

# TECHNICAL RESEARCH REPORT

Browsing Large Online Data Tables Using Generalized Query  
Previews (2001)

*by Egemen Tanin, Ben Shneiderman*

**TR 2005-18**



*ISR develops, applies and teaches advanced methodologies of design and analysis to solve complex, hierarchical, heterogeneous and dynamic problems of engineering technology and systems for industry and government.*

*ISR is a permanent institute of the University of Maryland, within the Glenn L. Martin Institute of Technology/A. James Clark School of Engineering. It is a National Science Foundation Engineering Research Center.*

**Web site <http://www.isr.umd.edu>**

# Browsing Large Online Data Tables Using Generalized Query Previews

Egemen Tanin      Ben Shneiderman\*  
{egemen, ben}@cs.umd.edu

Draft: February 5, 2002

Department of Computer Science,  
\*Human-Computer Interaction Laboratory,  
Institute for Advanced Computer Studies, and \*Institute for Systems Research  
University of Maryland at College Park

## Abstract

Companies, government agencies, and other organizations are making their data available to the world over the Internet. They often use large online relational tables for this purpose. Users query such tables with front-ends that typically use menus or form fillin interfaces, but these interfaces rarely give users information about the contents and distribution of the data. Such a situation leads users to waste time and network resources posing queries that have zero-hit or mega-hit results. Generalized query previews enable efficient browsing of large online databases by supplying data distribution information to the users. The data distribution information provides continuous feedback about the size of the result set as the query is being formed. Our paper presents a user interface architecture and discusses recent experimental findings. Our prototype system, ExpO, provides a flexible user interface for research and testing. The user study shows that for exploratory querying tasks, generalized query previews speed user performance and reduce network load.

**Categories and Subject Descriptors:** H.5.2 [Information Interfaces and Presentation]: User Interfaces - *Graphical user interfaces (GUI)*, H.3.5 [Information Storage and Retrieval]: Online Information Services - *Web-based services*.

**General Terms:** Design, Human Factors, Experimentation.

**Other Keywords:** Visual Data Mining, Information Visualization, Graphical User Interfaces, Relational Databases, Database Querying, Internet, World-Wide Web, Online Databases.

## 1. Problem

Companies, government agencies, and other organizations are making their data available to the public over the Internet. The United States Census Bureau hosts vast collections of economic, geographic, and demographic data (sample data retrieval system available at [ferret.bls.census.gov/cgi-bin/ferret](http://ferret.bls.census.gov/cgi-bin/ferret)) and the National Aeronautics and Space Association (NASA) has still larger collections of scientific and environmental data (e.g., [eos.nasa.gov/eosdis](http://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/](http://www.who.int/whosis/)). These are only a few of the organizations that are making vast data resources available over the Internet.

The user interfaces that serve as the database front-ends mostly utilize menus or form fillin interfaces [SBC97]. Generally, these user interfaces are activated within a browser. 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 in using computers.

Many of the database front-ends, instead of giving users information about the contents of the data, require users to fill lengthy electronic forms. The designers of such interfaces generally assume that the users are informed about the data and that they want to submit known-item queries rather than probing the database. However, unguided novice users frequently waste their time submitting queries that have zero-hit or mega-hit result sets. Users of online databases generally do not have the time or the willingness to learn a complex querying mechanism or the patience to fill in a lengthy form. Finally, users of online databases often have to access large amounts of data using a low bandwidth congested network.

The screenshot shows a Netscape browser window with the address bar displaying <http://ferret.bls.census.gov/>. The page title is "Select Variable Values". Below the title, there is a link to "FAQ Contents Help". The main content area is titled "Select Variable Values" and contains the following text: "You requested the following variables. Now select the values for these variables ([Link](#))".

There are three variable selection sections:

- Variable 1:** "# hours worked at home (not employed) (WHHRR)". It includes a dropdown menu with options: "ALL VALUES", "WHHRR = 9 (Not Reported)", and "WHHRR = 8 (Refused)". Below the dropdown, there are input fields for a range: " <= WHHRR <= ". A note states: "Values must be between 1 and 168".
- Variable 2:** "# hours worked at home (wage & salary) (WHHRRW)". It includes a dropdown menu with options: "ALL VALUES", "WHHRRW = 9 (Not Reported)", and "WHHRRW = 8 (Refused)". Below the dropdown, there are input fields for a range: " <= WHHRRW <= ". A note states: "Values must be between 0 and 168".
- Variable 3:** "# weeks with nothing reported/refused (WALLX)". It includes a dropdown menu with options: "ALL VALUES", "WALLX = 9 (Not Reported)", and "WALLX = 8 (Refused)". Below the dropdown, there are input fields for a range: " <= WALLX <= ". A note states: "Values must be between 1 and 99".

At the bottom of the form, there is a section for "# weeks since occupied as permanent home (HOPZWA)".

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

Figure 1.1 shows a form fillin interface that can be generated from the interactive United States Census Bureau homepages (i.e., [ferret.bls.census.gov/cgi-bin/ferret](http://ferret.bls.census.gov/cgi-bin/ferret)). This interface is a good example of the database front-ends that are currently available over the Internet. This lengthy form fillin interface has some guidance about what values can be selected on some of the fields, but other information such as the data distribution are not available. In this interface, users can easily generate queries that will return zero-hit or mega-hit result sets. Novice users have to probe the database until they find what they want or get tired of using this interface. A more effective, simple, and easy to learn approach for defining queries is needed for public online databases.

Section 2 of this paper introduces the related research. Section 3 describes our early work on the problem. Section 4 gives results for a previous user study performed on our early work. Section 5 describes our recent work on the problem. Section 6 reports results of our recent user study and Section 7 discusses the implementation issues. Sections 8 and 9 state our conclusions and possible future work.

## 2. Related Work

Sources of inspiration for query previews include information visualization and database systems. 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 similar visual explanations [Ber83] [Tuf83] [Tuf90] [Tuf97]. Examples of abstract information spaces are stock market performance data, bibliographic databases, organizational budgets, patient records from a medical database, web logs, etc.

Users continue to demand more powerful methods to visually mine and manipulate their data. Searching and browsing methods are becoming more challenging to develop and understand with the increasing data set sizes and diversity of access methods. Generalized query previews address many of these new features of contemporary data sets and support exploration of large online data tables stored in relational databases.

Heppe, Edmondson, and Spence [HES85] created one of the first menu-driven information retrieval systems. They used volume previews and progressive query formulation to help the users. Spence continued this line of work with his Attribute and Influence Explorers [TWS94] [TSD96] that used interactive histograms. The tools give an overview of the contents of the data and reduce the visual complexity by summarizing the distributions in a compact way. The selections on a histogram immediately update the other histograms. RABBIT is one of the first systems where progressive querying was used [Wil84] for helping users understand the complex querying process. In RABBIT, users form queries incrementally by showing the system what needs to be changed in the original query. These systems focused on making users understand and organize their sessions looking at the results of their previous actions and queries. More recently, the idea of parallel bargrams have become popular in online e-commerce applications (i.e., product selection) [WLH01]. The idea behind a bargram is to convert a histogram representation of an attribute into a stack of bars that looks like a compound single bar. Our work parallels these approaches in use of the distribution information to present the overview of the data. We extend the work to previewing data in large online databases.

Table Lens [RC94] is a multi-dimensional data visualization tool that uses a focus + context technique on tabular data. It is based on a fisheye view of a spreadsheet. In this technique a group of rows and columns are in focus and the rest of the spreadsheet is used to give an overview of a potentially very large table. These portions of the spreadsheet give an overview of the data with the help of histograms and color-coding. More recently, InfoZoom [BS99] is a tool that also provides an overview of tabular data sets. 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 stored as charts or as various interactive objects. Multiple columns with similar values can be collapsed to make room for and obtain an overview of a larger portion of the data. Inselberg and Dimsdale took a different approach to visualizing multi-dimensional tabular data [ID89]. Instead of viewing data using commonly used perpendicular multiple coordinate axes, they used a series



of parallel axes to display the data attributes. Rows from a data table became lines crossing over these axes. In contrast to our approach these techniques rely more closely on individual data rows, columns, and individual items of tabular data, making them less applicable to large online data.

A comprehensive approach, SDM [CRM95], gives a set of interactive techniques for 2D and 3D visualizations. Visualizations in SDM are linked. Real-time interactive animation techniques are used. In SDM, it is possible to use multi-dimensional visualizations to see a data set on and navigate over a landscape of color-coded bars. These bars can be linked together and indicate a certain relationship. Visage [RLS96] aims to coordinate the exploration of information across different types of visual aids. For example, users can drag and drop the spreadsheet view of some geographical data onto a map to display its distribution over the map. These systems focused on having more interchangeable, comprehensive, and interactive techniques than many others. Yet, their focus is less on overviews, previews, and pruning data.

Visualizing large amounts of abstract data has always been a challenging area of research. DEVise is an exploration system where users can visualize large data sets [LRB97]. Views are linked to each other and updates occur simultaneously. VisDB [KK94] used a single pixel of the screen to represent a single record from a database. It colors and organizes the database records with respect to the query result relevancies. Organizational features, i.e., spirals, also help VisDB show million record data sets. Volume rendering is a common method for displaying large scientific data (e.g., computer aided tomography images of a human brain). Becker applied this method to abstract information spaces [Bec97]. He rendered volumetric abstract data stored in a relational database. This let him show large abstract data sets on 3D spaces interactively, mostly in the form of 3D density clouds on large data sets. Goldstein and Roth [GR94] used aggregation with dynamic query definition methods to help users with large data sets. They used a mechanism called the Aggregate Manipulator and combined it with various dynamic query definition capabilities. The distribution of data is also visible to the users in this system. These systems share our vision of making more data available to the user in a more compact fashion. They also did not extend to large online databases due to their close links to individual data items.

The second source of related work is database-oriented research. For example, Tioga-2 (formerly DataSplash) is a database visualization environment [ACS96] that provides users with a variety of display objects to be used on canvases to explore the underlying complex database. It provides various browsing capabilities, but no systematic way to see previews. SeeData, a system for displaying the relational schema of a database [AEP96], uses 2D bar charts to show relationships between thousands of relations. Polaris [SH00] 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. Marmotta, a querying system for networked databases [CMP95], uses progressive querying. The main idea behind Marmotta is the formulation of queries with simple icons that represent actions and items. Users manipulate these graphical items instead of typing complex queries using a difficult querying language, but there is no notion of summarizing database distributions. Distribution data is available in HIBROWSE, a user interface that helps users interact with a database through a view-based searching mechanism, but it is not tied to the query process [Pol97]. More recent applications such as the Eureka database exploration system helps users continuously browse an Internet-based database [SA00]. It integrates querying with result browsing. It allows users to make explorations based on example records and has dynamic query definition capabilities. E-commerce sites form a good application area for this system.

### 3. NASA Experience and Early Work

Our early work on dynamic queries [WS92] [AS94] [AW95] [AWS92] used a direct manipulation approach to facilitate query formulation on multi-dimensional abstract data with a visual representation of query components and results. Dynamic queries allow rapid, incremental, and reversible control of the query. Results are presented visually and continuous feedback guides users in the query formulation process. The application of dynamic queries to large online data is attractive to the users of the Internet. Unfortunately, high system-resource demands make dynamic queries less applicable to large online data collections. Dynamic queries require immediate access to individual data items so that continuous feedback is given to the users. Yet, large online data cannot be immediately and continuously accessed.

In 1995, Human-Computer Interaction Laboratory of the 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. Using overviews and previews, researchers made dynamic queries applicable to large online data and allowed users to efficiently prune irrelevant data. This NASA project led to the idea of query previews.

Query previews give an overview of the data and a preview of the queries before the final queries are sent through the network. Query previews work only on a 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 storage of the client computer from the network. We applied the principles of this two-phase querying strategy (i.e. previews first and then refinements) for NASA's Earth Observing System Data Information System [DPS96] [DPS97] [PBD99]. This strategy was created as an experimental interface [GTP99] for the Global Change Master Directory and was the basis for the Global Land Cover Facility interfaces, all part of NASA. This paper carries query previews to a more general level with methods that help users work on any of the data attributes.

Query previews show a few discriminating (i.e., capable of selecting a few from a large set) attributes of the data so that the user selections on these attributes would immediately 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 provide aggregate information about the data. Distribution of data over different attribute values is shown graphically with 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 results before refining the queries, they are less likely to submit queries that return zero or mega hits. The system load is reduced since users do not waste their time with zero-hit queries or consume network and server resources in downloading and processing many useless results. While dynamic queries require attribute values of every record of the data to be downloaded, query previews only need 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. This is also a scalable approach, as aggregate data size remains fixed even as the number of records in a database table grows. Figure 3.1 shows a query preview interface formed using the three most commonly used attributes of the Global Change Master Directory (topic,

year, and area). The distribution of data over these attributes is shown with bar charts and the result set size is displayed as a separate bar at the bottom.

The query preview phase can be followed by a refinement phase, which can be implemented as a dynamic query interface, to further formulate/refine the query. After the refinement phase, when a desired final result set is obtained, the results can be visualized as a simple table or a complex visualization. The results can also be of different data types, i.e., images, values, etc. Other interfaces for the refinement phase are also possible (e.g., form fillin, menus).

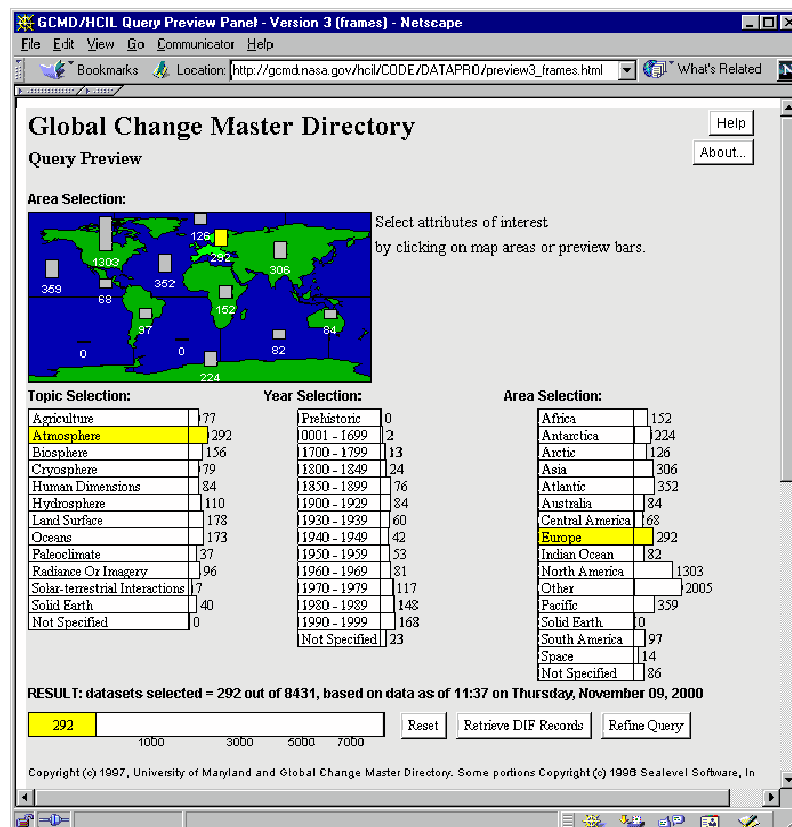


Figure 3.1: An example query preview interface. Topic, Year, and Area are the discriminating attributes for the 8431 data items. Here, the bars show the overview of the data distribution. When users select the attribute values (e.g., atmosphere for topics and Europe for area), the bars are updated immediately 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, and the query can be refined with additional attributes in the second phase. In this case, atmospheric data for Europe produces a set of 292 data items to be retrieved.

#### 4. Previous Study: Query Previews

Since query previews add another phase to query formulation, there is a possibility that user performance deteriorates and that users are frustrated by a two-phase approach. Moreover, query previews focus attention on only a few selected attributes that may not be useful in some of the 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. We identified the task types that would put

query previews into their best and worst light so that we could quantify their maximum benefits and drawbacks [TLH00].

Clearly specified tasks have a straightforward and an accurate definition (known-item search), e.g. "List all the State of Maryland employees from the employee database". Query previews should have no benefits for these tasks. In this case, users want a complete list regardless of the outcome of the query. 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 refinement stage e.g., the form fillin interface (tasks of this worst case scenario are called T1 tasks).

Unclearly specified tasks usually require a series of query submissions. User's constraints and preferences cannot be stated immediately. Information gained from the query previews will influence the user's choices, so query previews could be very useful. However, the relevance of the attributes used in the query preview will impact the usefulness of the 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).

This first study used a within subject counter-balanced design with twelve subjects. 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 reverse order.

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.

The independent variable was the user interface type and the treatments were: (1) Form fillin interface with a query preview, (2) form fillin interface without a query preview. The dependent variables were the time to complete the tasks in each interface (excluding the setup times) and the subjective preferences of the users.

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. The interface with the preview was only 10% slower.

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 needed for the query and users were performing known-item searches. However, users of the interface with the query preview performed only slightly worse. The users spent only a few seconds in the query preview, identified that it is not relevant for this type of task and continued to the refinement phase.

For unclearly specified tasks with partial relevance of the query preview attributes, 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 and 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.

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 many queries 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 (even more than T2's).

The users preferred the interface with the query preview (average of 7.1 for the interface with the preview versus average of 5.2 for the interface without it on a scale of 1 to 9 at the significance level of 0.05). They stated that the query preview was helpful, enabling them to search faster, and learn more about the data. We believe that this result 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.

This study provoked a challenge as to whether we had any data demonstrating the reduced number of queries submitted per task. Our recent work replicates the result and addresses this concern, while extending the work to show the benefits of the generalized query previews.

## 5. Generalized Query Previews

After our early work, field experience, previous user study, and user observations, we defined a general user interface architecture for efficient browsing of large online data. The generalized query previews user interface architecture has the following three components (Figure 5.1):

- *schema component*: presents the data table and attribute names augmented with a mechanism for selecting a user-defined view of the data attributes,
- *distribution information component*: displays and helps users work on the data distribution information,
- *raw data component*: displays and/or used for analysis of the retrieved records.

We allowed users to see the database schema with table and attribute names. Then, they can define their view by choosing some of the data attributes. Later, using the attached distribution information, they can analyze the overview of the data and define their queries. Finally, they can fetch the mapping results, and if desired, forward it to another program for further analysis. This pattern can be repeated if needed.

To demonstrate and later to experiment with the generalized query previews user interface architecture, we implemented a prototype system, ExpO, and applied it to 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 entries. The data is stored in four different relations on a networked relational database. Each relation represents a single table. All the tables share a unique identifier, 'report\_id', representing a unique report from a county. The remainder of this section analyzes the components of ExpO's generalized query previews user interface architecture.

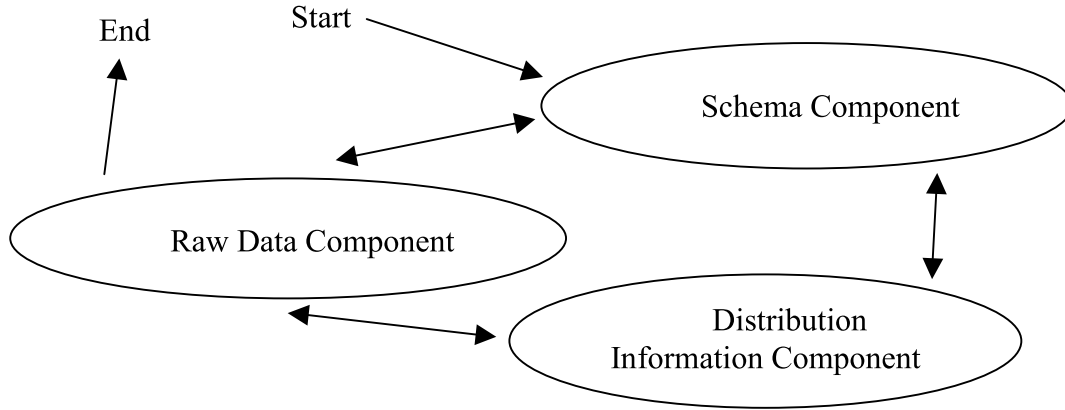


Figure 5.1: Interactions between the three generalized query previews components.

### 5.1. 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 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.

ExpO uses a hierarchical browser, the panel on the left in Figure 5.1.1, 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 need scalable approaches so that relations between database tables can be seen. 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 is important.

In generalized query previews, users can select attributes to form a view. The schema component plays the role of a selector for this purpose. In applications where there is no 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.

ExpO users can select the attributes that they want to define their queries on. This action triggers the insertion of that attribute to the user-defined view, which is presented in a separate panel and also by a hierarchical browser (Figure 5.1.2). The panel on the right 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.

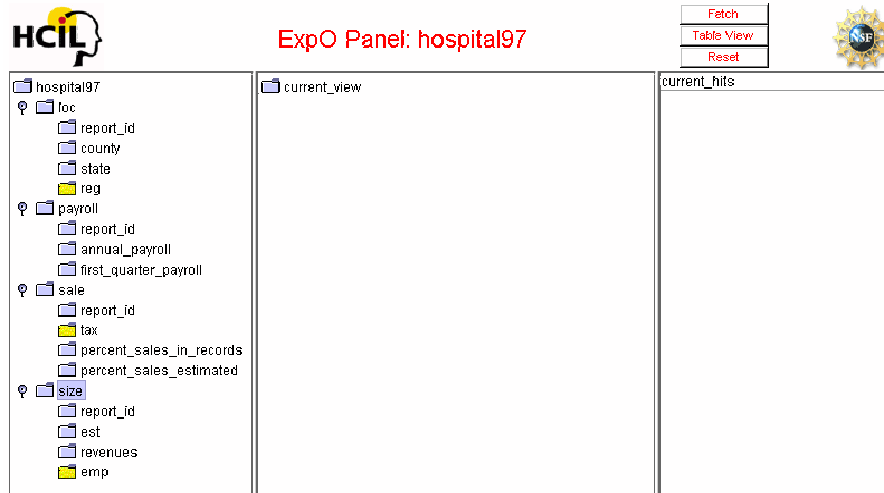


Figure 5.1.1: An example generalized query preview interface, ExpO, the left panel displays the table and attribute names of a relational database as an implementation of the schema component.

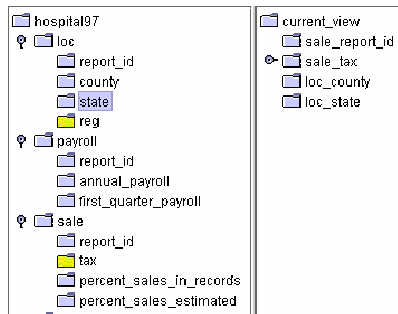


Figure 5.1.2: ExpO with a user-defined view after four attribute selections. The user-defined view is also a hierarchical browser.

## 5.2. Distribution Information Component

Distribution information plays an important role in the generalized query previews user interface architecture. The distribution information component shows an overview of the data and allows users to specify queries. User-defined views are used to attach the distribution information. In the example from Figure 5.1.2, a special icon in the user-defined 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 5.2.1).

The attributes are expandable into buckets and the data distribution information is attached to these buckets. Buckets are values over which the data can be aggregated. The data distribution information is attached to these buckets as some visual aids, such as the bar charts of this example. Here, 'taxable' (172 hits) and 'non\_taxable' (3080 hits) are the bucket names for the 'sale\_tax' attribute (Figure 5.2.2). Forming these buckets is the duty of the system designer, who

should gather information about the details of the data and the user needs. In some cases, e.g., gender of a person, the formation of the buckets may be trivial. Yet, in some other cases it may be quite difficult. Certain attributes, e.g., social-security number of a person, may not allow an easy formation at all. Although automatic generation of the buckets may be useful in some applications, a domain expert should work with the designer for many other applications, e.g., scientific classification of diseases, etc. In ExpO, the buckets are created by a batch program from the raw data and can be updated regularly if needed.

An important feature of the generalized query previews user interface architecture is the capability of visualizing a preview of the results. In ExpO, a separate bar on the top of the right panel shows the total number of distinct items mapping to a query. This is called the result bar (showing 3252 hits in Figure 5.2.1). Hence, users will be aware of the consequences of their query submissions, i.e. whether they are submitting mega-hit or zero-hit queries.

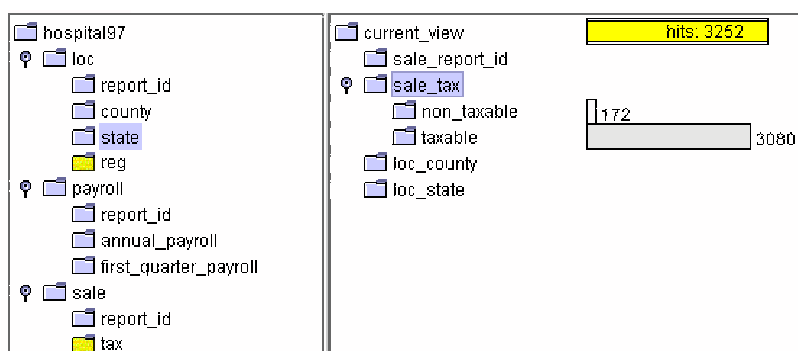


Figure 5.2.1: ExpO with the data distribution information attached to the user-defined view.

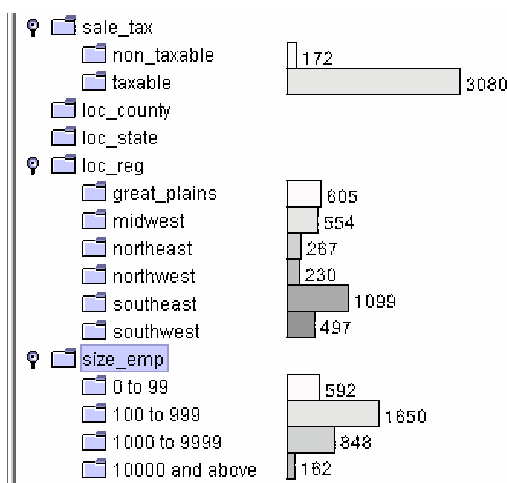


Figure 5.2.2: Information attached to the buckets of three attributes in the user-defined view.

In generalized query previews, queries are incrementally and visually formed by selecting items from a set of charts attached to the user-defined view. Users get continuous feedback on the data distribution information as they continue their selections (Figure 5.2.3). 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. They can work with these interactive charts as long as they want to explore the contents of the data, without incurring any network or server load. At this stage, only the number of hits but not the actual hits themselves are relevant.

We have experimented with data sets that contain hundreds of thousand of records (i.e. Environmental Protection Agency, Toxic Release Inventory with more 300,000 records) with the same efficiency as a data set of only a few hundred records. Clicking on the visual aids, bars at this stage, selects or deselects the items from the charts (Figure 5.2.4). Selections within a chart map to a disjunction operation. Selections between charts map to a conjunction operation. Other implementations are also possible. Yet, our implementation experiences show that the current ExpO implementation is one of the most intuitive implementations.

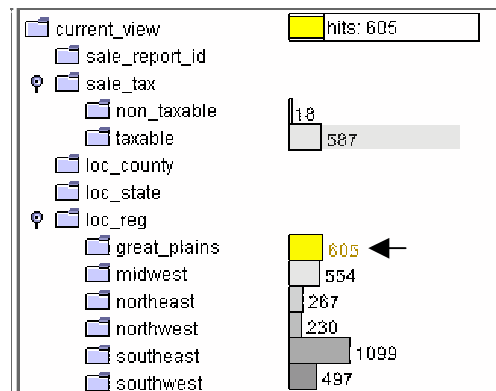


Figure 5.2.3: A selection is made on one of the buckets, 'great\_plains'.

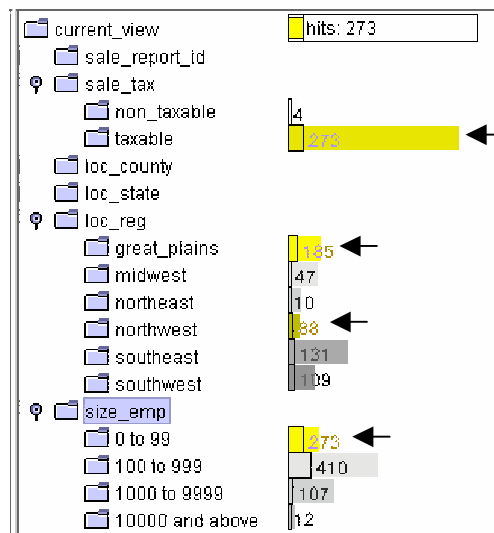


Figure 5.2.4: Multiple selections are made, 'taxable', 'great\_plains', 'northwest', and '0 to 99'.

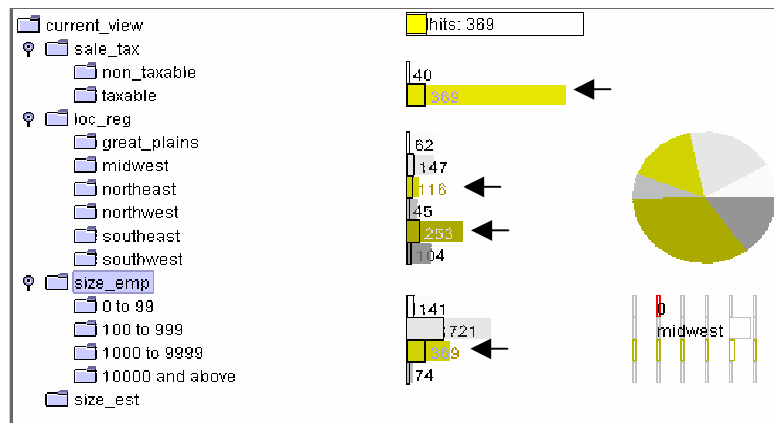


Figure 5.2.5: Other visual aids, such as a pie chart, may also be available to the users.

It is possible to display the distribution information on different types of visual aids. Figure 5.2.5 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 also be utilized.

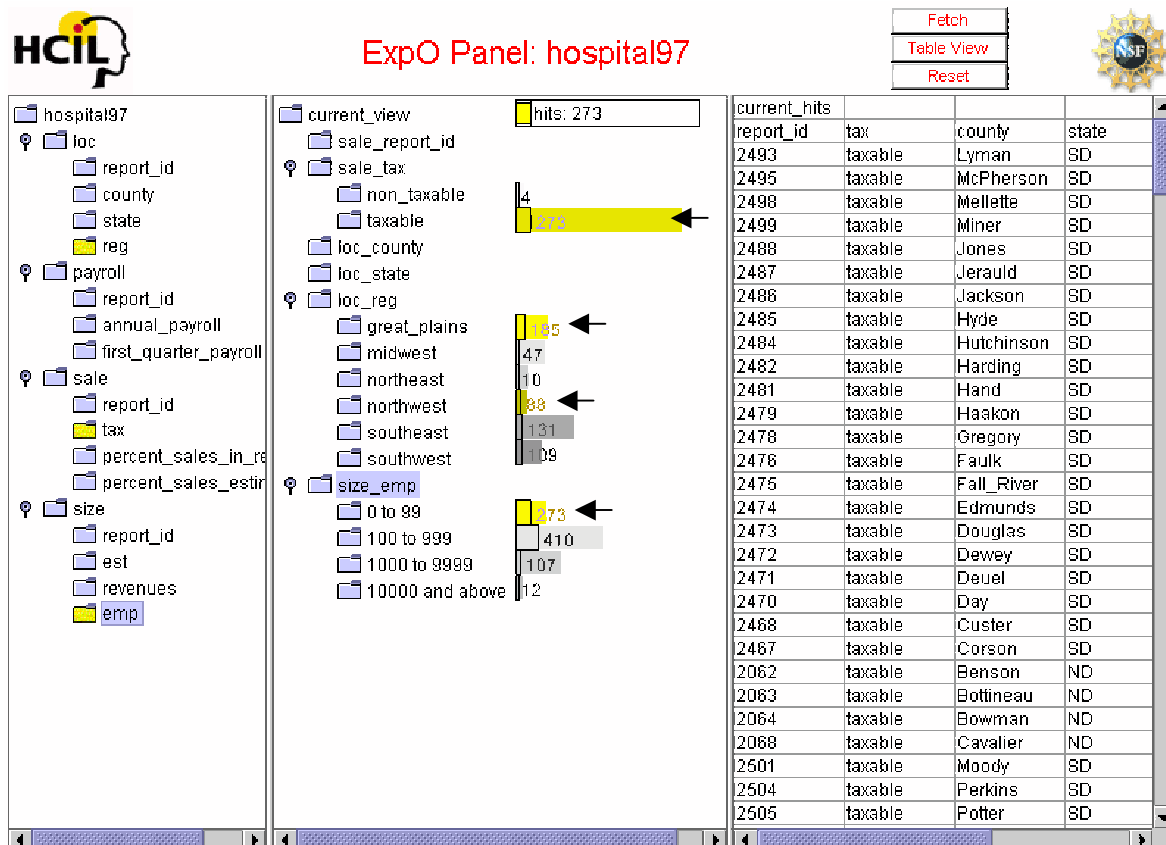


Figure 5.3.1: ExpO with results to a query displayed in a panel on the right, 273 hits are listed.

### **5.3 Raw Data Component**

After the investigative selections, users can fetch the desired portions of the data by sending their final selections over the network. They make informed queries, getting neither zero-hit nor mega-hit result sets is an issue. The benefit to users is in rapid exploration, and the benefit to system maintainers is the substantially reduced network and server load. For some implementations, the system designer can take drastic measures, such as preventing query submissions for zero-hit or mega-hit queries. Figure 5.3.1 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 (e.g., MS-Excel).

## **6. Current Study: Generalized Query Previews**

Generalized query previews are a step forward from the initial idea of query previews. Although they are more general, they also introduce some overheads such as defining a user-defined view and explicit expansions of charts. Hence, there is a possibility that user performance could be disturbed, and users may get confused and annoyed by the generalizations. Therefore, there was a need for a new user study to verify and quantify the benefits of generalized query previews and measure subjective user preferences.

### **6.1 Hypothesis**

In this user study, we identified the task types that would put generalized query previews into their best and worst situations, as was done in the previous user study. Clearly specified tasks and unclearly specified tasks were again used. However, for this study, only the full relevance of query attributes was an issue. Hence, there were only two task types. These two task types varied in terms of the clarity of the specifications they had.

Our hypotheses were: (1) For clearly specified tasks (T1') generalized query previews will lead to slower task performance, but with the same number 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 Independent and Dependent Variables**

The independent variable is the user interface type and the treatments are: (1) A form fillin interface (FFN') and (2) a generalized query preview interface (ExpO). The dependent variables are the time to complete the tasks in each interface (excluding the setup times), the number of query submissions by the users, and the subjective preferences of the users.

### **6.3 Subjects and Materials**

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. The user type is similar to the previous study. Eight subjects performed a set of tasks, once by using a generalized

query preview interface (i.e., ExpO) and once by using a form fillin interface. Another set of eight subjects worked in the reverse order.

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

The form fillin interface in Figure 6.1 was used to perform queries on a United States Census Bureau data set. The attributes in this sample data set were shown with a hierarchical browser. Attributes were selected by marking the toggles near them. This action also triggers the display of editable fields attached to these attributes. The output is a list of records matching the query.

The survey included six questions that determined the experience level of the subjects with computers. We also prepared a subjective preference questionnaire. This questionnaire included six questions to find out which of the two interfaces the subjects preferred.

county	reg	tax	emp
Benecio	midwest	non_taxa...	13227
Dana	midwest	non_taxa...	18037
Bookare	northwest	non_taxa...	14014
Plence	northwest	non_taxa...	17630

hits = 4

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

## 6.4 Tasks and Structure

The tasks given to the subjects were to find a list of the counties in the database satisfying certain constraints. In detail, the following two types of tasks were used:

- 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 this type of task, users can typically find the answer by submitting a single form fillin query. The ExpO user should have no specific advantage.
- T2': Vaguer 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".

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 combinations. Subjects received only introductory level training (not to exceed 10 minutes in total). They performed only two training tasks on each of the user interfaces.

## 6.5 Results

Figure 6.2 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. As hypothesized, 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. 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 6.3). The results show a statistically significant preference ( $t(15) = 6.37, p < 0.05$ ) for the ExpO System over the form fillin interface.

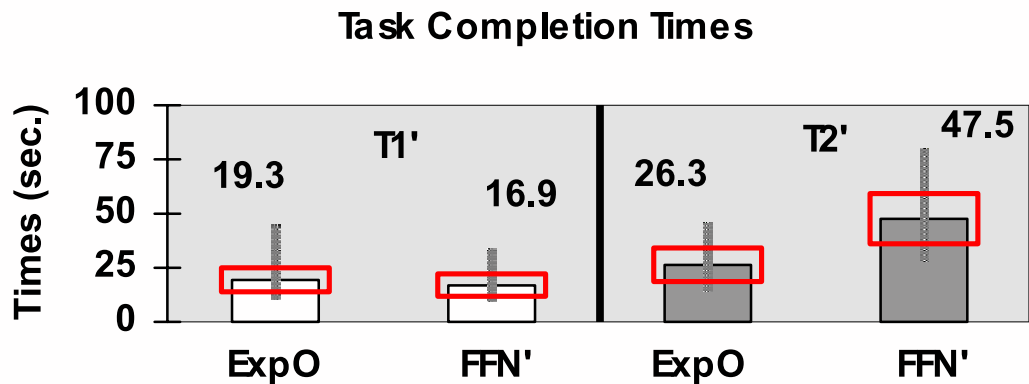


Figure 6.2: FFN' stands for the form fillin interface. Bars indicate the average values, rectangles indicate the standard deviations, lines indicate the range from the minimums to the maximums.

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 6.4. The scores for all of the questions were statistically significantly above the mid-point scale value of five ( $t(15) = 16.43, 5.84, 5.33, 13.49, \text{ and } 9.30$  respectively,  $p < 0.05$ ). An extra piece of data that was collected in this study was the number of queries submitted for each task (Figure 6.5). 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 ( $t(31) = 1.44, p < 0.05$ ).

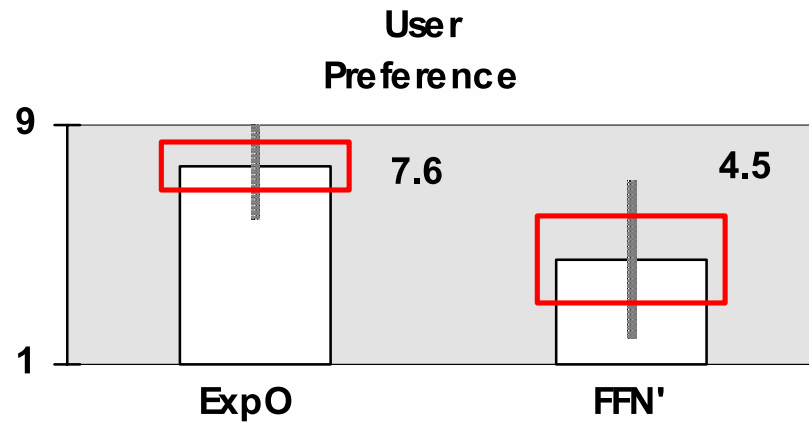


Figure 6.3: User preference for sixteen users (higher numbers indicate higher satisfaction).

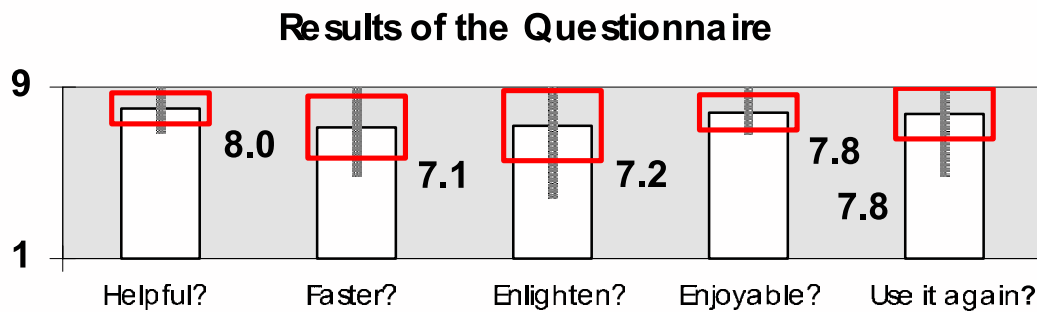


Figure 6.4: Subject questionnaire results (number of users is sixteen). Higher numbers indicate higher satisfaction for using the ExpO System.

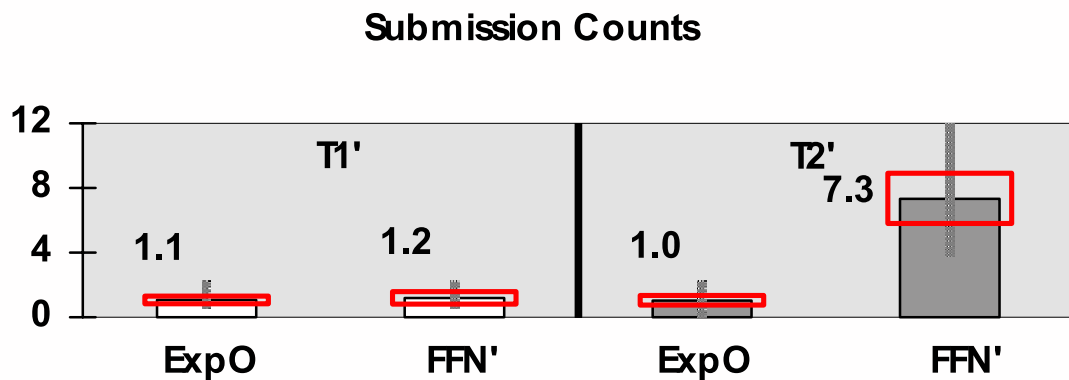


Figure 6.5: Number of queries submitted with form fillin (FFN') increases to 7.3 for T2 tasks (unclearly specified) while remaining small with the ExpO System.

## 6.6 Discussion of Results

Our findings support the hypothesis that for unclearly specified tasks, the generalized query previews yield 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 number of queries, a measure of network load, was more than 7 times lower. 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.

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. As expected, users of the ExpO System performed only slightly worse (14% slower). In addition, the number of queries submitted did not change.

For unclearly specified tasks, the generalized query previews enabled the users to see 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 greater than 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 specifications of each query caused significant differences in performance. In most working systems, network and database delays would substantially increase the advantage of the generalized query previews.

The users (statistically significantly) preferred the generalized query previews to the form fillin interface. They stated that the generalized query previews were very 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. 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.

This study supports the claim that benefits of generalized query previews exceed the overhead of the generalizations. The benefits will grow substantially in real-life situations where congested networks and servers are used. However, an overhead due to the generalizations still exists. The previous study showed up to 2.1 times performance improvement while this study showed 1.8 times improvement for similar tasks. This suggests that there may be some degradation in the user performance in the new user study due to the generalizations.

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

## **7. Implementation**

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. The first type, the database schema, is a hierarchy of database table and attribute names. It is requested from the server as soon as the program starts on the client. It loads rapidly because it rarely exceeds a few kilobytes of text information. The second type, the distribution information, is requested from the server only when needed, i.e., during the chart expansions, and used during

user selections (e.g., Figure 7.2). It can be created as a series of data structures for a set of attribute combinations or as a single large data structure that is manipulated at runtime to obtain the desired subsets of distribution information mapping to the user needs (i.e., for attributes that are being manipulated). The third type is the raw data that is fetched at the time of a user query submission, as it is the case in most other architectures.

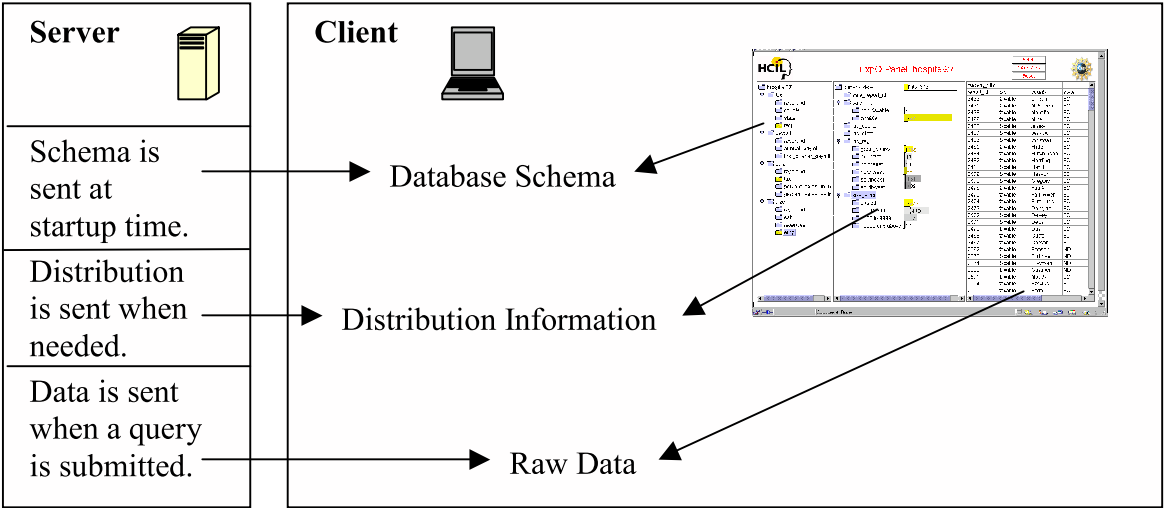


Figure 7.1: The architecture for transferring data in generalized query previews, shown on ExpO.

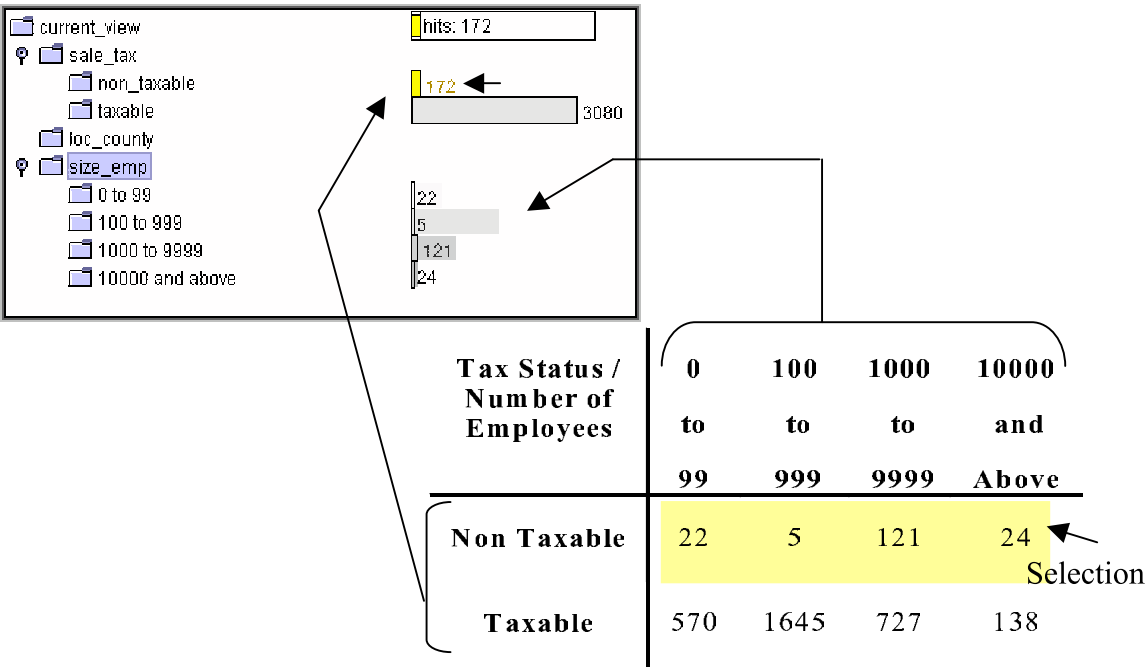


Figure 7.2: A sample selection operation and a related update on a chart.

The most critical piece of data that is transferred is the data distribution information, usually stored in multidimensional arrays. Although the size of the data structures to keep the distribution



information does not increase with the number of records, other features of the data may alter this size. First, for each attribute combination that can be manipulated by the user, if we keep a single separate array on the server, pre-calculation of these arrays may get cumbersome with the increasing number of attributes and their combinations. This situation can reduce the freshness of the distribution information. If we want to keep only one multidimensional array for all of the attributes of the data, addition of an attribute will increase the dimension of this large array. Hence, in time, dynamically updating the distribution information may again become difficult. Second, with the addition of a bucket to an  $n$ -dimensional array for an attribute, we will be adding a new  $n - 1$  dimensional slice to the array. Hence, the number of buckets for an attribute should be selected carefully. Third, for combinations of attributes, it may be difficult to gather the distribution information rapidly at runtime if a single large array for all the attributes of the data is used. Finding a small subset of this single large array can be difficult. If disjoint arrays in combinations of four to five attributes is kept, then this may need a large storage capacity from the server for all combinations. In both of these cases, a hash table to allocate the right subset would be needed. Keeping these efficient access paths up to date may gain importance, but may also prove to be difficult. Many of these issues have been investigated by the database researchers in the realm of data warehouses and online analytical processing (OLAP) [CD97]. Finally, as a well-known problem in databases, joining multiple tables at runtime for answering a user query with raw data gets difficult with the increasing number of tables, attributes, and user selections.

Formally, let's assume  $k$  denotes the total number of attributes of the universal relation of a database,  $l$  denotes the maximum number of attributes that can be manipulated simultaneously by the user,  $m$  denotes the maximum number of buckets for any attribute. Then, in the worst case, the size of a single multidimensional array that holds all the distribution information will be  $m^k$ . If a separate array for each combination of possible attribute selections is kept, then the size of a single array, in the worst case, will be  $m^i$  where  $i = 1 \dots l$  denoting the number of current attributes manipulated by the user. There will be a  $Combination(k, l) + Combination(k, l - 1) + \dots + Combination(k, 1)$  number of these arrays where the combination operation is calculated as  $Combination(k, l) = (k(k-1) \dots 1) / ((l(l-1) \dots 1)((k-l)(k-l-1) \dots 1))$ .

There are two challenges of the client-server architecture that require a more detailed analysis:

- Manipulation challenges,
- Representation challenges.

The manipulation challenges are observed when users want to make selections on more than a few attributes of the database at the same time. Tracking the updates on multiple charts, and also maintaining and transferring 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 querying on other attributes. In addition, 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 and contradicting with our initial design paradigm of not downloading/querying irrelevant parts of the raw data. The initial selections should be selective enough to prune the data down to a manageable size. This size depends on the client computer primary memory capacity as well as the network download speed. This approach should be implemented as a transparent operation to the user.

The representation challenges are more difficult to handle. The multidimensional arrays representing the distribution information are effective for some data types. On the other hand,

there are certain data types that cannot be handled easily with multidimensional arrays. The multi-valued data types have the most generalized version of these challenges. Temporal data forms a good example. 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 a multidimensional array. It is counted once for each date it spans. The distribution information represented with the array is no longer the same information represented with the data set.

Similar situations are observed with the NASA prototypes for the query previews idea, where a satellite picture maps onto multiple regions of the 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 multidimensional arrays must be enhanced. We introduced some approaches to address these challenges by creating various new data structures during our early work [BT98].

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 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 previous client modules,
- 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 ExpO (Figure 7.3), these modules were implemented in Java and were integrated as an applet. The client program connects to a Postgres Relational Database Server installed on a Sun Ultra-1 workstation. Querying is always done by using the general principles of the SQL querying language. This makes the implementation easily convertible to other server-database/client-applet pairs, if needed. The distribution information is created by simple batch queries.

## 8. Conclusions

Generalized query previews are a user interface architecture for browsing large online databases. 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 node of a slow network, accessing only the metadata can be very efficient. Metadata remains at a constant size even as the raw data grows. Showing the result set size before accessing the results is another beneficial idea. Users can immediately see the result sizes for their query submissions. The simple hierarchical display and creation of a user-defined view are two comprehensible strategies for defining (and confining) 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.

A controlled experiment clarified the application domain for generalized query previews, which proved the approach to be especially useful when users need to probe the data. Hence, generalized query previews help users refine their queries and guides them in the exploratory query formulation process, while reducing network and server load.

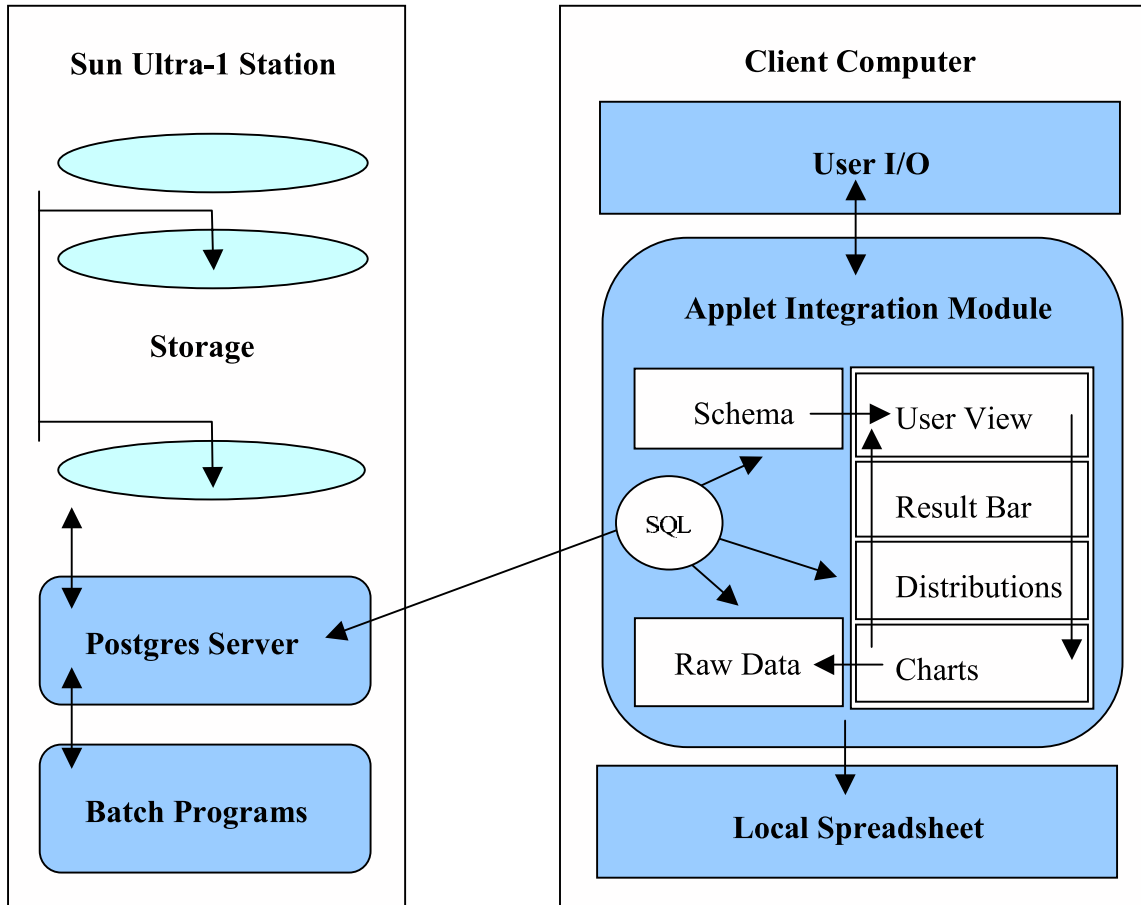


Figure 7.3: The software implementation and integration for the ExpO System.

## 9. Future Work

We envision using multiple views rather than a single user-defined view. In many cases, users may want to form two separate queries simultaneously. They may want to compare two different combinations of selections.

Second, researchers may want to explore using a hierarchy of charts rather than a single level of charts. 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 metadata may 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 [Shn94] mentions the need for a scatter plot widget for certain types of applications. 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 online can take time.

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.

Finally, applications for web page searches can also be considered as a separate research direction. The Internet, with its large collection of web pages, forms an unstructured large online database. Using the distribution information about date of creation, language, or country in tandem with the classical keyword search mechanism can be useful for the users of the Internet search engines.

## Acknowledgements

This work is supported in part by the United States Census Bureau, National Science Foundation, and NASA. Thanks to Catherine Plaisant for her contributions throughout the history of this project.

## 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., "TVEE: 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 Pyrcce, 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 Symposium*, pp. 87-90, (1997).
- [BT98] Beigel, R. and Tanin, E., "The Geometry of Browsing", *Proceedings of the Third Latin American Conference on Theoretical Informatics*, pp. 331-340, (1998).
- [BS99] Beilken, C. and Spence, M., "Visual, Interactive Data Mining with InfoZoom – the Medical Data Set", *Proceedings of the Workshop on Discovery Challenge in 3<sup>rd</sup> European Conference on Principles and Practice of Knowledge Discovery in Databases*, Prague, pp. 49-54, (September 1999).
- [Ber83] Bertin, J., "Semiology of Graphics", *University of Wisconsin Press*, Madison, Wisconsin, (1983).
- [CMP95] Capobianco, F., Mosconi, M., and Pagnin, L., "Progressive HTTP-Based Querying of Remote Databases within the Mammotta 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).
- [CD97] Chaudhuri, S. and Dayal, U., "An Overview of Data Warehousing and OLAP Technology", *ACM SIGMOD Record*, 26(1), (1997).
- [CRM95] Chuah, M. C., Roth, S. F., Mattis, J., and Kolojechick, 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).
- [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).
- [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).
- [Hear99] Hearst, M., "User Interfaces and Visualization", *Modern Information Retrieval*, ACM Press, Ricardo Baeza-Yates and Berthier Ribeiro-Neto, pp. 257-323, (1999).
- [HAC99] Hellerstein, J.M., Avnur, R., Chou, A., Hidber, C., Olston, C., Raman, V., and Roth, T., "Interactive Data Analysis: The Control Project", *IEEE Computer*, 32(8), pp. 51-59, (1999).
- [HES85] Heppel, 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).
- [KK94] Keim, D. A. and Kriegel, H. P., "VisDB: Database Explorations Using Multidimensional Visualization", *IEEE Computer Graphics and Applications*, pp. 40-49, (1994).
- [LRB97] Livny, M., Ramakrishnan, R., Beyer, K., Chen, G., Donjerkovic, D., Lawande, S., Myllymaki, J., and Wenger, K., "DEVis: Integrated Querying and Visual Exploration of Large Datasets", *Proceedings of the ACM SIGMOD '97*, (1997).
- [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).
- [Pol97] Pollitt, S., "Interactive Information Retrieval Based on Faceted Classification Using Views", *Proceedings of the 6th International Study Conference on Classification*, University College, London, (June 1997).
- [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).
- [RLS96] Roth, S. F., Lucas, P., Senn, J. A., Gomberg, C. C., Burks, M. B., Stroffolino, P. J., Kolojechick, J. A., and Dunnire, C., "Visage: A User Interface Environment for Exploring Information", *Proceedings of the IEEE Information Visualization Symposium*, pp. 3-12, (1996).
- [SA00] Shafer, J. and Agrawal, R., "Continuous Querying in Database-Centric Web Applications", *Proceedings of the 9th International World Wide Web Conference*, Amsterdam, (May 2000).
- [Shn94] Shneiderman, B., "Dynamic Queries for Visual Information Seeking", *IEEE Software*, 11(6), pp. 70-77, (1994).
- [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).
- [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).
- [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 (ACM-CIKM)*, pp. 80-85, (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).
- [Wil84] Williams, M. D., "What makes RABBIT run?" *International Journal of Man-Machine Studies* 21, pp. 333-335, (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).
- [WLH01] Wittenburg, K., Lanning, T., Heinrichs, M., and Stanton, M., "Parallel Bargrams for Consumer-based Information Exploration and Choice", *Proceedings of the ACM Symposium on User Interface Systems and Technology*, (2001).