dimension to the horizontal axis and focusing.

The data consists of a collection of spectra centered around the 21-cm wavelength for longitudes in the range of 10^o to 90^o and increments of 2.5^o (see figure B.17). The spectral intensities are collected into thirty bins. After moving the longitude dimension to the horizontal axis and opening two focal areas, one can see the raw and calibration profile spectra better (figure B.20).

We then calculate:

```
"net" = "raw" - "calibration"
```

Which adds a new “net” variable to the data set, representing the actual spectral data that we are interested in (figure B.21).



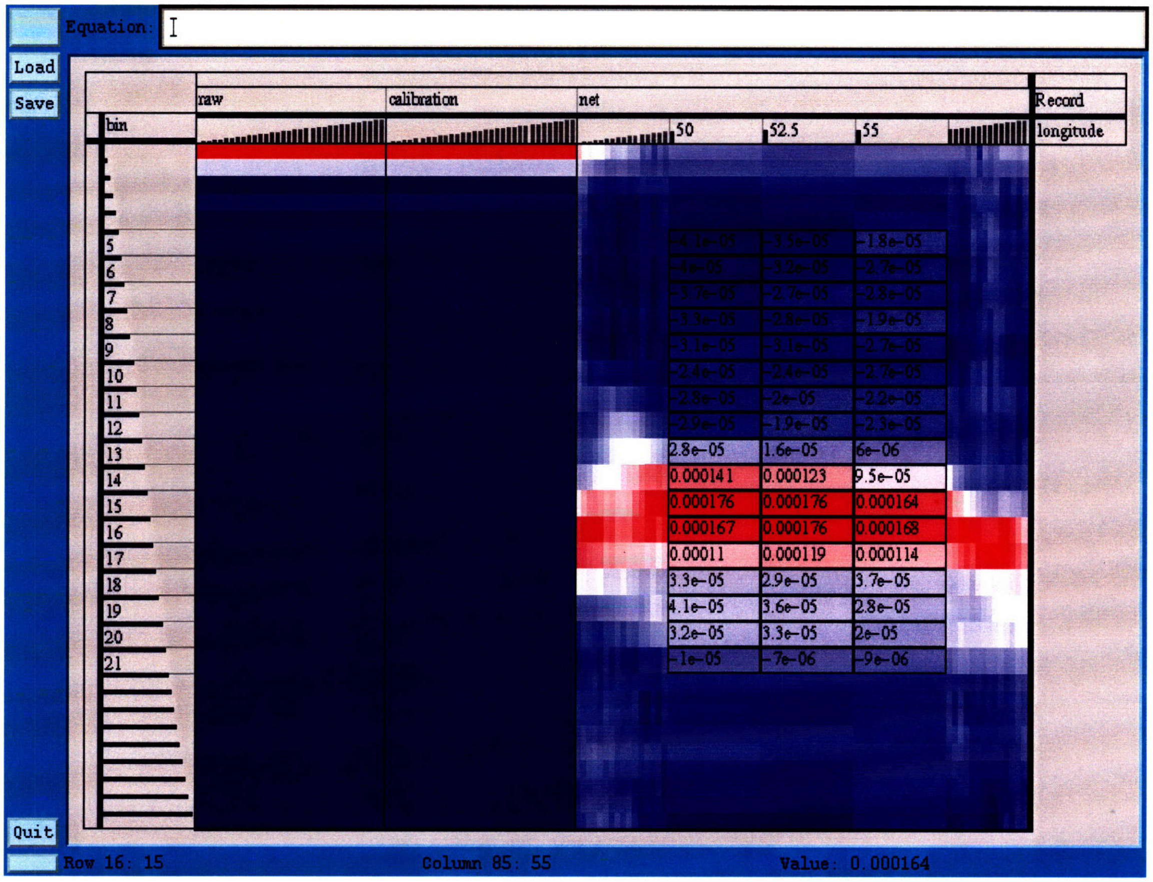
Equation: "net" = "raw"-"calibration"

bin	50	52.5	55
5	-4.1e-05	-3.5e-05	-1.8e-05
6	-4e-05	-3.2e-05	-2.7e-05
7	-3.7e-05	-2.7e-05	-2.8e-05
8	-3.3e-05	-2.8e-05	-1.9e-05
9	-3.1e-05	-3.1e-05	-2.7e-05
10	-2.4e-05	-2.4e-05	-2.7e-05
11	-2.8e-05	-2e-05	-2.2e-05
12	-2.9e-05	-1.9e-05	-2.3e-05
13	2.8e-05	1.6e-05	6e-06
14	0.000141	0.000123	9.5e-05
15	0.000176	0.000176	0.000164
16	0.000167	0.000176	0.000168
17	0.00011	0.000119	0.000114
18	3.3e-05	2.9e-05	3.7e-05
19	4.1e-05	3.6e-05	2.8e-05
20	3.2e-05	3.3e-05	2e-05
21	-1e-05	-7e-06	-9e-06

Row 16: 15 Column 85: 55 Value: 0.000164

Figure B.21: Calculating the "net" Doppler shift.

The visualization on figure B.21 represents each spectrum as an individual profile, by grouping the spectra into sticks. By moving the horizontal merge mark up, we can view the spectral profiles as group, i.e. the spectral intensity is visualized as a color saturation with respect to bin number and longitude (figure B.22).

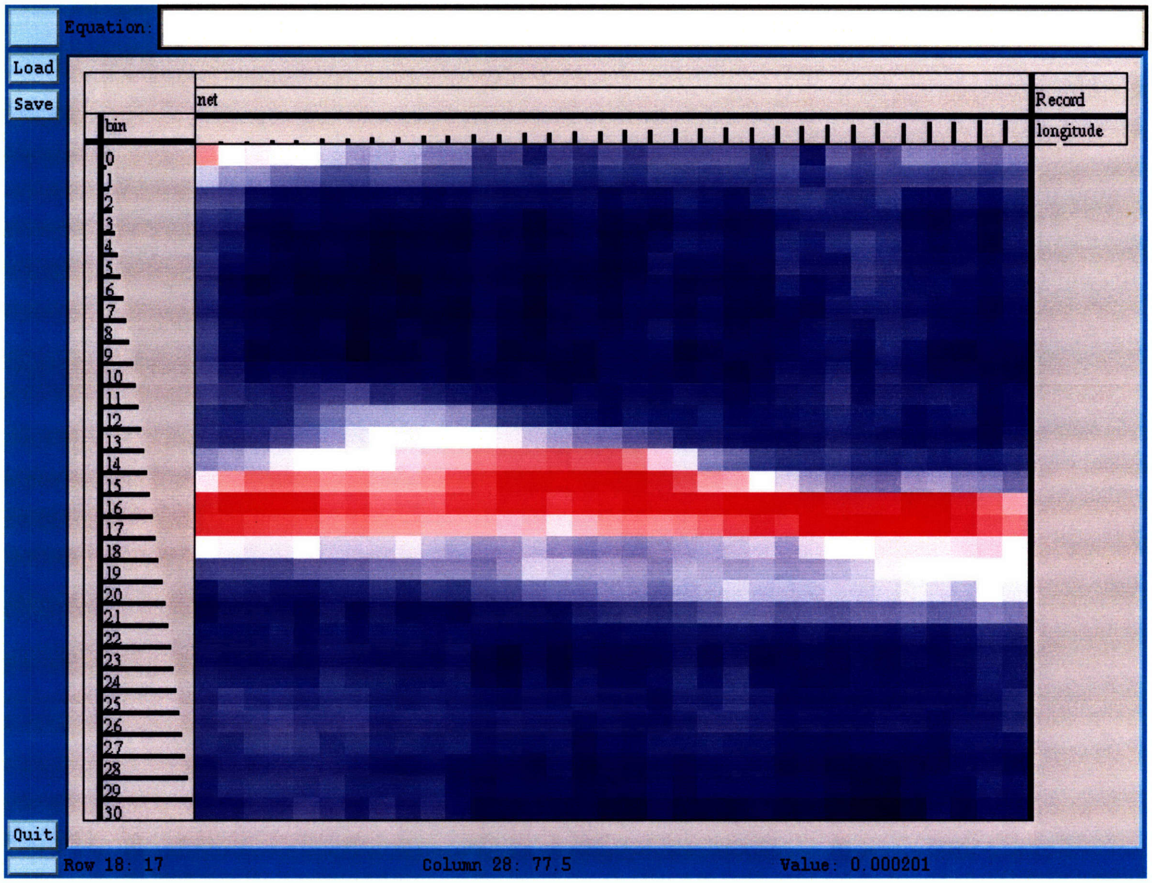


bin	50	52.5	55
5	-4.1e-05	-3.5e-05	-1.8e-05
6	-4e-05	-3.2e-05	-2.7e-05
7	-3.7e-05	-2.7e-05	-2.8e-05
8	-3.3e-05	-2.8e-05	-1.9e-05
9	-3.1e-05	-3.1e-05	-2.7e-05
10	-2.4e-05	-2.4e-05	-2.7e-05
11	-2.8e-05	-2e-05	-2.2e-05
12	-2.9e-05	-1.9e-05	-2.3e-05
13	2.8e-05	1.6e-05	6e-06
14	0.000141	0.000123	9.5e-05
15	0.000176	0.000176	0.000164
16	0.000167	0.000176	0.000168
17	0.00011	0.000119	0.000114
18	3.3e-05	2.9e-05	3.7e-05
19	4.1e-05	3.6e-05	2.8e-05
20	3.2e-05	3.3e-05	2e-05
21	-1e-05	-7e-06	-9e-06

```
Row 16: 15          Column 85: 55          Value: 0.000164
```

Figure B.22: Moving the horizontal merge mark up.

We can now remove the "raw" and "calibration" variables which are no longer necessary and only clutter the view (figure B.23).



Row 18: 17 Column 28: 77.5 Value: 0.000201

Figure B.23: Removing the “raw” and “calibration” variables.

At this point we would like to isolate the portion of the data which corresponds to greatest doppler shift, as in the case of point 2 on figure B.19. We do so by holding down the middle mouse button and drawing a loop around the area which we wish to isolate. This process is captured on figure B.24.

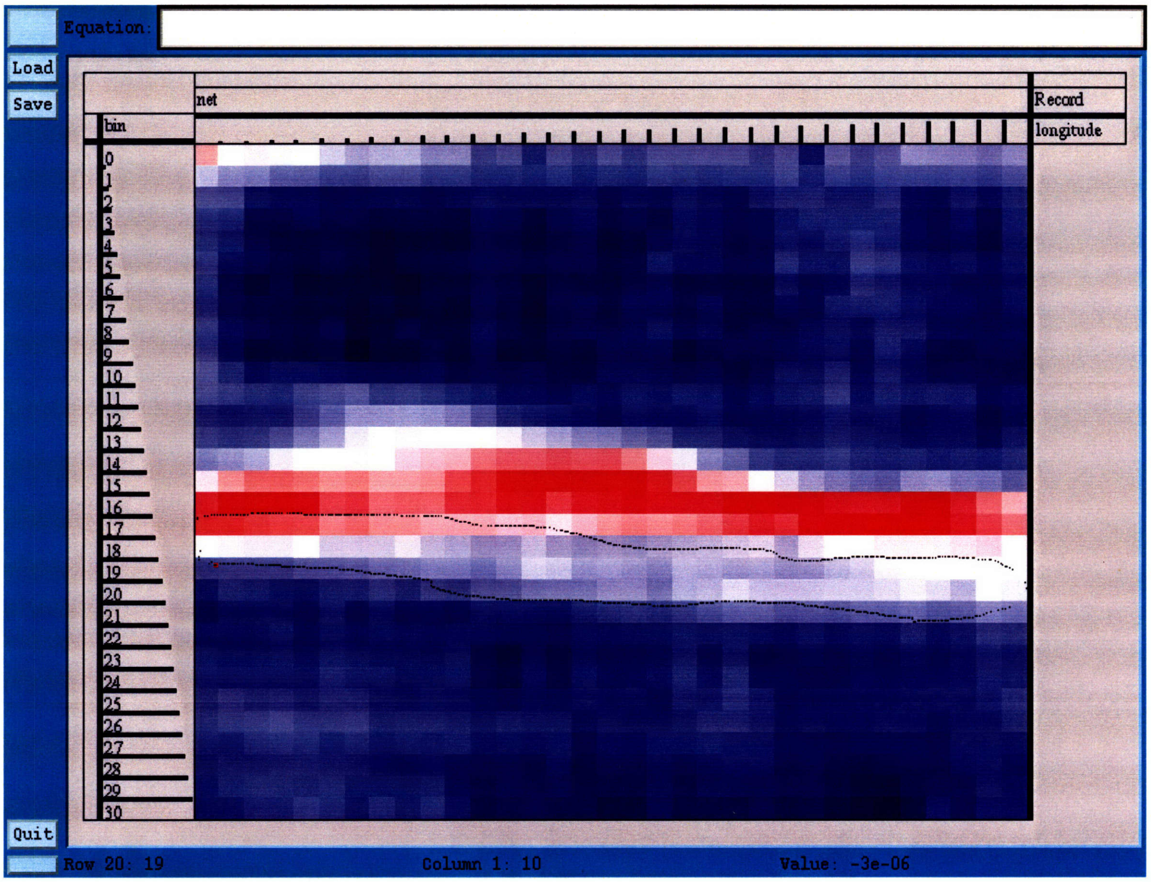


Figure B.24: Selecting the region of maximum Doppler shift

As a result of the selection, a new selection variable is created (figure B.25).

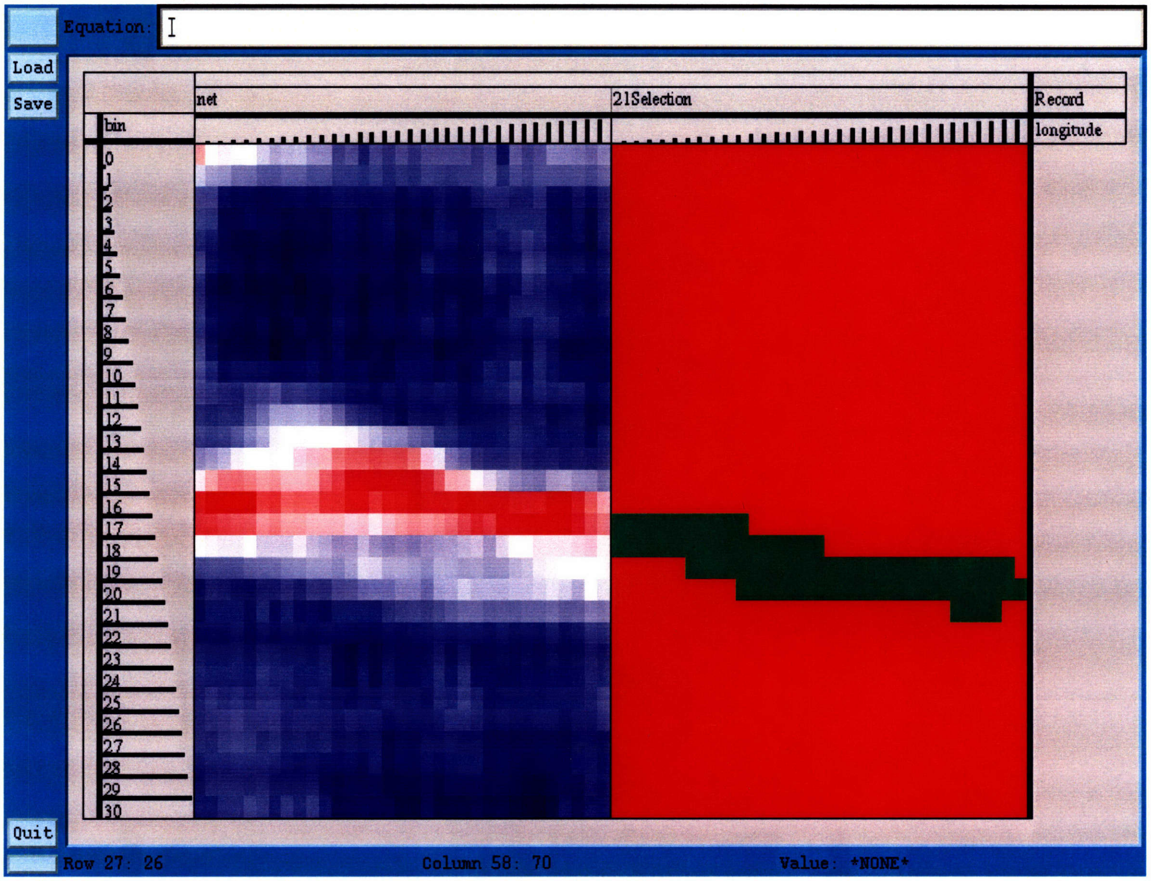


Figure B.25: The result of the selection from figure B.24.

After promoting the selection variable we arrive at figure B.26.

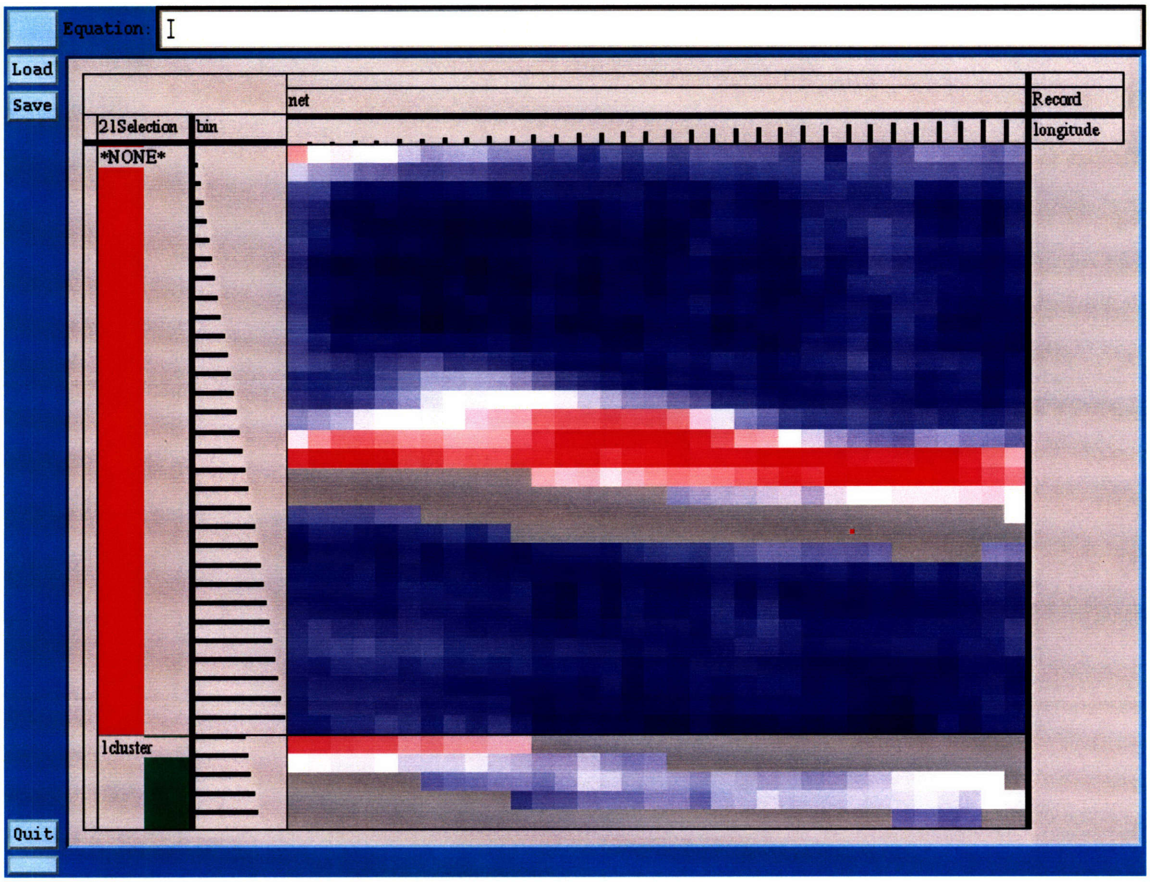


Figure B.26: Promoting the selection variable

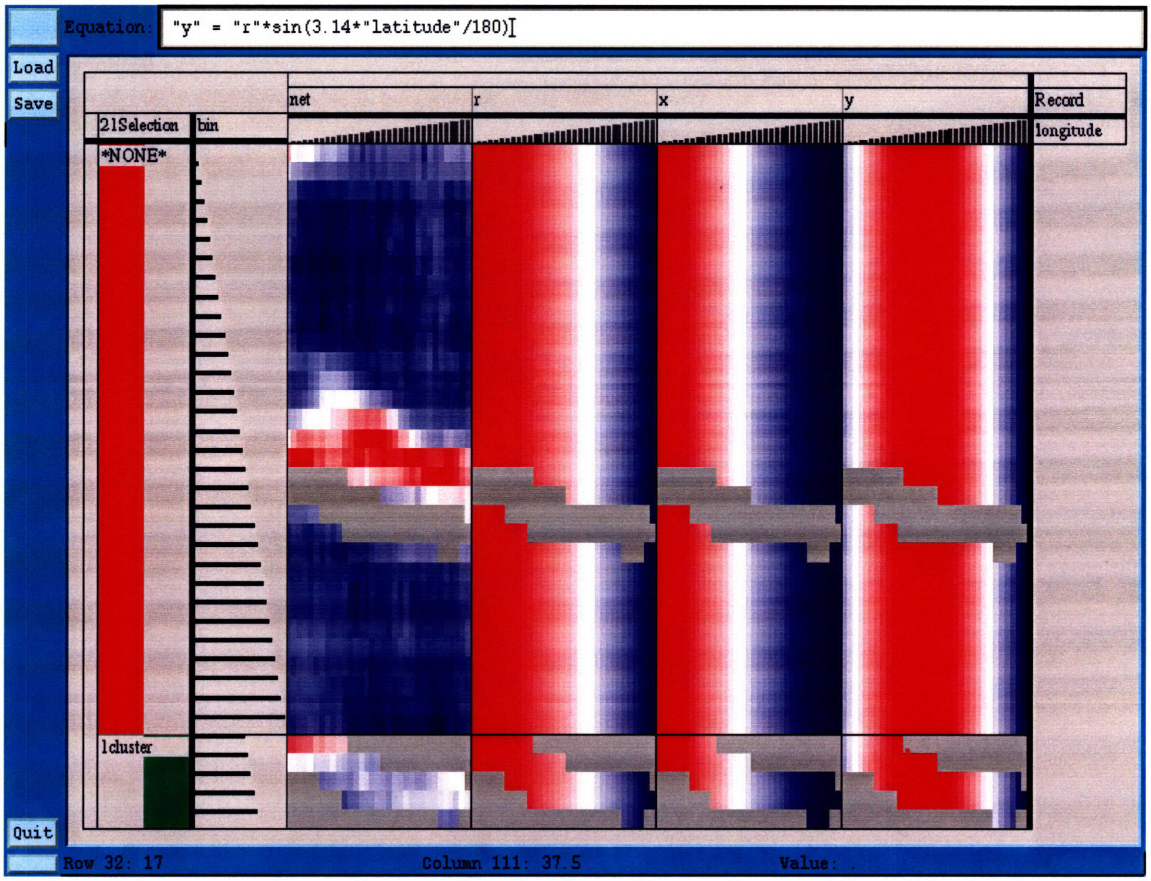
The gray areas on figure B.25 represent non-existent values.

We are interested in plotting the selection in polar coordinates. For that we first calculate the distance "r" between the Solar system and a hydrogen cloud which is tangential to the line of sight. From figure B.18 we can conclude that:

$$r = r_o \cos(l) \tag{B.4}$$

where r_o is the distance between the Solar system and the Galactic center. For the purposes of this demonstration we will assume that $r_o = 1$ in some units. From "r" and "longitude" we can compute the Cartesian coordinates: "x" and "y":

$$\begin{aligned} x &= r\cos(l) \\ y &= r\sin(l) \end{aligned} \tag{B.5}$$



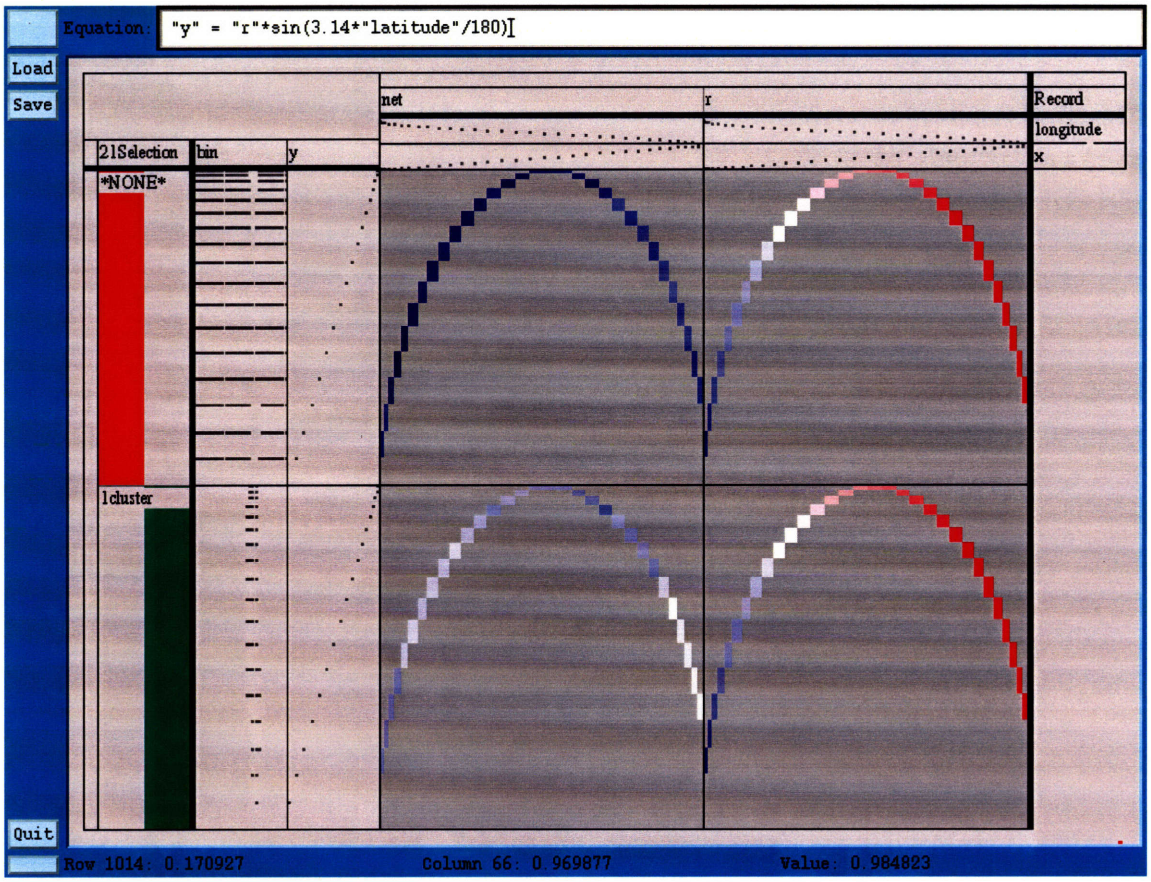
Equation: "y" = "r"*sin(3.14*"latitude"/180)

Row 32: 17 Column 111: 37.5 Value:

Figure B.27: Computing “r”, “x” and “y” coordinates.

The illustration on figure B.27 shows the stage of calculating “y” as a function of “r” and “l”.

Now we can promote “x” and “y” to the horizontal and vertical axis respectively in order to arrive at a polar plot (figure B.28).

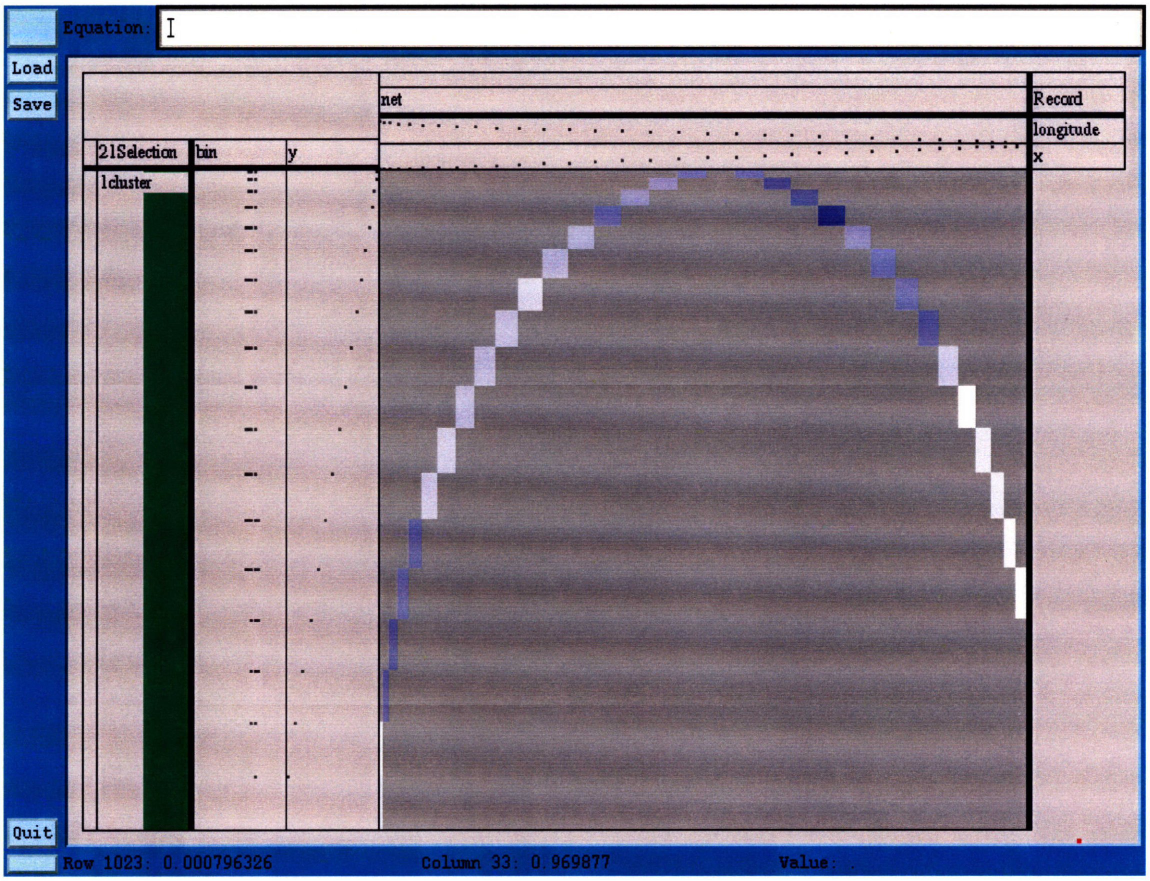


Equation: "y" = "r"*sin(3.14*"latitude"/180)

Row 1014: 0.170927 Column 66: 0.969877 Value: 0.984823

Figure B.28: Promoting the "x" and "y" variable and switching to graph mode.

On figure B.28 there are four plots, but only the lower left interests us. To get rid of the other three, we slice through the "1cluster" and "net" dimension keys to arrive at the visualization on figure B.29.



Row 1023: 0.000796326

Column 33: 0.969877

Value: .

Figure B.29: Slicing through the “net” and “cluster1” keys.

The plot on figure B.29 represents the density of hydrogen clouds in the Galaxy at the points tangential to the line of sight. The curve which can be observed on this plot corresponds to the red curve on the sketch on figure B.30. The color intensities correspond to hydrogen densities along this curve.

The position of the Solar system is at the lower left corner of the plot on figure B.29 and the Galactic center is at the lower right. From this plot we can see that indeed the hydrogen density appears to be largest near the Galactic center.

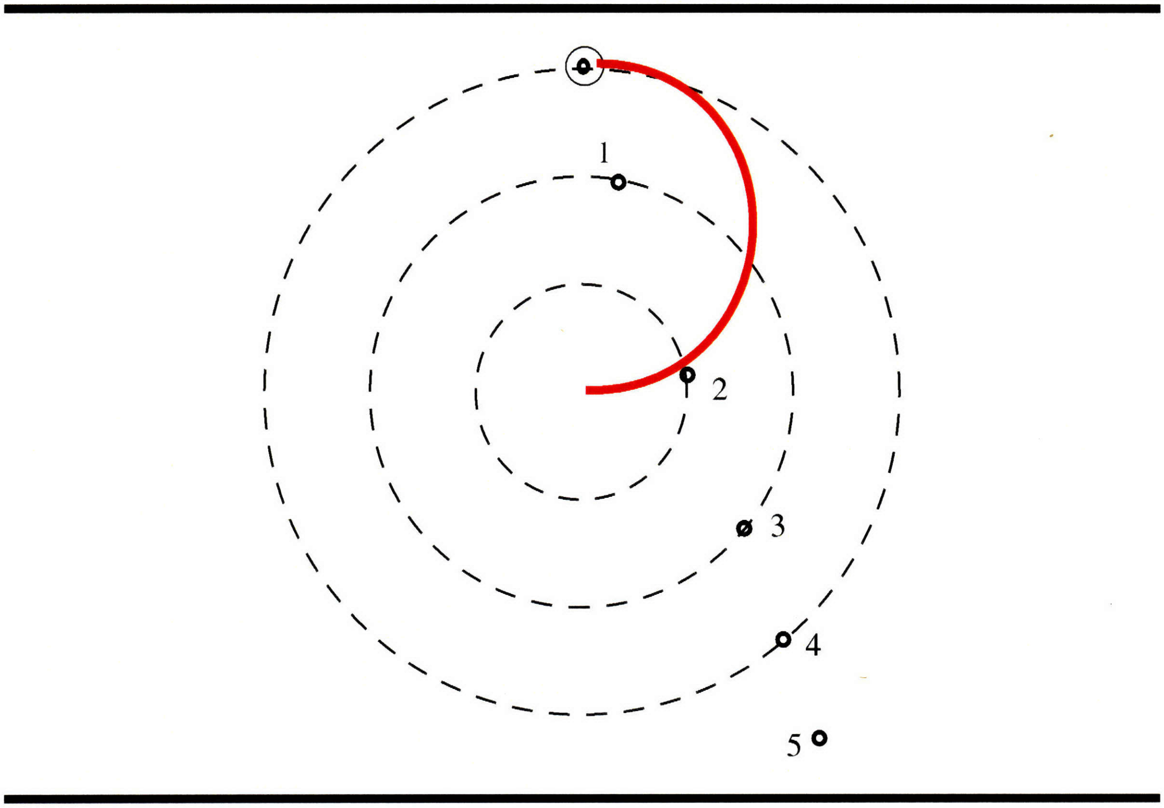


Figure B.30: A sketch of what the plot on figure B.29 shows.