BROWSING LARGE ONLINE DATA USING GENERALIZED QUERY PREVIEWS

Egemen Tanin, Doctor of Philosophy, 2001

Companies, government agencies, and other organizations are making their data available to the world over the Internet. These organizations store their data in large tables. These tables are usually kept in relational databases. Online access to such databases is common. Users query these databases with different front-ends. These front-ends use command languages, menus, or form fillin interfaces. Many of these interfaces rarely give users information about the contents and distribution of the data. This leads users to waste time and network resources posing queries that have zero-hit or mega-hit results.

Generalized query previews forms a user interface architecture for efficient browsing of large online data. Generalized query previews supplies distribution information to the users. This provides an overview of the data. Generalized query previews gives continuous feedback about the size of the results as the query is being formed. This provides a preview of the results.

Generalized query previews allows users to visually browse all of the attributes of the data. Users can select from these attributes to form a view. Views are used to display the distribution information. Queries are incrementally and visually formed by selecting items from numerous charts attached to these views. Users continuously get feedback on

the distribution information while they make their selections. Later, users fetch the desired portions of the data by sending their queries over the network. As they make informed queries, they can avoid submitting queries that will generate zero-hit or mega-hit results.

Generalized query previews works on distributions. Distribution information tends to be smaller than raw data. This aspect of generalized query previews also contributes to better network performance.

This dissertation presents the development of generalized query previews, field studies on various platforms, and experimental results. It also presents an architecture of the algorithms and data structures for the generalized query previews.

There are three contributions of this dissertation. First, this work offers a general user interface architecture for browsing large online data. Second, it presents field studies and experimental work that define the application domain for generalized query previews. Third, it contributes to the field of algorithms and data structures.

BROWSING LARGE ONLINE DATA USING GENERALIZED QUERY PREVIEWS

by

Egemen Tanin

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2001

Advisory Committee:

Professor Ben A. Shneiderman, Chair
Assistant Professor Ben Bederson
Associate Professor Kent L. Norman
Associate Research Scientist Catherine Plaisant-Schwenn
Associate Professor Amitabh Varshney

To
NİŞVAN ERKAL

ACKNOWLEDGEMENTS

I first would like to thank Ben A. Shneiderman for being my advisor. He is a very patient and inspirational advisor. He encouraged me for many years and helped me stay on track. I learned many valuable things about research, science, and life from him.

I also would like to thank all of my dissertation members. They gave useful leads through out this work. Their objectivity made this dissertation a better one.

I thank all the members of the Human-Computer Interaction Laboratory. I am very happy to work with them. Specifically, thanks to Catherine Plaisant-Schwenn, Anne Rose, Harry Hochheiser, Hyunmo Kang, Chris North, and Eser Kandoğan for their inputs.

I also thank members of National Science Foundation, United States Census Bureau, and National Aeronautics and Space Association for supporting this work. Without their belief, I would not be at this point.

I finally would like to thank my family and friends for enduring me during these years of research.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1:
INTRODUCTION

1.1 Problem

Companies, government agencies, and other organizations are making more of their data available over the Internet. International Business Machines Corporation (IBM) hosts a collection of millions of patents (www.patents.ibm.com) accessible to the public over the Internet. United States Census Bureau is a government agency hosting vast collections of economic, geographic, and demographic data (e.g., ferret.bls.census.gov). National Aeronautics and Space Association (NASA) is another government agency with still larger collections of scientific and environmental data (e.g., eos.nasa.gov/eosdis). The World Health Organization (WHO) is an international organization that shares medical and population related information over the Internet (e.g., www.who.int/whosis/). These are only a few of the organizations that are making vast data resources available to the public on the Internet.

Many organizations store their data in large tables. They typically use multiple tables that are correlated to represent different aspects of the data. These tables can have many attributes and rows. Typically, these tables are kept in relational databases. A table can be a view or simply a relation of a database. Online access to many such databases is now common.

Users all around the world can use their browsers to access online databases. People of various ages, genders, and backgrounds are now forming the user domain for such

databases. Some of these users have no background on these databases. They may also be inexperienced with computers.

Users can access different types of user interfaces to work with such databases. The user interfaces that serve as the database front-ends are typically command languages, menus, or form fillin interfaces [SBC97]. Generally, these user interfaces are activated within a browser.

Many of these interfaces, instead of giving users information about the contents of the data, require users to fill lengthy electronic forms. The designers of such interfaces assume that the users are informed about the data that they are working on or they can directly submit known-item queries rather than probing the database. However, unguided novice users may waste their time submitting queries that have zero-hit or mega-hit result sets. Sometimes they assume that the users know or have the will to understand a querying environment or fill a lengthy form. However, users of online databases generally do not have the time or the will to learn a query language or they are annoyed when they have to fill a lengthy form. In some other cases, they assume that users will have the bandwidth or the time to access such large databases remotely. On the contrary, users of a public online database have to access large amounts of data using a low bandwidth congested network. A more effective, simple, and easy to learn approach for defining queries is needed for public online databases.

Figure 1.1 shows a form fillin interface from the Unites States Census Bureau (i.e., www.ferret.bls.census.gov). This interface forms a good example for the database front-ends that are currently available over the Internet. This is a lengthy form fillin interface. There is some guidance about what values can be selected on some of the fields, but the

data distribution information is not available. In this interface, users can easily generate queries that will return zero-hit or mega-hit result sets. Mostly, the users that have enough background on the data will be able to easily form useful queries. Other types of users will simply probe the database until they find what they want or get tired of working with this interface. Even when users find the parts of the data that they were looking for, most of them would be annoyed with the experience of filling in a lengthy form and blindly probing a remote database.

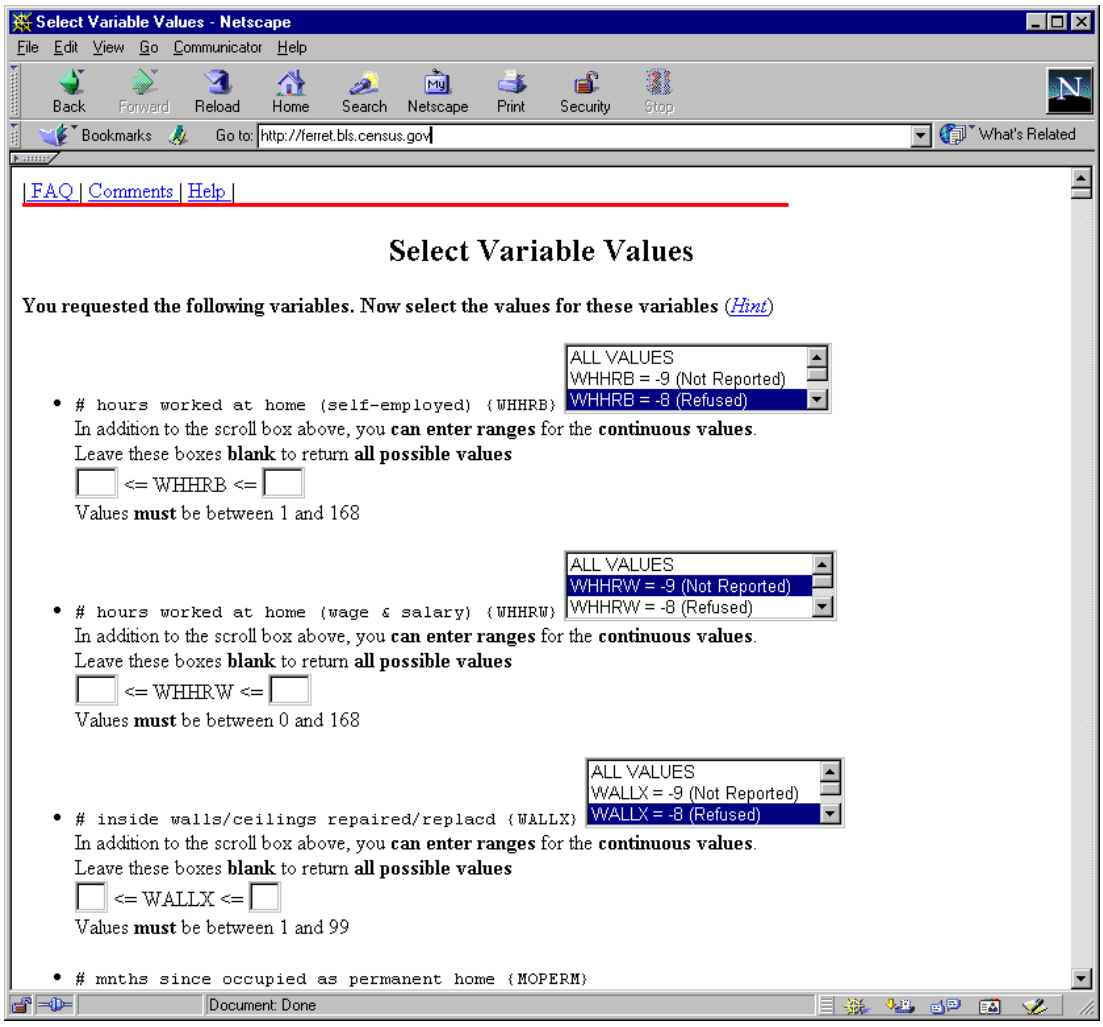


Figure 1.1: A form fillin interface from the United States Census Bureau homepage

1.2 Generalized Query Previews

One approach to overcoming the hidden nature of data is to provide some form of easily understood overview of the data. Generalized query previews forms such a user interface architecture for efficient browsing of large online data. Generalized query previews supplies data distribution information to the users. This is an overview of the data. By looking at this overview users can immediately see what is in and what is not in the data. Generalized query previews gives continuous feedback about the size of the result set as the query is being formed. This is a preview of the result set. Queries that can have zero-hit or mega-hit result sets will be visible to the user and they may be avoided easily. This will increase the performance of the overall system.

An example generalized query preview interface, the ExpO System, is shown in Figure 1.2. This interface is a sample implementation of the generalized query previews user interface architecture and the paradigms behind it. The ExpO System is generated by using a data set from the 1997 United States Economic Census collections. In this example, the data contains information about hospitals located in each of the United States Counties. This sample data contains about ten attributes and 3000 rows in its universal relation. The data is stored as four different relations. Each relation represents a single table. All the tables share a unique identifier, ‘report_id’, representing a unique report from a county.

Generalized query previews allows users to visually browse all the attributes and tables of the data. In the ExpO example, the schema for the data is presented with a hierarchical browser (the panel on the left in Figure 1.2). The root of this panel is tagged

with the name of the database, ‘hospital97’. The first level in the hierarchy displays the tables. In this data, ‘loc’, ‘payroll’, ‘sale’, and ‘size’ are the tables. The second level displays the attributes.

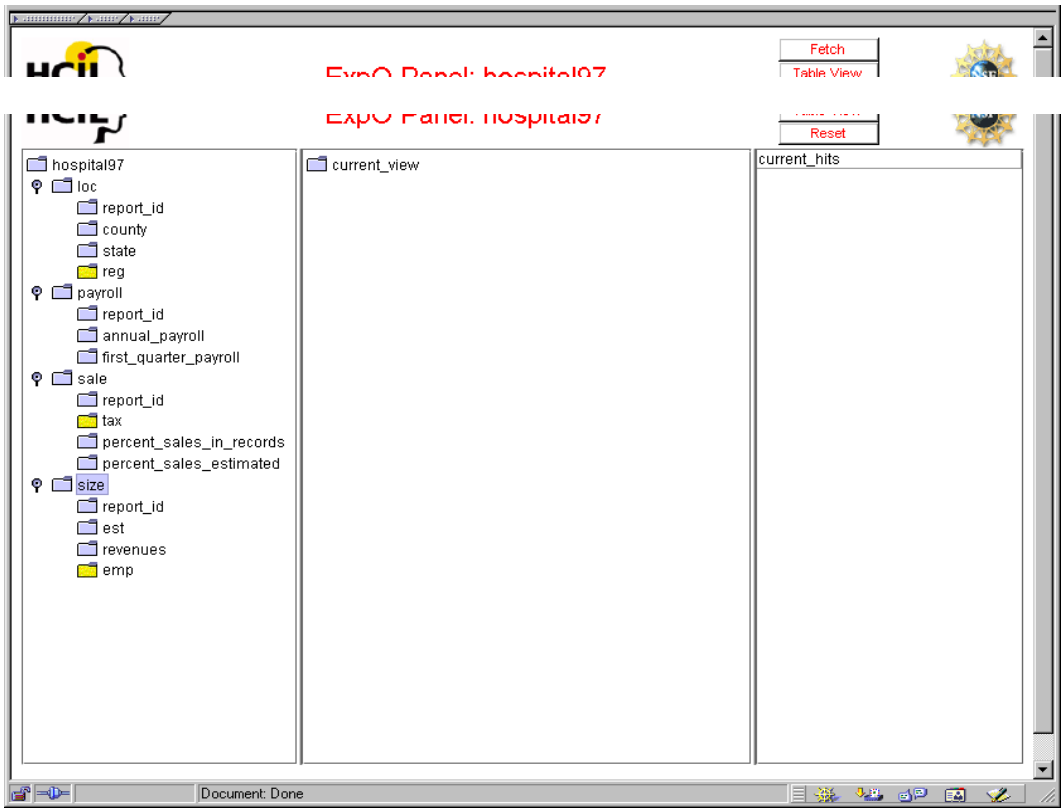


Figure 1.2: An example generalized query previews interface, ExpO, the schema for the data is presented with a hierarchical browser (the panel on the left)

In generalized query previews, users can select some of the attributes of the data to form a user view. In the ExpO example, this view is presented in a separate panel and is also represented by a hierarchical browser (Figure 1.3). The panel in the middle depicts a few user selections and a user-defined view of the data. The selected attributes are tagged

with the name of the tables that they are selected from. For example, ‘tax’ attribute of the ‘sale’ table forms the tagged name of ‘sale_tax’. Joining the relations representing these tables is automatically done in the background. Hence, only the tables with common attributes can be joined to form a view. This example contains only four tables, each represented by a single relation, all carrying the same identifier. Tables can be predefined system views of the data and need not directly map to the relations of the database.

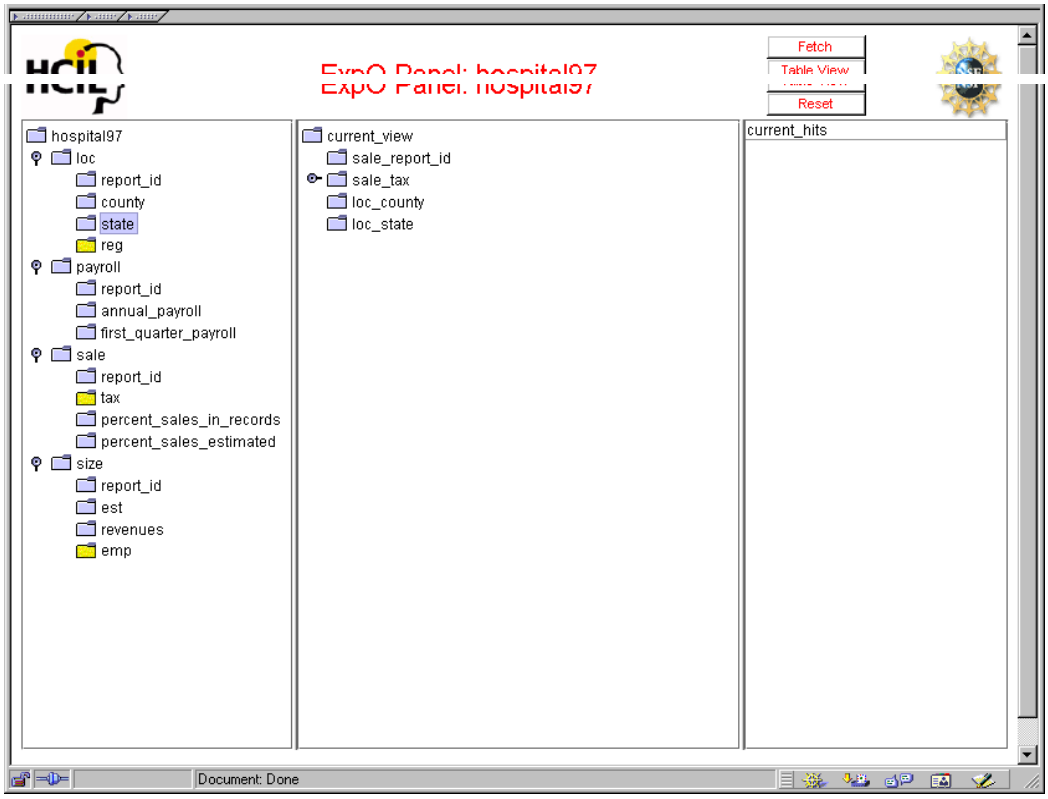


Figure 1.3: ExpO with a user-defined view of four attributes, this view is presented in a separate panel as a hierarchical browser (the panel in the middle)

Views are used to display the distribution information. In the example from Figure 1.3, a special icon in the user view shows that one of the attributes of this view,

'sale_tax', can be expanded. The same attribute can also be displayed with a different marker or colored icon on the hierarchical browser of the database schema. Figure 1.4 shows the expansion. The attributes are expandable into buckets. The data distribution information is attached to these buckets. Buckets are values where the data can be aggregated over. The data distribution information is attached to these buckets as some visual aids such as the bar charts of this example. Here, 'taxable' and 'non_taxable' are the bucket names. Further expansions on other attributes are shown in Figure 1.5.

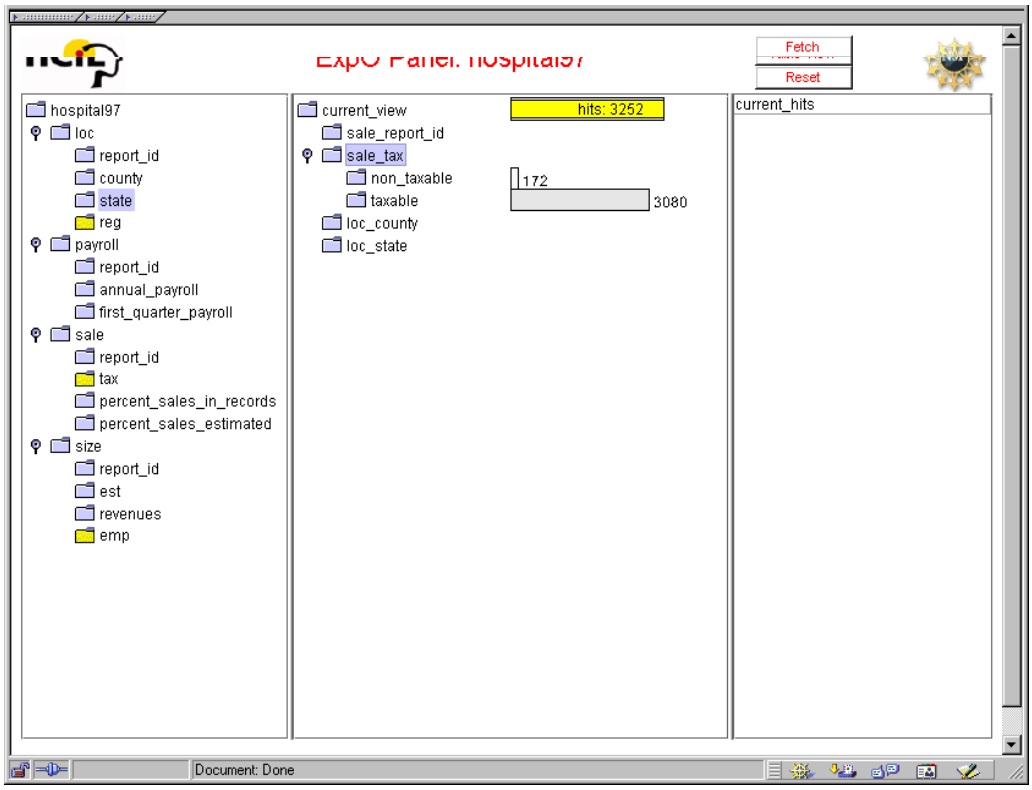


Figure 1.4: ExpO with the data distribution information attached to the buckets of a single attribute expanded in the user view. Two buckets are shown (total 3,252).

In generalized query previews, a preview of the results is displayed. In the ExpO example, a separate bar on the top of the middle panel shows the total number of distinct items mapping to all of the buckets. This is called the result bar (showing 3252 hits). It is a preview for the result set and shows the size of it. Hence, users will be aware of the consequences of their query submissions, i.e. whether they are submitting mega-hit or zero-hit queries, or not. Thus, the result bar helps prevent useless query submissions.

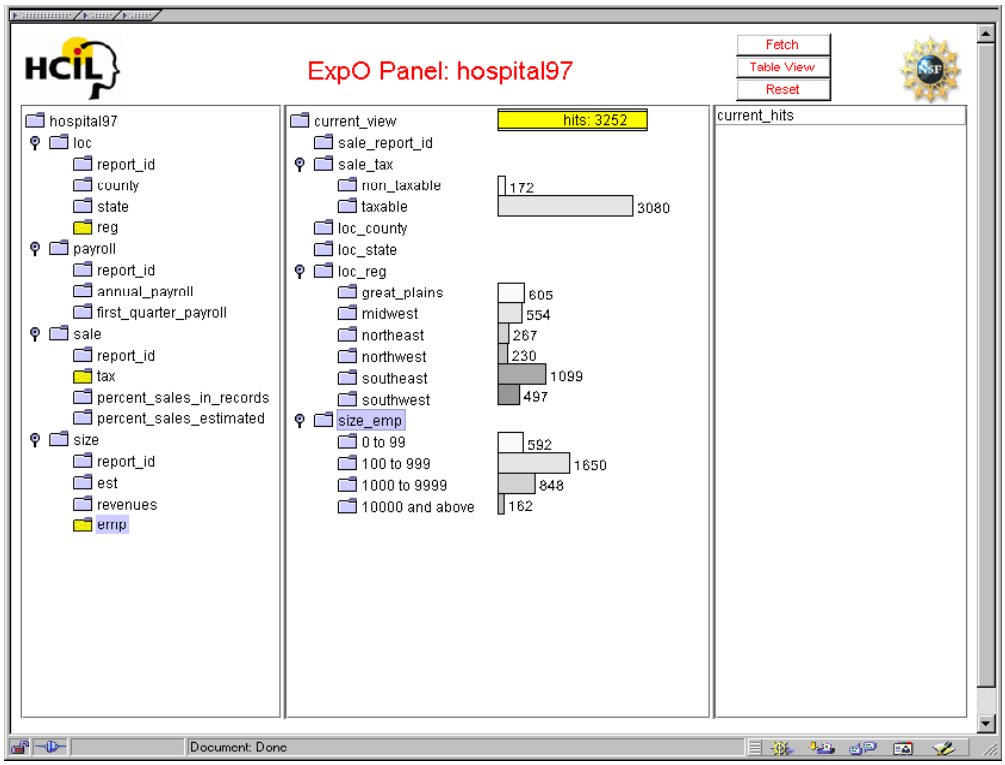


Figure 1.5: ExpO with the data distribution information attached to the buckets of three attributes expanded in the user view (tax, region, employee count)

In generalized query previews, queries are incrementally and visually formed by selecting items from a set of charts attached to the user view. Users continuously get

feedback on the data distribution as they continue their selections. For our example, Figure 1.6 shows a selection. As soon as the selection is made, other charts and the result set preview are updated to reflect the new data distribution satisfying this selection. This is called tight coupling. Possible zero-hit queries immediately become visible to the users. Users also see where data is and how it is distributed over different values even before manipulating the bars. They can play with these interactive charts as long as they want to investigate the contents of the data. Clicking on the visual aids, bars in this case, selects or deselects them. Figure 1.7 shows some further selections on the charts. Selections within a chart map to a disjunction operation. Selections between charts map to a conjunction operation.

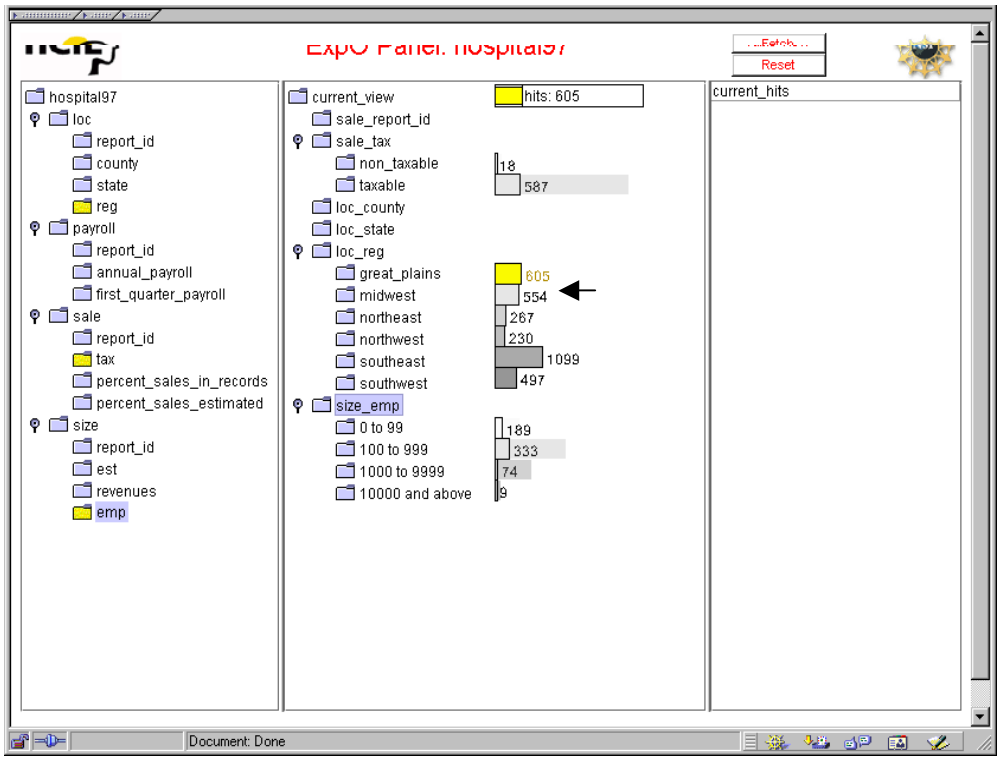


Figure 1.6: The distribution information is attached to the buckets of three attributes expanded in a view and a selection is made on one of them, the 'great_plains' region.

After the investigative selections, users can fetch the desired portions of the data by sending their final selections over the network. As they make informed queries, getting neither zero-hit nor mega-hit result sets is an issue. Hence, the problem of blind formation of queries is solved.

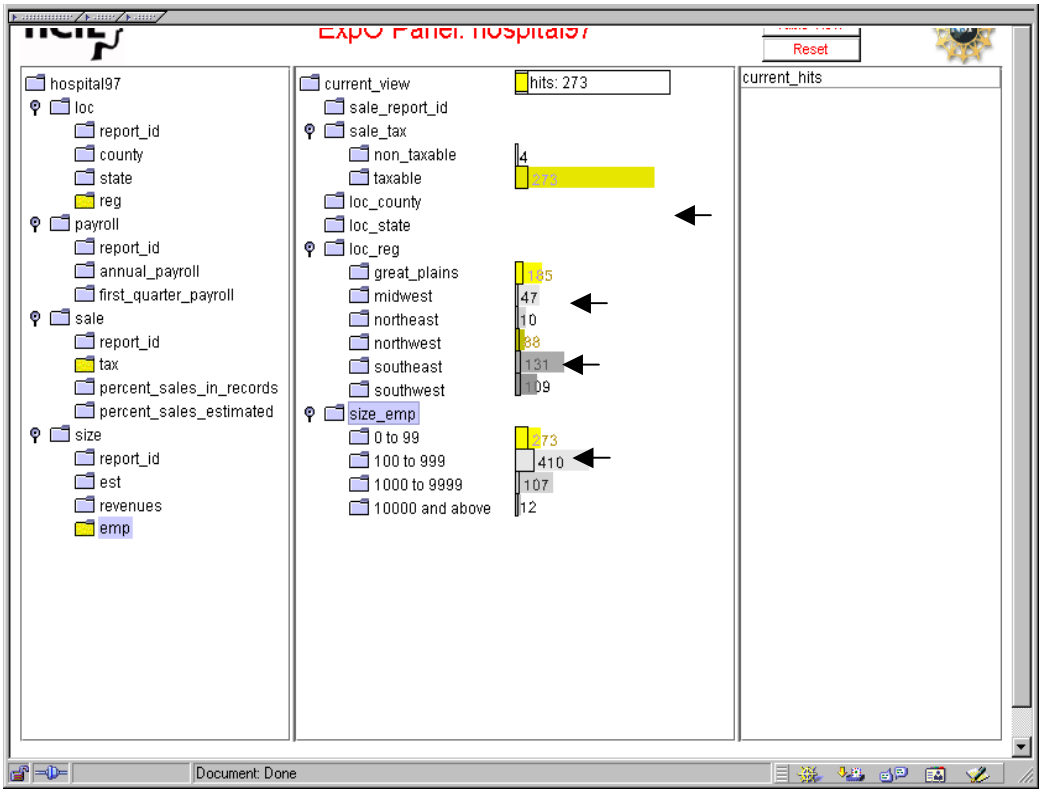


Figure 1.7: The data distribution information is attached to the buckets of three attributes expanded in the user view and multiple selections are made on these buckets, ‘taxable’, ‘great_plains’, ‘northwest’, and ‘0 to 99’.

For some other sample implementations, the designer of the system can even take more drastic measures such as preventing the query submissions for zero-hit or mega-hit

queries, by utilizing a threshold. It is important to note that the information given by the charts is not the probabilistic distribution of data, but the real one.

Figure 1.8 shows a result set displayed on the right side of the ExpO frame as a separate panel. Users can load this result set into a local tool for further analysis of this portion of the data (e.g., Excel). The whole process of pruning and loading a portion of the data can be repeated as long as the user desires. Generalized query previews can enable access to the results in multiple ways. In our example, only the list is shown.

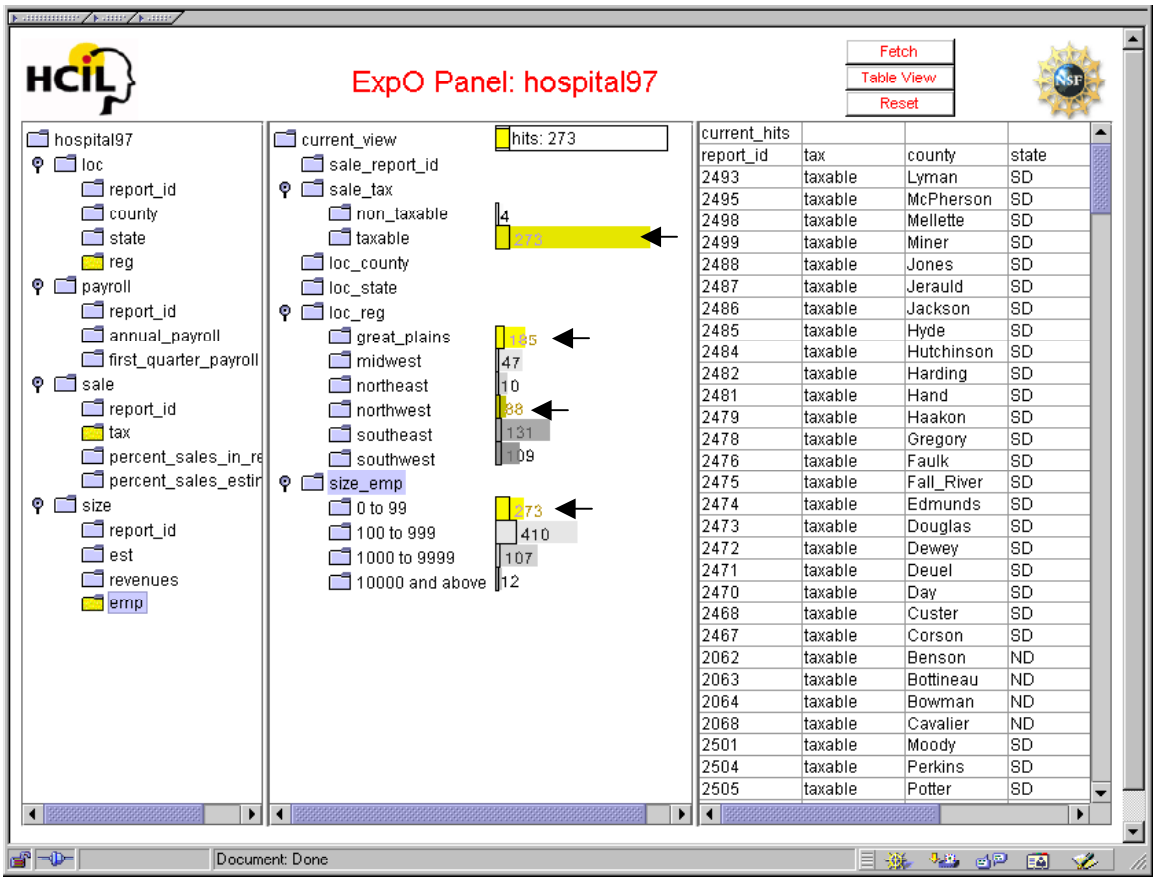


Figure 1.8: ExpO with a result set to a query displayed in a panel that is on the right side of the main frame, 273 hits

Generalized query previews works on data distributions. Expansions of charts generate the requests for the distribution information from the server where the database is kept. The distribution information can be updated periodically on the server.

Distribution information tends to be much smaller than the raw data and does not scale up with the size of the raw data. This aspect of generalized query previews also contributes to better network performance. Only metadata is downloaded from the network until an informed query by the user is made. Raw data is loaded only at this final stage of querying.

1.3 Contents

This dissertation presents the development of generalized query previews. The first ideas started to develop in the Human-Computer Interaction Laboratory of University of Maryland in 1995. Field studies on various platforms including real case applications for NASA helped the development of the ideas. Experimental results showed the strong and weak points of the architecture. As a result of this process generalized query previews has evolved.

This dissertation also presents a detailed architecture of the algorithms and data structures developed for generalized query previews. Development of generalized query previews triggered research on algorithms and data structures. The presentation of the algorithms and data structures is also required for the completeness of this dissertation.

There are three main contributions of this dissertation to the field of visual data mining and information visualization and to the field of algorithms and data structures. First, with this work a general architecture for browsing large online data is formed and

presented. Generalized query previews is not a user interface, but actually an architecture for efficient browsing of large online data. Storage of data distributions, accessing, and viewing these distributions, defining a user view, the algorithms and data structures running behind the scenes, all form this architecture. Different implementations of this general architecture are possible (e.g., ExpO). Second, through field studies and experimentation the application domain for generalized query previews is defined. This dissertation presents not only the strong points of the architecture, but also the shortcomings of it. Third, contributions to the field of algorithms and data structures are made. These contributions are also covered in detail in this dissertation.

Chapter 2 is an introduction to the related work, mostly on the field of visual data mining and information visualization. Chapter 3 presents the roots of generalized query previews and initial implementations. Chapter 4 presents user feedback and the first experimental findings on the initial implementations. Chapter 5 gives a more detailed explanation of generalized query previews. Chapter 6 discusses the algorithms and data structures that support the architecture. Chapter 7 gives the second experimental results. Chapter 8 concludes with a summary of this work, its contributions, benefits, and some possible future work.

CHAPTER 2:
RELATED WORK

2.1 Field of Research

Visual data mining and information visualization researchers are working on effective visual methods for browsing and manipulating abstract information spaces [CMS99] [GE97] [Hear99] [Shn98]. Most of these methods rely on bar charts, scatter plots, and other means of visual explanations [Ber83] [Tuf83] [Tuf90] [Tuf97]. Generalized query previews is a part of this field of research.

This chapter presents a general introduction to the field of visual data mining and information visualization (2.2). It provides an overview of some of the known taxonomies and references to some of the key papers of this field. Then, since generalized query previews deals mostly with multi-dimensional data, it focuses on browsing and manipulating this type of data. Multi-dimensional data visualization started to form a separate category of research under the field of visual data mining and information visualization [BCS96] (2.3). Thus, papers of this category are discussed in a separate section.

Finally, as generalized query previews relies on the following three pillars of research, I present the papers that are relevant to each of these pillars under the related sections (2.4, 2.5, 2.6), summarizing with a section on future directions (2.7):

- Online Data Visualization (2.4): Generalized query previews introduces an architecture for browsing ‘online’ data,

- Large Data Visualization (2.5): Generalized query previews helps users browse ‘large’ data,
- Database Schema, Tables, and Query Visualization (2.6): Generalized query previews uses data stored as multiple tables of a ‘database’.

2.2 Visual Data Mining and Information Visualization

Visual data mining and information visualization researchers are interested in abstract information spaces. Examples of such spaces are stock market data, document databases (e.g., a patent database), a database of films, patient records from a hospital database, computer logs, web logs, etc. Abstract information spaces cannot be easily mapped onto a 3D world coordinate system unlike other information spaces, such as the landscape of a country, structure of a machine, or a 3D scan of a human brain. For example, a patent database cannot use a reference coordinate system (like a map of a city) that makes it easily comprehendible to the users. The non-abstract information spaces are generally considered to fall within the research realm of the field of scientific visualization [Kau91] or geographical information systems.

Visual data mining is a term that is commonly used to refer to the action of finding something interesting in information spaces (e.g., as gaps, clusters, trends, or sometimes just a single item). Information visualization is a term that is generally used to refer to the methods utilized to help users see what these spaces look like.

Recent work at the University of Maryland and Xerox PARC introduced the first taxonomies of the field of visual data mining and information visualization [Chi00] [Shn96]. In these taxonomies researchers used data types, tasks, and data states to define the different categories of the field. The first popular taxonomy by [Shn96] introduces the

visualization categories of 1D, 2D, 3D (e.g., [ESS92] [Hear95] [RM93]), tree (e.g., [Fei88] [JS91] [KPS97] [LR96] [RMC91] [SFR00]), multi-D (multi-dimensional or multi-variate) (e.g., [Rot00] [Shn94]), temporal (e.g., [PMR96]), and network (e.g., [BEW95] [Eic93]) visualizations. Some researchers also include workspaces as a separate category (e.g., [CRM91] [KS98]). Among these categories, generalized query previews falls under the category of multi-D (multi-dimensional) visualizations.

2.3 Multi-dimensional Data Visualization

Multi-dimensional data contains multiple equal weight attributes. Unlike temporal or network data, the relation between attributes is not implicit and does not dominate the organization of the data. Hence, many network and temporal data sets can also be considered as multi-dimensional data sets by rearranging them depending on the application domain.

Common tasks that can be performed with multi-dimensional data are understanding or obtaining an overview of the whole or a part of the multi-dimensional data by finding patterns, relationships, clusters, gaps, and outliers of the data or finding a specific item in the data by zooming and filtering the data.

Generalized query previews uses data that is stored in multiple relations of a database. Hence, the relation between the attributes of the data is explicit. Generalized query previews uses the same methods for all the attributes of the data with no specific precedence. Hence, all the attributes of the data are assumed to have equal weights. Generalized query previews uses metadata to give an overview of the data (e.g., data distributions). Metadata is used to understand the trends and patterns of the data. Data can be filtered and portions of it can be downloaded. All of these aspects of generalized query

previews brings it closer to the research category of multi-dimensional data visualizations.

The rest of this section will discuss some popular multi-dimensional data visualization systems. Most of these systems assume that local access to data is available and they work on random access memory rather than the secondary storage devices. Hence, networked access to large data sets is not addressed by many of these systems.

Visage: Carnegie Mellon University developed this multi-dimensional data visualization system in the early 1990's [RLS96]. Visage aims to coordinate the exploration of information across different types of visual aids regardless of the source and type of the information. For example, as shown in Figure 2.1, users can drag and drop the spreadsheet view of some geographical data onto a map to display its distribution over the map.

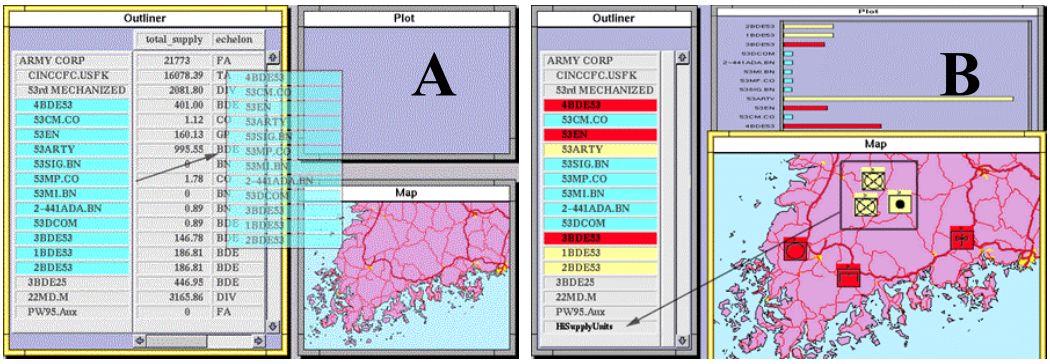


Figure 2.1: Visage from Carnegie Mellon University, in snapshot 'A' user selects and drags a list of values onto a map, in snapshot 'B' user views the values on the map

Table Lens: At the same time Rao and Card of Xerox PARC were working on the Table Lens [RC94]. This multi-dimensional data visualization tool used a focus+context technique based on a fisheye view of a spreadsheet, Figure 2.2. In this technique a group

of rows and columns are in focus while the rest of the spreadsheet appears to be out of focus. Yet the out of focus parts of the data give an overview of the data with the help of histograms and color-coding.

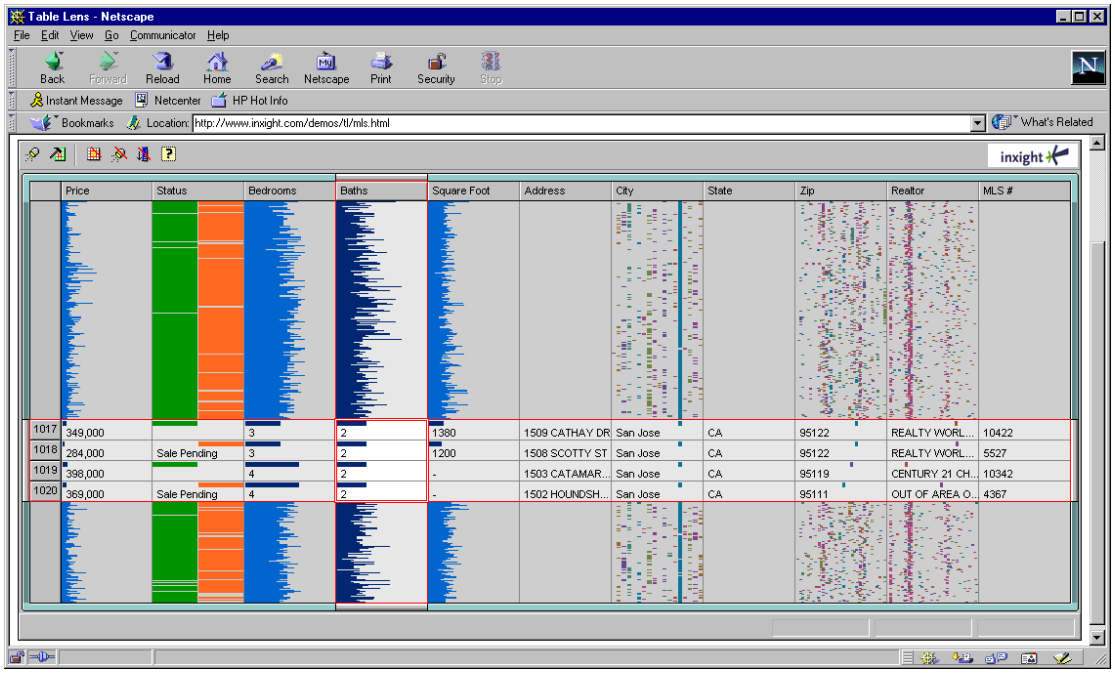


Figure 2.2: Table Lens of Xerox PARC, four rows from a large spreadsheet are in focus while the rest of the spreadsheet is represented by histograms and charts

Magic Lens: Again at the same time, Stone, Fishkin, and Bier of Xerox PARC developed the Magic Lens [SFB94]. Magic Lens views the data using a tool that works like a magnifying glass over a text or a graphical document. In Figure 2.3, the user views a map with two separate lenses to identify two different portions of the map, major roads and water canals.

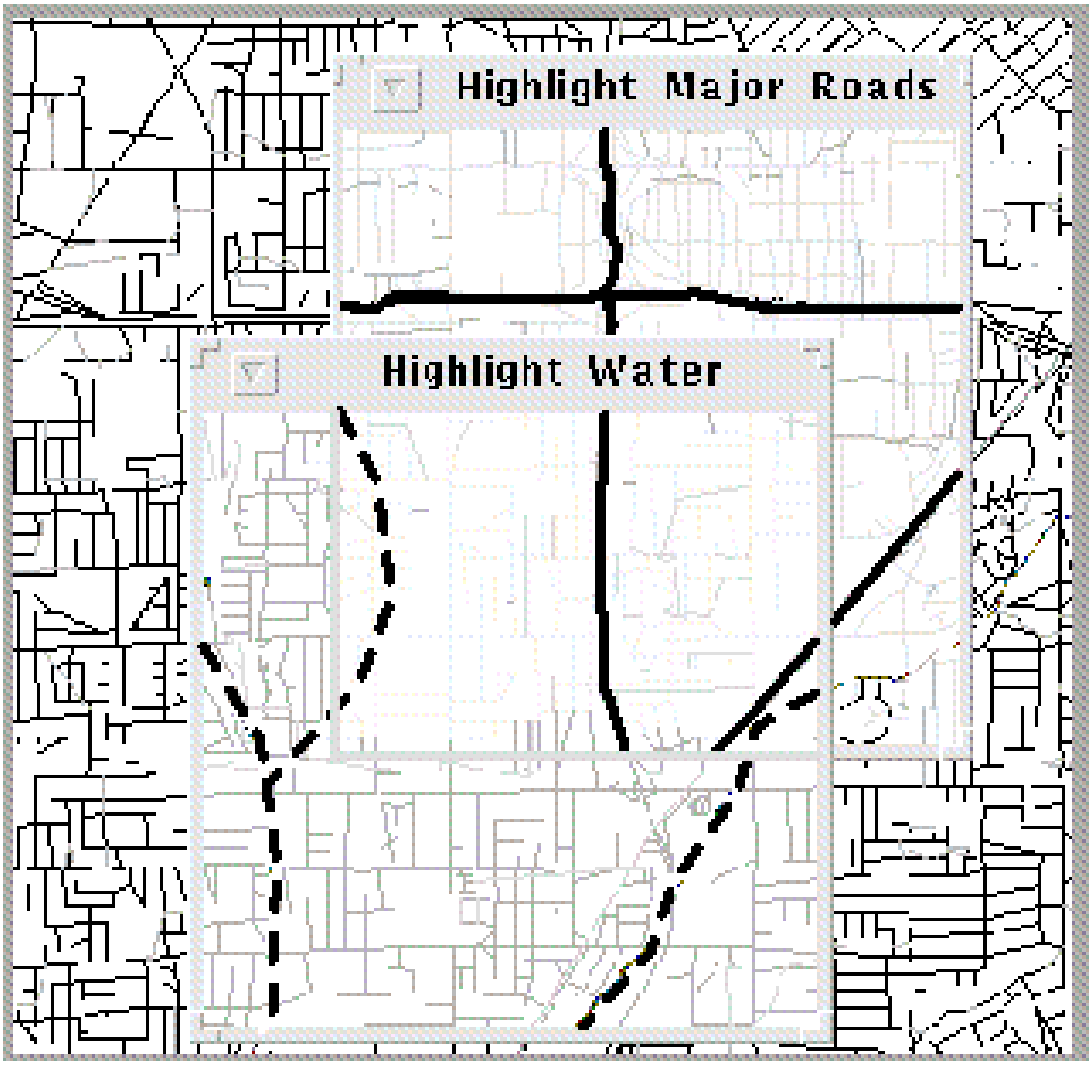


Figure 2.3: Magic Lens of Xerox PARC, the user is using two lenses on a city map, one for highlighting the major roads and the other one for highlighting the water lines

XGobi: In 1990's, Swayne, Cook, and Buja came up with the XGobi system [SCB98]. XGobi shows various visualizations of the same data set simultaneously and dynamically, Figure 2.4. It also takes various projections of a multi-dimensional data set on some of the attributes of the data and then animates these projections.

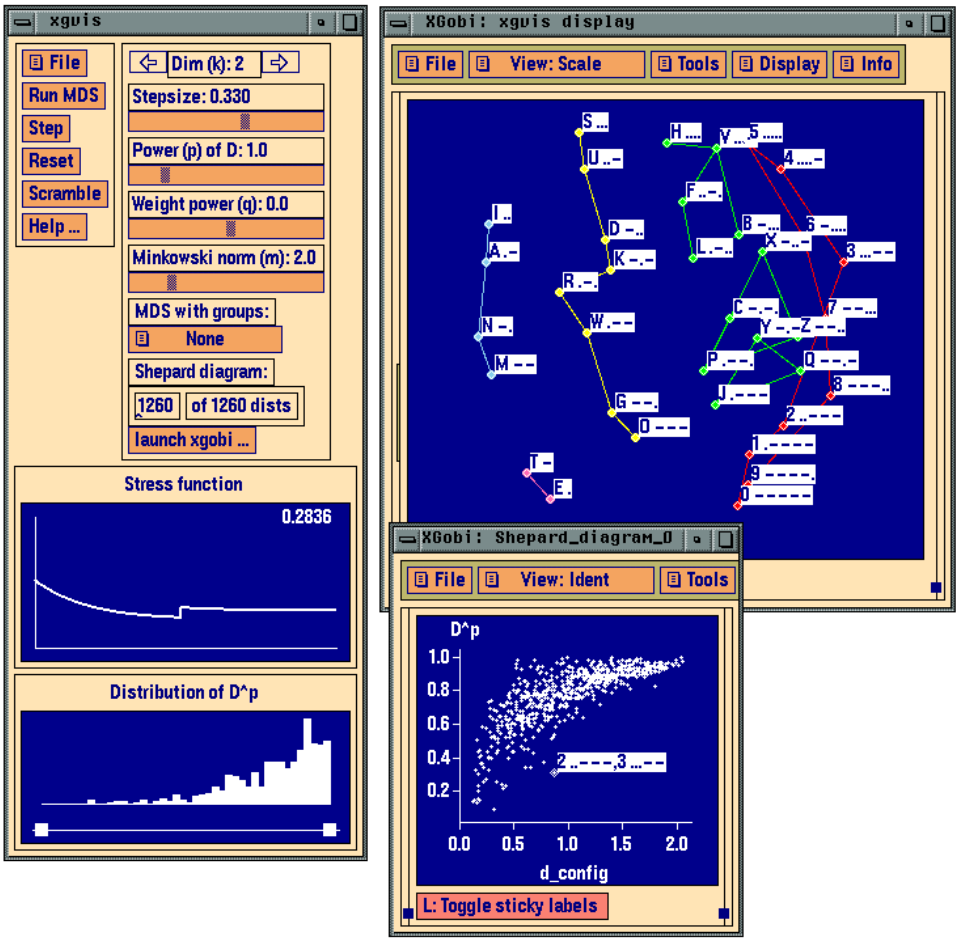


Figure 2.4: XGobi shows multiple projections of a data set (e.g., histograms, scatter plots, etc.)

Attribute and Influence Explorers: In Imperial College, U.K., researchers developed the Attribute and the Influence Explorers in mid 1990's [TWS94] [TSD96]. The idea behind both of these systems is to have interactive histograms to view the contents of the data while, giving an overview about the contents, Figure 2.5 and 2.6. The selections on a histogram immediately update others. A similar visual representation and feedback mechanism is also used for the generalized query preview example implementation, the ExpO system.

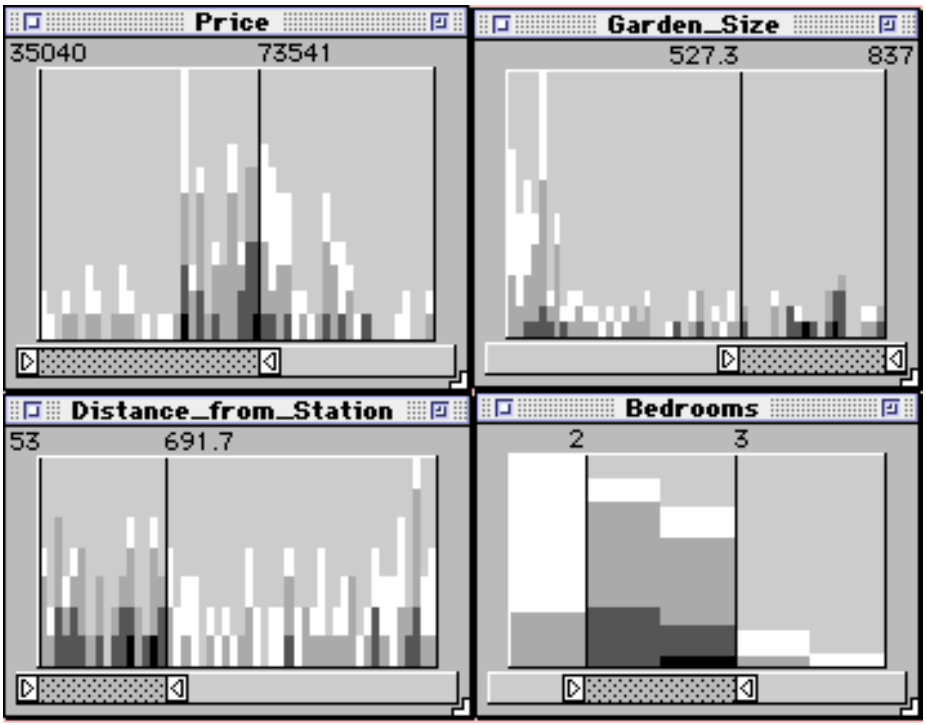


Figure 2.5: Attribute Explorer, with four interactive histograms

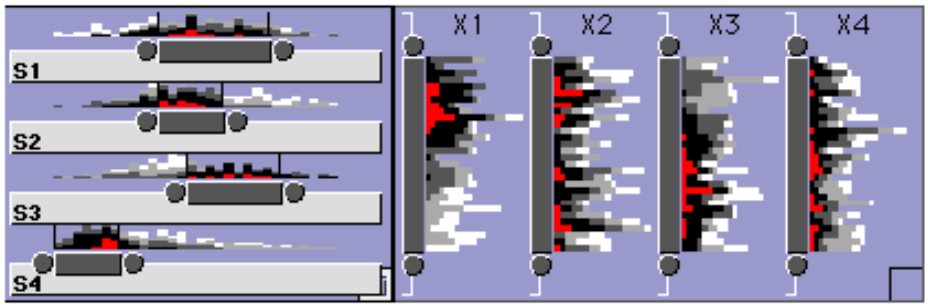


Figure 2.6: Influence Explorer, with many histograms

Parallel Coordinates: Inselberg and Dimsdale, in late 1980's, took a thoroughly different approach to visualizing multi-dimensional data [ID89]. Instead of viewing data using perpendicular multiple coordinate axis, they used a series of parallel axis, Figure 2.7.

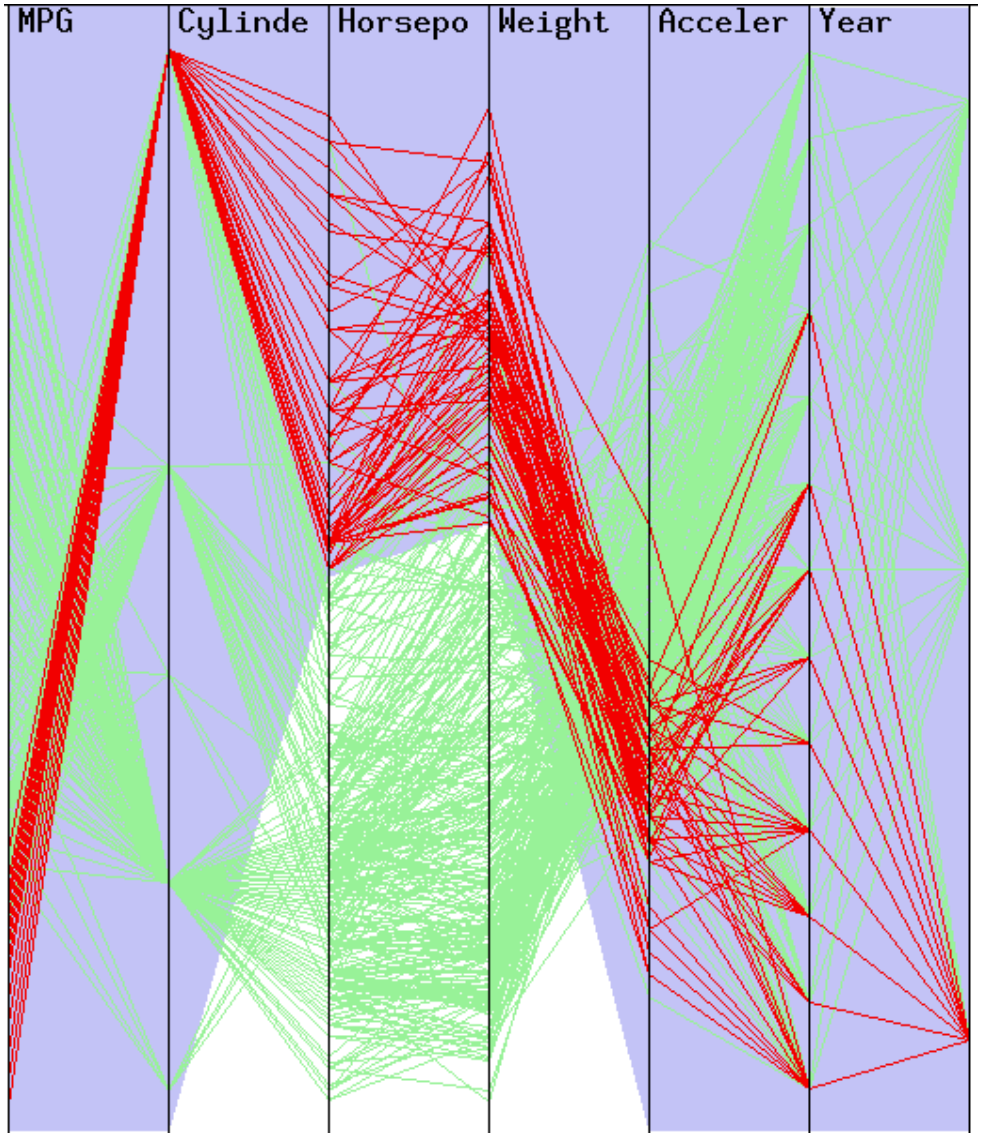


Figure 2.7: Parallel Coordinates, cars from a car database are represented by lines crossing through six vertical axes representing the six dimensions of the data

Worlds within Worlds: Feiner and Besher of Columbia University developed this system. The system is designed for the exploration of multi-dimensional coordinate systems containing arbitrary functions using nested coordinate axis. They show users parts of the whole world in a simple but effective manner [FB90], Figure 2.8.

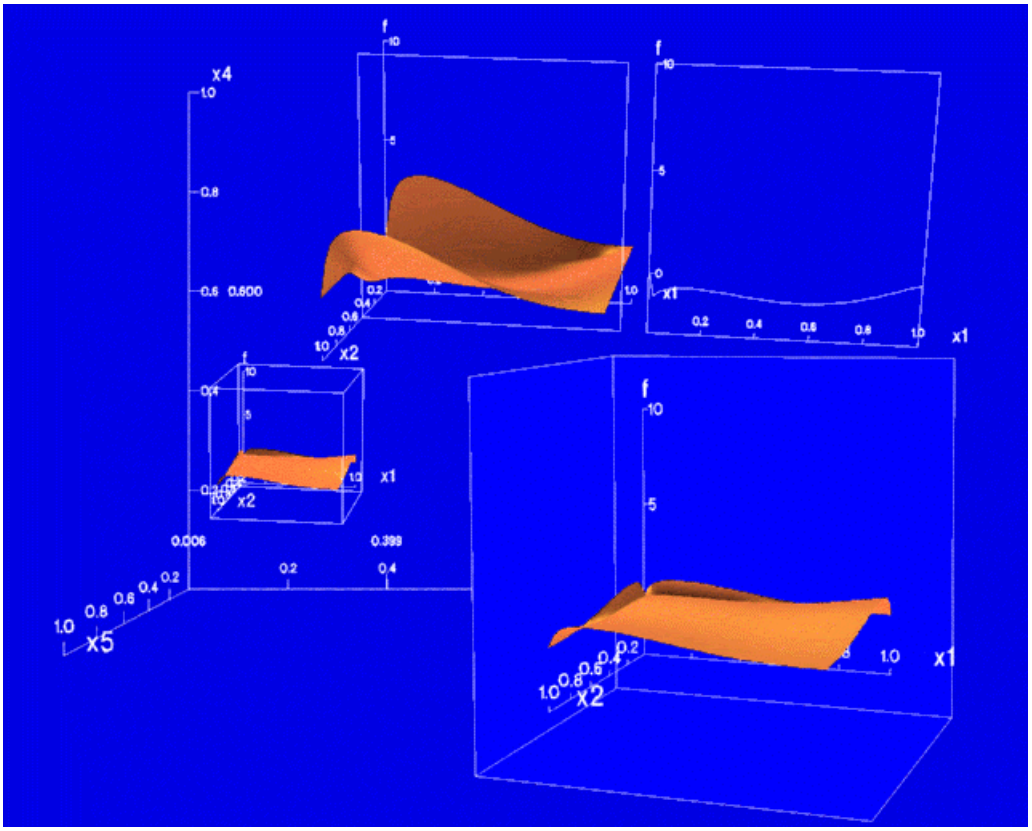


Figure 2.8: Worlds within Worlds, 3D projections of a multi-dimensional world

Data Visualization Sliders: Eick, in 1994, used sliders not only as a mechanism for user input, but also as a means to show data distributions in the form of density plots [Eic94].

Selective Dynamic Manipulation of Visualizations (SDM): SDM is also from Carnegie Mellon University [CRM95]. SDM is actually a set of interactive techniques for 2D and 3D visualizations. Visualizations in SDM are linked, and real-time interactive animation techniques are used. Figure 2.9 shows a popular view from SDM where a multi-dimensional data set is displayed as a landscape of color-coded bars that are linked together, indicating a certain relationship between certain groups of bars.

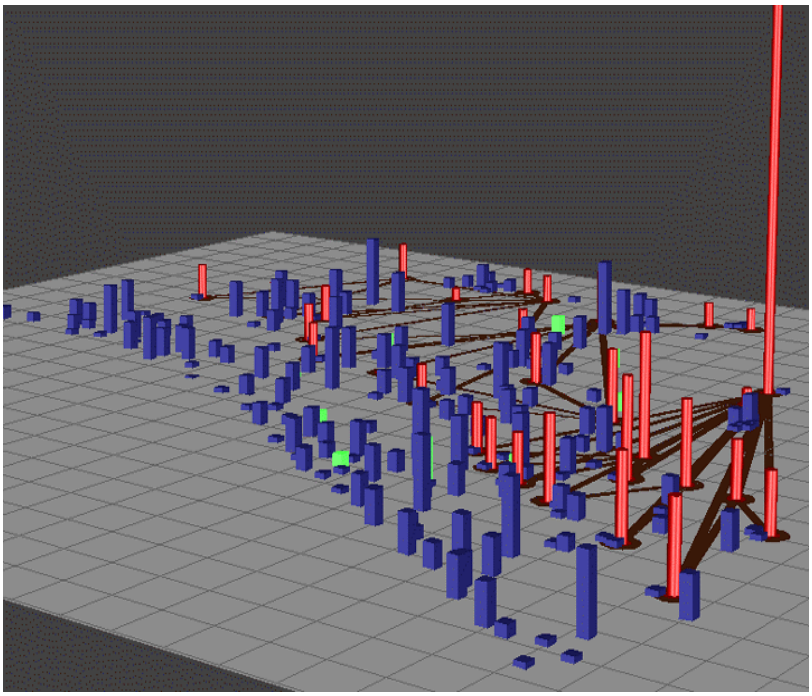


Figure 2.9: SDM from Carnegie Mellon University showing a landscape of data distribution

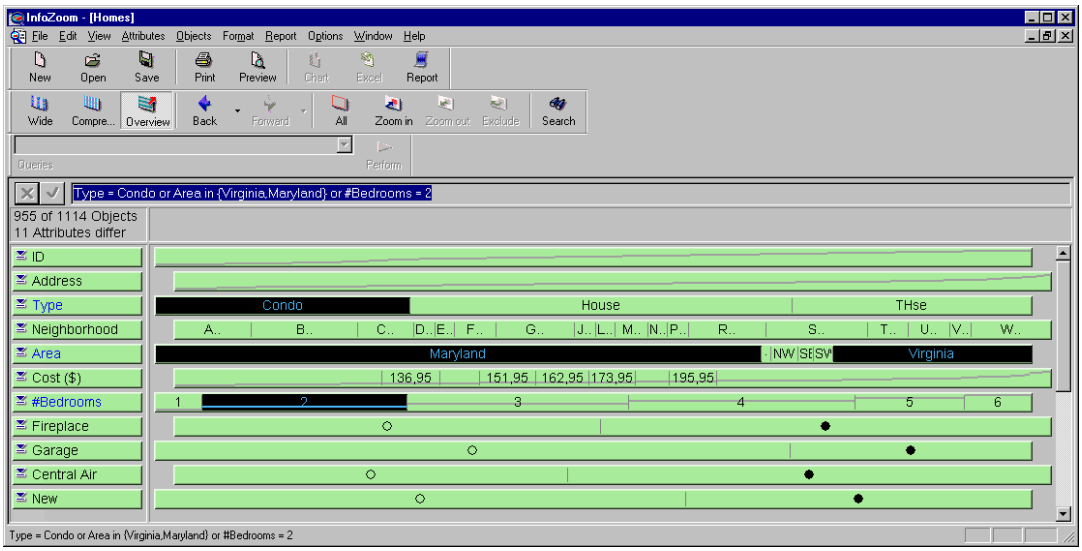


Figure 2.10: InfoZoom (available from www.humanit.de) showing the overview of data by size-coding and simple graphs on multiple bars representing the attributes of data

InfoZoom: This is a tool that also provides an overview of the data (Figure 2.10). Any part of the data can be selected, filtered out, or zoomed in using highly interactive controls. Queries can be captured and replayed. Results can be saved as charts or as interactive objects.

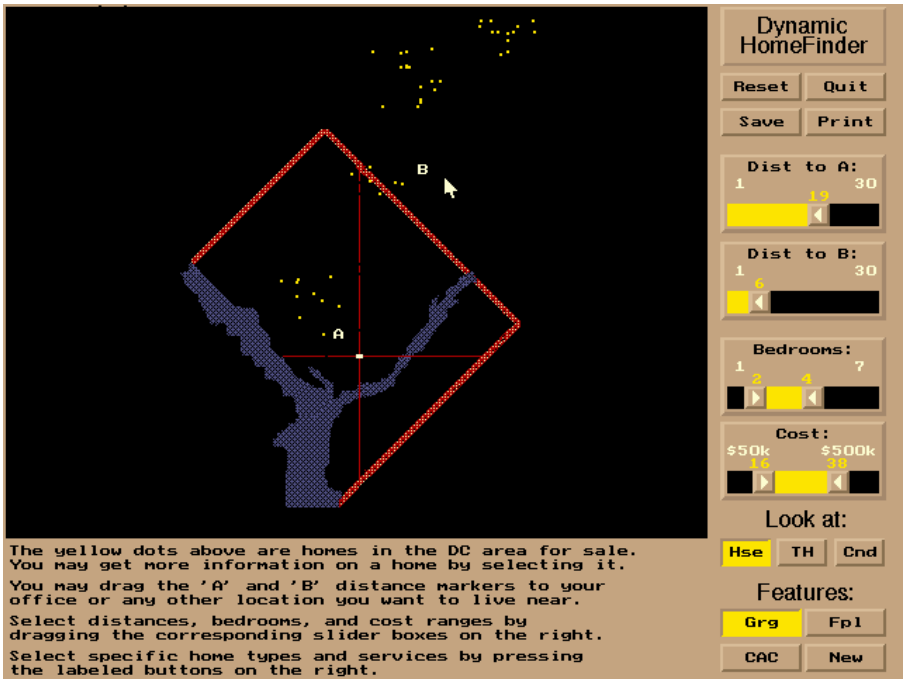


Figure 2.11: The first dynamic query example, Home Finder, using a real estate data set from Washington, D.C., users can adjust the widgets on the right to manipulate the six different dimensions of the data, updates are immediate on the map

Dynamic Queries: Dynamic queries from the Human-Computer Interaction Laboratory at the University of Maryland emerged in the early 1990's. First, Williamson and Shneiderman in 1992 developed a tool to explore some real estate data set [WS92], Figure 2.11. Later, this idea evolved and triggered a product called Spotfire (available from www.spotfire.com) [AS94] [AW95] [AWS92]. Spotfire is a more general tool to

explore multi-dimensional data, Figure 2.12. In dynamic queries, users formulate queries with graphical widgets, such as sliders. Users can see a graphical visualization of the data and their search results. They can also filter and zoom into parts of the data. Actions are easily reversible. Feedbacks on users' selections are immediately available on all the widgets and charts of the interface.

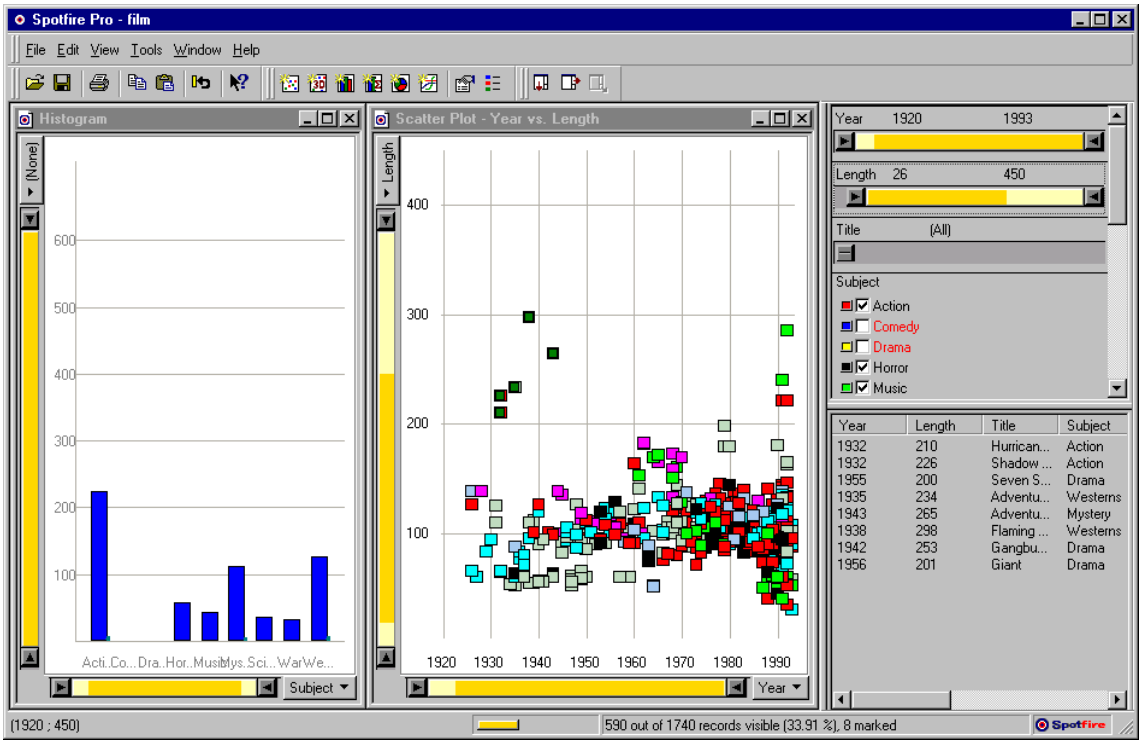


Figure 2.12: The general dynamic query tool, Spotfire, from www.spotfire.com, showing multiple views of a data set

Data Desk: This is a statistical data analysis package available from www.datadesk.com (Figure 2.13). Data Desk provides interactive tools for data analysis. Visualizations are linked together and actions on a single view update the other views immediately.

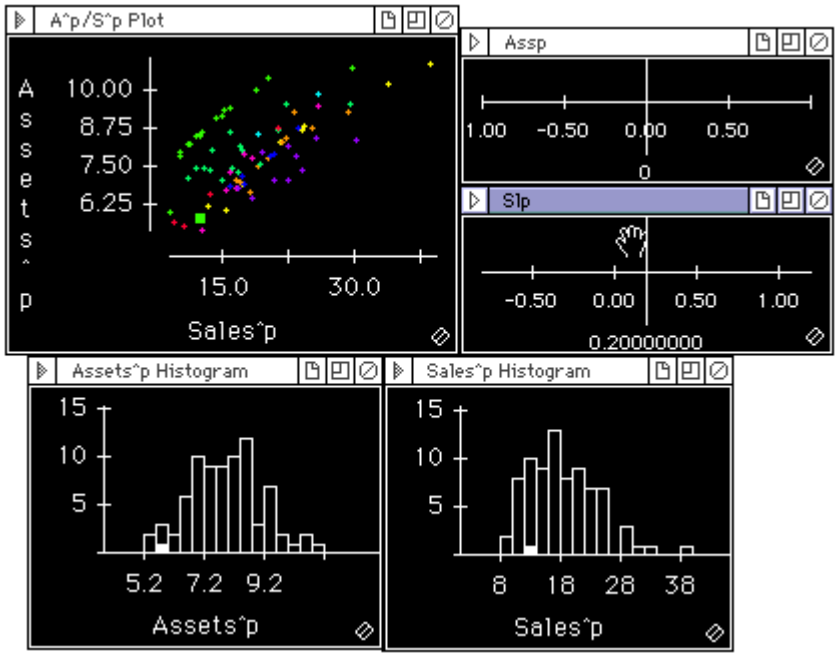


Figure 2.13: Data Desk showing multiple linked views of a data set

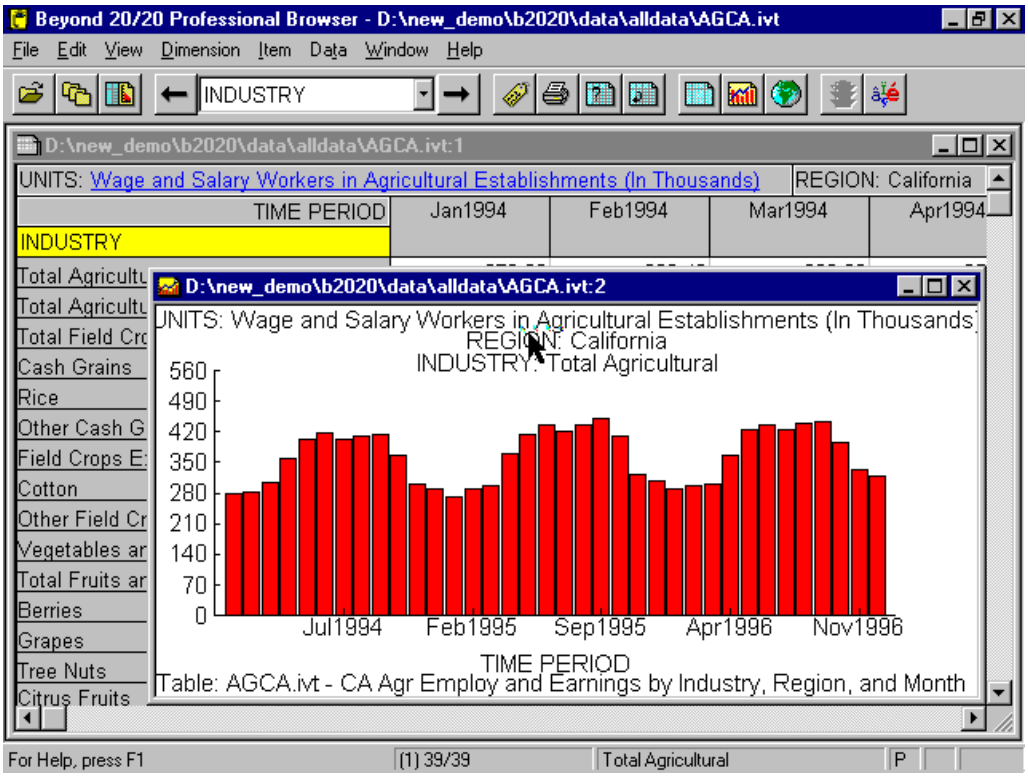


Figure 2.14: Beyond 20/20 showing a statistical data set with a bar chart

Beyond 20/20: This is another tool that can be used for statistical data analysis (available from www.ivation.com), Figure 2.14. This tool is designed to be an advanced spreadsheet (in comparison to the other tools, e.g., Data Desk).

2.4 Online Data Visualization

Online data has certain features that require a special treatment. It is distributed over a network and cannot be immediately accessed at all times. The following are a few known systems that work with online data (mostly text-based data):

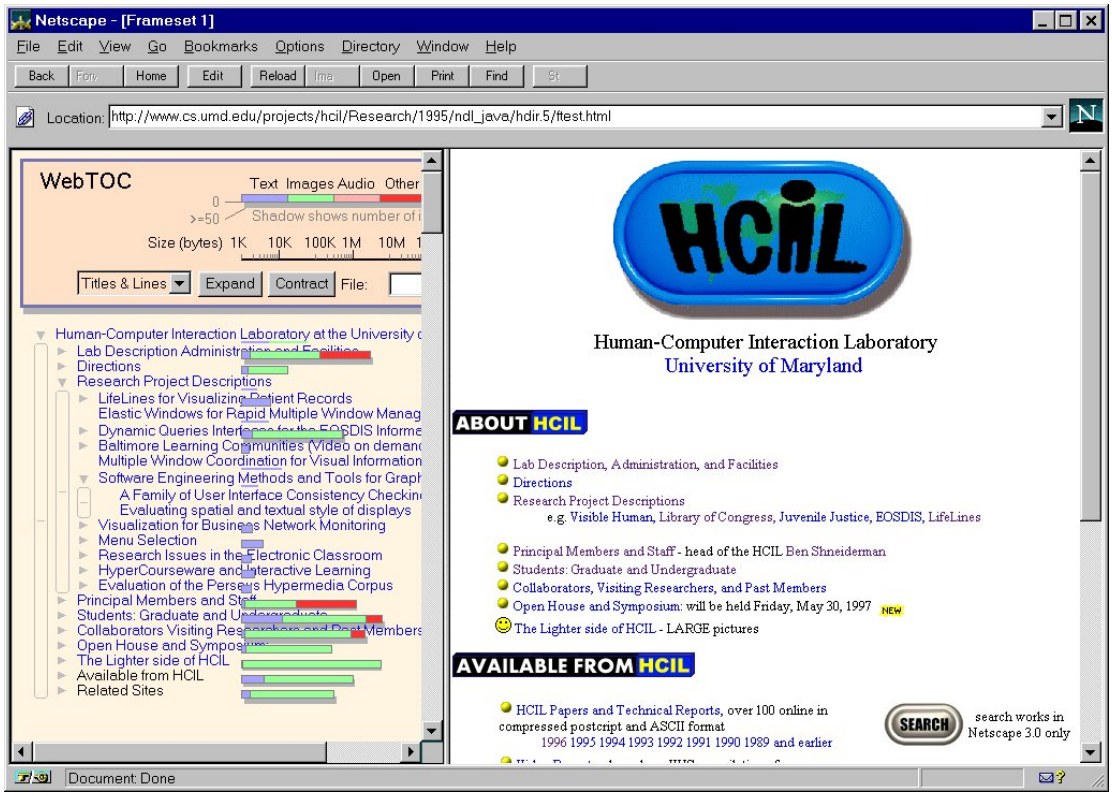


Figure 2.15: WebTOC uses a hierarchical browser to show the contents, type, and the size of a website.

WebTOC: This system is developed in the Human-Computer Interaction Laboratory of the University of Maryland. Nation, Plaisant, Marchionini, and Komlodi worked on this system to visualize websites using a hierarchical table of contents browser [NPM97], Figure 2.15. The ExpO System also uses the same method, but for visualizing the schema of a relational data set.

Butterfly: The Butterfly System by Mackinlay, Rao, and Card [MRC95] is a system for searching citation links across the Internet. Butterfly integrates the action of searching, browsing, and access management. Visualization is used in displaying retrieved information and combines the search and browsing tasks. Citations and links between citations is displayed, Figure 2.16.

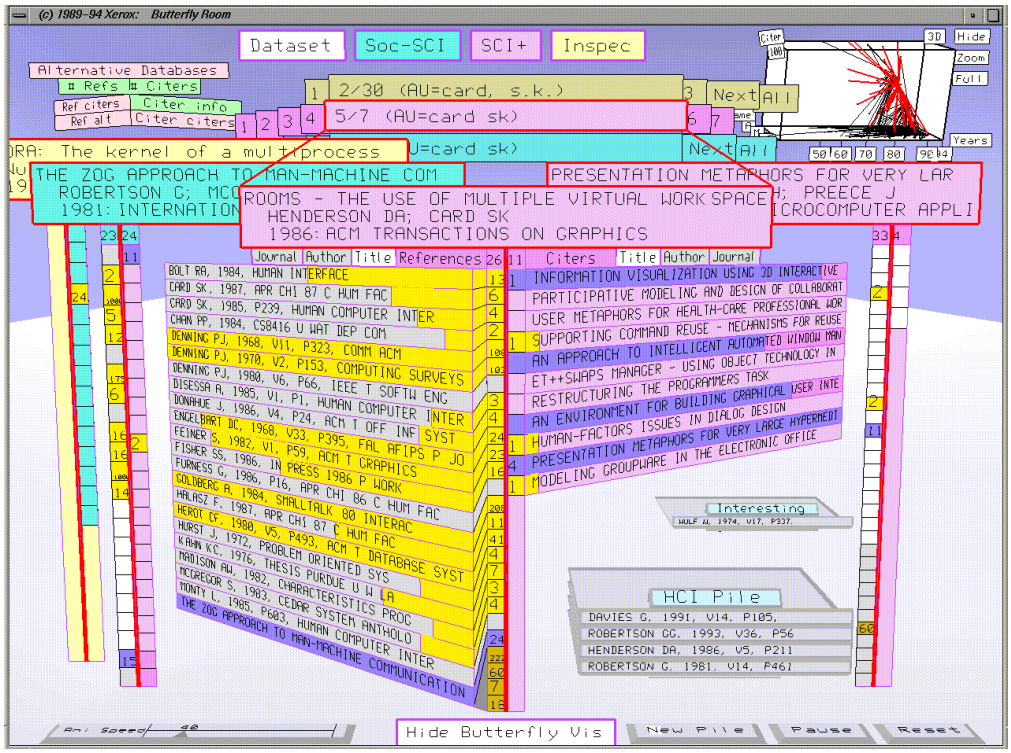


Figure 2.16: Butterfly Citation Search System displaying some citation search results

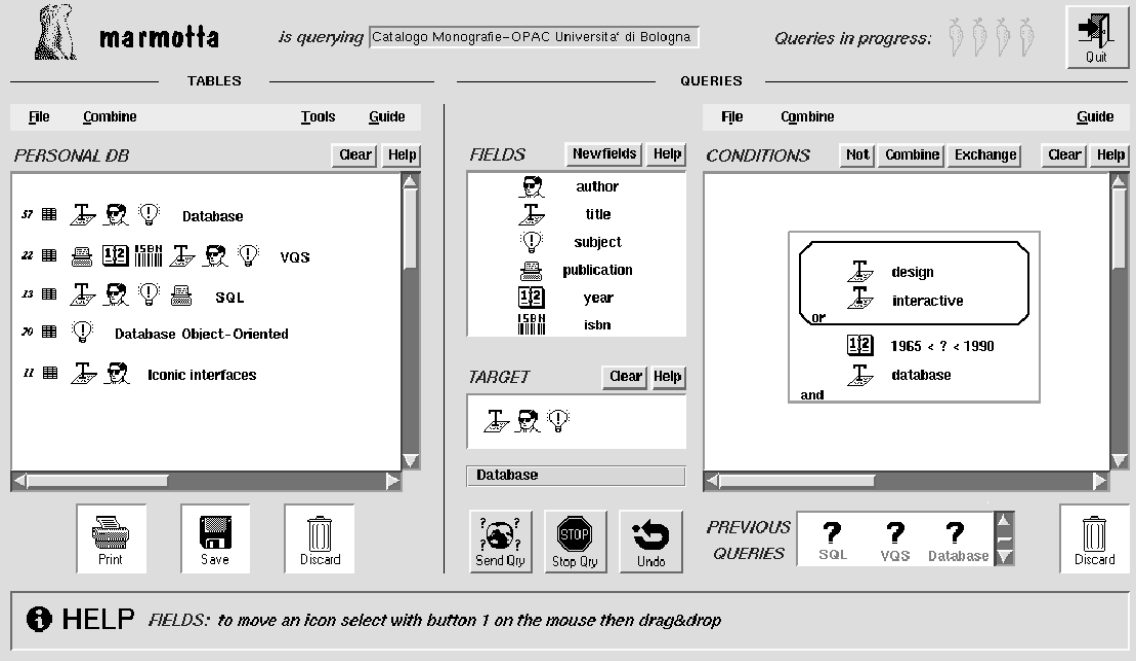


Figure 2.17: Marmotta Iconic System where queries are formed using simple icons

Harvest: Bowman, Danzig, Hardy, Manber, and Schwartz introduced the Harvest System in 1994 [BDH94]. Harvest works on the Internet and provides users with customizable tools to collect information from different websites. However, Harvest uses a very common querying technique for the Internet, the keyword searching technique.

Marmotta: Marmotta is a querying system for networks [CMP95]. It uses progressive querying and works on the Internet. The main idea behind Marmotta is the formulation of queries via simple icons representing actions and items, Figure 2.17.

Envision: Envision is a project from the Virginia Polytechnic Institute and State University. The project developed a large digital library of computer science publications. It is available over the Internet [Heat95], Figure 2.18. It has a highly interactive results screen.

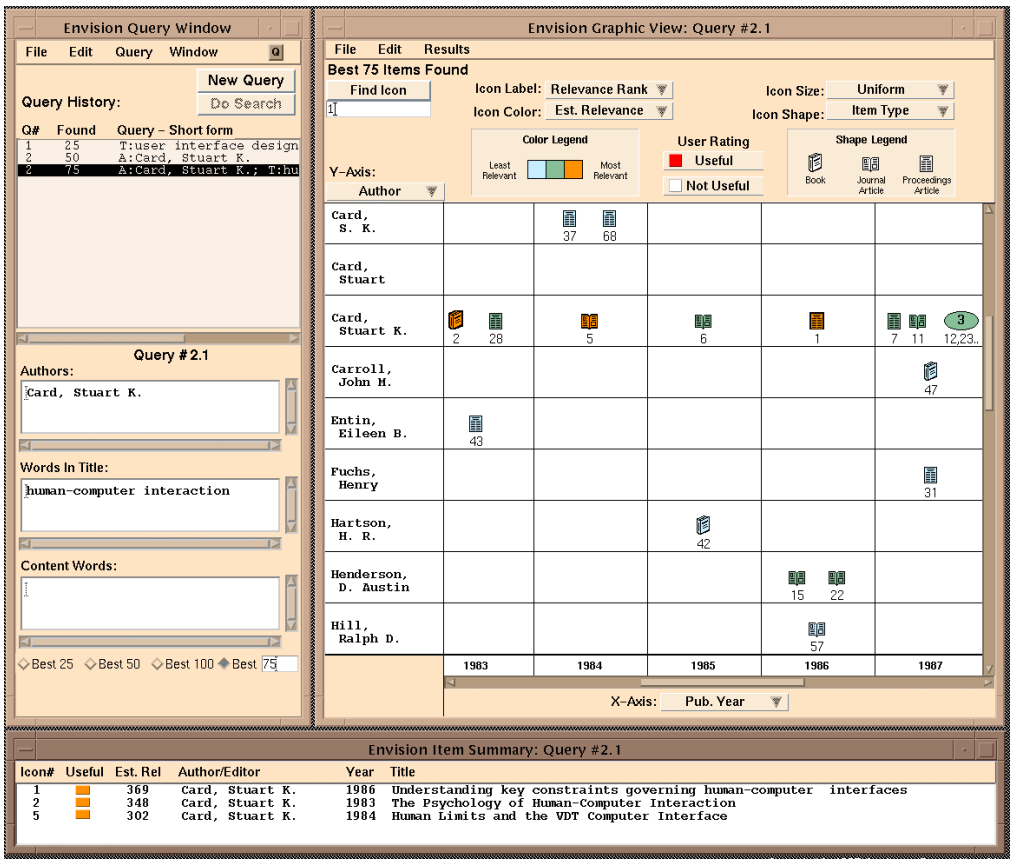


Figure 2.18: Envision, search results, authors are on the y-axis, years are on the x-axis

SuperBook: SuperBook [ERG89] is an early hypertext browsing system. It precedes many of the current web-based systems. It is designed to improve the usability of conventional documents. Especially, the later versions of the SuperBook was implemented to improve the search accuracy and speed.

2.5 Large Data Visualization

Visualization of large data sets forms a challenging field of research. Users requiring highly interactive systems to explore large data sets face many difficulties. For example, having a large data set causes several operations to be slow. Not only querying, storing, and accessing these systems is difficult, but also visualizing cluttered information spaces

becomes a cumbersome task. Many such data sets are stored in large relational databases on secondary storage devices, where, in certain cases, loading even a part of the data into the main memory of a personal computer may be problematic. A few example systems that work with large data are:

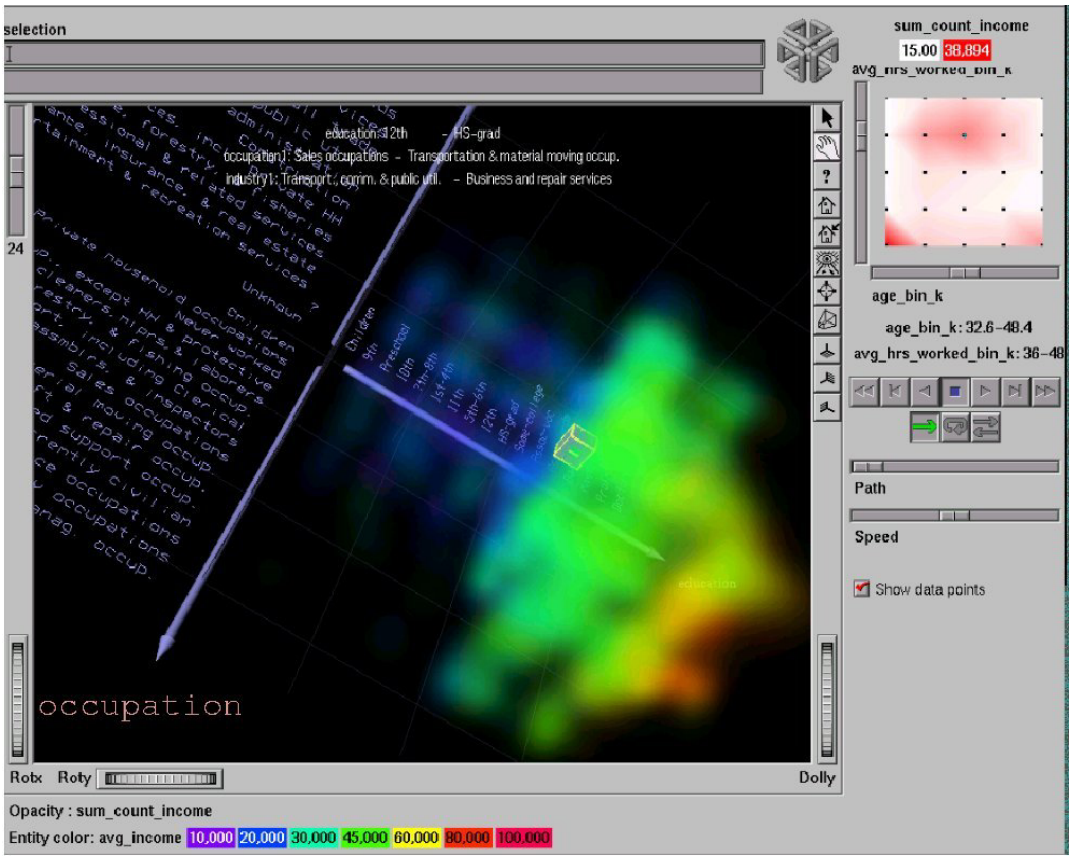


Figure 2.19: Volume rendering of relational data over three dimensions and color-coding is used

Volume Rendering: Volume rendering is a common method for displaying large scientific data (e.g., computer aided tomography images of a human brain). In late

1990's, Becker applied this method to abstract information spaces [Bec97]. He rendered volumetric abstract data stored in a relational database, Figure 2.19.

VisDB: Keim and Kriegel used a single pixel of the screen to represent a record from a database [KK94]. Their system is called VisDB. VisDB colors and organizes the database records with respect to the query result relevancies, e.g. spirals, Figure 2.20.

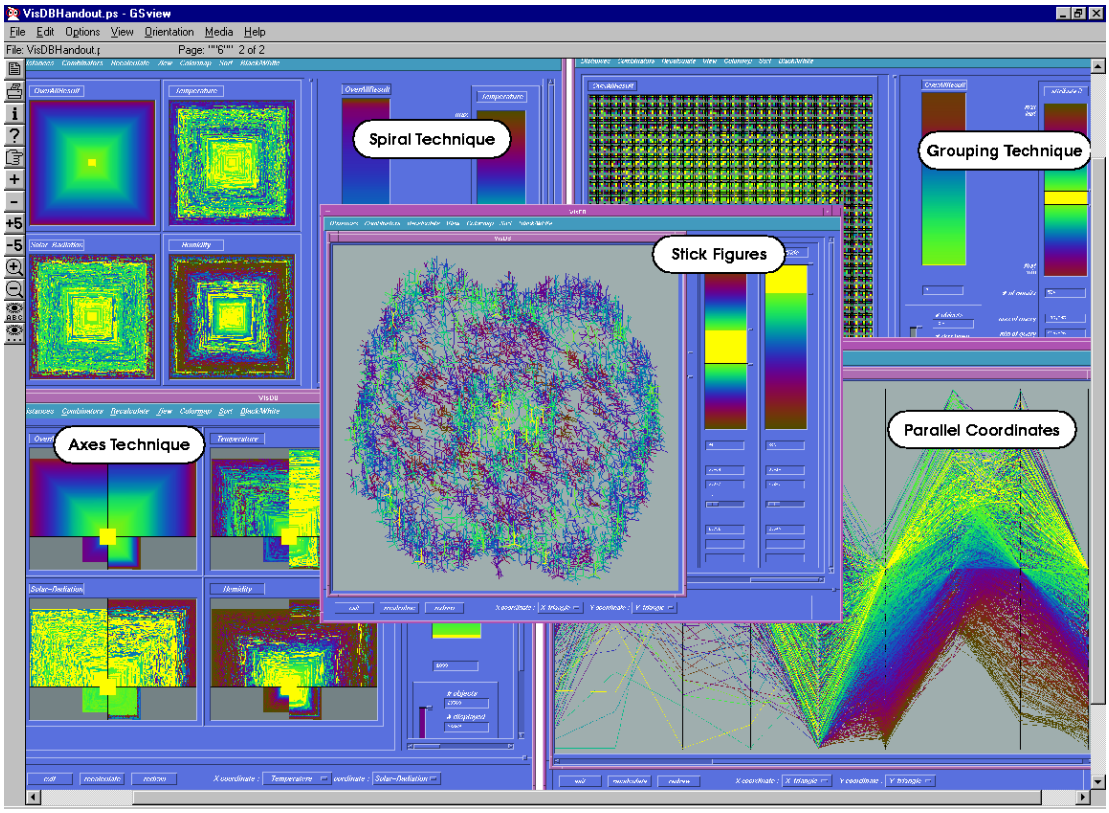


Figure 2.20: VisDB uses spirals and color-coding to show the relevance of results to the query and gives multiple views of these results (e.g., parallel coordinates).

Aggregation and Dynamic Queries: Goldstein and Roth [GR94] used aggregation with dynamic queries to help users with large data sets. They used a mechanism called the Aggregate Manipulator and combined it with dynamic queries, Figure 2.21. The

distribution of data is available to the users in this system. Generalized query previews also uses aggregation to make large amounts of data explorable over a congested network. This approach is very scalable, as the distribution information on data does not drastically change with the data itself.

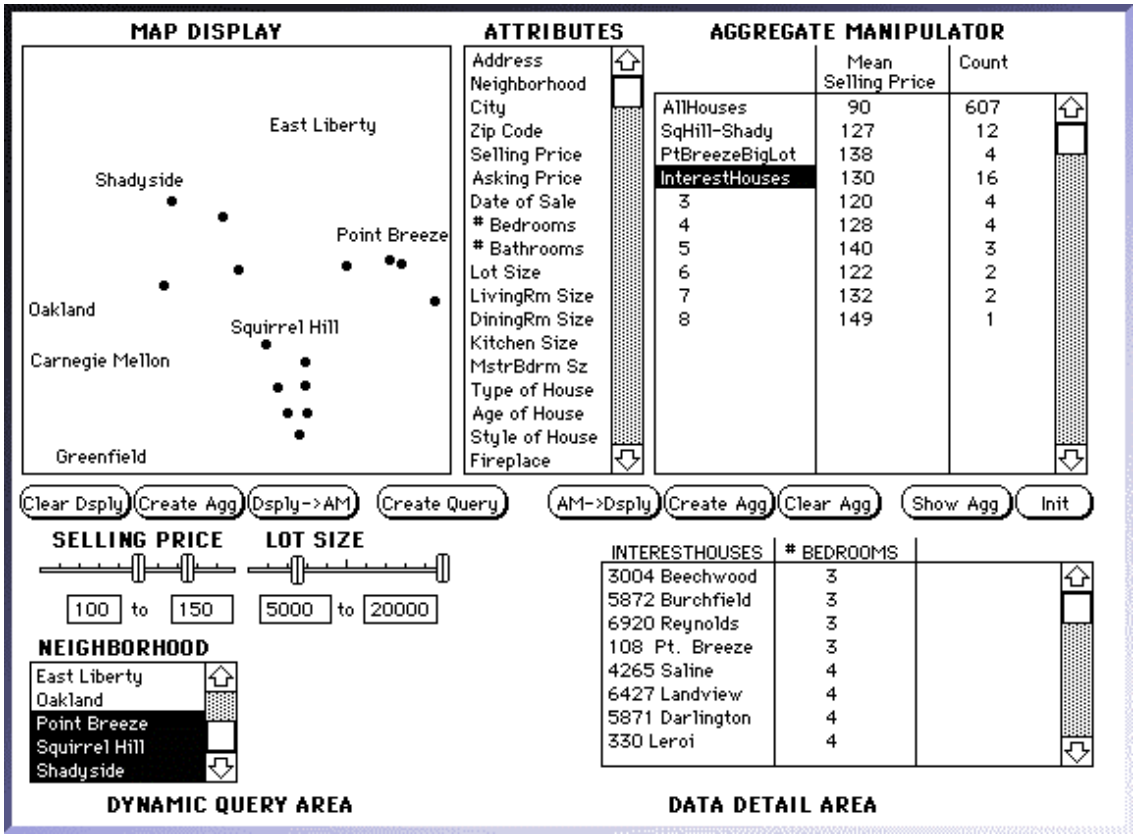


Figure 2.21: Using aggregation in tandem with dynamic queries, users can see how much of each type of an item exists in the data

LinkWinds: Berkin and Orton developed LinkWinds as a visual data exploration tool for large multi-dimensional multi-disciplinary data sets [BO94]. It is developed at the Jet Propulsion Laboratory of NASA. Linking visualizations is also used with this system.

Since users of this system mostly view large scientific data, this tool falls between the fields of scientific visualization and abstract information visualization.

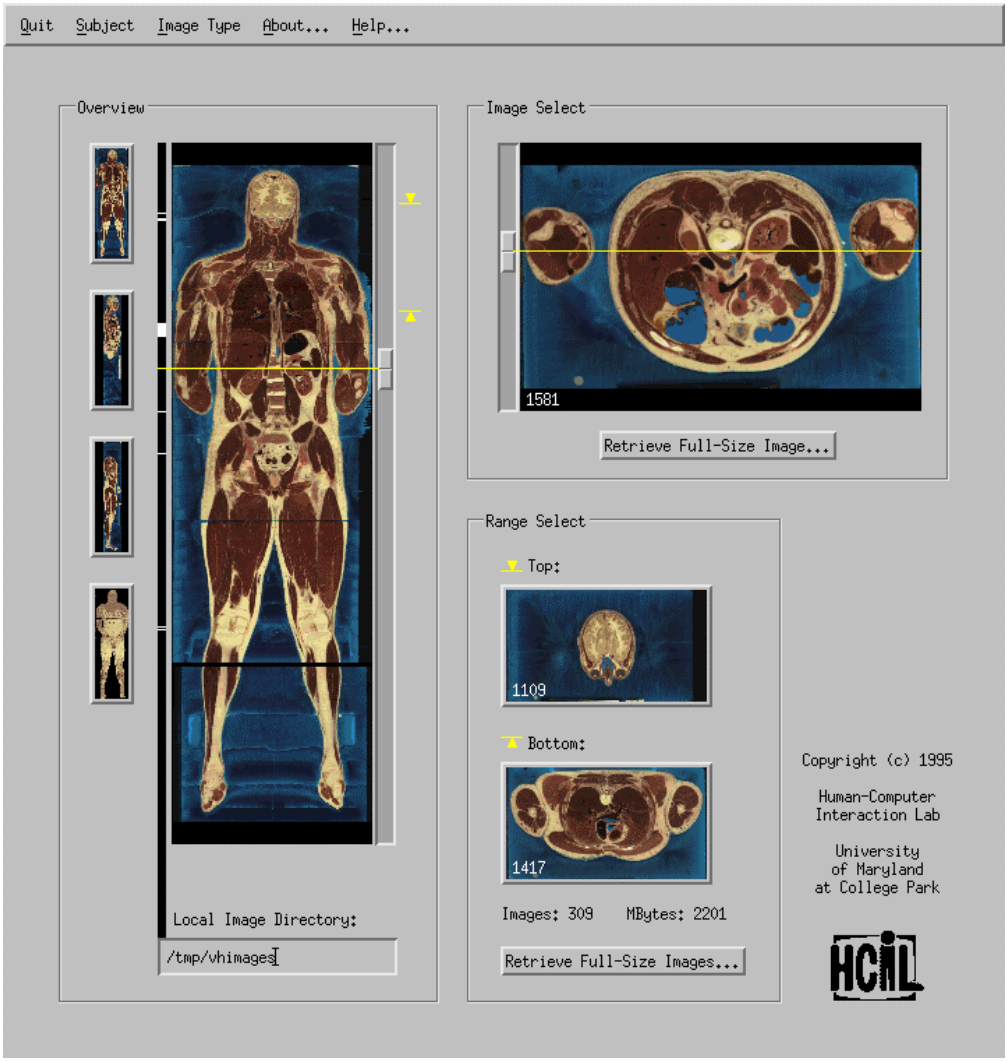


Figure 2.22: The Visible Human Explorer of the University of Maryland where multiple tightly coupled views of the data is available to the users

Visible Human: North, Shneiderman, and Plaisant developed an image library browser for a project known as the Visible Human Project [NSP96]. Researchers of this project, in multiple organizations, worked on methods to visualize and analyze the human

body represented by thousands of images. North, Shneiderman, and Plaisant worked on a browser where tightly coupled views of the images help users browse a large image library. Selections in the preview image of a human body are immediately reflected on the other views, Figure 2.22.

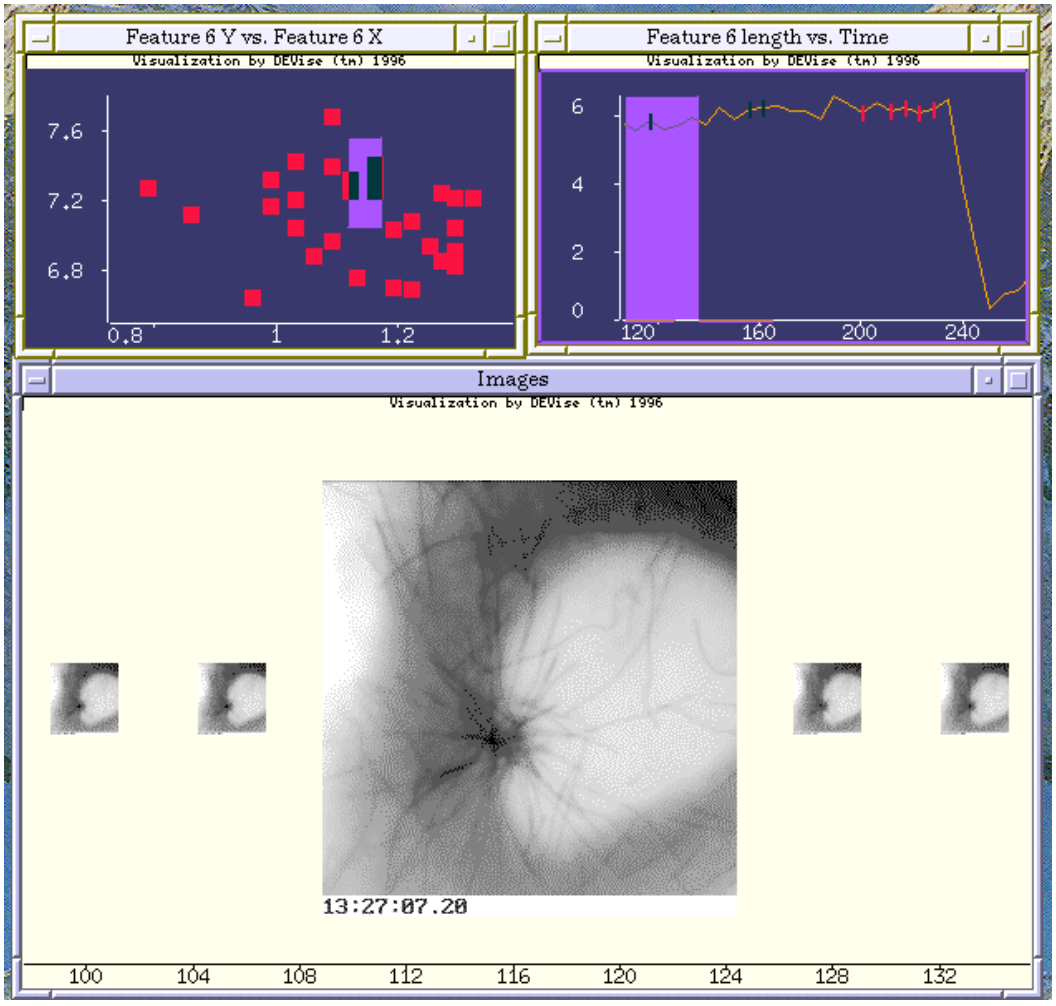


Figure 2.23: A sample DEVise display with multiple views, a scatter plot, a 2D chart showing a function, and a series of images related to that data item

DEVise: DEVise is an exploration system where users can browse and share visual presentations of large data [LRB97]. Multiple linked views are also available with this system, Figure 2.23.

2.6 Database Schema, Tables, Queries, and Results Visualization

Visual Querying Systems (also known as VQSs) is a known research topic under the field of databases. Many systems exist in this field. Most of these systems work for relational databases and provide visualizations for the database schema, tables, queries, and results. Catarci, Costabile, Levialdi, and Batini [CCL95], and Reisner [Rei91] did the first surveys of this topic. They mention the main approaches and give an assessment of various usages of these systems. Many of the systems discussed in this section can also be mentioned under some of the previous sections, but certain database related features of these systems make them more appropriate for this section. The following are a few example database schema, tables, queries, and results visualization systems:

Tioga-2: Formerly known as DataSplash, Tioga-2 is a database visualization environment from Berkeley [ACS96]. It provides users with a variety of display objects to be used on canvases to explore the underlying database. Navigation between canvases are provided by means of portals, Figure 2.24.

SeeData: SeeData is a system for displaying the relational schema of a database [AEP96]. This is a visualization system that uses 2D visualizations of a database schema where relationships between objects are shown via bar charts. SeeData can display large database schemata containing over thousands of relations, Figure 2.25.

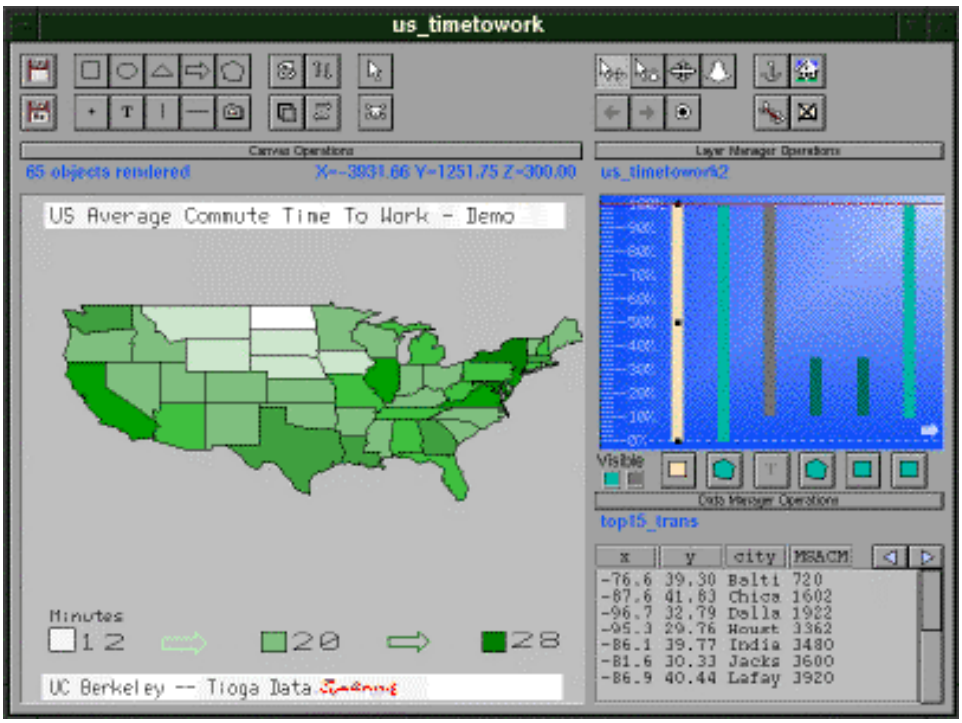


Figure 2.24: Tioga-2 display with data mapped onto the United States

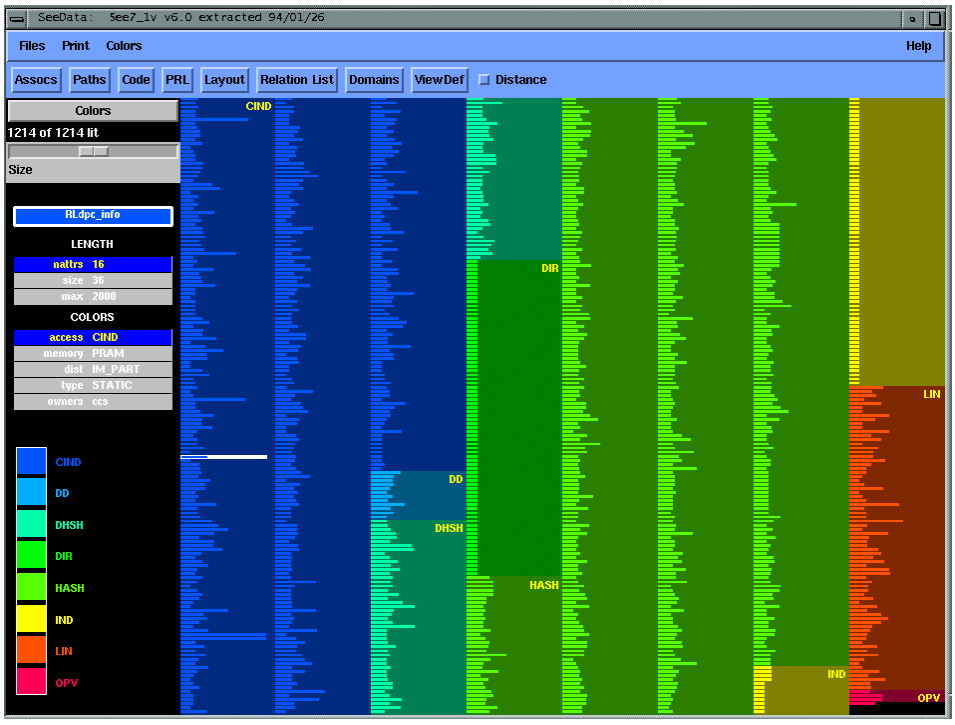


Figure 2.25: SeeData relational schema browser, each line is a relation and each pixel in a line is an attribute (dimension) of that relation

InfoCrystal: This is a tool used to visualize sub-results of a query and their relations to the query terms. It helps users understand and form complex queries on a database [Spo93].

Polaris: Polaris is a recent system from Stanford University [SH00]. It is a visualization system for relational databases that extends the concept of pivot tables. Visual querying is possible, and these visual specifications can be rapidly and incrementally developed.

Giving Ranked Outputs: Veerasamy and Navathe [VN95] introduced a digital library catalog browser in 1995. This interface gives ranked output information in the form of a histogram. This system is for complex query formulation on a document database, Figure 2.26.

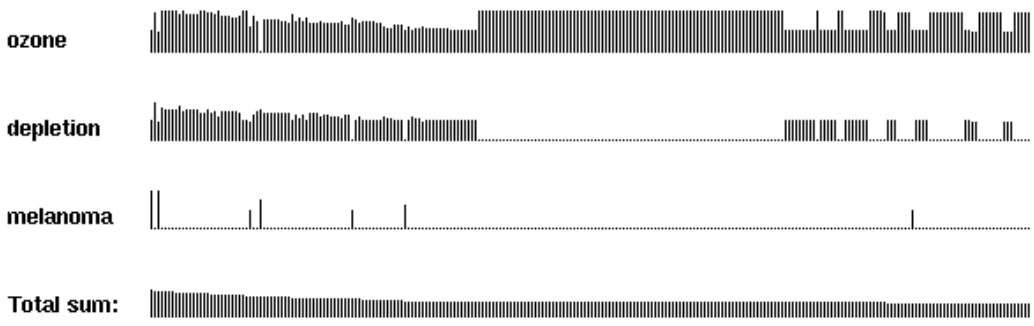


Figure 2.26: Veerasamy and Navathe used histograms to rank results of a query

RABBIT: RABBIT is the oldest of these systems [Wil84]. RABBIT presents a query to the users and allows them to modify it by showing RABBIT what has to be changed in it. It is one of the first systems where progressive querying is used on databases.

Menu-driven information retrieval: Heppe, Edmondson, and Spence [HES85] also introduced one of the first VQSs. They used volume previews in the database search process. They also demonstrated some progressive querying capabilities in their systems.

ADVIZOR: This is a system available from www.visualinsights.com. ADVIZOR presents the information that is stored as pivot tables of a database for online analytical processing (OLAP) by using interactive charts and graphs (Figure 2.27).

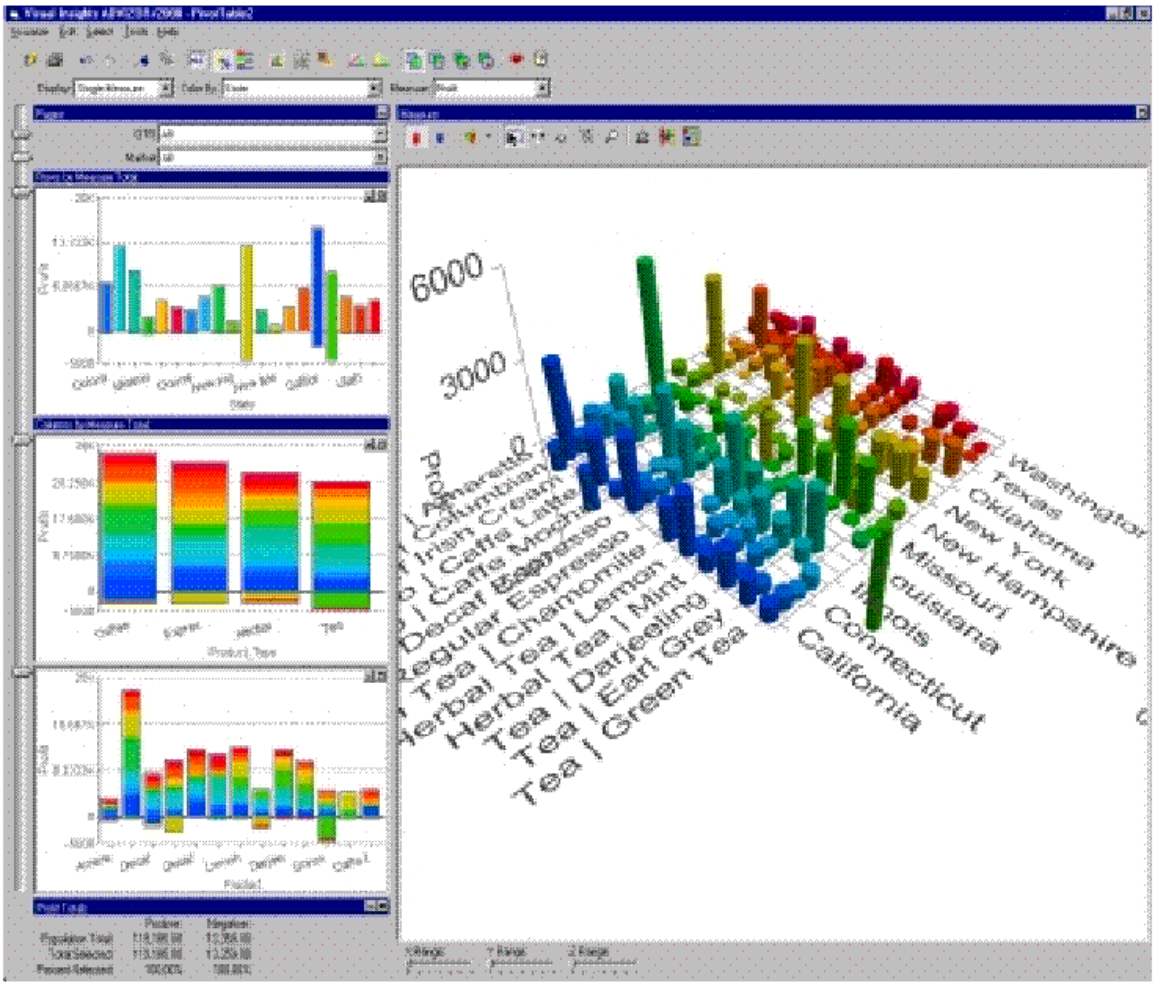


Figure 2.27: ADVIZOR shows a landscape visualization of a pivot table.

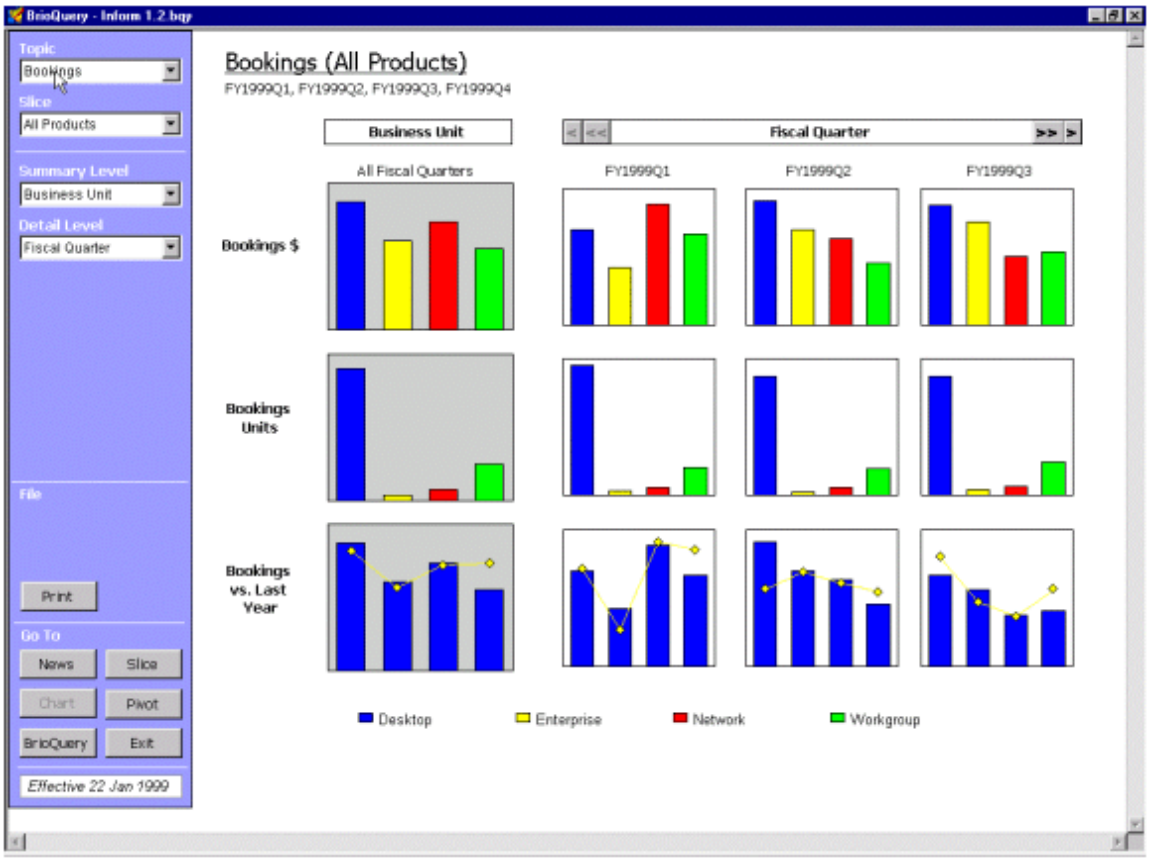


Figure 2.28: Brio shows query results graphically.

Brio: Brio is a database front-end that works efficiently over a network. It works with OLAP servers and helps users generate online business analysis reports efficiently (Figure 2.28). It is available from www.brio.com.

dynaSight: dynaSight is available from www.arcplan.com. It is also a database front-end and works with OLAP servers to help the users generate business reports (Figure 2.29). It can connect to multiple types of servers.

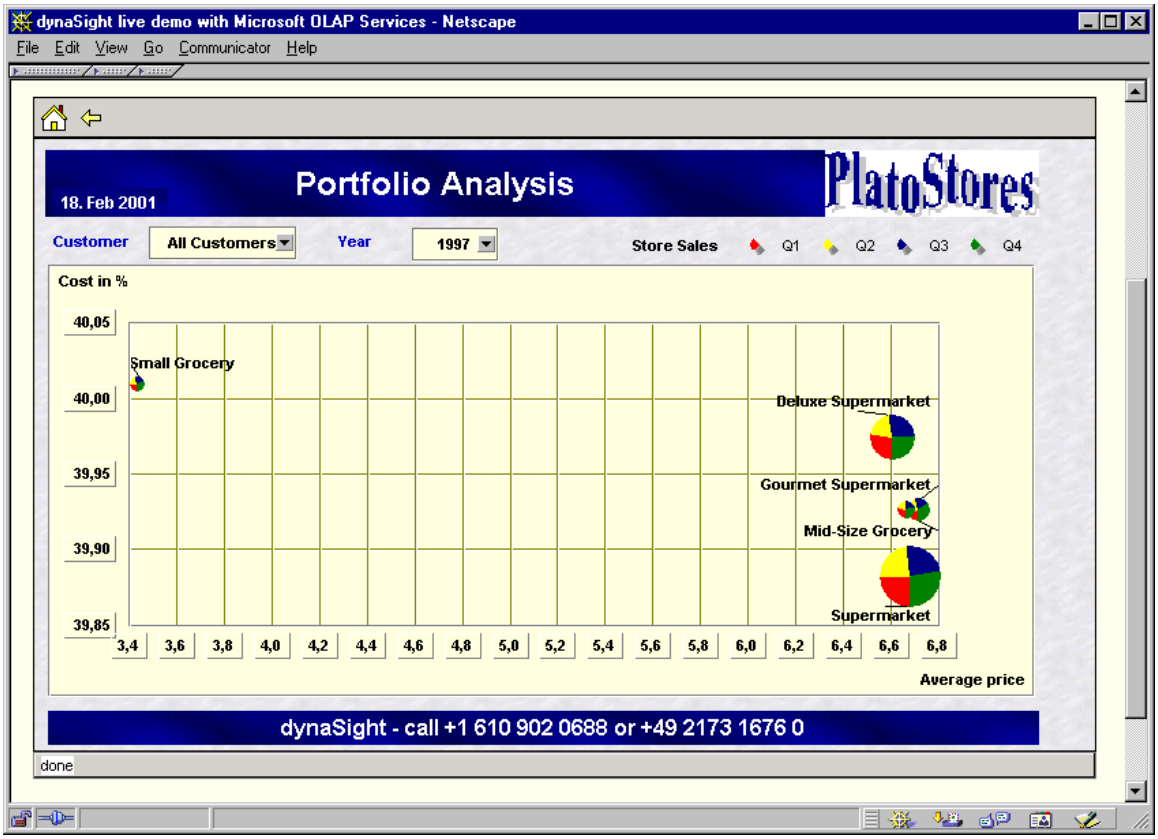


Figure 2.29: dynaSight showing some portfolio analysis graphically

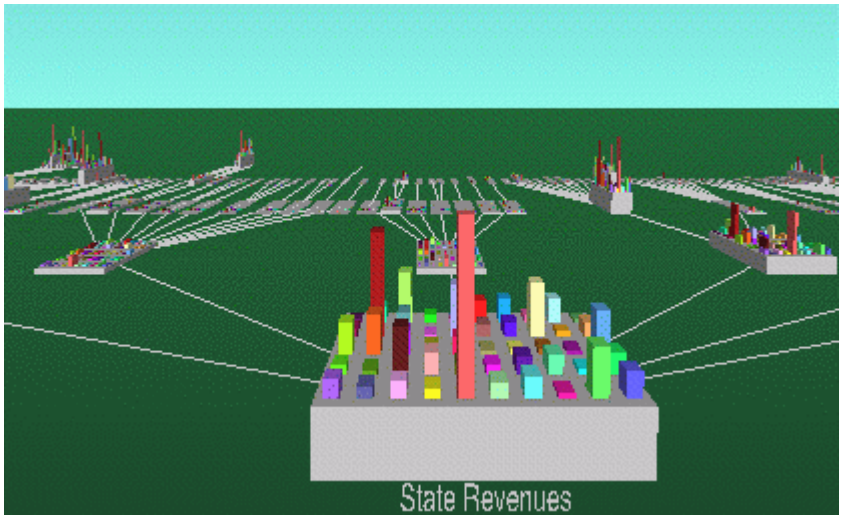


Figure 2.30: MineSet showing a hierarchy by using a 3D browser

MineSet: MineSet is available from www.mineset.sgi.com. It is another a database front-end that can also work with OLAP servers (Figure 2.30). It has strong data-mining capabilities through some classification and association rule generators. It can connect to multiple types of database servers.

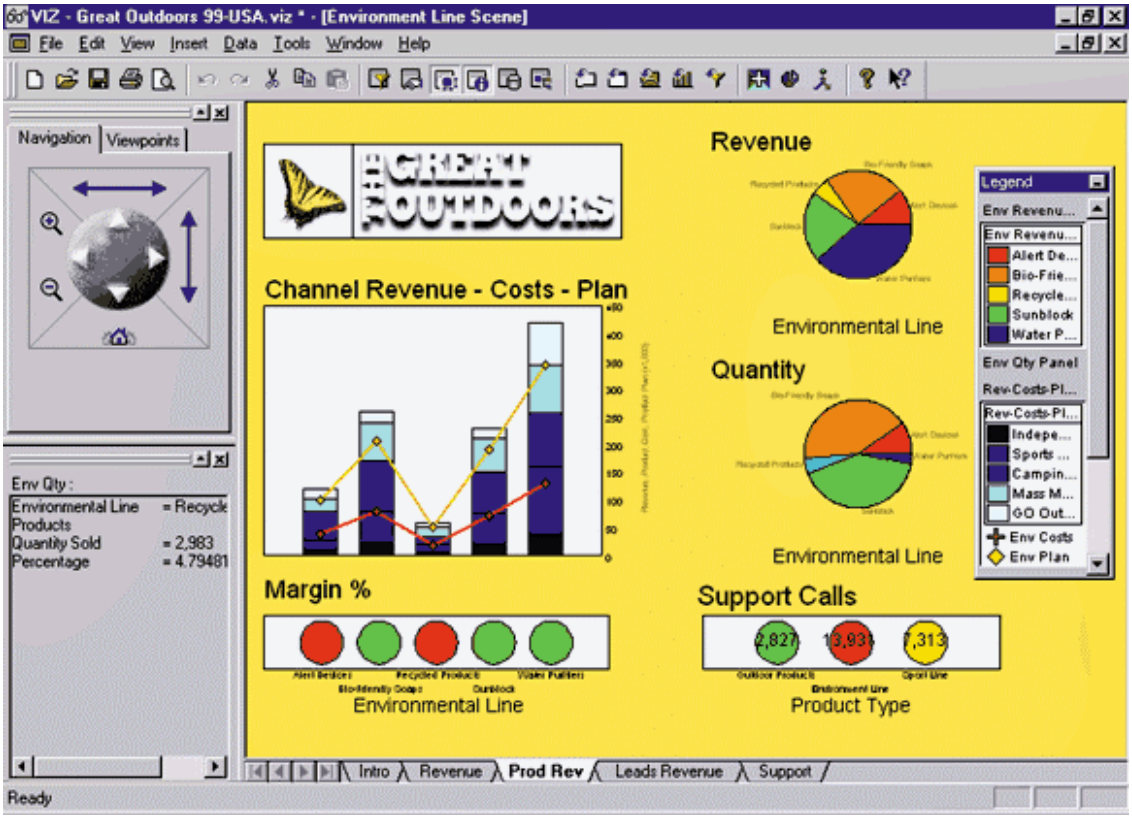


Figure 2.31: Cognos showing multiple interactive charts

Cognos: Cognos (available from www.cognos.com) implements interactive abstract data visualization capabilities that help users gain insight about their data (Figure 2.31). This system also works very efficiently on networked OLAP servers.

2.7 Directions

Multi-dimensional data visualization continues to draw increasing interest from visual data mining and information visualization researchers. Users of various data sets continue to demand more elaborate methods to visually mine and manipulate their multi-dimensional data. Search for these methods is becoming more challenging with the increasing data set sizes, types, and complexity of access methods.

More varied types of data (e.g., images, sound, etc.) with larger data storage facilities has become a reality. Methods to visualize large amounts of more varied data are required.

Data sets are not confined to a single system or network anymore. It can easily be stored and accessed over the Internet. Visualization of such data sets continues to be an interesting research topic.

Data sets are not formed of simple text files or a simple list of items. Databases have long become a common storage and maintenance facility for data. Databases can be formed of many complex relations, objects, and attributes.

It is difficult to consider a direction of research for visual data mining researchers without addressing most of these features of data sets (size, type, access methods). Researchers should not only support large data sets but also support complex access mechanisms through networks to complex data storage facilities (such as relational database management systems). All of this should also be done in a transparent fashion to the users.

Generalized query previews tries to address many of these new features of the data. It claims success in browsing large online data stored in relational databases.

CHAPTER 3:
NASA EXPERIENCE AND EARLY WORK

3.1 Dynamic Queries with Large Online Data

Dynamic queries [WS92] [AS94] [AW95] [AWS92] uses a direct manipulation approach to facilitate query formulation on multi-dimensional data with a visual representation of query components and results. Dynamic queries allows rapid, incremental, and reversible control of the query. Results are presented visually. Continuous feedback guides users in their query formulation process. Figure 2.11 shows an example dynamic query interface.

The application of dynamic queries to large online data is an exciting idea. Unfortunately, high system-resource demands make dynamic queries inapplicable to large online data collections. Dynamic queries requires immediate access to data so that continuous immediate feedback is always given to the user. Yet, large online data cannot be immediately and continuously accessed.

In 1995, Human-Computer Interaction Laboratory of University of Maryland with support from NASA's Goddard Space and Flight Center started to work on NASA's large online data collections. These collections are stored in vast distributed data archive centers and contain various types of data (e.g., documents, images, numerical values, etc.). The attempt to apply dynamic queries to these collections required researchers to look for solutions to make dynamic queries work with large online data.

Researchers of the laboratory decided to use overviews and previews to make dynamic queries applicable to large online data. They were looking for methods to

efficiently prune irrelevant data and help users focus and form their queries. At the same time, algorithms and data structures that would enable dynamic queries to work with large data collections were developed [TBS97].

The project led to the idea of query previews. Query previews forms the basis for this dissertation. The paradigm of query previews is to give an overview of the data and a preview of the queries to the users before the final queries are sent through the network or details are visualized with an interface. Query previews works on a very few selected attributes of the data. It divides the querying process into two steps to reduce the resources needed to form the final query. Hence, a smaller and more interesting portion of a larger data set can be downloaded to the local memory of a computer from the network.

We applied the principles of this two-phase querying strategy (i.e. previews first then refinements) for NASA's Earth Observing System Data Information System [DPS96] [DPS97]. This strategy is now available as an experimental interface [GTP99] for the Global Change Master Directory (gcmd.nasa.gov) and is the basis for the Global Land Cover Facility interfaces (glcf.umiacs.umd.edu), all part of NASA. This dissertation carries query previews to another more general stage.

3.2 Two-Phase Querying

For the two-phase approach, the designer first has to choose a few discriminating attributes of the data, usually the most commonly used, for the initial phase. This is the query preview phase. The other attributes are kept for the second phase that will include all of the attributes of the data for further querying. When the querying environment is activated the query preview appears first. Users make some decisions on this first

interface and then move to the second one, the refinement phase, to complete the query. The first phase is used to prune irrelevant data while preventing the user from submitting zero-hit and mega-hit queries. The second phase, the refinement, can be implemented as a dynamic query interface to help users finish their querying sessions by guiding them in further refining their query definitions.

3.2.1 Query Previews

Query previews shows the discriminating attributes of the data so that any selection would lead to a smaller subset of the data. Commonly used attributes of the data are selected to form a query preview. This is essential to address the needs of most of the users. In order to guide users in the query formulation process, query previews provides aggregate information about the data. Distribution of data over different attribute values is shown graphically as bar charts. When users select a value on any of the attributes of the interface, the rest of the interface is updated. Therefore, for each action users take, feedback is given immediately. As users see the potential size of their query result before refining the queries, they are less likely to submit queries that return zero or mega hits. The system load will be reduced if users do not waste their time with zero-hit queries or consume network resources in downloading useless results.

While dynamic queries requires attribute values of every record of the data to be downloaded, query previews only needs aggregate information about the data. So whatever the data size, only the distribution information of the data is needed to form a query preview interface. However, this may have a disadvantage. Only the buckets of data distributions will be available to the users, but not the details of a single item. The details of the data can become visible only in the second phase.

Figures 3.1 and 3.2 show a query preview interface formed using the three most commonly used attributes of the Global Change Master Directory of NASA (topic, year, and area). The distribution of data over these attributes is shown with bar charts and the result set size is displayed in the result bar at the bottom.

3.2.2 Query Refinement

If needed, the query preview phase can be followed by a refinement phase, which can be implemented as a dynamic query interface, to further change the query. At the refinement phase, when a desired final result set size is obtained, the results can be retrieved from a remote data collection. These can be images, values, etc. Other types of interfaces for the refinement phase are also possible (e.g., form fillin, menus, etc).

Figure 3.3 shows a possible implementation for the refinement phase with dynamic queries (implemented for the Earth Observing System Data Information System of NASA). In this example, the time-span of the data items versus the size of the items (e.g., image size) are shown on a scatter plot that is tightly coupled with a number of scrollable menus that enable selections on the remaining attributes of the data. Color-coding is used to present an attribute from the data (i.e., processing level of images). An interactive geographical map is also available. After the completion of the query, users can submit it to access the final set of results. Figure 3.4 shows this as a list of items presented as a web page. The option of skipping the refinement phase and directly jumping into the query results is also a viable alternative for certain implementations.

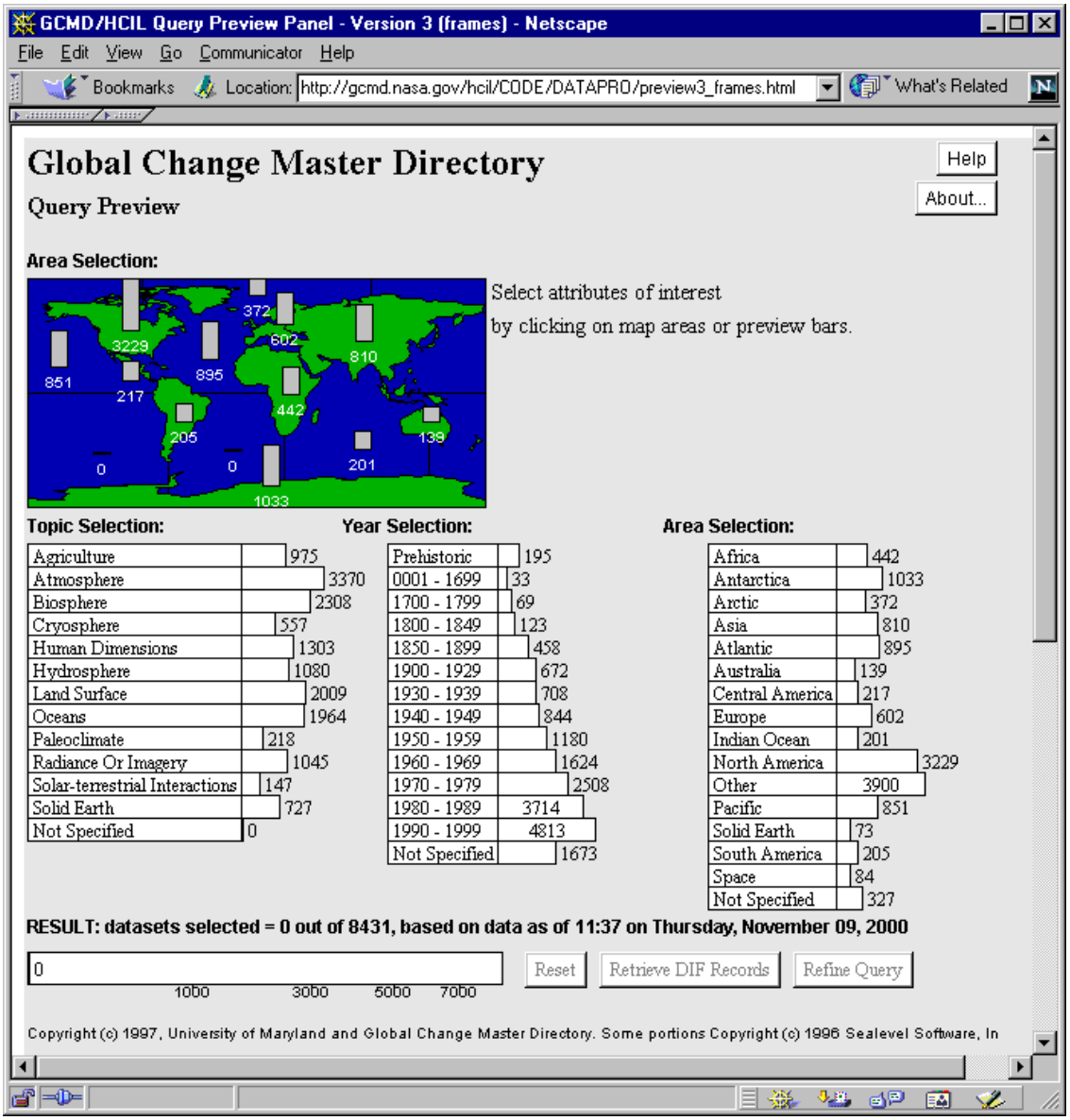


Figure 3.1: An example query preview interface developed at the Human-Computer Interaction Laboratory of University of Maryland, for NASA's Global Change Master Directory. Topic, Year, and Area are the discriminating attributes for the 8431 scientific data items of the NASA archives. In this screen shot, the bars show the overview of the data distribution. Recent versions of this interface are available at the Global Change Master Directory (gcmd.nasa.gov).

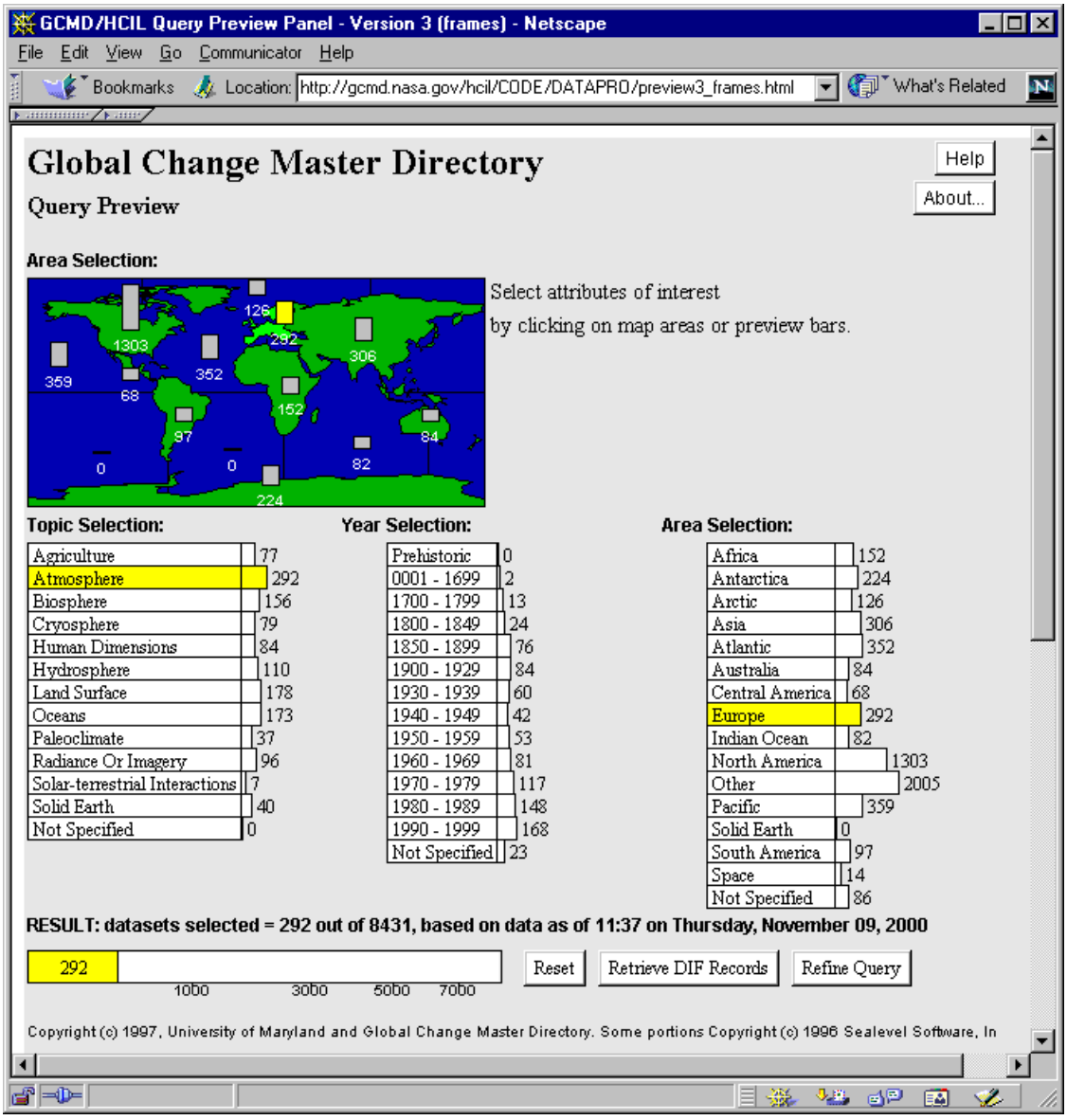


Figure 3.2: When users select the attribute values (e.g., here atmosphere for topics and Europe for area), the bars are updated immediately to reflect the new distribution of the data that satisfies the query. When users are satisfied with their initial query, the results can be retrieved, or the query can be refined with additional attributes in the second phase with another interface. In this case, atmospheric data for Europe produces a set of 292 data items to be retrieved.

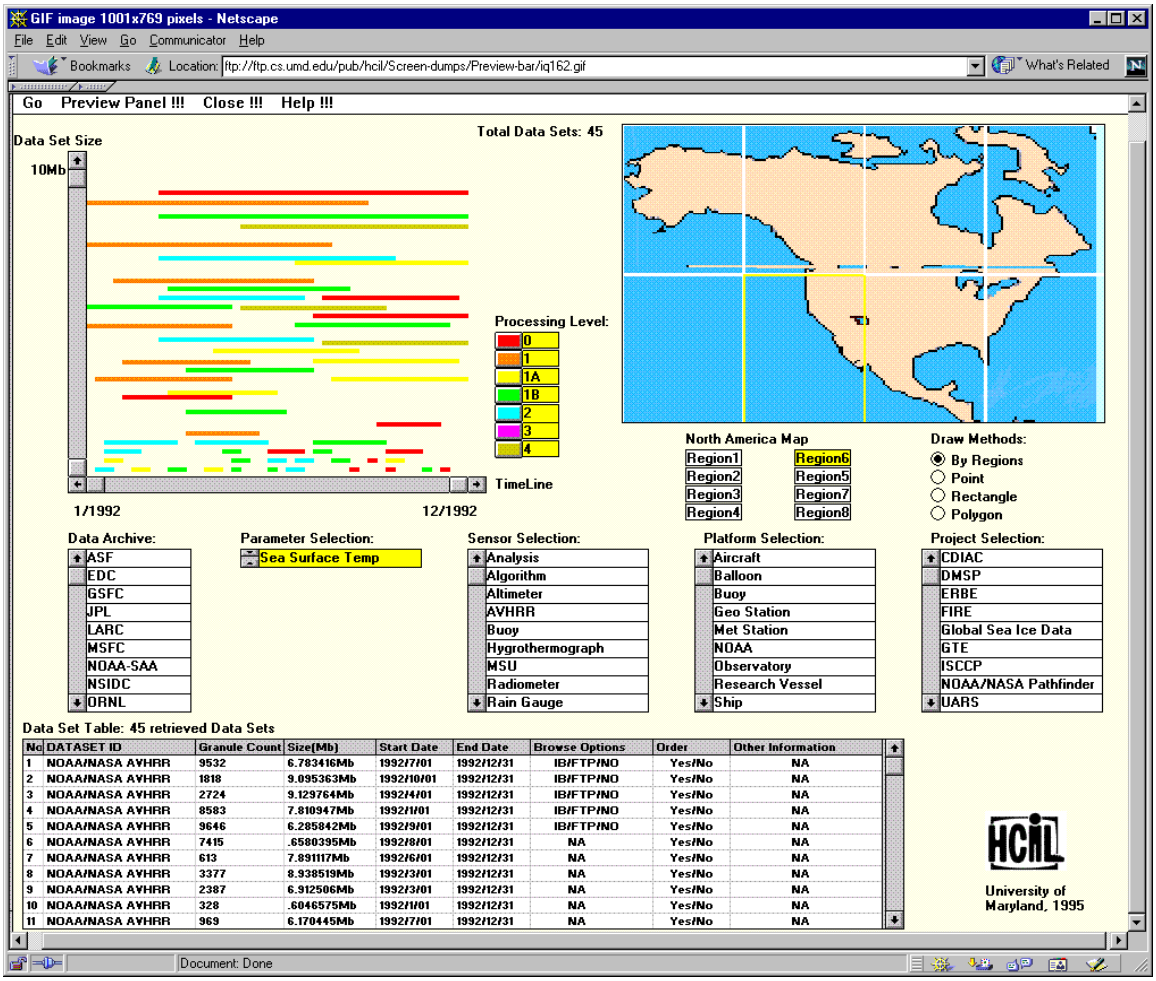


Figure 3.3: The refinement phase, implemented as a dynamic query interface. In this example, the time-span of the data items versus the data size (i.e., image size) are shown on a scatter plot that is tightly coupled with a number of scrollable menus that enable selections on the remaining attributes of the data. Color-coding is used to present an attribute from the data (i.e., processing level of images). An interactive geographical map is also available.

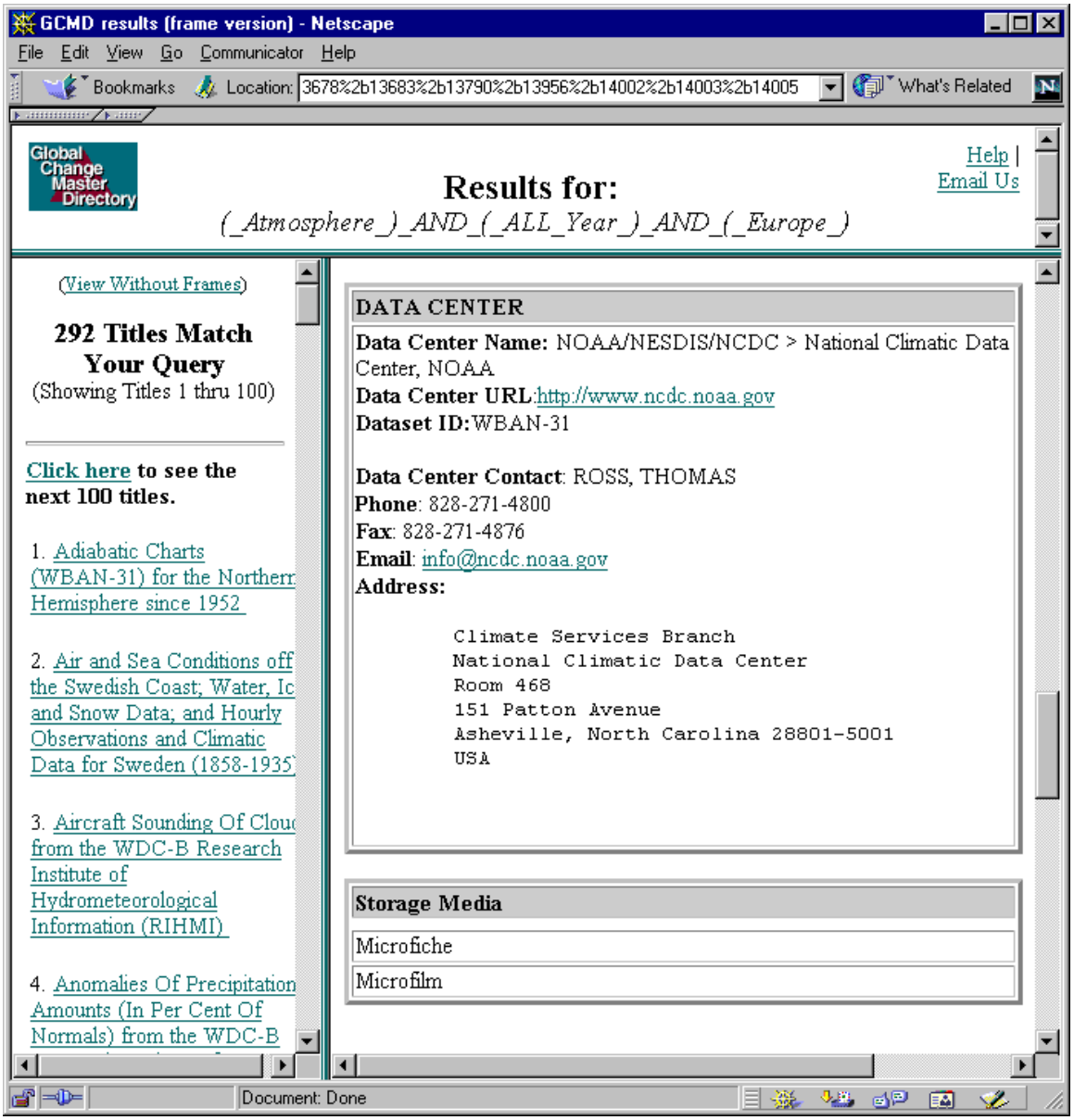


Figure 3.4: Results presented as a hit list on the left, details are presented for a single item in a frame on the right, and the query is displayed on the top as a conjunct of disjuncts

3.2.3 Extensions and Recent Prototypes

The query previews work continued with implementations of new prototypes using different types of previews and overviews. Figure 3.5 shows a query preview interface where the overview of data is presented as a binary preview. In this example, the availability of data, instead of the amount of data, is used to show the data distribution.

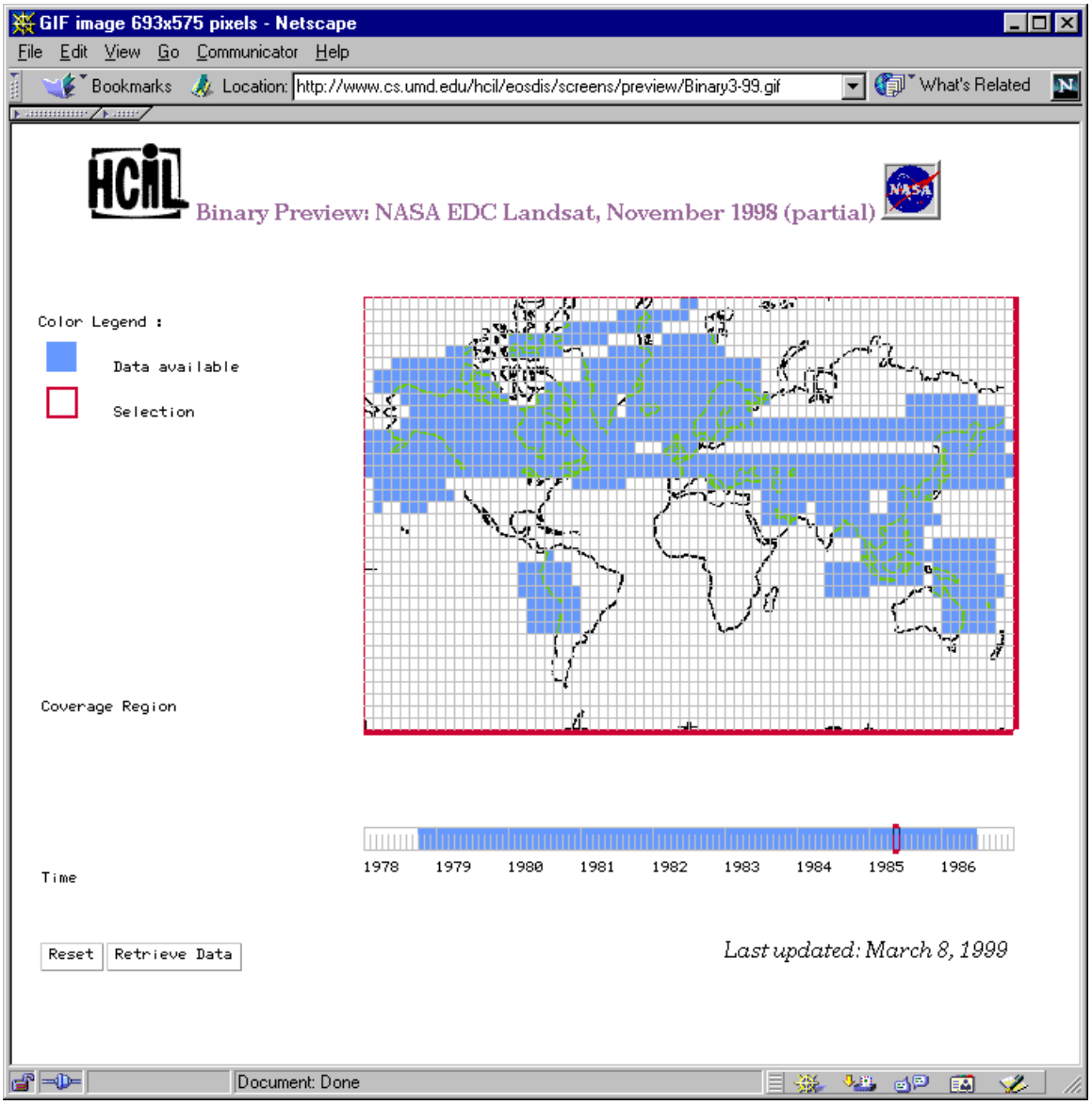


Figure 3.5: Binary previews, dark regions show locations where the data is available

Figure 3.6 shows another query preview prototype. In this example, the data items contain a special attribute, the time attribute (i.e., start and end dates of a certain event). This attribute contains ranges of values rather than points of values that require special attention. This new type of preview, range preview, must use special algorithms and data structures to calculate and display the distribution information as the data items may expand multiple dates. Overlaps can occur in counting these items in the bars.

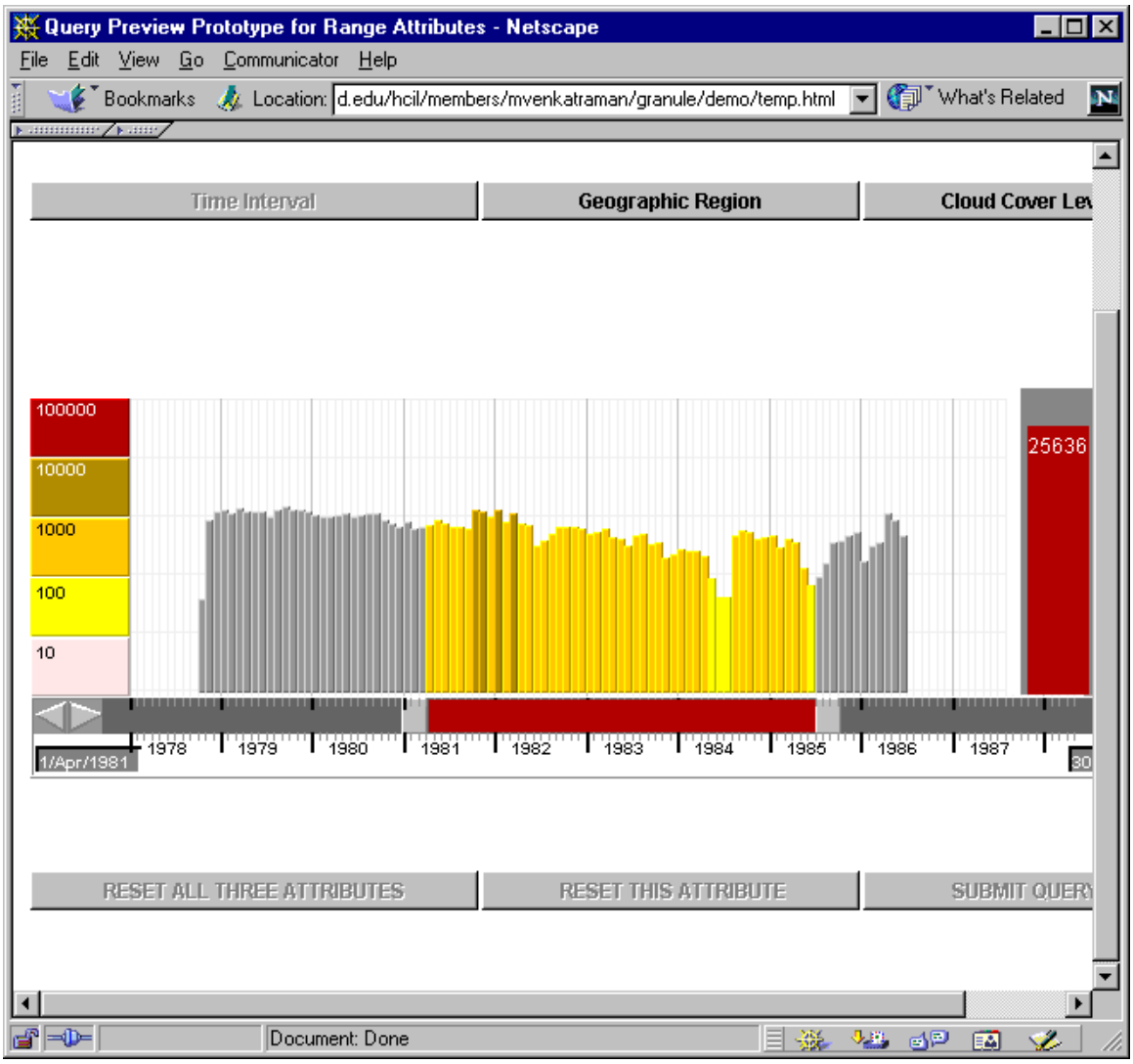


Figure 3.6: Range previews, the distribution of data is shown over years

Finally, the query previews work has been expanded to cover the task of analysis of statistical data. Figure 3.7 shows an example for this special type of query previews, table previews, where analysis of the distribution of data may be as important as or sometimes more important than retrieving the matching data items for the query.

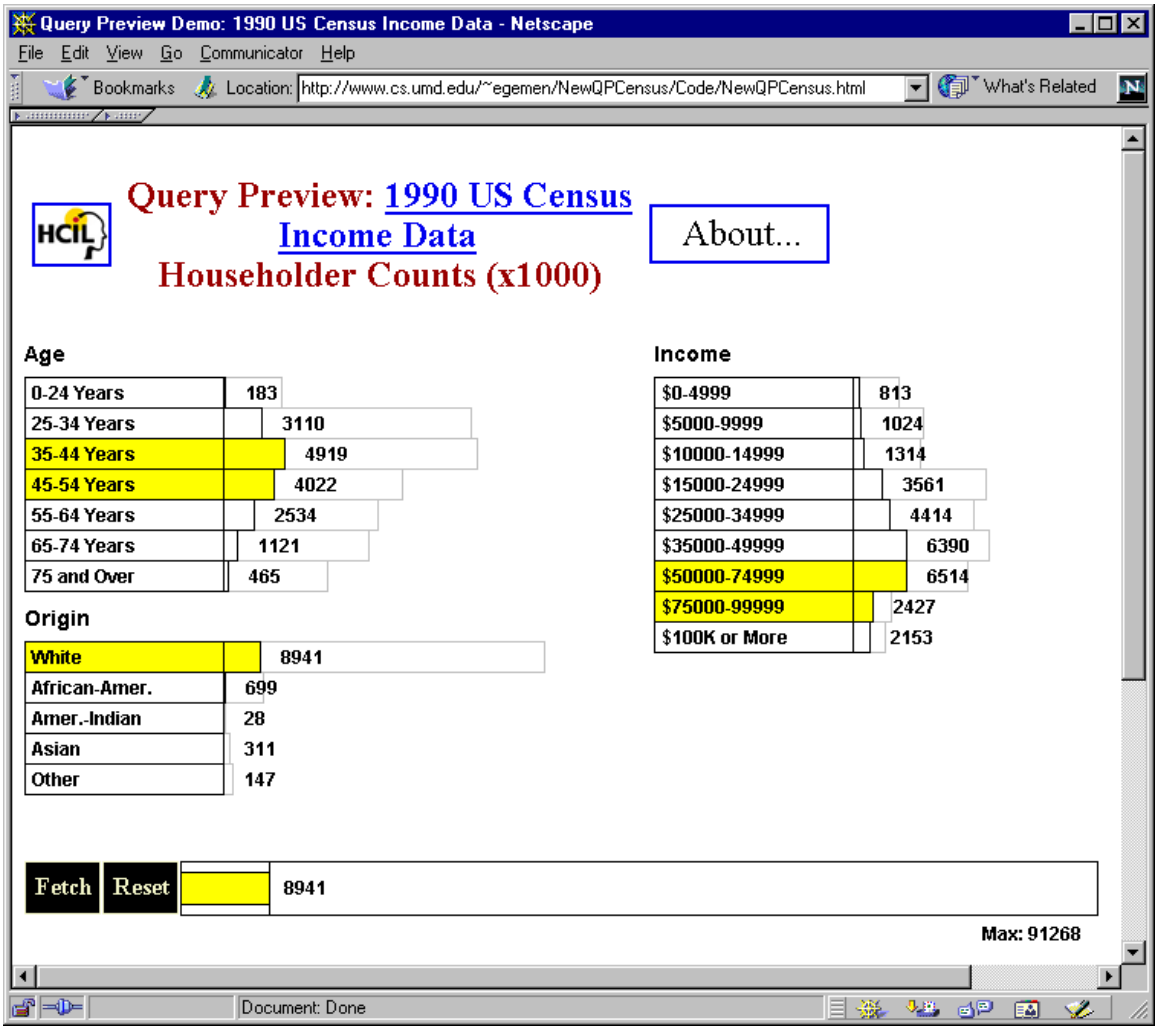


Figure 3.7: Continuing work with table previews, the 1990 United States Census Bureau Income Survey Data is used for this example, 8,941 hits selected

3.3 Summary

Query previews shows the data distribution information on some of the commonly used discriminating attributes of the data so that any selection would lead to a smaller subset of interest. It helps users form informed queries preventing zero or mega hits. Distribution of data over different aggregated attribute values is shown graphically by using tightly coupled bar charts.

Especially, the recent work with the Global Change Master Directory of NASA suggests that query previews forms a promising option for browsing operational large online data collections.

CHAPTER 4:
INITIAL USER STUDY

4.1 Motivation for a User Study

Since query previews adds another phase to query formulation, there is a possibility that user performance would deteriorate and that users would be annoyed by a two-phase approach. Moreover, query previews focuses attention on only a few selected attributes that may not be useful in some queries. Therefore, there was a need for a user study to verify and quantify the benefits of query previews and measure the subjective user preferences.

4.2 User Study Methods

4.2.1 Introduction to the Study

In this user study, we identified the task types that would put query previews into their best and worst situation so that we could quantify the maximum benefits and drawbacks of this technique [TLH00].

Clearly specified tasks have a straightforward and an accurate definition (known-item search), e.g. "List all the Maryland employees from the employee database". Query previews has no benefits for this task. In this case, users want the complete list regardless of the outcome of the query. For this case and in general for clearly specified tasks, the relevance of the preview attributes to the query is not an influential factor since users are best served by going directly to the form fillin interface (tasks of this worst case scenario are called T1 tasks).

Unclearly specified tasks usually require a series of submissions. User's constraints and preferences cannot be stated immediately. Information gained from the query previews will influence their series of choices, so query previews should be very useful. However, the relevance of the attributes used in the query preview will impact the usefulness of the user interface. Suppose that a user is looking for some software engineers from the Washington, D.C. area using an employee database. If the query preview shows the number of employees per state and some other attribute values of the data such as the age distribution, then the preview is only partially relevant to the task (middle case scenario: T2). On the other hand, if the query preview shows the number of employees per state and their job types, then the query preview becomes fully relevant (best case scenario: T3).

The three task types in the study varied in terms of their clarity of the specifications they have and in terms of their degree of relevance to the attributes they used to the query preview attributes. Six subjects performed a set of tasks, once by using an interface that included a query preview followed by a form fillin interface and once by only using a form fillin interface. Then, another set of six subjects worked in the opposite order. The task completion times and the subjective preferences of the subjects were measured.

4.2.2 Hypothesis on Query Previews

Our hypotheses were: (1) For clearly specified tasks (T1), adding the query preview step will lead to slower task performance, (2) for unclearly specified tasks (T2 and T3), the addition of a query preview step will lead to faster performance, and (3) users will always prefer query preview interfaces.

4.2.3 Independent and Dependent Variables

The independent variable was the user interface type and the treatments were:

- Form fillin interface *with* a query preview.
- Form fillin interface *without* a query preview.

The dependent variables were the *time to complete* the tasks in each interface (not including setup times) and the *subjective preferences* of the users.

4.2.4 Tasks

We examined the two interfaces using the three types of tasks that are formally defined as:

- T1: Clearly specified tasks in which the query preview attributes are not relevant to the task.
- T2: Unclearly specified tasks in which some of the query preview attributes are relevant to the task.
- T3: Unclearly specified tasks in which all of the query preview attributes are relevant to the task.

4.2.5 Subjects

Twelve computer science graduate students were used as subjects. All of them use computers almost every day and have at least five years of experience in using computers.

4.2.6 Materials

The materials include a form fillin interface for querying a film data set (including 500 films), a query preview panel for the same data, a set of tasks to be performed by the subjects, a subject background survey, and a subjective preference questionnaire.

4.2.6.1 Form Fillin Interface

A form fillin interface (Figure 4.1) was used to perform queries on a film data set. There are ten attributes for a film in our sample data set: category (horror, action, comedy, etc.), award winner (yes or no), rating (R, PG-13, PG, and G), year of production, length, popularity, lead actress, director, lead actor, and title. The output of a query is a list of films matching the specifications of the query. Vertical and horizontal scroll-bars can be used for scanning the list.

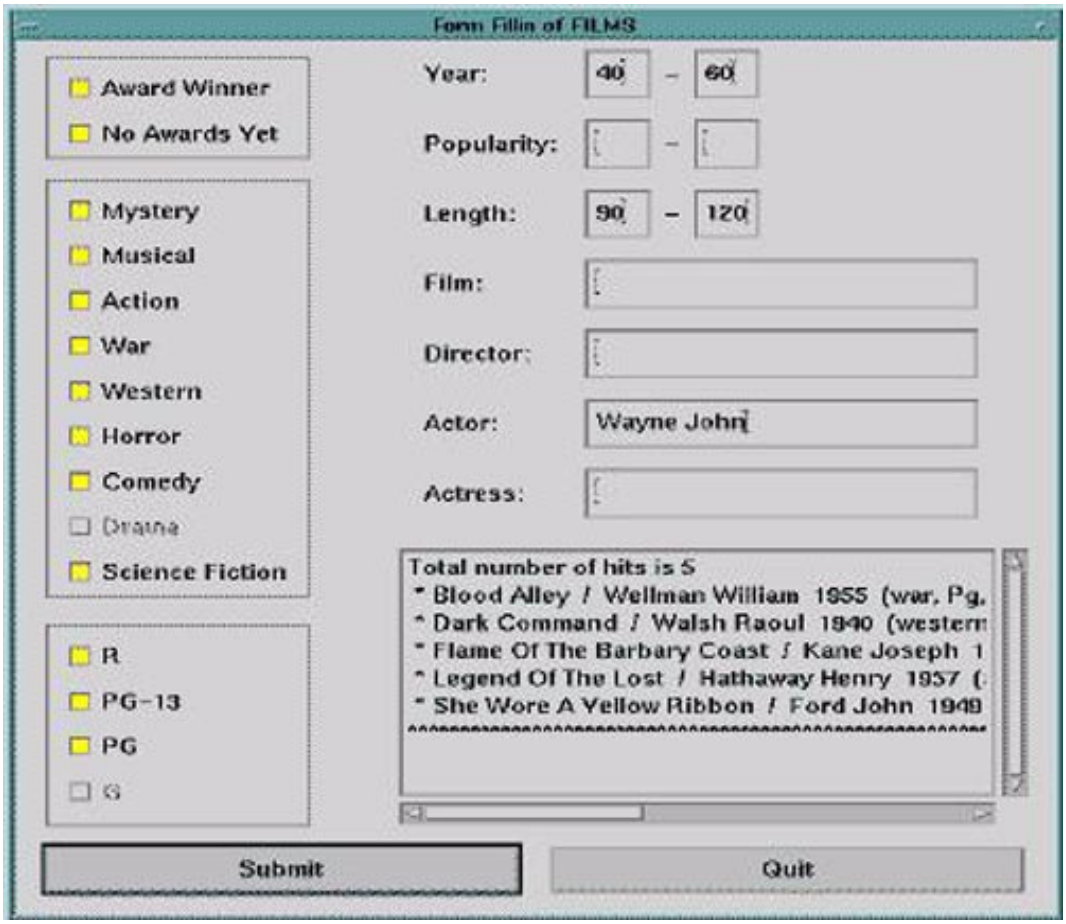


Figure 4.1: The form fillin interface used in the study. The rectangle on the right bottom corner is used for displaying the result list to a query. The list of fields allows users to enter values for the attributes of the data set. The three attributes on the left side are the ones that are also available in the query preview.

4.2.6.2 Query Preview

In the query preview interface (Figure 4.2) users can select values for three attributes of the data set: the category (horror, action, comedy, etc.), whether the film won an award or not, and the rating (R, PG-13, PG, and G). Multiple selections are available for each of these attributes. The number of films for each attribute value is shown on a separate bar. Each bar consists of a frame and an internal rectangle (gauge). The length of the frame is proportional to the number of films in the data set that match this specific value of the corresponding attribute. The length of the gauge is proportional to the portion of the films that match the query specified (the number of matches appears to the left of the bar). Users formulate queries by selecting the attribute values. As each value is selected, the bars of the other attributes adjust to reflect the number of films available for that selected specific values. For example, users may be interested only in films that won awards. By selecting "Award Winners", the gauges of the bars of the selected categories and ratings change immediately to reflect only the films with awards. The query preview bar at the bottom of the screen changes its length to illustrate the total number of films that match the current conditions.

When the "Refine" button is pressed, the query preview submits the specified partial query to the search engine and all the data about films that satisfy the query are downloaded for the refinement phase. The query preview is closed and the form fillin interface is loaded to refine the query (displaying initially all the films selected in the query preview in the result box).

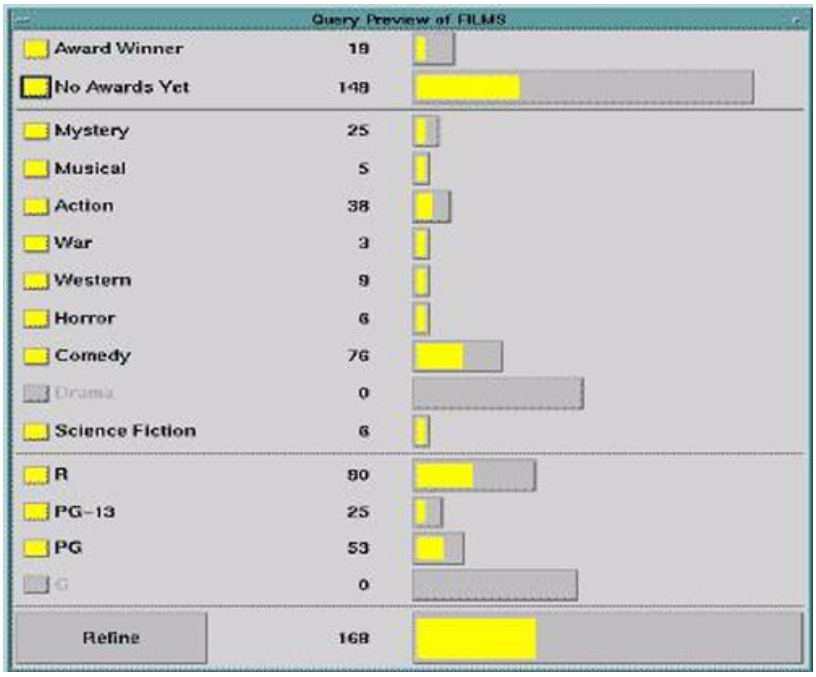


Figure 4.2: The query preview used in the study. The toggles on the left are used to choose attribute values and form the query. The counts and bars show the distribution of the result set for a query corresponding to the current settings of the toggles. The larger bar at the bottom shows the total number of hits, here 168.

4.2.6.3 Task Examples

The tasks given to the subjects were to find a film or a list of films in the database satisfying the constraints that is provided. Three types of tasks were used for this purpose:

- *T1*: a clearly specified task in which none of the query preview attributes is relevant for the task, e.g. "Find the latest film by Alfred Hitchcock" (a known-item search). For that type of task, users can typically find the answer by submitting a single form fillin query. The query preview has no specific advantage since its attributes are not relevant to the query.

- *T2*: desired films are vaguely specified. In this type of task, some of the query preview attributes are relevant, e.g. "Find a PG-13 musical which was produced between years 1990 and 1995, if no such film is available, find a war film from the same years with the same rating, if not, try a musical or a war film from 1970-90, and as the last possibility, try a comedy from 1970-95". This type of task is typical when users have a complex set of acceptable results, with some preferences. To perform such a search in the form fillin interface users must issue several queries, i.e., when the preferred choice is not available in the data. In the preview, users can get some insight about what is available in the data and what is not and hence can make more informed queries. However, since not all the attributes in the specification appear in the query preview, the form fillin is required to refine the query.
- *T3*: formed in a similar way to T2. A series of preferences for films are specified. In this case however, the query preview attributes are fully relevant to the task specifications. Example: "Find at least 30 films of the same category which are R rated and have no awards" (for example, in order to organize a film festival or make a collection). In the form fillin interface this task requires several queries to examine the number and rating of films in each category. The query preview on the other hand, gives an immediate picture of the relevant categories. The form fillin interface is required only to get an explicit list of the films.

For each of the above task types, six example tasks were prepared (eighteen tasks in total).

4.2.6.4 Subject Background Survey and Preference Questionnaire

The survey included eight questions that determined the experience level of the subjects with computers and with search engines. We also prepared a preference questionnaire. The subjective preference questionnaire included six questions that aimed to find out which of the two interfaces (a form fillin with or without a query preview) the subjects preferred and what their attitudes were toward adding query previews to the interface.

4.3 User Study Design

The study used a within subject counter-balanced design with twelve subjects. Each subject was tested on both of the interfaces, but the order of the interfaces was reversed for half of the users. A parallel set of tasks (similar but not the same set of tasks) was used on the second interface to reduce the chance of performance improvement. Each set of tasks included the three types of tasks (T1, T2, T3), with three tasks for each of these types. The order of the task types within a task set was also reversed (each of the six permutations was used twice). The order of the tasks within each task type was fixed.

4.4 Procedure and Administration

The subjects signed a consent form, filled out a background survey, received a brief demo of the form fillin interface and the query preview, and a ten minute training session during which they used the two interfaces (again similar to but not the same tasks with the actual tasks were used). During the study each subject performed eighteen tasks (nine in each of the interfaces). At the end of the study the subjects completed the preference questionnaire. The study took 50-60 minutes including the training and the questionnaires.

Two administrators were present. One of them administered the study, performed the demo, presented the tasks, and measured the task execution times. The other administrator recorded notes about the way subjects coped with the tasks and about problems that occurred during the study, and verified the procedures that were followed. The time that the subjects spent in using each of the interfaces was recorded (successful completion time of a task). These times did not include program startup time.

4.5 Results

4.5.1 Time for Completing Tasks

Figure 4.3 summarizes the times for completing each of the task types for our subjects (clearly specified: T1, unclearly specified and partially relevant: T2, unclearly specified and fully relevant: T3) for each of the user interfaces (with and without a preview). For T1 tasks, the user interface with the query preview yielded slower performance than the user interface without the query preview ($t(35) = 2.44$, $p < 0.05$). For T2 and T3 tasks, the interface with the query preview yielded faster performance than the interface without the query preview ($t(35) = 8.77$, $p < 0.05$, and $t(35) = 14.70$, $p < 0.05$, respectively). The statistical analysis used two-tailed paired two-sample t-test for means. Each task was considered separately leading to the degrees of freedom of 35.

4.5.2 Subjective Satisfaction

The subjects answered six questions about their preferences on a one to nine scale (with higher numbers indicating stronger preferences). The first question examined the general preference of subjects for using a form fillin interface with or without the query preview (Figure 4.4). The results show a statistically significant preference ($t(11) = 2.82$, $p < 0.05$) for the interface with the query preview over the interface without the query

preview. The rest of the questions asked what the subjects thought about the user interfaces. The results (average scores, standard deviations, minimums, and maximums) appear in detail in Figure 4.5.

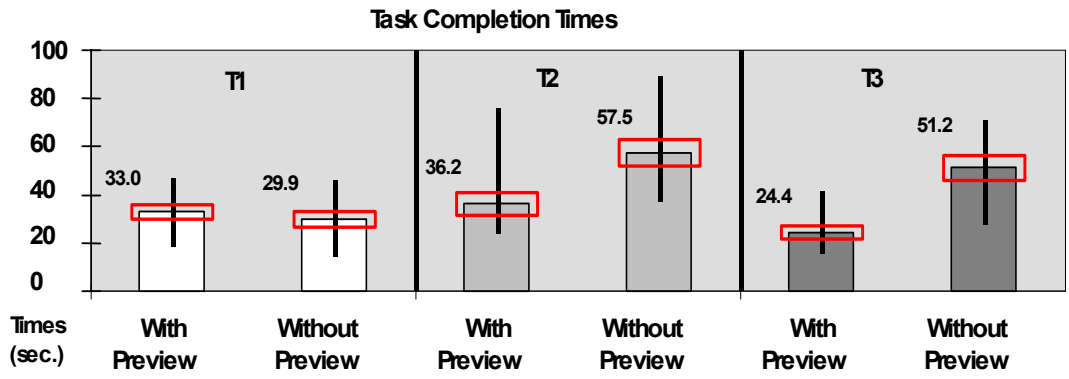


Figure 4.3: Average task completion times for T1, T2, and T3 (the rectangles show the standard deviations and the vertical lines indicate the ranges)

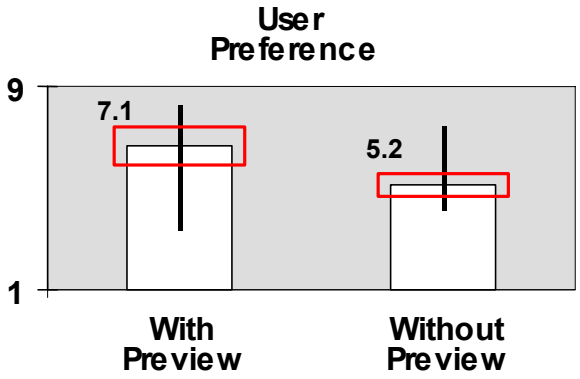


Figure 4.4: User preference for twelve users

The scores for all of the questions were statistically significantly above the mid-point scale value of five ($t(11) = 3.86, 6.20, 7.71, 2.24$, and 2.58 respectively, $p < 0.05$).

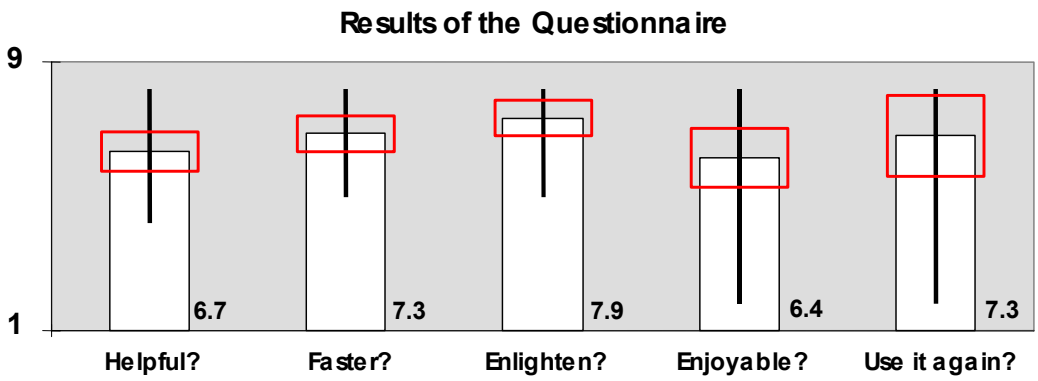


Figure 4.5: Subject questionnaire results (number of users is twelve) for the interface with the preview. Higher numbers indicate higher satisfaction with using the query preview.

4.6 Discussion on the Query Preview Study Results

Our findings support the hypothesis that for unclearly specified tasks, the interface with the query preview yields better performance times than the interface without the query preview. For both types of the unclearly specified tasks the improvement in performance was significant (at the level of 0.05): 1.6 times faster for T2 tasks and 2.1 times faster for T3 tasks. For the clearly specified tasks (T1), as expected, the form fillin only interface performed slightly better.

4.6.1 Clearly Specified Tasks (T1)

As expected, users of the form fillin interface for clearly specified tasks performed more rapidly since they were able to find the answer by submitting a single form fillin query. The query preview had no advantage since its attributes were not relevant to the query and users were performing known-item searches. However, users of the interface with the query preview performed only slightly worse (10% slower). The users spent

about two seconds in the query preview, identified that its attributes are not relevant for the task and continued to the refinement phase.

4.6.2 Unclearly Specified Tasks, with Partial Relevance of the Query Preview Attributes (T2)

Although not all the attributes in the task specification could be specified using the query preview, the insight gained from the query preview enabled users to eliminate some potential zero-hit queries in advance, concentrating in the refinement phase on a much smaller set of possible queries. The query preview enabled the users to reduce the search space significantly so that they could find the answer more quickly.

4.6.3 Unclearly Specified Tasks, with Full Relevance of the Query Preview Attributes (T3)

For unclearly specified tasks with full relevance of the query preview attributes, the full power of the query preview was utilized. The query preview enabled the users to see immediately which of the possible queries should be submitted. The users loaded the refinement phase only for submitting the query and viewing the results. The users performed the refinement phase with a high confidence that they would get the expected results. On the other hand, in the user interface without the query preview, the users had no clue about which of the possible queries will give the expected results. They had to try several possible queries, submitting five to six queries on average until they got a satisfactory answer. Although the response time for each such query was immediate (about one second), the time for filling in the right specifications of each query (five to ten seconds) caused significant differences in performance (even more than T2's).

4.6.4 Performance Improvement

Building models is a useful way to understand how the querying process works. Many different models exist in the database literature (i.e., for SQL and QBE) for this purpose. Reisner [Rei91] lists many of these in a survey paper from a human-factors point of view. The following simple model for performance times in the refinement stage can be used to explain some of our results:

$$performance\_time = no\_of\_queries \times query\_time$$

where:

$$query\_time = fillin\_time + response\_time + analysis\_time$$

The *fillin_time*, *response_time,* and *analysis_time* are the average times for filling in a query, getting a response, and analyzing the results, respectively. The response time is a function of several parameters such as the complexity of the query, the size of the data, the load on the server, the number of the retrieved entities, and the load on the network. The time for analyzing the results is determined by the number of retrieved elements. In our study the response time was short (about one second), the average analysis time was also small (e.g., analysis of a zero hit is almost zero seconds, and analysis of a mega-hit is only the time to decide to resubmit). Thus, the main factors were *no_of_queries* and *fillin_time*. For the T3 and T2 tasks, the query preview achieved the performance improvement by reducing the *no_of_queries*, yielding a situation in which:

$$preview\_time + (no\_of\_queries_{refinement} \times query\_time) <$$

$$no\_of\_queries_{form\_fillin} \times query\_time$$

In a more common situation where the access to the data would be through a network, the response time would be typically much larger than one second and the performance improvement that is achieved for T2 and T3 tasks would be even greater.

The results show that for different types of tasks the query preview achieves different rates of performance improvement in comparison with the traditional form fillin interface (from 0.1 times slower in T1 to 2.1 times faster in T3). The performance improvement which follows from the reduction in the number of required queries depends on several parameters. One parameter is the *clarity* of the task specifications. In clearly specified tasks the number of queries required in a form fillin interface is small, hence there is almost no potential for improvement. Another important parameter is the *relevance* of the query preview attributes to the task. Two additional parameters are the *significance* of the query preview attributes in pruning the search space and the *resolution* of the attribute values.

For example, if rating *R* is used and almost all the films in the data are of rating *R*, this attribute, although relevant, has insignificant contribution to the performance improvement for some queries. When numeric attributes such as year of production or length of the film are presented in a query preview, the possible values for these attributes are presented using some pre-defined resolution (for example, a ten-year resolution). Tasks that require higher resolution for an attribute than the one provided in the query preview will benefit less from the query preview.

In the study, the query preview yielded more performance improvement for T3 tasks (full relevance of the query preview attributes) than for T2 tasks (partial relevance of the query preview attributes). This result may support the assumption that better

relevance of the query preview attributes to the task yields more performance improvement.

During the study and in the pilot study we observed almost an order of magnitude of difference between the number of queries submitted among the two interfaces. Although the model uses number of submissions, we believe that the time to completion for each task suggests a parallel and more reasonable comparison among the two treatments.

4.6.5 Learning to Use a Query Preview

We found that it was easy for users, with experience in querying a database using a form fillin interface, to learn the query preview interface and take advantage of the information it supplies. However, some of the users, during training and, in a few cases, during the study, continued with the refinement phase immediately, skipping the examination of some of the relevant attributes. That happened when not all the task attributes could be found in the query preview. For example, when performing a task with conditions on *rating* (in the query preview), *year* (not in the query preview) and *category* (in the query preview), the fact that the *year* could not be specified in the query preview caused some of the subjects to continue to the refinement stage without examining the information for the *category* attribute. This problem seemed to diminish quickly with some experience.

4.6.6 Subjective Satisfaction

The users (statistically significantly) preferred the interface with the query preview, to the interface without it. They stated that the query preview was helpful, enabling them to search faster, and learn more about the data (scores for these questions were statistically

significantly above the mid-point value). We believe that this subjective satisfaction comes not only from the improvement in performance time which is experienced by the subjects but also from gaining better control in performing the tasks.

The suggested improvements related to user interfaces are: supplying a way to clear a group of related check boxes in one step, or easily resetting or setting all of them, a more immediate refresh operation on the bars for visual accuracy when changing the attribute values of the query preview panel, etc. The significant preference that subjects showed for including query previews in their current systems (in addition to the objective performance improvement for two of the task types) encourages further efforts in understanding, improving, and developing query preview interfaces.

4.7 Summary

This study supports the claim from the field experiences that query previews forms a realistic option for and can be extended to help users browse large online data collections.

Query previews is not suggested as a useful technique for all types of query interfaces and all types of tasks but this first study confirms that the benefits of query previews exist for several tasks.

CHAPTER 5:
GENERALIZED QUERY PREVIEWS

5.1 Motivation for Generalizing the Query Previews

Query previews is a simple method to eliminate most of the zero-hit and mega-hit queries and help users prune data efficiently. Unfortunately, query previews works only on a few selected attributes of the data as a separate phase in a querying session. This situation introduces a drawback on their applicability and performance. Many data sets are formed of numerous attributes. The designer of the preview panel can select the most frequently desired attributes of the data to form the preview panel. Yet, selecting only a few frequently used attributes may not be a possible option for many of the data sets. Even when it is possible, it may not be enough to satisfy some of the users with only a restricted number of attributes. Hence, a generalization to relax this restriction is needed.

5.2 The First Generalization Attempt on Query Previews

The first generalization attempt led to a new family of query previews [TPS00]. We called this new family of query previews the generic query previews. We combined the query previews approach with a method to present all of the attributes of the data to let users manipulate these attributes simultaneously. With this generalization, all of the appropriate attributes can be used to display the data distribution information. This new generalized approach could be used as a standalone query formulation mechanism, or like the query previews, it can be used as a preceding interface to another query formulation interface.

Figure 5.1 presents a sample implementation of this generalization. For this implementation a hierarchical browser is used to display all of the attributes and tables of the data. In this sample prototype, we used the Environmental Protection Agency's (EPA) Toxic Release Inventory as our sample data set. It contains approximately 400,000 reports of toxic material releases to the environment from various facilities in the United States. There are four tables in this data. They are Contact Info, Release Info, Chemical Info, and Facility Info tables. Each table contains a few attributes. For example, the Contact Info table contains Contact Phone and Contact Name as its attributes.

The root of our browser is tagged with the name of the data set. Each table is represented by a separate branch. Each branch may also have leaves representing different attributes of that branch. The result bar is visible on top of the panel showing the total number of hits in the result set for the current query definition. At any time, the users can fetch these results by simply pressing the fetch button to the left of the result bar.

We attached the distribution information next to the related branch of an attribute. Some of the attributes do not have the distribution information attached to them. For example, Contact Name of the Contact Info table of Figure 5.1 does not have anything attached to it. The nature of the Contact Name attribute does not allow a useful representation. There are almost as many names in the data set as the number of data points. In this work, we focused on other types of attributes, e.g., gender and age. These have useful representations.

Figure 5.1 shows some selections and updates on the bars showing the distribution information. Selections are visible in textual form below the result bar. The session can

continue as long as the users want to explore the data. When users need to see the results for their query, they can fetch the desired hits from the server matching their selections and they can view this result set as a simple list, or they can continue querying on it using some other local tools (e.g., Excel, Access, etc).

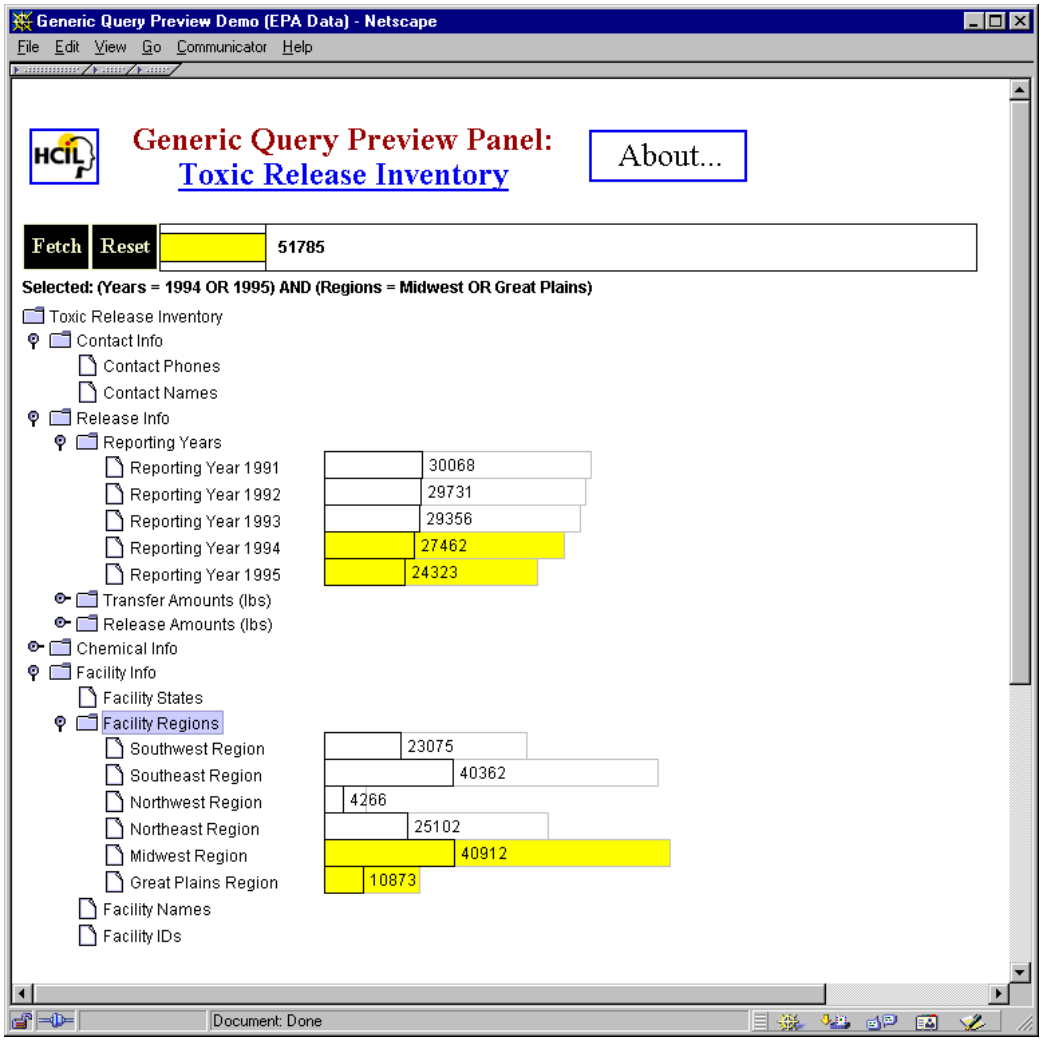


Figure 5.1: A sample implementation of the generic query previews approach. A hierarchical browser is used to display the data attributes and tables (51,785 hits)

5.3 The Next Step

After the first generalization attempt, we realized that there was also a need in our designs to give users the ability to define their personalized *views* of the data (i.e., a group of attributes of interest to the user). In query previews and later in generic query previews users could only fetch their results projected on all of the attributes of the data. This is a redundant activity if the users want only a portion of the data in terms of the number of attributes they fetch from the server. For example, in Figure 5.1 the users may only want the contact information related attributes of the data, but not the other ones. Therefore users should have the capability to define their desired-attributes list, formally their views of the data.

Hence after our early work, field experience, initial user study, user observations, and some generalization attempts, we decided to define a general user interface architecture to form a high level mechanism for efficient browsing of large online data. The result was the generalized query previews user interface architecture.

5.4 Generalized Query Previews User Interface Architecture

The generalized query previews user interface architecture has the following three components:

- A presentation component for the data table and attribute names augmented with a mechanism for selecting a user view, the *schema component*,
- A manipulation component for displaying and selecting the data distribution information, the *distribution information component*,
- A presentation component for displaying or analyzing the results, the *raw data component*.

Figure 5.2 shows how these components are attached to each other. First, users can see the database table and attribute names. Then, they can define their view by choosing some attributes. Later, using the distribution information attached to the view, they can analyze the overview of the data and make their selections. Finally, they can fetch the mapping results, and if desired, forward it to another program. The session can continue in a cyclic manner as long as the users want.

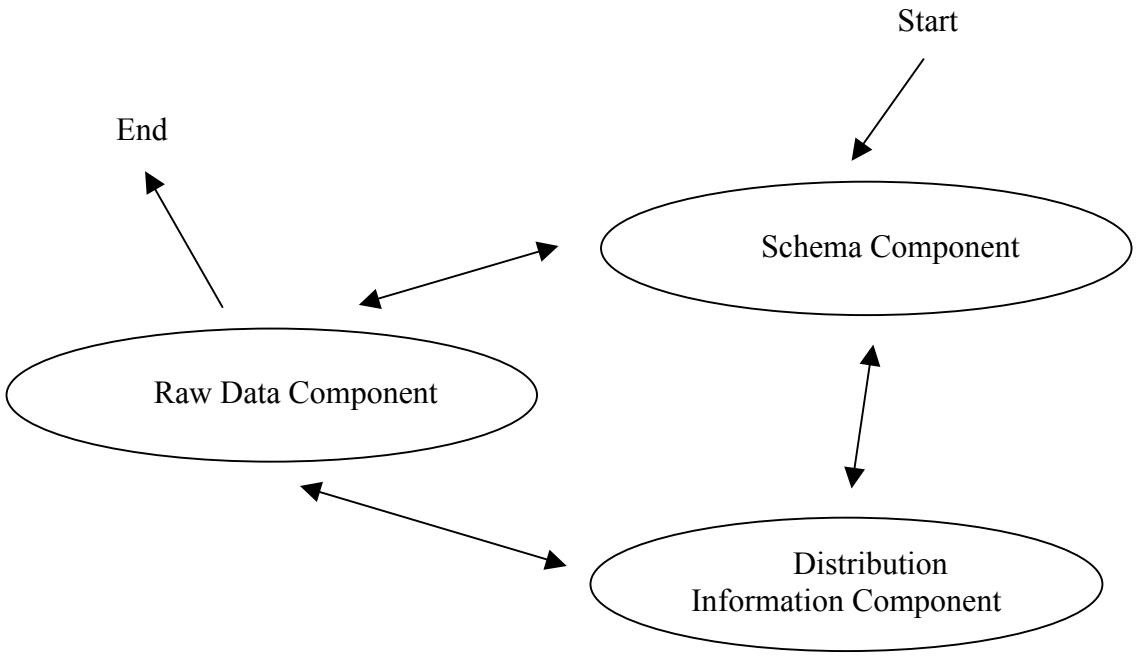


Figure 5.2: Interactions between the three generalized query previews components

To demonstrate and later to experiment with the generalized query previews user interface architecture, I have implemented a sample program called the ExpO System. The ExpO System is implemented on a data set from the 1997 United States Economic Census collections. The collection contains information about hospitals located in each of the United States Counties. It has about ten attributes and approximately 3000 rows. The

data is stored in four different relations on a networked relational database management system. Each relation represents a single table. All the tables share a unique identifier, ‘report_id’, representing a unique report from a county. The remaining sections of this chapter analyze each component of the generalized query previews user interface architecture by using the ExpO System as the sample implementation.

5.5 Schema Component

Users need to visually browse all the attribute and table names to begin understanding the data. The schema component of the generalized query previews user interface architecture serves to this purpose. For simple data sets, a simple list of attributes would generally suffice to show the attribute names. In more complex environments, such as large relational databases, more scalable approaches may be needed. SeeData [AEP96] proposes such an approach to visualize attributes and tables from a large database.

For our ExpO example, I have used a hierarchical browser, the panel on the left in Figure 5.3, to present the attribute and the table names. The root of this panel is tagged with the name of the database, ‘hospital97’. The first level in the hierarchy displays the table names. In this data, ‘loc’, ‘payroll’, ‘sale’, and ‘size’ are the four tables that share a common identifier. The second level displays the attribute names.

The schema component should be implemented after a careful analysis of the database schema size, user needs, and the relations between the schema entities. For small schemata, it may be unnecessary to implement and moreover can become annoying to use a complex visualization. However, large databases may need scalable approaches. Although in some cases the database schema may be large, the information that needs to be displayed to the users may be relatively small. In addition to these, relations between

database tables may be important for the presentation. In some applications, only a few tables sharing a set of common attributes may be of interest to the user. In others, many clusters containing such attributes may exist. Viewing these clusters in an organized manner may be important. Designers should consider all of these issues before starting to work on the schema component.

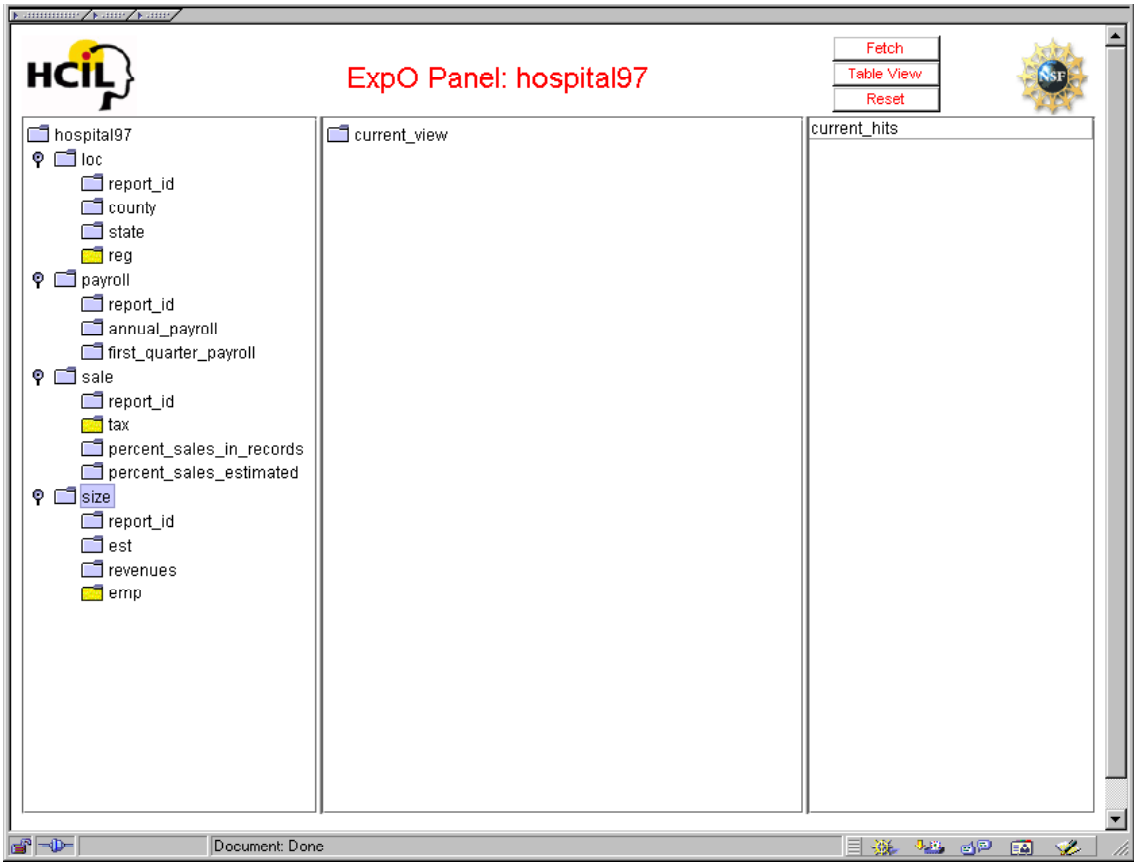


Figure 5.3: An example generalized query preview interface, ExpO, where the left panel displays the table and attribute names of a relational database as an example implementation of the schema component

In generalized query previews, users should be able to select some of the attributes of the data to form a view. The schema component also plays the role of a selector for this purpose. Again, different implementations are possible, depending on the user needs, possible view sizes, and the database schema layout. For example, in certain applications where there does not exist a single universal relation, selections of two attributes from two disjoint tables without any common attributes should be prevented. Otherwise, this can cause conflicts in implicit join operations during query processing.

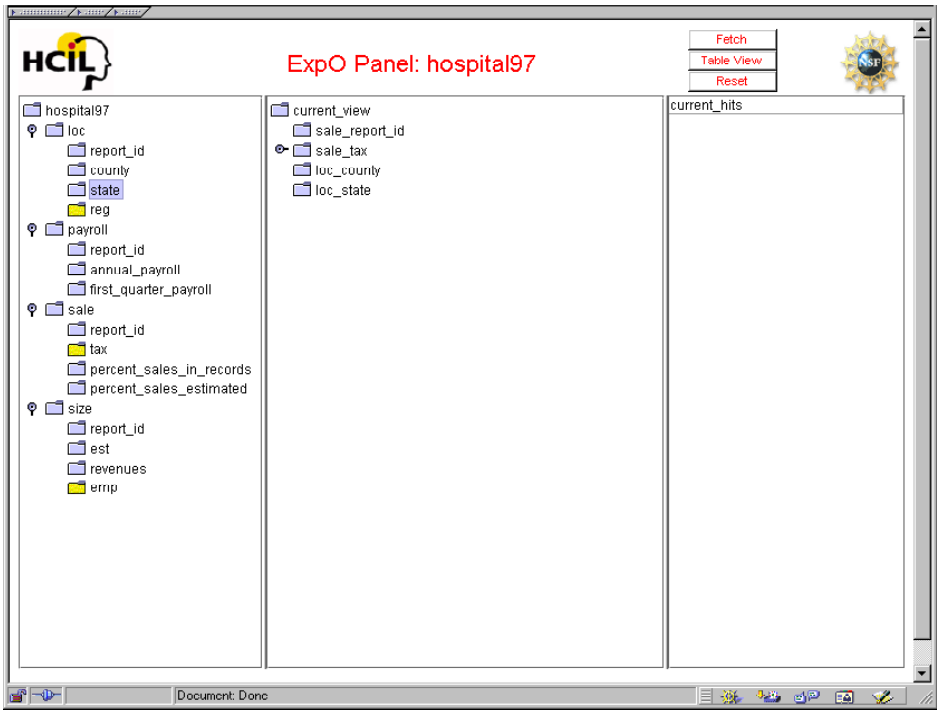


Figure 5.4: ExpO with a user-defined view after four attribute selections. The user-defined view is also a hierarchical browser.

In the ExpO example, users can select the attributes that they want to define their queries on. This action triggers the insertion operation of that selected attribute to the user view. This view is presented in a separate panel and it is also represented by a hierarchical browser (Figure 5.4). The panel in the middle depicts a few user selections and a user-defined view of the data. The selected attributes are tagged with the name of the tables that they are selected from. For example, 'tax' attribute of the 'sale' table forms the tagged name of 'sale_tax'. Joining the relations representing these tables is automatically done in the background. Hence, only the tables with common attributes can be joined to form a view. The example shown contains only four tables, each represented by a single relation. Tables can be predefined views from the data, and need not directly map to the relations of the underlying database.

5.6 Distribution Information Component

At the core of the generalized query previews architecture, distribution information plays an important role. The distribution information component of the architecture is a means to see an overview of the data and to define queries on it. User views are used to attach the distribution information. In the example from Figure 5.4, a special icon in the user view shows that one of the attributes of this view, 'sale_tax', can be expanded to show the distribution of data on this attribute. The same attribute may also be displayed with a different icon on the hierarchical browser of the database attribute and table names.

Figure 5.5 shows the expansion. The attributes are expandable into buckets. The data distribution information is attached to these buckets. Buckets are values where the data can be aggregated over. The data distribution information is attached to these buckets as

some visual aids, such as the bar charts of this example. Here, ‘taxable’ and ‘non_taxable’ are the bucket names for that attribute. Further expansions on other attributes are shown in Figure 5.6. Forming these buckets is the duty of the designer of the system who should gather information about the details of the data and the user needs.

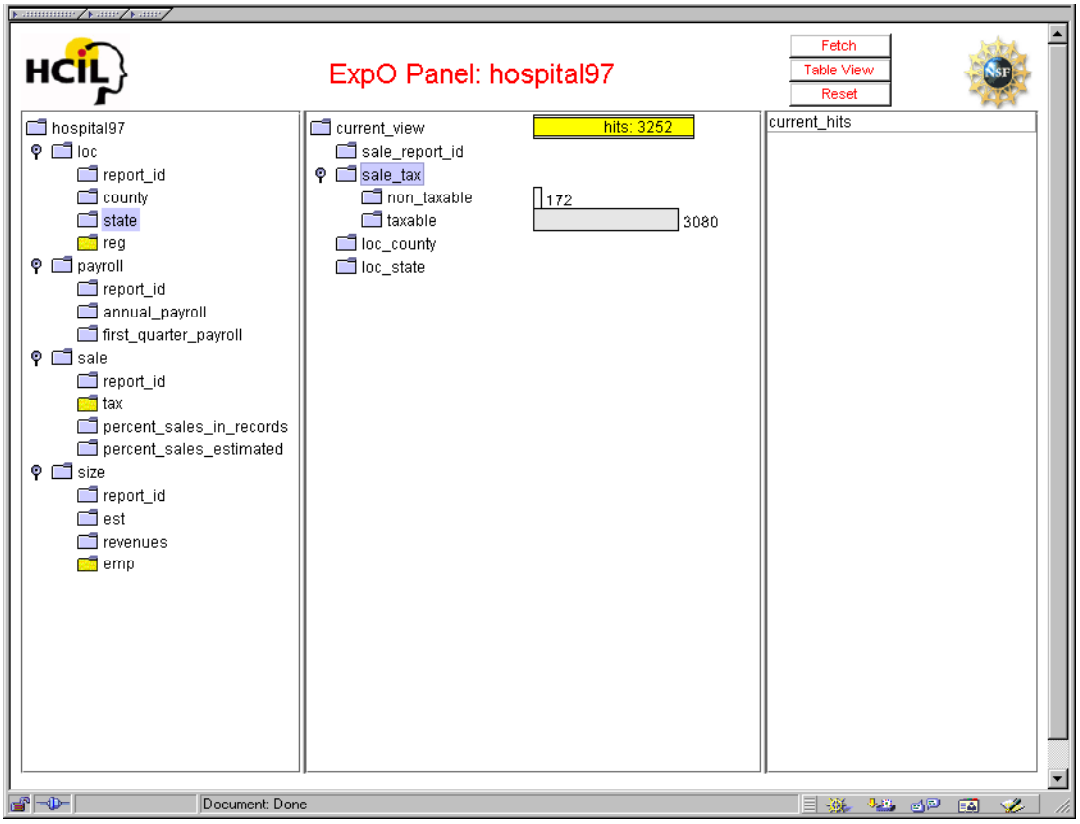


Figure 5.5: ExpO with the data distribution information attached to the user-defined view

An important feature of the generalized query previews is the capability of visualizing a preview of the results. In the ExpO example, a separate bar on the top of the middle panel shows the total number of distinct items mapping a query. This is called the result bar (showing 3252 hits in Figure 5.6). It is a preview for the result set and shows the size

of it. Hence, users will be aware of the consequences of their query submissions, i.e. whether they are submitting mega-hit or zero-hit queries. Thus, the result bar helps prevent useless query submissions.

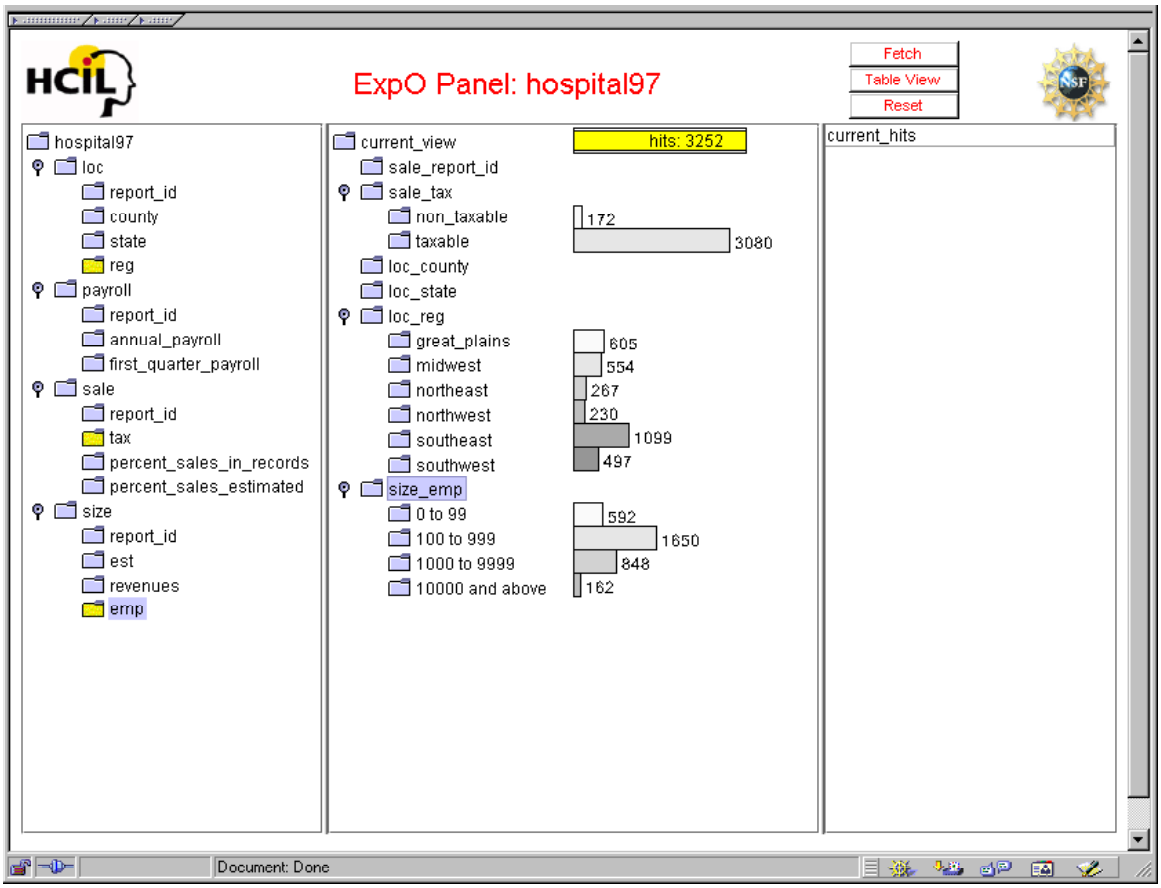


Figure 5.6: ExpO with the data distribution information attached to the buckets of three attributes expanded in the user view

In generalized query previews, queries are incrementally and visually formed by selecting items from a set of charts attached to the user view. Users continuously get feedback on the data distribution information as they continue their selections. For our example, Figure 5.7 shows an example selection. As soon as the selection is made, other

charts and the preview of results are updated to reflect the new data distribution satisfying this selection. This is called tight coupling. Possible zero-hit queries immediately become visible to the users. Users also see where the data is and how it is distributed over different values even before manipulating the bars. They can play with these interactive charts as long as they want to investigate the contents of the data. Clicking on the visual aids, bars in this case, selects or deselects them. Figure 5.8 shows some further selections on different charts. Selections within a chart map to a disjunction operation. Selections between charts map to a conjunction operation. Other types of implementations and interpretations are also possible.

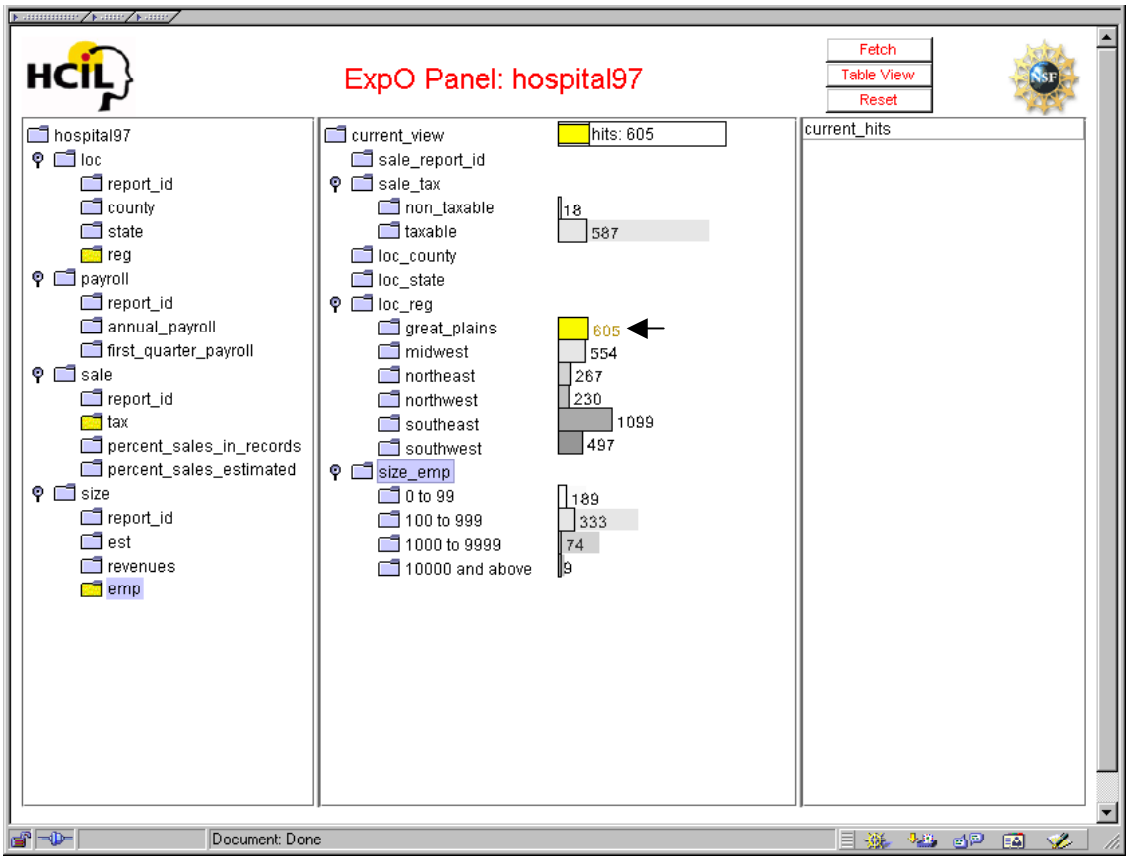


Figure 5.7: A selection is made on one of the buckets, ‘great_plains’.

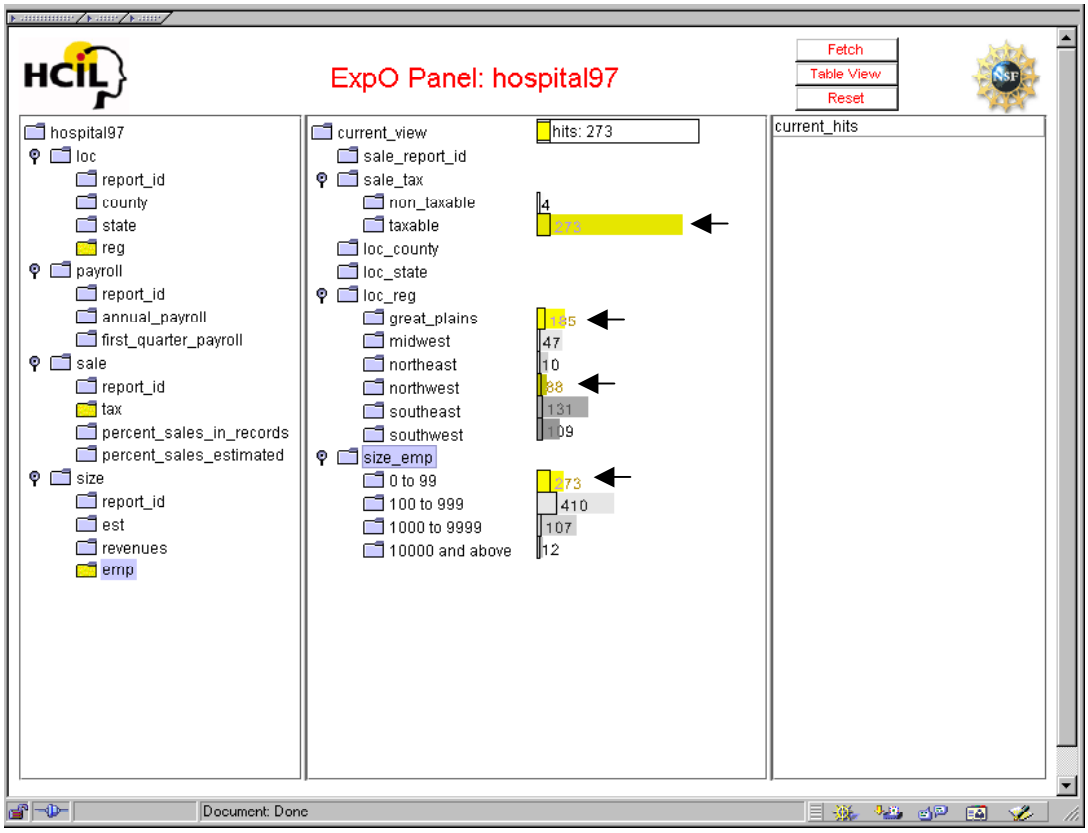


Figure 5.8: Multiple selections are made, ‘taxable’, ‘great_plains’, ‘northwest’, and ‘0 to 99’.

It is possible to display the distribution information on different types of visual aids. Figure 5.9 shows another snapshot of the ExpO system where a pie chart version of a corresponding bar chart and a series of bars mapping to another bar chart are shown. Other representations, such as a color-coded stack of bars instead of a single bar, can be implemented for different applications.

5.7 Raw Data Component

After the investigative selections, users can fetch the desired portions of the data by sending their final selections over the network. As they make informed queries, getting

neither zero-hit nor mega-hit result sets is an issue. Hence, the problem of blind formation of queries is solved.

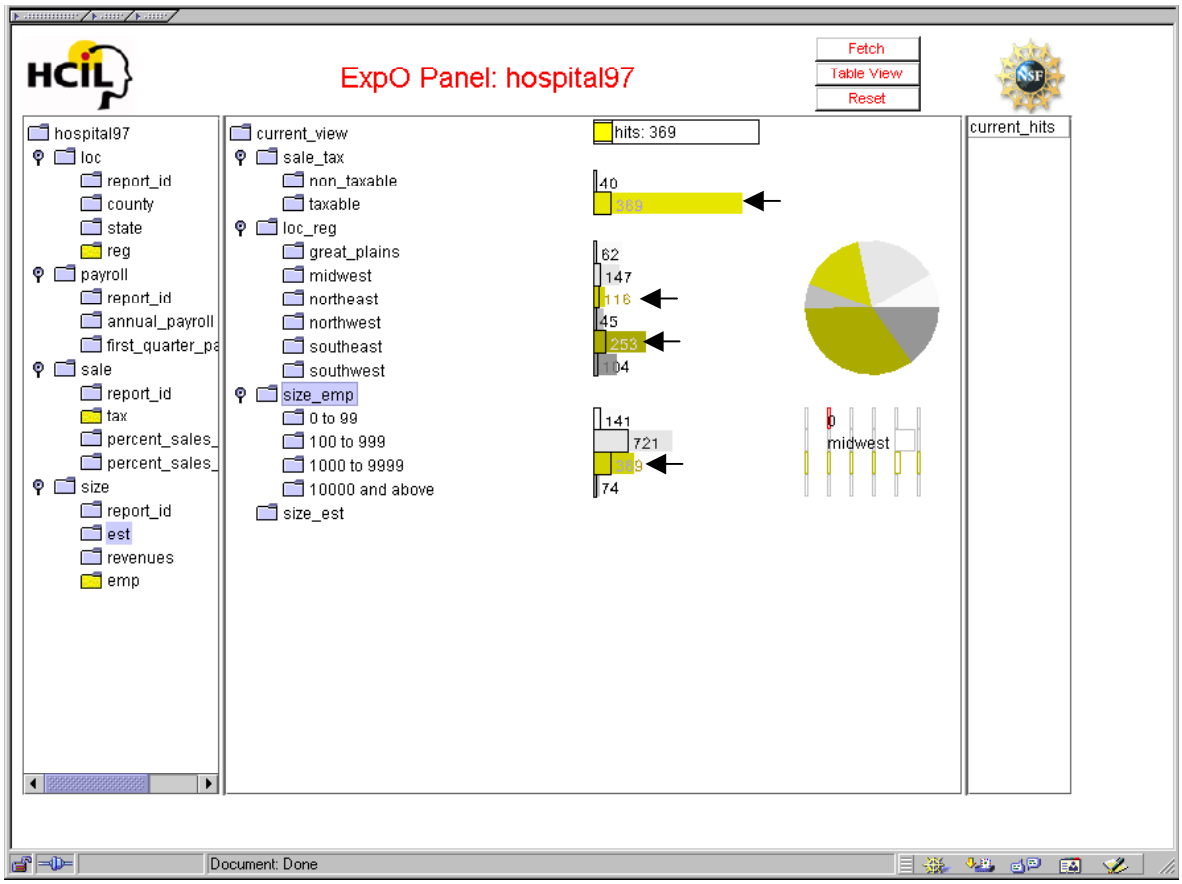


Figure 5.9: Other visual aids, such as a pie chart, may also be available to the users.

For some implementations, the designer of the system can take some drastic measures, such as preventing the query submissions for zero-hit or mega-hit queries by utilizing a threshold. It is important to note that the information given by the charts is not the probabilistic distribution of data, but the real one.

Figure 5.10 shows a result set displayed on the right side of the ExpO frame as a separate panel. Users can load this result set into a local tool for further analysis of this portion of the data (Figure 5.11, e.g., Excel). The whole process of pruning and loading a portion of the data can be repeated as long as the user desires. Generalized query previews can enable access to the results in multiple ways. In our example, only a list is shown.

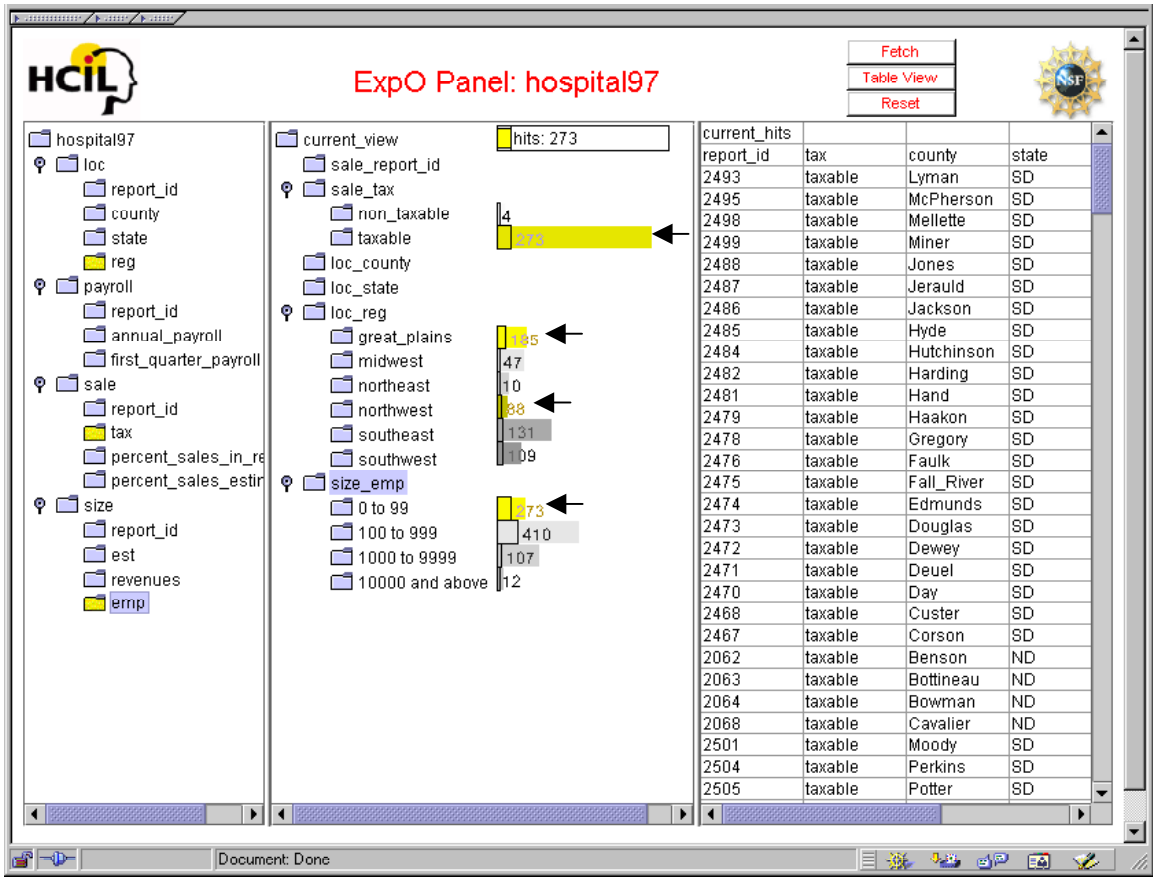
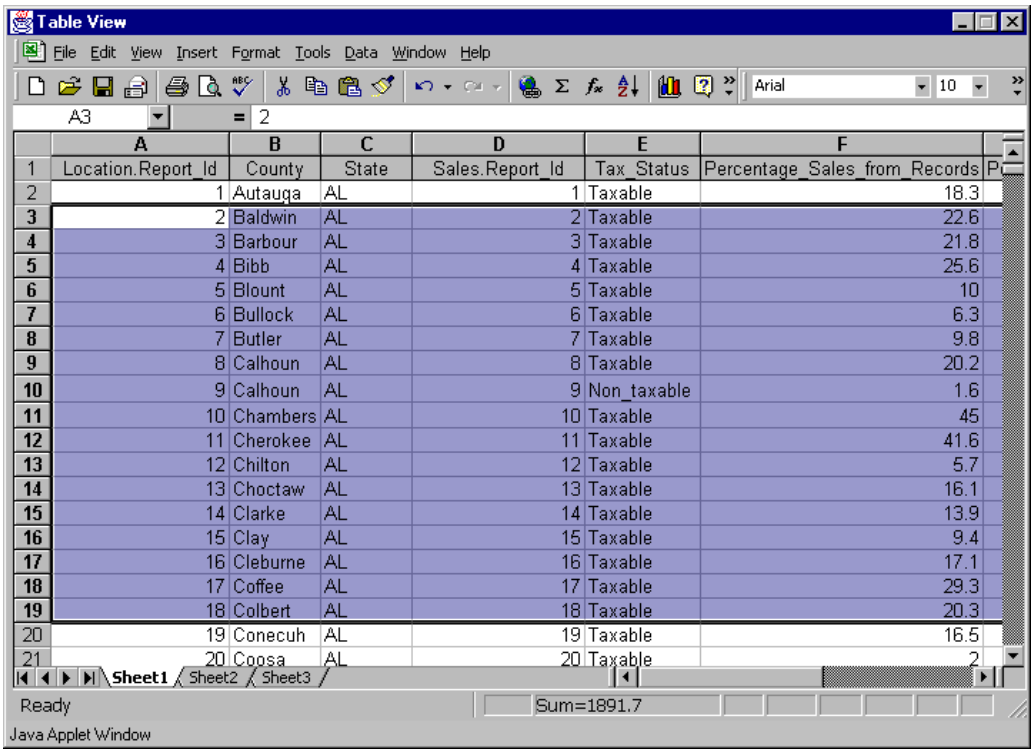


Figure 5.10: ExpO with a result set to a query displayed in a separate panel on the right, 273 hits are listed

5.8 Summary

Although query previews form a simple and effective way to prune large online data, a more general method is needed. The generalized query previews user interface architecture is designed for this purpose. This new user interface architecture consists of three components. A presentation component exists for displaying data table and attribute names augmented with a mechanism for selecting a user view, the *schema component*. A manipulation component exists for displaying and selecting the data distribution information, the *distribution information component*. Finally, another presentation component exists for displaying or analyzing the results, the *raw data component*.



	A	B	C	D	E	F
1	Location.Report_Id	County	State	Sales.Report_Id	Tax_Status	Percentage_Sales_from_Records
2	1	Autauga	AL	1	Taxable	18.3
3	2	Baldwin	AL	2	Taxable	22.6
4	3	Barbour	AL	3	Taxable	21.8
5	4	Bibb	AL	4	Taxable	25.6
6	5	Blount	AL	5	Taxable	10
7	6	Bullock	AL	6	Taxable	6.3
8	7	Butler	AL	7	Taxable	9.8
9	8	Calhoun	AL	8	Taxable	20.2
10	9	Calhoun	AL	9	Non_taxable	1.6
11	10	Chambers	AL	10	Taxable	45
12	11	Cherokee	AL	11	Taxable	41.6
13	12	Chilton	AL	12	Taxable	5.7
14	13	Choctaw	AL	13	Taxable	16.1
15	14	Clarke	AL	14	Taxable	13.9
16	15	Clay	AL	15	Taxable	9.4
17	16	Cleburne	AL	16	Taxable	17.1
18	17	Coffee	AL	17	Taxable	29.3
19	18	Colbert	AL	18	Taxable	20.3
20	19	Conecuh	AL	19	Taxable	16.5
21	20	Coosa	AL	20	Taxable	2

Figure 5.11: A result list is loaded to a local program, i.e., Excel

CHAPTER 6:
SECOND USER STUDY

6.1 Motivation for a Second Study

Generalized query previews is hopefully a forward step from the query previews approach. Although it is a more general architecture, it also introduces some overhead such as defining a view and explicit expansions of charts. Hence, there is a possibility that user performance could be disturbed and that users could be annoyed by the generalization. Therefore, there is a need for a second user study to verify and quantify the benefits of generalized query previews and measure the subjective user preferences.

6.2 User Study Methods

6.2.1 Updates for the Second Study

In this new user study, I identified the task types that would put generalized query previews into their best and worst situations, as was the case for the first user study.

Clearly specified tasks and unclearly specified tasks were again used. However, for the new study, only full relevance of query attributes was an issue. There was only a single-phase querying session. Hence, there were only two task types. These two task types varied in terms of the clarity of the specifications they have. Eight subjects performed a set of tasks, once by using a sample generalized query preview interface (i.e., ExpO System) and once by only using a form fillin interface. Another set of eight subjects worked in the opposite order. The task completion times, the number of query submissions, and the subjective preferences of the subjects were measured.

6.2.2 Hypothesis on Generalized Query Previews

Our hypotheses were: (1) For clearly specified tasks (T1') generalized query previews will lead to slower task performance, but with the same amount of query submissions, (2) for unclearly specified tasks (T2'), generalized query previews will lead to faster performance and fewer query submissions, and (3) users will always prefer generalized query previews.

6.2.3 Independent and Dependent Variables

The independent variable is the user interface type and the treatments are:

- A form fillin interface (FFN') and
- A sample generalized query preview interface, the ExpO System (EO).

The dependent variables are the *time to complete* the tasks in each interface (not including setup times), the *number of query submissions* by the users, and the *subjective preferences* of the users.

6.2.4 Subjects

Sixteen computer science graduate students were used as subjects. All of them use computers almost every day and have at least five years of experience in using computers. Hence, the user type is similar to the previous study.

6.2.5 Materials

The materials include a form fillin interface (FFN') for querying a United States Census Bureau data set (including information on approximately 3000 counties), a sample generalized query preview interface (i.e., ExpO System: EO) for the same data, a set of tasks to be performed by the subjects, a subject background survey, and a subjective preference questionnaire.

6.2.5.1 Form Fillin Interface

The form fillin interface in Figure 6.1 was used to perform queries on a United States Census Bureau data set. There are more than ten attributes in this sample data set. They are listed by using a hierarchical browser. Attributes are selected by marking the toggles near them. This action also triggers the display of editable fields attached to these attributes. The output of a query is a list of hits matching the specifications of the query. Scroll-bars can be used for scanning the list.

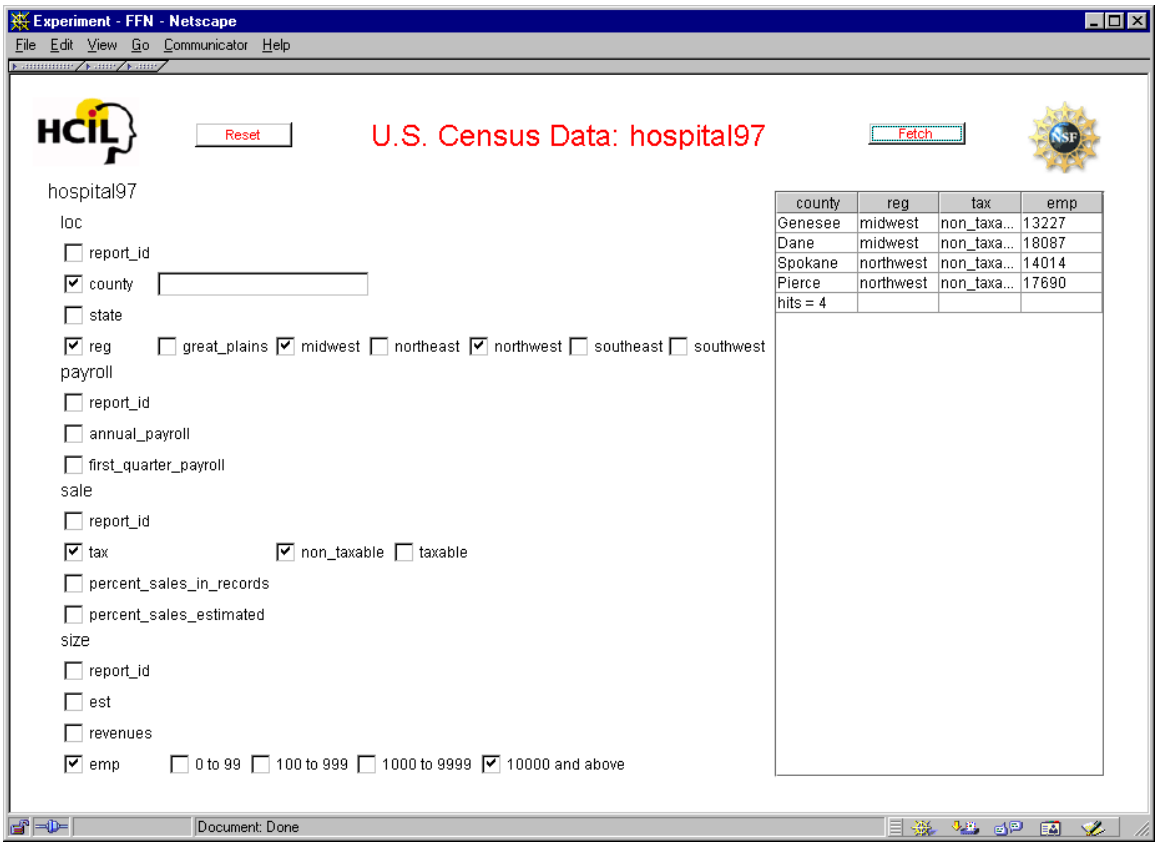


Figure 6.1: The form fillin interface used in the study. The rectangle on the right is used for displaying the result list to a query, four hits for this query.

6.2.5.2 The Sample Generalized Query Previews Interface

The ExpO System is the sample generalized query preview interface (Figure 6.2).

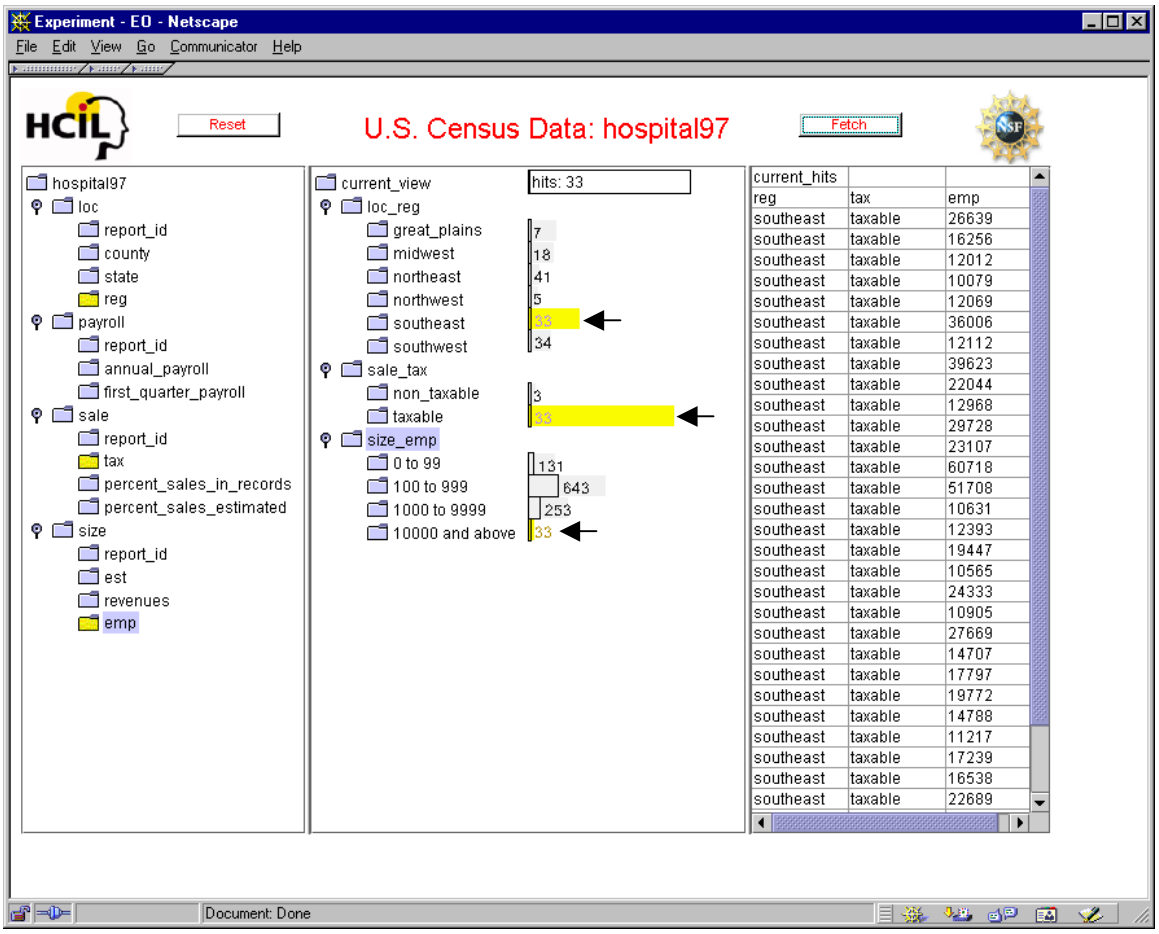


Figure 6.2: The ExpO System. The name of the system is hidden from the subjects to avoid bias towards anyone of the systems until the end of the study

6.2.5.3 Task Examples

The tasks given to the subjects were to find a list of the counties in the database satisfying the constraints that were provided. Two types of tasks were used for this purpose:

- T1': a clearly specified task, e.g. "Please get a list of all the counties that are in the northwest region and have less than 100 employees working in taxable hospitals" (a known-item search). For that type of task, users can typically find the answer by submitting a single form fillin query. The ExpO System has no specific advantage.
- T2': More vague query definitions were used, e.g. "Please get a list of all the counties from the region that has the smallest number of counties with less than 100 employees working in taxable hospitals".

For each of the above task types, four example tasks were prepared.

6.2.5.4 Subject Background Survey and Preference Questionnaire

The survey included six questions that determined the experience level of the subjects with computers. I also prepared a subjective preference questionnaire. This questionnaire included six questions that aimed to find out which of the two interfaces the subjects preferred. Similar questions to the ones in the first user study were used.

6.3 The Second Study Design

The study used a within subject counter-balanced design with sixteen subjects. Each subject was tested on both of the interfaces, but the order of the interfaces was reversed for half of the users. A parallel set of tasks (similar but not the same set of tasks) was used on the second interface to reduce the chance of performance improvement. Each set of tasks included the two types of tasks (T1', T2'), with two tasks for each of these types. The order of the task types within a task set, the order of the tasks within each task type, and the task set orders were all reversed, leading to sixteen different compositions.

6.4 Procedure and Administration

The subjects signed a consent form, filled out a background survey, received a brief demo of the interfaces, and a ten-minute training session during which they used the two interfaces (similar to but not the same tasks with the actual tasks were used). During the study each subject performed eight tasks (four in each of the interfaces). At the end of the study the subjects completed the preference questionnaire. The study took 30 minutes, including the training and the questionnaires.

The time that the subjects spent in using each interface was recorded (successful completion time of a task) along with the number of queries submitted per task. The times did not include program startup times.

6.5 Results

6.5.1 Time for Completing Tasks

Figure 6.3 summarizes the times for completing each of the task types for our subjects (clearly specified: T1', unclearly specified T2') for each of the user interfaces. For T1' tasks, the ExpO System yielded slower performance than the form fillin interface ($t(31) = 2.17$, $p < 0.05$). For T2', the ExpO System yielded faster performance than the form fillin interface ($t(31) = 9.46$, $p < 0.05$). The statistical analysis used two-tailed paired two-sample t-test for means. Each task was considered separately leading to a degrees of freedom of 31.

6.5.2 Subjective Satisfaction

The subjects answered six questions about their preferences on a 1 to 9 scale (with higher numbers indicating stronger preferences). The first question addressed the general preference of subjects for using either of the interfaces (Figure 6.4). The results show a

statistically significance preference ($t(15) = 6.37$, $p < 0.05$) for the ExpO System over the form fillin interface. The rest of the questions asked what the subjects thought about the user interfaces. The results (average scores, standard deviations, minimums, and maximums) appear in detail in Figure 6.5.

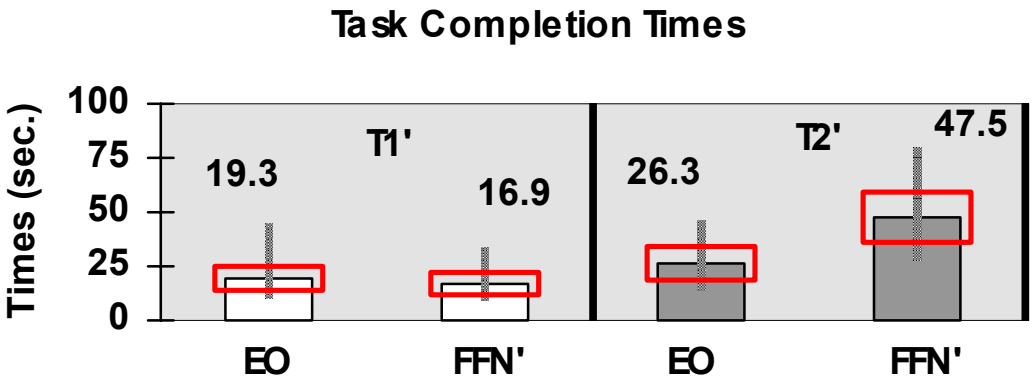


Figure 6.3: Average task completion times where the rectangles show the standard deviations and the vertical lines indicate the ranges. EO stands for the ExpO System and FFN' stands for the form fillin interface. The number of subjects used is sixteen.

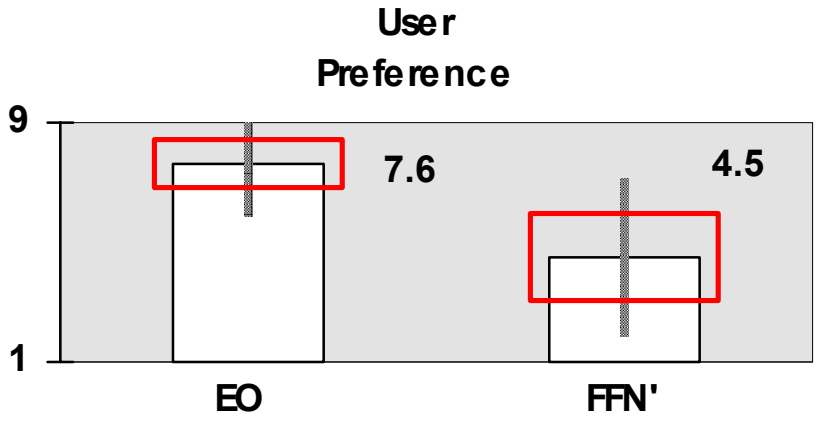


Figure 6.4: User preference for sixteen users

The scores for all of the questions were statistically significantly above the mid-point scale value of five ($t(15) = 16.43, 5.84, 5.33, 13.49$, and 9.30 respectively, $p < 0.05$).

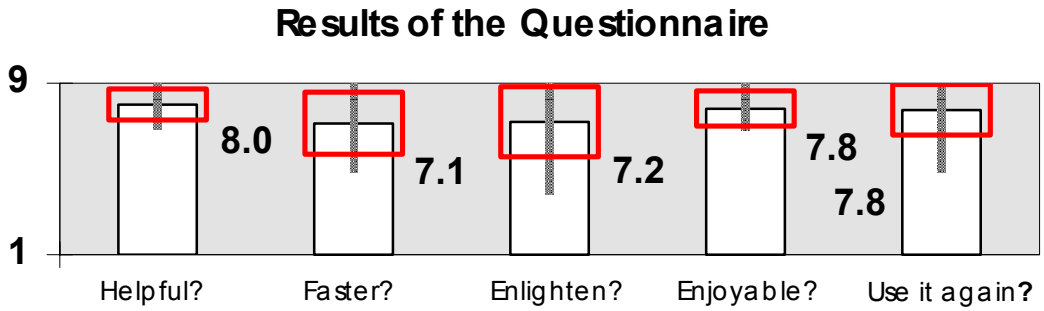


Figure 6.5: Subject questionnaire results (number of users is sixteen). Higher numbers indicate higher satisfaction for using the ExpO System.

6.5.3 Query Submission Counts

An extra piece of data that was collected in the second study was the number of queries submitted for each task (Figure 6.6). The results show a statistically significant difference ($t(31) = 22.39$, $p < 0.05$) for the ExpO System with the T2' tasks. For T1', the difference was not significant ($t(31) = 1.44$, $p < 0.05$).

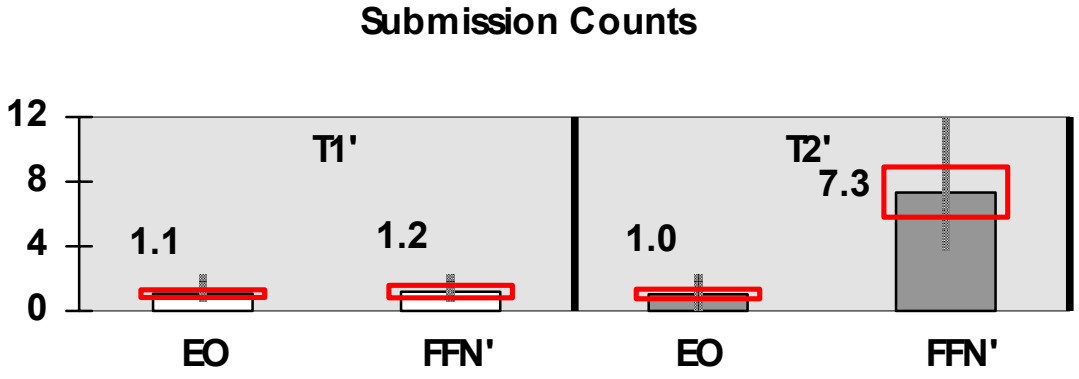


Figure 6.6: Number of queries submitted

6.6 Discussion on the Results

Our findings support the hypothesis that for unclearly specified tasks, the generalized query previews yields better performance times and counts than the form fillin interface. For the unclearly specified tasks the improvement in performance was significant (at the level of 0.05): 1.8 times faster. The counts were also more than 7 times better. For the clearly specified tasks (T1'), as expected, the form fillin interface performed slightly better in performance time, but no statistically significant difference was observed for the submission counts.

6.6.1 Clearly Specified Tasks (T1')

As expected, users of the form fillin interface for clearly specified tasks performed more rapidly since they were able to find the answer by submitting a single form fillin query. The generalized query previews had no advantage as users were performing known-item searches and they did not require an overview of the data. However, users of the sample ExpO System performed only slightly worse (14% slower). In addition to this, the number of queries submitted did not change.

6.6.2 Unclearly Specified Tasks (T2')

The generalized query previews enabled the users to see immediately which of the possible queries should be used. On the other hand, in the form fillin interface, the users had no clue about which of the possible queries will give the expected results. They had to try several possible queries, submitting many queries (on average 7 times more) until they got a satisfactory answer. Although the response time for each such query was immediate, the time for filling in the right specifications of each query caused significant differences in performance.

6.6.3 Subjective Satisfaction

The users (statistically significantly) preferred the generalized query previews to the form fillin interface. They stated that the generalized query previews was helpful, enabling them to search faster and learn more about the data (scores for these questions were statistically significantly above the mid-point value). I believe that this subjective satisfaction comes not only from the improvement in performance time which is experienced by the subjects but also from gaining better control in performing the tasks. Yet, many of the users experienced some problems understanding the concept of a view and adopting to the bar expansions when they first started using the ExpO System. These problems seem to diminish quickly with user experience.

6.6.4 User Comments

Users specifically stated that the ExpO System:

- Made them feel more familiar with the data,
- Gave them more information about the data,
- Let them define queries in a more interactive fashion,
- Is a better candidate for long term usage, especially on categorical data, but
- Can get confusing and burdensome to use for simple (know-item) queries,
- Can get confusing if there are many selections on many attributes, and
- Can get difficult to use for defining general (SQL like) queries.

The users also stated that the form fillin interface they used in the study:

- Was a simple to use but boring interface,
- Required them to understand and remember the data and their actions,
- Made them probe data blindly, and

- Could only be used if you know what you are looking for.

6.7 Summary

This study supports the claim that benefits of generalized query previews exceed the overhead of the generalizations. The benefits will amplify on real-life situations where congested networks over long distances are used. However, an overhead due to the generalizations still exists. The first study showed up to 2.1 times performance improvement while this study showed 1.8 times improvement for similar tasks. Hence, we can claim that there is some degradation in the user performance in the second study due to the generalizations. Yet, the performance improvement remained to be significant.

Implementers of generalized query previews interfaces should continue to be cautious about their application domain in terms of the query types. Generalized query previews was not meant to attack the issues that may appear in known-item searches or efficient formulation of general SQL queries. Although the number of query submissions did not change in the worst-case, the time to analyze an overview panel should still be considered before implementing an application.

In this study, we observed almost an order of magnitude of difference between the number of queries submitted among the two interfaces for some tasks. This was also observed during the first study without any statistical analysis. The second study shows that this difference is in fact statistically significant. This result also confirms the notions of the predictive model from our first study. The number of query submissions is crucial and can be reduced by using overviews and previews.

The major limitation of both of the user studies remains to be the lack of variety of the task types. More varied tasks should be used in the future studies to investigate and quantify the relationship between task types and the performance time improvements.

CHAPTER 7:
ALGORITHMS AND DATA STRUCTURES

7.1 Internal Architecture

The generalized query previews user interface architecture utilizes a client-server approach for storing, computing, and transferring data (Figure 7.1). It works with three different types of data within this approach. The first type is the database schema. This is a hierarchy of database table and attribute names. It is requested from the server as soon as the program starts on the client. The second type is the distribution information. This is requested only when needed, i.e., during the chart expansions. The third one is the raw data that is fetched up on a user query submission.

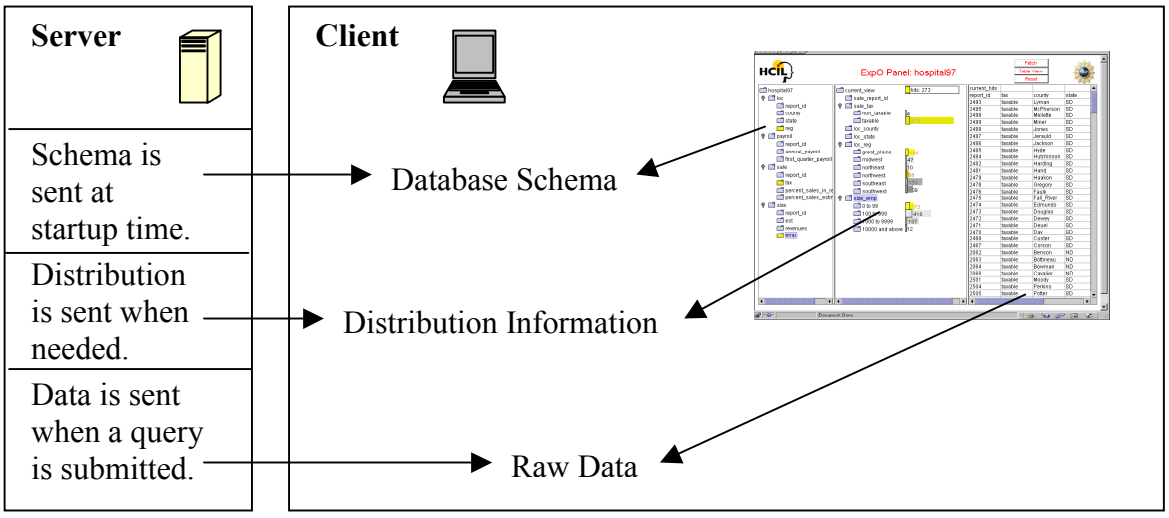


Figure 7.1: The architecture for storing, computing, and transferring data internally in generalized query previews, shown on the sample ExpO System

7.1.1 Database Schema

As soon as the generalized query previews implementation (e.g., ExpO) starts working on the client, a request for the database schema is sent to the server. Unless there is a connection error this information is sent back to the client instantaneously from the server. The size of the schema is generally very small and delays during this connection are unnoticeable by the user. The database schema is brought only once and stored on the client through out the session.

Generalized query previews uses only a part of the database schema. The database schema is a shallow tree for a generalized query preview implementation. The root is tagged with the name of the database. The first level represents a list of the names of the database tables and the second level represents a list of the names of the attributes forming these tables. The schema also contains the tags for tracking various types of other information, i.e., whether an attribute is a primary key for that table or not, whether any distribution information for that attribute exists or not, etc. A sample schema is shown in Figure 7.2 from a generalized query preview implementation (i.e., ExpO) point of view. The internal representation for the schema is a table containing a hierarchy of database table and attribute names, stored as a list (also shown in Figure 7.2).

Updates on the database schema are not visible to the client during an open session. The updates can only be viewed when the user restarts the program and hence the connection to the server. Assuming that the database schema is a fairly stable entity of the database, updating it rarely should not cause major consistency problems for the users.

Like all the other data communications with the server, the database schema is just a read-only entity and cannot be altered by the users. Security is maintained by an internal

password mechanism triggered at the startup time. This is a transparent operation for the user.

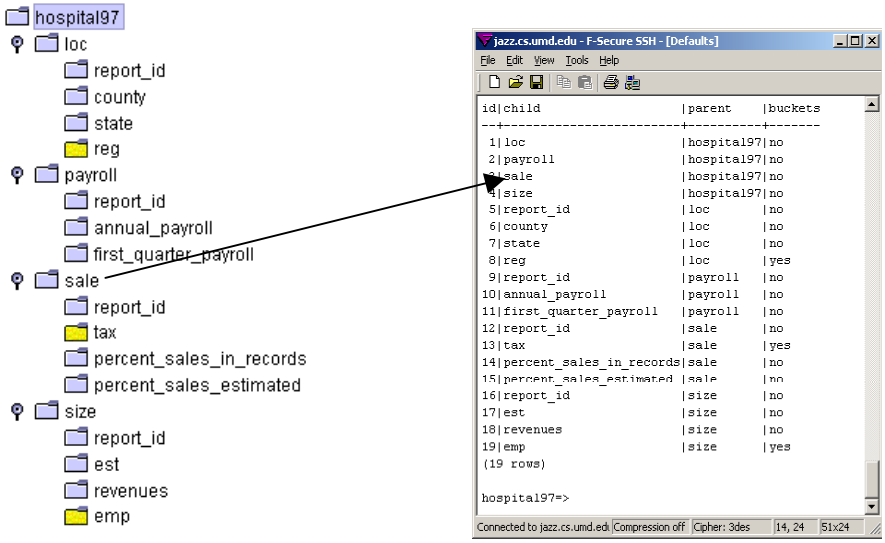


Figure 7.2: A sample database schema (in ExpO System) showing the table, the attribute names of a database (left), and a sample internal representation of this schema as a simple list (right)

7.1.2 Raw Data

When the user decides to obtain the raw data for a query, the fetch button can be used to trigger a parsing operation. This is virtually transparent to the user. The operation is performed on the user-defined view where the chart selections are made. The query is immediately converted to an SQL query and then sent to the server. Depending on the size of the result set, complexity of the query, and the network workload, the result set is returned (Figure 7.3). Some special case handling may be required at this stage to avoid unnecessary network problems and loading delays depending on the application. For example, queries requesting more than a certain maximum size of the result set may be

avoided, or results can be truncated after the submission. Multi-threaded implementations may be needed.

The result set is a user-defined view of the database. It is presented as a simple list of hits mapping to the most recent query. Later on, if desired, it can be forwarded to a local program such as a spreadsheet, visualization tool, etc. Updates on the raw data are visible from one query submission to another.

current_hits		
tax	emp	report_id
non_taxable	619	30
non_taxable	1	60
non_taxable	1	99
non_taxable	972	134
non_taxable	1	152
non_taxable	409	164
non_taxable	1	939
non_taxable	1	1395
non_taxable	1	1410
non_taxable	1	1438
non_taxable	1	1454
non_taxable	1	1490
non_taxable	1	1742
non_taxable	1	1983
non_taxable	1	2032
non_taxable	1	2067
non_taxable	1	2121
non_taxable	1	2154
non_taxable	1	2274
non_taxable	79	2381
non_taxable	1	2448
non_taxable	598	2452
non_taxable	1	2879
non_taxable	1	2891
non_taxable	320	3119
non_taxable	1	3165
non_taxable	1	3183

Figure 7.3: A sample result set presented as a simple list. It is also represented as a simple list internally.

7.1.3 Distribution Information

Distribution information forms the core of the data transfers for the generalized query previews. Although the size of the distribution information is generally negligible in

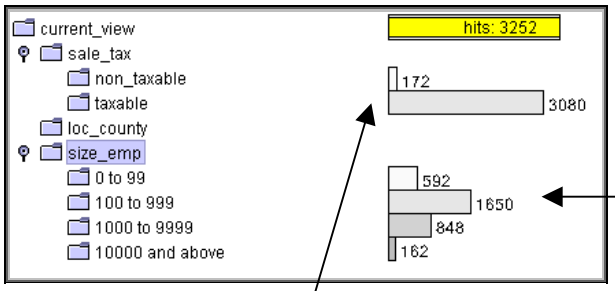
comparison to the size of the raw data (and hence more efficient to transfer), it plays a dominant role in the network connections and during the query formulation process of generalized query previews.

The database should automatically maintain this information via some triggers and batch processes. Recent database research led to the creation of a new generation of databases, data warehouses. Data warehouses enable users perform online analytical processing (OLAP) of large amounts of data. These use similar types of aggregate metadata (e.g., sums, maximums, minimums, etc.) to the distribution information of the generalized query previews. They maintain the aggregates regularly and efficiently via specialized methods [BS96] [CD97] [Gra97] [Rou97]. Hence, creation and maintenance of aggregates has become easier and more efficient with these advanced methods.

The distribution information is required to compute and update the charts that are attached to the user-defined views. Views can be formed of numerous attributes. These attributes can only come from tables that have some common attributes (i.e., identifiers). The common attributes allow the tables to be joined to form a user-defined view. This operation is thoroughly transparent to the users.

Distribution information is stored as multi-dimensional arrays on the server. For each chart combination, there may be a separate multi-dimensional array and this array may be requested from the server when that certain set of charts is expanded. Figure 7.4 displays a sample combination and its related multi-dimensional array using a snapshot from the ExpO System. These arrays are highly scalable. As the size of the database increases only the count information changes, but the sizes of the multi-dimensional arrays does not increase. During my field experience with NASA, I observed that increases in the raw

data could lead to large differences between the sizes of the multi-dimensional arrays and the raw data. For example, all of my prototypes for NASA used arrays of size 100Kbytes or less where the raw data ranged from a few megabytes to many tens of megabytes. This experience increased my confidence on the applicability and scalability of the multi-dimensional arrays.



Tax Status / Number of Employees	0 to 99	100 to 999	1000 to 9999	10000 and Above
Non Taxable	22	5	121	24
Taxable	570	1645	727	138

Figure 7.4: A sample two-dimensional array that represents the distribution information for a certain combination of charts, i.e., tax status vs. number of employees

Yet, the size of the multi-dimensional arrays can change in size with the number of dimensions used in these arrays and buckets used in each of these dimensions. For example, the array in Figure 7.4 has two dimensions. It also has four buckets for the number of employees dimension and two buckets for the tax status dimension. Hence, the

size of this particular multi-dimensional array will be the storage size of eight integers. The increase in any one of these dimensions in terms of the number of buckets they require will cause a linear increase in the size of the multi-dimensional array. If we were to increase the number buckets for the number of employees from four to five, the new size of the array will increase by two integers (for the two new tax status counts). Unfortunately, adding a new dimension creates a more dramatic change in the size of the multi-dimensional array. If we were to add another dimension to the array of Figure 7.4 that contains four new buckets, then the array size will quadruple immediately.

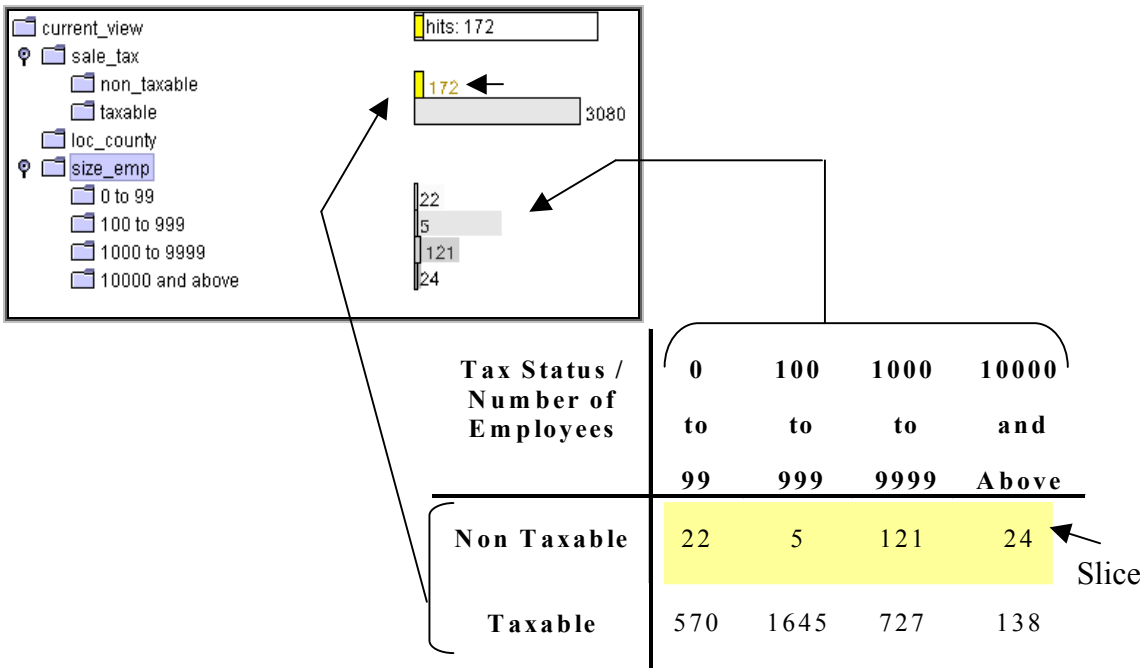


Figure 7.5: A sample selection operation and a related update on the second chart

Selections on a chart trigger the updates on the other charts. The multi-dimensional array is traversed to find the mapping slices of distribution information to these

selections. These are continuously printed on the screen as the users continue their selections. Figure 7.5 shows such an update.

7.2 Challenges of the Internal Architecture

The challenges of the internal client-server architecture can be divided into two categories:

- Manipulation challenges,
- Representation challenges.

The manipulation challenges are observed when the user wants to make selections on more than a few attributes of the database at the same time. Not only tracking the updates on multiple charts, but also maintaining the distribution information for these charts becomes difficult.

Experience shows that, users are not comfortable in tracking the updates on four or more charts. Hence, users need to collapse some of the charts to continue their queries on the other attributes.

In addition to this, arrays of four or more dimensions can easily become cumbersome to transfer over the network. This problem can be bypassed by downloading the raw data itself after the first few selections. This brings a solution to the number of dimensions manipulated simultaneously without downloading large amounts of raw data. The initial selections are generally selective enough to prune the data down to a manageable size. This approach should be implemented as a transparent operation to the user.

The representation challenges are more difficult to handle. The multi-dimensional array representation for the distribution information is effective for many of the data

types. On the other hand, some of the data types cannot be handled easily with the multi-dimensional array approach.

The multi-valued data types have the most generalized version of these challenges. Temporal data forms a good example to show it. A record covering a range of dates in a temporal database is a record that contains a multi-valued attribute. This may result in the duplication of the same record in the multi-dimensional array. It is counted once for each date it spans. Hence, the distribution information represented with the array is no longer the same information represented with the actual data set itself.

Similar situations are observed with the NASA prototypes of the query previews, where a satellite picture maps onto multiple regions of earth and not to a single longitude and latitude (i.e., a geographical point). The situation is more dramatic when large and multiple ranges of values are used in a database. This may result in more erroneous representations of the data with high levels of duplication. Therefore, to accommodate multi-valued data types and hence attack the representation challenges, the multi-dimensional array approach must be extended.

7.2.1 Focusing on the Multi-valued Data Types

Multi-dimensional arrays are fundamental for storing the distribution information. Figure 7.6 shows a sample array used for this purpose. This sample will be used throughout this section to traverse the multi-valued data type challenges in depth.

		K	L	M	N	Attribute #1
	W	0	1	2	0	
	X	0	0	1	1	
	Y	0	0	1	1	
Attribute #2	Z	0	0	0	0	Q

Figure 7.6: A sample multi-dimensional array. Each of the dimensions of this array represents an attribute. The counts give the number of records that map to the associated values of the two attributes. An example range query is shown as a colored rectangle, Q.

The distribution information shown with these arrays do not thoroughly represent the multi-valued attributes. For example, the array presented in Figure 7.6 could be used to accurately display the seven records given in Figure 7.7, but not the ones given in Figure 7.8. The reason for this is that we lose track of some of the overlapping information in the second data set. We can never know whether the last four items of the multi-dimensional array has come from a single record that has some range values (Figure 7.8) or from four separate records (Figure 7.7). The intersection information is lost in the conversion.

The range query shown in Figure 7.6 matches to two separate records of the single-valued data set shown in Figure 7.7. However, there is only one record that actually maps to both of the data points when the multi-valued data from Figure 7.8 is used.

Id	Attribute #1	Attribute #2
1	L	W
2	M	W
3	M	W
4	M	X
5	N	X
6	M	Y
7	N	Y

Q

Figure 7.7: A sample single-valued data set. Records 6 and 7 form the answer for the range query shown in Figure 7.6 (colored rectangle Q).

Id	Attribute #1	Attribute #2
1	L	W
2	M	W
3	M	W
4	M - N	X - Y

Q

Figure 7.8: A sample multi-valued data set. Record 4 forms the only answer for the range query presented in Figure 7.6 (colored rectangle Q).

We can always duplicate the multi-valued record of Figure 7.8 into multiple separate records. However, this may increase the data size without adding much useful information. Similar problems were also observed for the data structures presented by [TBS97] for dynamic queries.

7.2.2 Ranges

Our research on multi-valued data types revealed some solutions for some of the subsets of the multi-valued data types [BT98]. If the data is range data consisting of single intervals and if the queries are range queries consisting of continuous single ranges, then some modifications on the multi-dimensional arrays may solve the problems. For range queries that are defined in two dimensions and for range data with two attributes, Figure 7.9 gives such a modification on the multi-dimensional arrays to form a new data structure. This approach can make the arrays work without transferring the actual data over the network and without drastically increasing the querying times. It is also generalizable to any number of dimensions.

Using Euler's well-known formula as a basis for this approach, the changes can be explained more easily from a geometrical point of view, as presented in Figure 7.10. This also gives an insight for the generalization of the approach to multi-dimensions. F represents the faces, V represents the vertices, and E represents the edges. F is equal to 2 for our example query in Figure 7.9. Similarly, E is 1 and V is 0. Therefore, the answer is 2 - 1 = 1, as it is the case in Figure 7.6 where we only have a single multi-valued record as an answer to our range query.

The idea in this new approach is to keep the counts for the intersecting boxes along with the actual duplicated counts of the data. Note that, we can only find the non-duplicated cardinality for a query using these intersections, if the record is continuously covering a single region and similarly the query contains a single continuous interval.

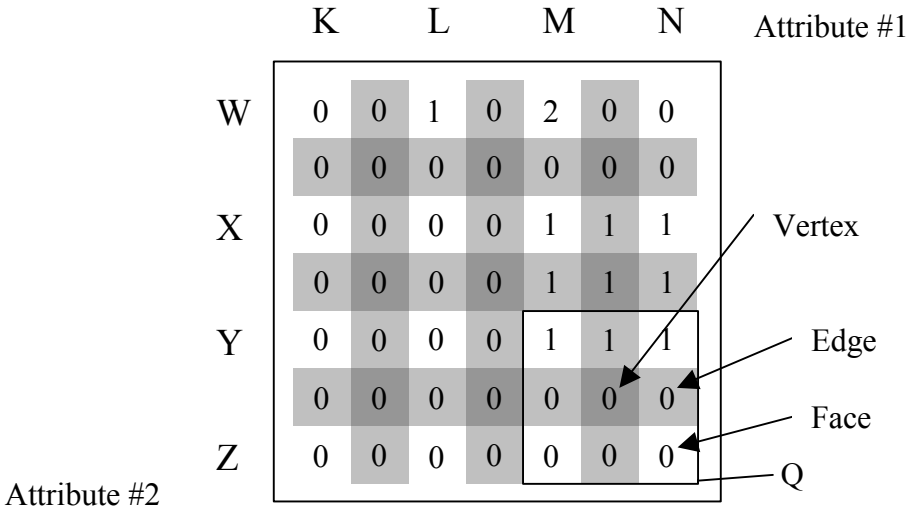


Figure 7.9: The counts in the colored boxes are the cardinalities of the intersections between any two neighboring-boxes of the original array. The same query rectangle Q from Figure 7.6 is used for this figure. The data is from Figure 7.8.

Result = F + V - E

F = Sum of all face counts inside the query
V = Sum of all vertex counts inside the query
E = Sum of all edge counts inside the query

Figure 7.10: Representation of the approach formulated by using Euler's formula

Additions and subtractions for a small array may not be very costly. On the other hand, large arrays should be preprocessed using a prefix sum approach, shown in Figure 7.11. Then, the results for any range query could be found more efficiently per query.

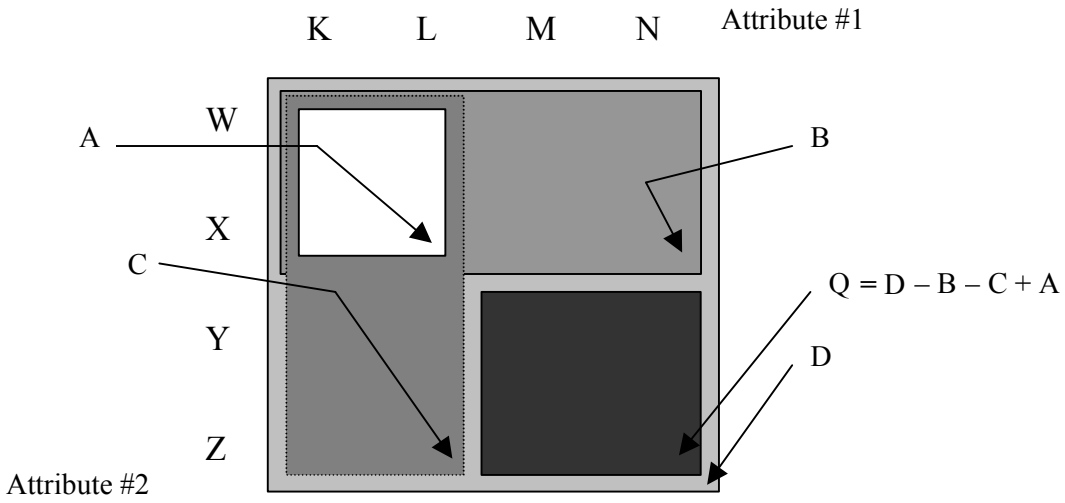


Figure 7.11: The prefix summed version of the data in Figure 7.10

Prefix sums are taken from left to right and then from top to bottom using the counts on the faces, edges, and vertices, of the boxes in Figure 7.10. Q denotes the range query region. This is the same region with Figure 7.6. The answer to the query is computed using the formula given for Q in Figure 7.11. The prefix summed counts on the right bottom corner box of each rectangle of Figure 7.11 should be used for the calculation of the answer for Q. If the same data from Figure 7.8 is used, then A is 1, B is 3, C is 1, and D is 4. Hence, 4 - 3 - 1 + 1 = 1. This is the right answer to our query Q. Note that this value is equal to the value found using the formula derived by using the Euler's formula.

A batch process can compute the prefix sums. Hence, at querying time, the server needs to process only the single addition and the two subtraction operations to find the answer.

The alterations have changed the size of the multi-dimensional arrays. However, this increase is only by a small factor and will not dramatically effect the transfer times for the arrays.

7.2.3 Generalized Multi-valued Data Types

The alterations for single ranges can easily be generalized to any number of dimensions. Unfortunately, any number of ranges with any number of dimensions is a much harder problem to solve.

By using exhaustive tables for all possible intersections between the multi-dimensional array cells, the problem with any number of ranges can be solved in a less scalable manner than the single ranges.

In this section, a formal definition of the multi-valued data type issues will be presented. For detailed proofs and generalizations please refer to [BT98].

Formally, lets say a record from some data set with d attributes creates a d-dimensional multi-valued attribute problem when all of the attributes are of multi-valued data types. And lets define:

- $N = \{0,1,\ldots\}$, the set of natural numbers,
- N^d is the set of all d-dimensional lattice points in the 1st quadrant,
- An *interval* is a set $\{a,a + 1,\ldots,b\}$ of consecutive natural numbers,
- A *generalized interval* is a subset of N,
- A *rectangle* in N^d is a cross-product $I_1 \times \ldots \times I_d$ of d intervals $I_1 , \ldots , I_d$,
- A *generalized rectangle* in N^d is a cross-product $I_1 \times \ldots \times I_d$ of d generalized intervals $I_1 , \ldots , I_d$.

So the two main problems of interest are:

- Rectangle Intersection Problem:
    - Data to be preprocessed: A list of D rectangles in N^d,
    - Problem Instance: A single rectangle Q in N^d,
    - Question: How many elements of D intersect Q ?
- Generalized Rectangle Intersection Problem:
    - Data to be preprocessed: A list of D generalized rectangles in N^d,
    - Problem Instance: A single generalized rectangle Q in N^d,
    - Question: How many elements of D intersect Q ?

Hence, formally, the Generalized Rectangle Intersection Problem maps to our general multi-valued data type problem on a d-dimensional space, and the Rectangle Intersection Problem maps to the single range version of it.

Now, consider the worst-case scenario. Let R denote a fixed rectangle in N^d that contains each element of D and with the intersection alterations we are trying to solve:

- The Rectangle Intersection Problem with $O(\rho)$ preprocessing time per element of D, using tables of size ρ, in time $O(d2^d)$ per query,
- And similarly, the Generalized Rectangle Intersection Problem with exhaustive tables of size $2^{n_1}2^{n_2}\ldots2^{n_d}$ for all possible intersections.

Where:

- $\rho = (2n_1\text{-}1)\ldots(2n_d\text{-}1) < 2^d |R|$ and
- R is an $n_1 \times \ldots \times n_d$ rectangle.

Therefore, we can observe that the Rectangle Intersection Problem scales much better than the generalized one.

For each d-dimensional cube c, let #(c) denote the number of rectangles r in the list D such that the interior r intersects the interior of c. In particular, the answer to the query Q is #(Q). We have:

$$\#(Q) = \Sigma_{0 \le k \le d} (-1)^{d-k} \Sigma_{c \text{ is a } k\text{-dimensional unit cube in the interior of } Q} \#(c)$$

Why? Because each rectangle in D that does not intersect Q contributes '0' to the sum, and each rectangle in D that intersects Q contributes '1' to the sum. The derivation of this general formula for any number of dimensions using Euler's theorem is in [BT98].

Let dim(r) denote the dimension of a rectangle. If we stored $(-1)^{d-\dim(c)} \#(c)$ for each unit cube c, then we could compute the sum specified in #(Q) by summing over each unit cube contained in Q. Better yet, if we stored d-dimensional prefix sums, we can evaluate that sum in constant time. For each unit cube a we store $\Sigma_{b \le a} (-1)^{d-\dim(b)} \#(b)$, where the inequality must hold on every coordinate. Given such a table, the sum specified in #(Q) may be obtained with $2^d - 2$ additions and subtractions, by the principle of inclusion and exclusion. The total storage needed is the number of unit cubes interior to R whose dimension is d or less, which is exactly ρ.

7.2.4 Optimizations

Different data types may require different versions and sizes of multi-dimensional arrays to be used for representing the distribution information. Although the sizes of these arrays do not change with the updates on the raw data, some of these arrays can get considerably large with the increasing number of dimensions, buckets, etc. There are various methods to optimize the access mechanisms and sizes of such arrays:

- Materialization of frequently asked queries is a standard database optimization technique. Similarly, only a subset of the large arrays can be sent over a network

to optimize the transfer times. Portions of the array that are known to be used with high frequency and queries that are common on such portions of the arrays can be used to materialize some of the answers for the distribution update queries on the server.

- Another method is to reduce the number of buckets used per attribute for some of the less frequently used dimensions of the arrays.
- One method is to restrict the number of selections and selection patterns over an attribute. This may reduce the number of the intersection counts kept for a multi-valued data set.
- Analyzing the data for the intersecting array cells may be used to restrict the need for a large set of exhaustive tables. Sparse data sets may contain fewer intersections or only a few clusters of intersections that can lead to compressed representations of the multi-dimensional arrays.
- Restricting the number of attributes manipulated simultaneously is another simple but effective method.
- Cases where the availability of data, instead of the exact distribution of data is needed can lead to other optimization methods. The size of the tables can be reduced dramatically by compressing the information kept in these arrays. For example, using the similar methods to the ones presented in 3.2.3, i.e., binary previews, only the boolean representations of the distribution information can be considered. Bitmaps can be used instead of integer arrays.

These methods can be combined to obtain more optimal solutions than a single method would provide.

7.3 Software Implementation Issues

The generalized query previews user interface architecture requires the following modules to be implemented, installed, and maintained for a sample application:

- A client module that contains the database schema related communication protocols and implements the presentation and manipulation code for the schema component of the generalized query previews user interface architecture,
- A client module that contains the raw data related communication protocols and implements the presentation and the transportation (to a local program) code for the raw data component of the generalized query previews user interface architecture,
- A client module that contains the distribution information related communication protocols and implements the presentation and query formulation code for the distribution information component of the generalized query previews user interface architecture,
- A client module that integrates all the client modules and transports information from one module to another,
- A database management system and a series of database server programs that will create and update the related parts of the schema and the distribution information.

In the context of my sample implementation, the ExpO System, the following modules were implemented in Java (using JDK-1.2) that are integrated as an applet (Figure 7.12):

- A client module was implemented to obtain the database schema information such as the table and attributes names and some distribution information tags from the

database server. This module uses simple general SQL queries to obtain the information from a relational database server. It uses a hierarchical representation to present and let users manipulate the information,

- A client module was implemented to download the raw data, i.e., results of a query and present them as a list. The results can be forwarded to a spreadsheet,
- A client module that help users form queries using the distribution information. It displays and helps users manipulate:
    - A user-defined view with a hierarchical browser. This structure is formed of attribute names and buckets for different attribute values,
    - A series of charts attached to the user-defined view. These are implemented by using Java-Swing. This modular implementation enables a simple path for future advancements. Different types of charts can easily be implemented and plugged into the current system with minor changes,
    - A result bar showing a preview for a query,
    - A sub-module to maintain the current user query and download the distribution information when needed,
- A client module to integrate the three components of the applet, reset them when needed, and pass information between them such as the results of a user query.

The client program connects to a Postgres Relational Database Server installed on a Sun Ultra-1 workstation. Querying is always done by using the simple general principles of the SQL querying language. This makes the implementation easily convertible to another database/client-applet pair when needed. The distribution information is created

by some simple batch queries. It is assumed to be static for the current database and the applet implementation.

Three versions of the applet were implemented (about 5,000 lines of code excluding the comments). The coding effort for query previews is not included in these versions.

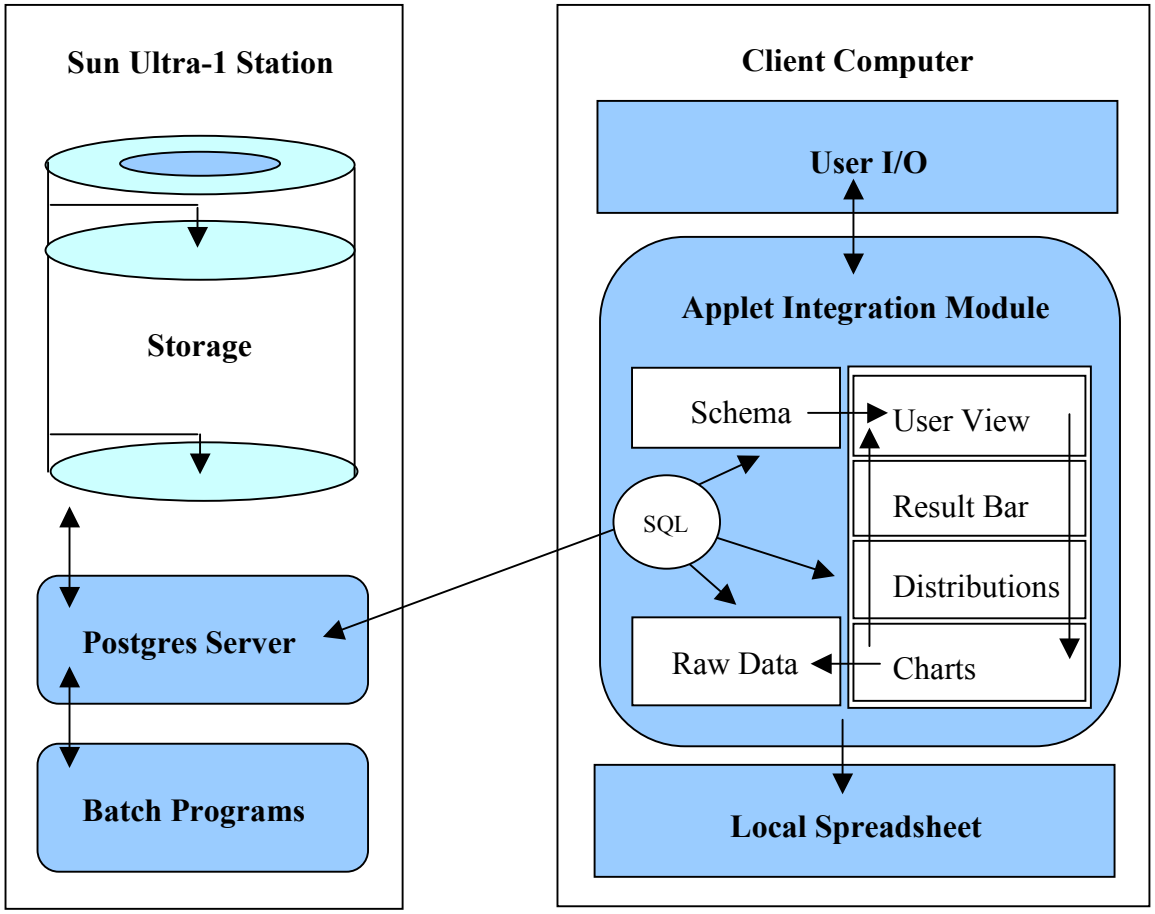


Figure 7.12: The software implementation and integration for the ExpO System

7.4 Summary

In this chapter, the internal architecture for generalized query previews and some software implementation issues are described. The internal architecture is a client-server

architecture, where the database schema, the raw data, and the distribution of data form the basis of all the communications.

The challenges for manipulating and representing multi-dimensional arrays are presented for the internal architecture. Some approaches for attacking these challenges are introduced.

Multi-valued data types form the root for most of the challenges. This dissertation contributes to the field of algorithms and data structures by introducing a method for partially solving the problems introduced by the multi-valued data types.

CHAPTER 8:
CONCLUSION

8.1 Contributions and Benefits

There are three main contributions of this dissertation to the field of visual data mining and information visualization and to the field of algorithms and data structures. First, the generalized query previews work introduces a general user interface architecture for browsing large online data sets. Using metadata (i.e., the distribution of data) for browsing and pruning raw data is an intriguing idea. Especially, when the data is stored in a remote location, accessing only the metadata can be very efficient. Metadata also does not grow with the size of the raw data. Only the distribution information, but not the size of the distribution information is updated. Showing the results set size before accessing the results is another intriguing idea. Users can immediately see what they should expect from their query submissions. Also, simple hierarchical display and creation of a user-defined view is a very intuitive idea for defining queries. In summary, using overviews and previews enables efficient and intuitive browsing of large online data and is often a dramatically faster alternative to traditional approaches for accessing such types of data.

Second, field studies and experimentation clarified the application domain for generalized query previews and led to a cognitive model that predicts user performance with a range of tasks. Generalized query previews is especially useful when users need to probe the data. The approach helps users refine their queries and guides them in the query formulation process. Situations where users know what they want from the data are not

the strong cases for generalized query previews. Identifying these strengths and weaknesses will make the generalized query previews more applicable to real-life situations.

Third, this dissertation makes contributions to the field of algorithms and data structures. Usage of multi-dimensional arrays for storing data distributions is a common approach and especially multi-valued data tends to weaken it. New data structures and algorithms will be useful in strengthening the applicability of generalized query previews and other such approaches.

8.2 Future Work

Many avenues can be investigated as future work on generalized query previews. First, working on multiple views rather than a single user view can be considered. In many cases, users may want to form two separate queries simultaneously.

Second, using a hierarchy of charts rather than a single level of charts can be investigated. This may help users to drill-down into the data (e.g., from years to months, months to weeks, etc).

Third, using more varied approaches for displaying the data distribution may be worth exploring. For example, scatter plots may be more useful for some types of data in comparison to other visual aids such as the bar charts. Shneiderman in [Shn94] mentions the need for a scatter plot widget for certain types of applications, Figure 8.1. Figure 8.2 shows another similar scatter plot. Hence, utilizing a variety of visualizations may be helpful in understanding and querying different types of data.

Fourth, working with user-defined buckets rather than the predefined ones may form a promising idea. Users may want to set the border values for the buckets rather than

using the predefined ones. Unfortunately, some types of attributes, like the gender of a person, may not be suitable for this idea. Plus, creation of such buckets can take time.

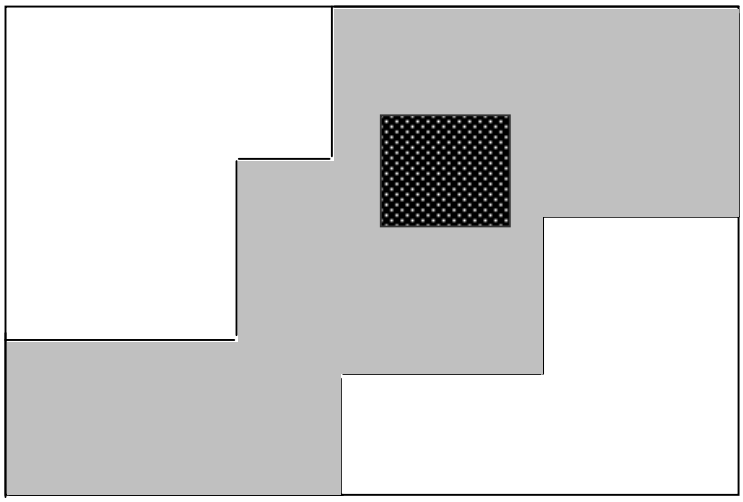


Figure 8.1: A two-dimensional widget for displaying and selecting data. In this example white regions indicate the areas where there is not any available data. The gray region indicates the area where the data is available. The dark gray box indicates a sample user query that will not return zero-hits.

Fifth, a wide avenue of ideas can be investigated for creating methods to efficiently manage some of the server functions (e.g., distribution information creation and design, efficient online updates with dynamic data, etc).

Sixth, creating a history keeping mechanism for generalized query previews is a useful idea. Users may want to look at their previous queries and results (not only within a session, but also between sessions). They may want to compare them to the current ones since this may show them some insight about the trends in data.

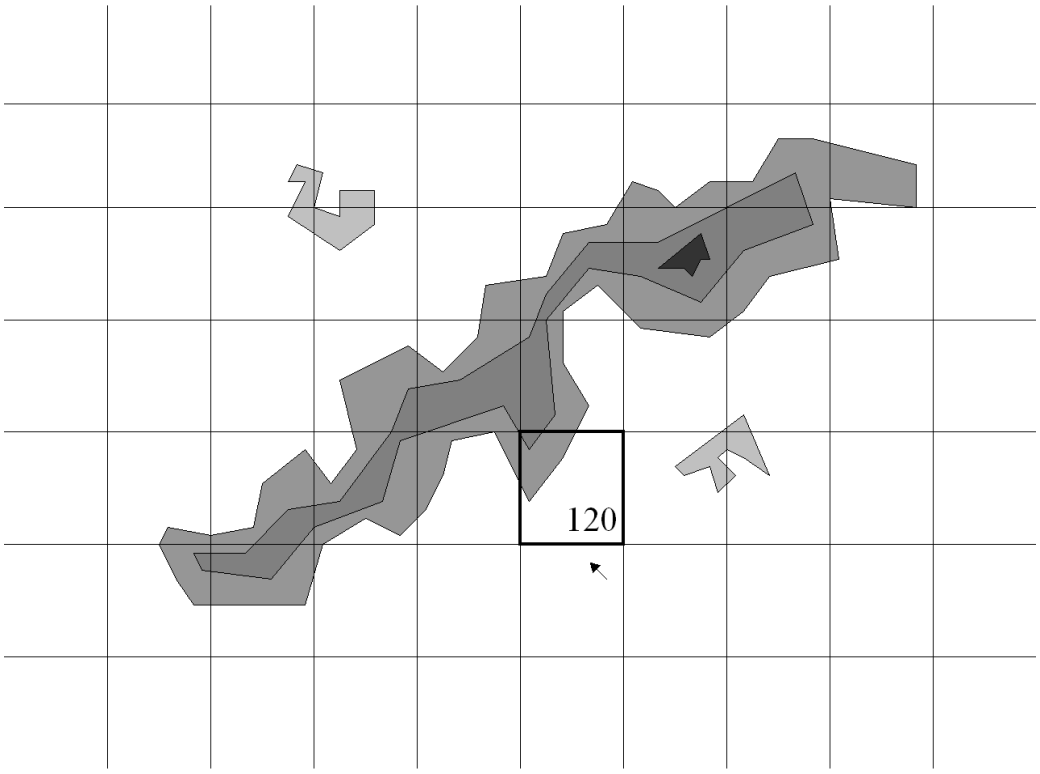


Figure 8.2: A scatter plot is shown using three concave hulls presenting the borders of the three main clusters of the underlying data. Multiple iso-surfaces are used in the large cluster to display the data distribution information. The exact counts mapping to a certain grid location are visible upon a user request.

Finally, applications on web-page searches can also be considered as a separate research direction. Internet with a large collection of web-pages forms an unstructured large online database. Using the distribution information in tandem with the classical keyword search mechanisms can be useful for the users of the Internet search engines. Similar ideas have already started to appear as prototype applications for certain subsets of the Internet. Figure 8.3 shows such an application for displaying the results of a search operation.

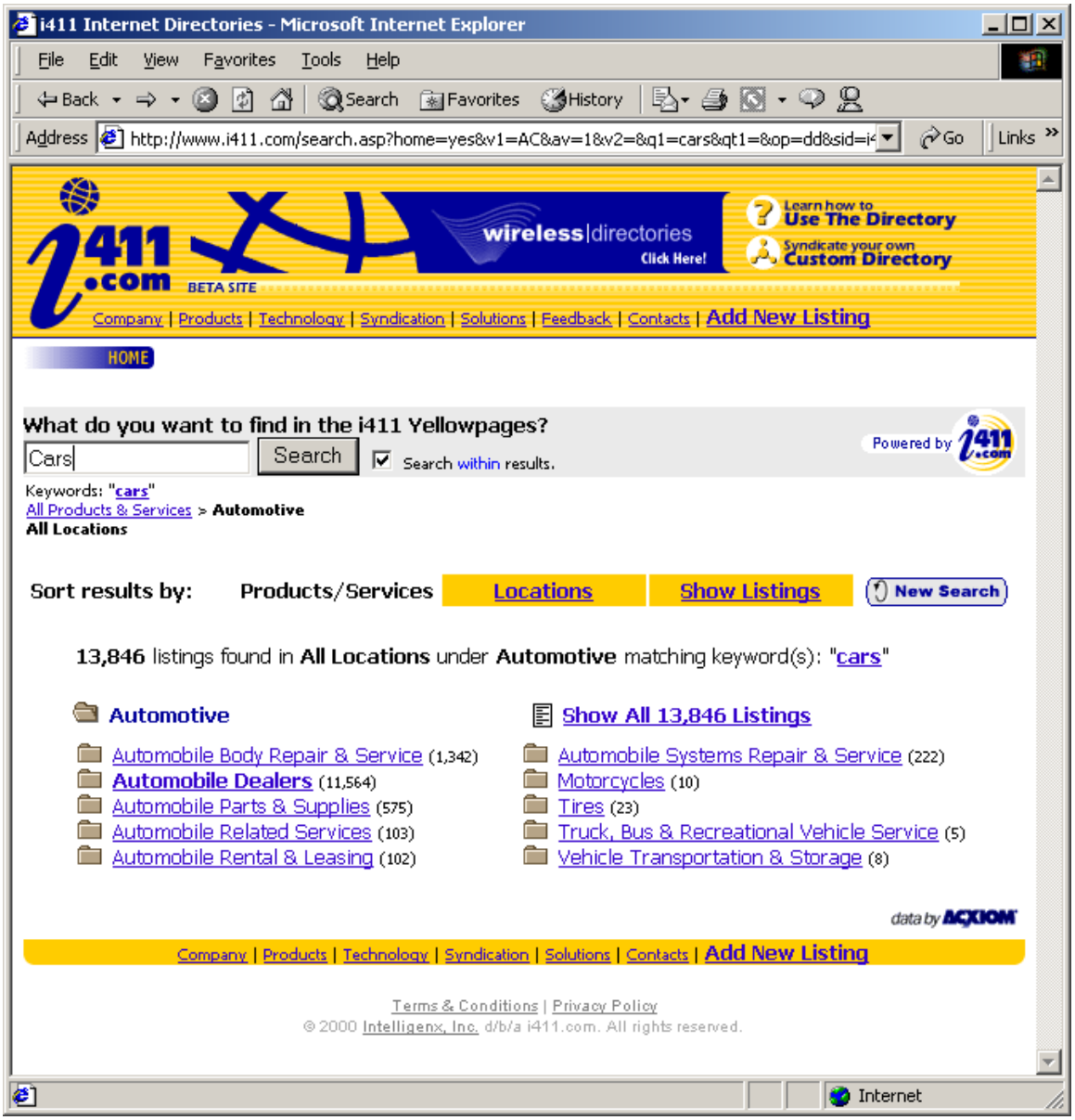


Figure 8.3: The i411.com web-site contains a keyword search mechanism that is concatenated with a list of intermediate search result categories. The number of hits is shown for each of these categories. Users can select a certain category in this window that will confine the result set to a single category.

8.3 Summary

Generalized query previews forms a user interface architecture for efficient browsing of large online data. It supplies data distribution information to the users. It also gives continuous immediate feedback about the size of the result set as the query is being formed. Generalized query previews works on metadata that is generally much smaller than the raw data. As users make informed queries by seeing this metadata, they can easily avoid submitting zero-hit and mega-hit queries over a network. Users can find and see what they want in the data and how it is organized. Hence, the problem of blind formation of queries is solved in an efficient and simple manner.

Field experience and controlled experimentation also supports this claim that generalized query previews form a promising architecture for efficient browsing of large online data. Except for the cases of known-item searches the idea has merits in investigative querying. In all cases, users are happy to see the overview of the data.

Future implementations and the research of visual data mining and information visualization for browsing large online data will hopefully benefit from this work. Software engineers can use the generalized query previews architecture to improve their tools. Researchers can work on the suggested future work to enhance the underlying ideas.

REFERENCES

[AS94] Ahlberg, C. and Shneiderman, B., "Visual Information Seeking: Tight Coupling of Dynamic Query Filters with Starfield Displays", *Proceedings of the ACM CHI'94 Conference,* pp. 313-317, (1994).

[AWS92] Ahlberg, C., Williamson, C., and Shneiderman, B., "Dynamic Queries for Information Exploration: An Implementation and Evaluation", *Proceedings of the ACM CHI'92 Conference*, pp. 619-626, (1992).

[AW95] Ahlberg, C. and Wistrand, E., "IVEE: An Information Visualization and Exploration Environment", *Proceedings of the IEEE Information Visualization Symposium*, pp. 66-73, (1995).

[ACS96] Aiken, A., Chen, J., Stonebraker, M., and Woodruff, A., "Tioga-2 (DataSplash): A Direct Manipulation Database Visualization Environment", *Proceedings of the 12th International Conference on Data Engineering*, pp. 208-217, (1996).

[AEP96] Antis, J., Eick, S. G., and Pyrce, J., "Visualizing the Structure of Relational Databases", *IEEE Software*, 13(1), pp. 72-79, (1996).

[Bec97] Becker, B. G., "Volume Rendering for Relational Data", *Proceedings of the IEEE Information Visualization Symposiu*m, pp. 87-90, (1997).

[BEW95] Becker, R. A., Eick , S. G., and Wilks, A. R., "Visualizing Network Data", *IEEE Transactions on Visualization and Computer Graphics*, 1(1), pp. 16-28, (March 1995).

[BT98] Beigel, R. and Tanin, E., "The Geometry of Browsing", *Proceedings of the Third Latin American Conference on Theoretical Informatics*, pp. 331-340, (1998).

[BO94] Berkin, A. L. and Orton, M. N., "LinkWinds: Interactive Scientific Data Analysis and Visualization", *Communications of the ACM*, 37(4), pp. 42-52, (1994).

[Ber83] Bertin, J., "Semiology of Graphics", *University of Wisconsin Press*, Madison, Wisconsin, (1983).

[BDH94] Bowman, C. M., Danzig, P. B., Hardy, D. R., Manber, U., and Schwartz, M. F., "The Harvest Information Discovery and Access System", *Proceedings of the Second International Conference on the World Wide Web*, pp. 763-771, (1994).

[BCS96] Buja, A., Cook, D., and Swayne, D., "Interactive High-Dimensional Data Visualization", *Journal of Computational and Graphical Statistics*, 5(1), pp. 78-99, (1996).

[BS96] Byard J. and Schneider, D., "The Ins and Outs (and everything in between) of Data Warehousing", *Proceedings of the ACM SIGMOD'96*, (1996).

[CMP95] Capobianco, F., Mosconi, M., and Pagnin, L., "Progressive HTTP-Based Querying of Remote Databases within the Marmotta Iconic VQS", *Proceedings of the IEEE Information Visualization Symposium*, pp. 122-125, (1995).

[CMS99] Card, S., Mackinlay, J. D., Shneiderman, B., "Readings in Information Visualization Using Vision to Think", *Morgan Kaufmann*, San Francisco (1999).

[CRM91] Card, S., Robertson, G., and Mackinlay, J. D., "The Information Visualizer, An Information Workspace Information Visualization", *Proceedings of the ACM CHI'91 Conference*, pp. 181-188, (1991).

[CCL95] Catarci, T., Costabile, M. F., Levialdi, S., and Batini, C., "Visual Query Systems for Databases: A Survey", *Technical Report*, SI/RR-95/17, Dipartimento di Scienze dell'Informazione, Universita' di Roma La Sapienza, (1995).

[CD97] Chaudhuri, S. and Dayal, U., "An Overview of Data Warehousing and OLAP Technology", *ACM SIGMOD Record*, 26(1), (1997).

[Chi00] Chi, E. H., "A Taxonomy of Visualization Techniques using the Data State Reference Model", *Proceedings of the IEEE Information Visualization Symposium*, pp. 69-75, (2000).

[CRM95] Chuah, M. C., Roth, S. F., Mattis, J., and Kolojejchick, J., "SDM: Selective Dynamic Manipulation of Visualizations", *Proceedings of the ACM UIST'95 Conference*, pp. 61-70, (1995).

[DPS96] Doan, K., Plaisant, C., and Shneiderman, B., "Query Previews in Networked Information Systems", *Proceedings of the Forum on Advances in Digital Libraries*, IEEE Society Press, pp. 120-129, (1996).

[DPS97] Doan, K., Plaisant, C., Shneiderman, B., and Bruns, T., "Query Previews in Networked Information Systems: A Case Study with NASA Environmental Data", *ACM SIGMOD Record*, 26(1), pp. 75-81, (1997).

[ERG89] Egan, D. E., Remde, J. R., Gomez, L. M., Landauer, T. K., Eberhardt, J., and Lochbaum, C. C., "Formative Design Evaluation of Superbook", *ACM Transactions on Information Systems*, 7(1), pp. 30-57, (1989).

[Eic94] Eick, S. G., "Data Visualization Sliders", *Proceedings of the ACM UIST'94 Conference*, pp. 119-120, (1994).

[ESS92] Eick, S. G., Steffen, J. L., and Sumner, E. E., "SeeSoft - A Tool For Visualizing Line Oriented Software Statistics", *IEEE Transactions on Software Engineerin*g, 18(11), pp. 957-968, (1992).

[Eic93] Eick, S. G. and Willis, G. J., "Navigating Large Networks with Hierarchies", *Proceedings of the IEEE Visualization'93 Conference*, San Jose, California, pp. 204-210, (1993).

[Fei88] Feiner, S., "Seeing the Forest for the Trees: Hierarchical Displays of Hypertext Structures", *Proceedings of the ACM Office Information Systems Conferenc*e, pp. 205-212, (1988).

[FB90] Feiner, S. and Beshers, C., "Worlds within Worlds: Metaphors for Exploring n-Dimensional Virtual Worlds", *Proceedings of the ACM UIST'90 Conference*, pp. 76-83, (1990).

[GE97] Gershon, N. and Eick, S. G., "Information Visualization", *IEEE Computer Graphics and Applications*, (August 1997).

[GR94] Goldstein, J. and Roth, S. F., "Using Aggregation and Dynamic Queries for Exploring Large Data Sets", *Proceedings of the ACM CHI'94 Conference*, pp. 23-29, (1994).

[Gra97] Gray, J., "Data Cube: A Relational Aggregation Operator Generalizing Group-by, Cross-Tab, and Sub-Totals", *Data Mining and Knowledge Discovery Journal*, 1(1), pp. 29-53, (1997).

[GTP99] Greene, S., Tanin, E., Plaisant, C., Shneiderman, B., Olsen, L., Major, G., and Johns, S., "The End of Zero-Hit Queries: Query Previews for NASA's Global Change Master Directory", *International Journal of Digital Libraries*, 2(2), pp. 79-90, (1999).

[Heat95] Heath, L., "Envision: A User Centered Database of the Computer Science Literature", *Communications of the ACM*, pp. 52-53, (1995).

[Hear95] Hearst, M., “Tilebars: Visualization of Term Distribution Information in Full Text Information Access”, *Proceedings of the ACM CHI’95 Conference*, pp. 59-66, (1995).

[Hear99] Hearst, M., “User Interfaces and Visualization”, *Modern Information Retrieval*, ACM Press, Ricardo Baeza-Yates and Berthier Ribeiro-Neto, pp. 257-323, (1999).

[HES85] Heppe, D., Edmondson, W. S., and Spence, R., “Helping both the Novice and Advanced User in Menu-driven Information Retrieval Systems”, *Proceedings of the HCI ’85 Conference*, British Computer Society Press, pp. 92-101, (1985).

[ID89] Inselberg, A. and Dimsdale, B., “Visualizing Multi-Variate Relations with Parallel Coordinates”, *Proceedings of the Third International Conference on Human-Computer Interaction*, pp. 460-467, (1989).

[JS91] Johnson, B. and Shneiderman, B., “Tree-maps: A Space-filling Approach to the Visualization of Hierarchical Information Structures”, *Proceedings of the IEEE Visualization’91 Conference*, pp. 284-291, (1991).

[KS98] Kandogan, E. and Shneiderman, B., “Elastic Windows: Design, Implementation, and Evaluation of Multi-Window Operations”, *Software Practice & Experience*, 28(3), pp. 225-248, (1998).

[Kau91] Kaufman, A., "Introduction to Volume Visualization", *Volume Visualization*, IEEE Computer Society Press, Los Alamitos, California, pp. 1-18, (1991).

[KK94] Keim, D. A. and Kriegel, H. P., “VisDB: Database Explorations Using Multidimensional Visualization”, *IEEE Computer Graphics and Application*s, pp. 40-49, (1994).

[KPS97] Kumar, H., Plaisant, C., and Shneiderman, B., “Browsing Hierarchical Data with Multi-Level Dynamic Queries and Pruning”, *International Journal of Human Computer Studies*, 46, pp. 103-124, (1997).

[LR96] Lamping, J. and Rao, R., “The Hyperbolic Browser: A Focus + Context Technique for Visualizing Large Hierarchies”, *Journal of Visual Languages and Computing*, 7(1), pp. 33-55, (1996).

[LRB97] Livny, M., Ramakrishnan, R., Beyer, K., Chen, G., Donjerkovic, D., Lawande, S., Myllymaki, J., and Wenger, K., “DEVise: Integrated Querying and Visual Exploration of Large Datasets”, *Proceedings of the ACM SIGMOD’97*, (1997).

[MRC95] Mackinlay, J. D., Rao, R., and Card, S. K., "An Organic User Interface for Searching Citation Links", *Proceedings of the ACM CHI'95 Conference*, pp. 67-73, (1995).

[NPM97] Nation, D. A., Plaisant, C., Marchionini, G. and Komlodi, A., "Visualizing Websites Using a Hierarchical Table of Contents Browser: WebTOC", *Proceedings of the Third Conference on the Human Factors and the Web*, http://www.uswest.com/webconference/proceedings/nation.html, (1997).

[NSP96] North, C., Shneiderman, B., and Plaisant, C., "User Controlled Overviews of an Image Library: A Case Study of the Visible Human", *Proceedings of the First ACM International Conference on Digital Libraries*, pp. 74-82, (1996).

[PBD99] Plaisant, C., Bruns, T., Doan, K., and Shneiderman, B., "Interface and Data Architecture for Query Previews in Networked Information Systems", *ACM Transactions on Information Systems*, 17(3), pp. 320-341, (1999).

[PMR96] Plaisant, C., Milash, B., Rose, A., Widoff, S., and Shneiderman, B. "LifeLines: Visualizing Personal Histories", *Proceedings of the ACM CHI'96 Conference*, pp. 221-227, (1996).

[RC94] Rao, R. and Card, S., "The Table Lens: Merging Graphical and Symbolic Representations in an Interactive Focus+Context Visualization for Tabular Information", *Proceedings of the ACM CHI'94 Conference*, pp. 318-322, (1994).

[Rei91] Reisner, P., "Human Factors Studies of Database Query Languages: A Survey and Assessment", *ACM Computing Surveys*, 13(1), (1991).

[RMC91] Robertson, G., Mackinlay, J. D., and Card, S., "Cone Trees: Animated 3D Visualizations of Hierarchical Information", *Proceedings of the ACM CHI'91 Conference*, pp. 189-194, (1991).

[RM93] Robertson, G. and Mackinlay, J. D., "The Document Lens", *Proceedings of the ACM UIST'93 Conference*, pp. 101-108, (1993).

[Rot00] Roth, S.F., "The SAGE Project", *http://www.cs.cmu.edu/Web/Groups/sage/sage.html*

[RLS96] Roth, S. F., Lucas, P., Senn, J. A., Gomberg, C. C., Burks, M. B., Stroffolino, P. J., Kolojejchick, J. A., and Dunmire, C., "Visage: A User Interface Environment for Exploring Information", *Proceedings of the IEEE Information Visualization Symposium*, pp. 3-12, (1996).

[Rou97] Roussopoulos, N., Kotidis, Y., and Roussopoulos, M., "Cubetree: Organization of and Bulk Incremental Updates on the Data Cube", *Proceedings of the ACM SIGMOD'97*, pp. 89-100, (1997).

[Shn94] Shneiderman, B., "Dynamic Queries for Visual Information Seeking", *IEEE Software*, 11(6), pp. 70-77, (1994).

[Shn96] Shneiderman, B., "The Eyes Have It: A Task by Data Type Taxonomy of Information Visualizations", *Proceedings of the IEEE Symposium on Visual Languages'96*, pp. 336-343, (1996).

[Shn98] Shneiderman, B., "Designing the User Interface: Strategies for Effective Human-Computer Interaction", *Addison Wesley*, (1998).

[SBC97] Shneiderman, B., Byrd, D., and Croft, W.B., "Clarifying Search: A User-Interface Framework for Text Searches", *D-Lib Magazine*, http://www.dlib.org/dlib/january97/ retrieval, (January 1997).

[SFR00] Shneiderman, B., Feldman, D., Rose, A., and Ferre Grau, X., "Visualizing Digital Library Search Results with Categorical and Hierarchical Axes", *Proceedings of the 5th ACM International Conference on Digital Libraries*, pp. 57-66, (2000).

[Spo93] Spoerri, A., "InfoCrystal: A Visual Tool for Information Retrieval", *Proceedings of the IEEE Visualization'93 Conference*, pp. 150-157, (1993).

[SH00] Stolte, C. and Hanrahan, P., "Polaris: A System for Query, Analysis, and Visualization of Multi-Dimensional Relational Databases", *Proceedings of the IEEE Information Visualization Symposium*, (2000).

[SFB94] Stone, M. C., Fishkin, K., and Bier, E. A., "The Movable Filter as a User Interface Tool", *Proceedings of the ACM CHI'94 Conference*, pp. 306-312, (1994).

[SCB98] Swayne, D. F., Cook, D., and Buja, A., "XGobi: Interactive Dynamic Visualization in the X Window System*", Journal of Computational and Graphical Statistics*, 7, pp. 113-130, (1998).

[TBS97] Tanin, E., Beigel, R., and Shneiderman, B., "Design and Evaluation of Incremental Data Structures and Algorithms for Dynamic Query Interfaces", *Proceedings of the IEEE Information Visualization Symposium,* pp. 81-86, (1997).

[TLH00] Tanin, E., Lotem, A., Haddadin, I., Shneiderman, B., Plaisant, C., and Slaughter, L., "Facilitating Data Exploration with Query Previews: A Study of User Performance and Preference", *Behaviour & Information Technology*, 19(6), pp. 393-403, (2000).

[TPS00] Tanin, E., Plaisant, C., and Shneiderman, B., "Browsing Large Online Databases with Query Previews", *Proceedings of the Symposium on New Paradigms in Information Visualization and Manipulation (CIKM)*, available from ACM, New York, (2000).

[Tuf83] Tufte, E., "The Visual Display of Quantitative Information", *Graphics Press*, Cheshire, Connecticut, (1983).

[Tuf90] Tufte, E., "Envisioning Information", *Graphics Press*, Cheshire, Connecticut (1990).

[Tuf97] Tufte, E., "Visual Explanations: Images and Quantities, Evidence and Narrative", *Graphics Press*, Cheshire, Connecticut, (1997).

[TSW94] Tweedie, L., Spence, R., Williams, D., and Bhogal, R., "The Attribute Explorer", *Video Proceedings of ACM CHI'94 Conference*, pp.435-436, (1994).

[TSD96] Tweedie, L., Spence, R., Dawkes, H., and Su, H., "Externalising Abstract Mathematical Models", *Proceedings of the ACM CHI'96 Conference*, pp. 406-412, (1996).

[VN95] Veerasamy, A. and Navathe, S., "Querying, Navigating and Visualizing a Digital Library Catalog", *Proceedings of the Second International Conference on the Theory and Practice of Digital Libraries*, http://www.csdl.tamu.edu/DL95, (1995).

[Wil84] Williams, M. D., "What Makes RABBIT Run", *International Journal of Man-Machine Studies*, 21(4), pp. 333-352, (1984).

[WS92] Williamson, C. and Shneiderman, B., "The Dynamic Home Finder: Evaluating Dynamic Queries in a Real-Estate Information Exploration System", *Proceedings of the ACM SIGIR'92 Conference*, pp. 338-346, (1992).