Available online at www.sciencedirect.com

ScienceDirect

Information Systems 32 (2007) 402–423

www.elsevier.com/locate/infosys

Browsing large online data tables using generalized query previews

Egemen Tanin[a,b,*], Ben Shneiderman[b], Hairuo Xie[a]

[a]*Department of Computer Science and Software Engineering, University of Melbourne, Australia*
[b]*Department of Computer Science, Human–Computer Interaction Laboratory, Institute for Advanced Computer Studies, University of Maryland at College Park, USA*

Received 5 September 2002; received in revised form 14 December 2005; accepted 18 December 2005

Recommended by Y. Ioannidis

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/server resources posing queries that have zero- or mega-hit results. Generalized query previews enable efficient browsing of large online data tables by supplying data distribution information to 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 new user interface architecture and discusses three controlled experiments (with 12, 16, and 48 participants). Our prototype systems provide flexible user interfaces for research and testing of the ideas. The user studies show that for exploratory querying tasks, generalized query previews can speed user performance for certain user domains and can reduce network/server load.

Keywords: Information visualization; Internet; World-wide web; Online querying

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 (one of their sample data retrieval systems is available at 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). The World Health Organization (WHO) is an international organization that shares medical- and population-related information over the Internet (e.g., www3.who.int/whosis/menu.cfm). These are only a few of the organizations that are making vast data resources available over the Internet.

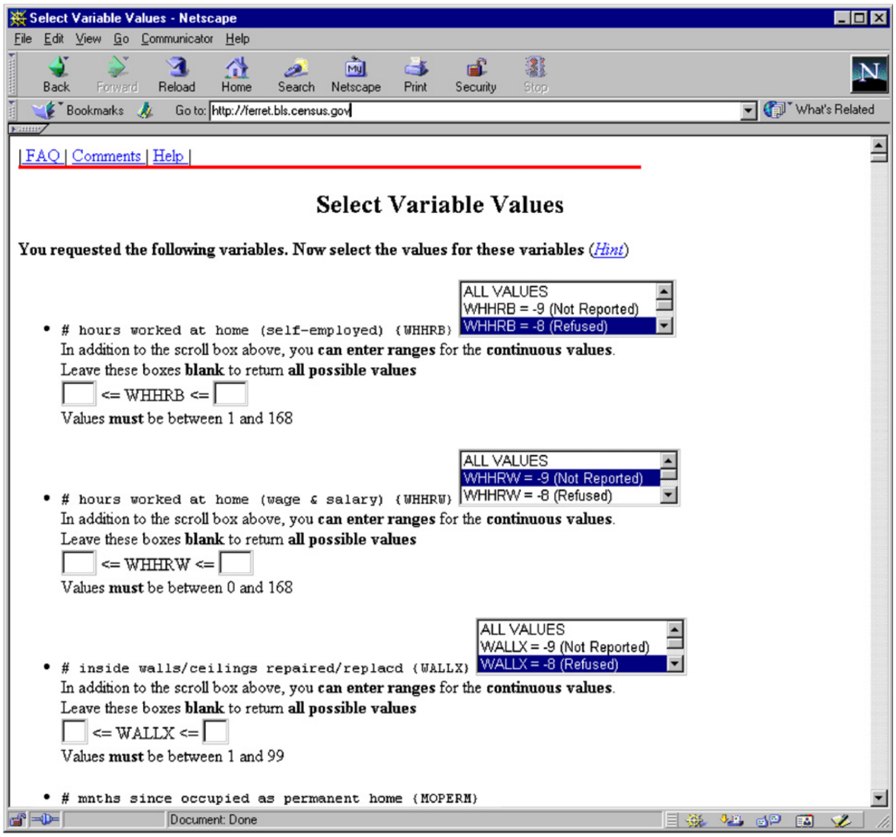
[*] Corresponding author. Tel.: +61 3 8344 1350.
E-mail addresses: egemen@cs.mu.oz.au (E. Tanin), ben@cs.umd.edu (B. Shneiderman), h.xie@pgrad.unimelb.edu.au (H. Xie).

0306-4379/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.is.2005.12.006

The user interfaces that serve as the database front-ends mostly utilize menus or form fillin interfaces [1]. 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. Many of these users have no background on these databases. Some 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 users are informed about the data and that they want to submit known-item queries rather than probing the data. However, unguided novice users of these data centers frequently waste their time submitting queries that have zero- 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 congested network or server.

Fig. 1 shows a form fillin interface that was generated using the interactive United States Census Bureau homepages. This interface is a good example of the database front-ends that are commonly 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- or mega-hit result sets. Novice users of this tool/database have to probe the data 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 describes our early work and states the results from our first user study. Section 3 describes the approach introduced with this paper. Sections 4 and 5 report results of our second and third user studies and Section 6 discusses the implementation issues. Section 7 documents the related research. Section 8 states our conclusions and possible future work.



http://ferret.bls.census.gov

Select Variable Values

You requested the following variables. Now select the values for these variables (*Hint*)

- # hours worked at home (self-employed) (WHHRB)

  ALL VALUES
WHHRB = -9 (Not Reported)
WHHRB = -8 (Refused)

  In addition to the scroll box above, you **can enter ranges** for the **continuous values**.
  Leave these boxes **blank** to return **all possible values**
  □ <= WHHRB <= □
  Values **must** be between 1 and 168

- # hours worked at home (wage & salary) (WHHRW)

  ALL VALUES
WHHRW = -9 (Not Reported)
WHHRW = -8 (Refused)

  In addition to the scroll box above, you **can enter ranges** for the **continuous values**.
  Leave these boxes **blank** to return **all possible values**
  □ <= WHHRW <= □
  Values **must** be between 0 and 168

- # inside walls/ceilings repaired/replacd (WALLX)

  ALL VALUES
WALLX = -9 (Not Reported)
WALLX = -8 (Refused)

  In addition to the scroll box above, you **can enter ranges** for the **continuous values**.
  Leave these boxes **blank** to return **all possible values**
  □ <= WALLX <= □
  Values **must** be between 1 and 99

- # mnths since occupied as permanent home (MOPERM)

Fig. 1. A form fillin interface from the United States Census Bureau homepages.

2. Early work and the first user study

Our early work on dynamic queries [2–5] used a direct manipulation approach to facilitate query formulation on multi-dimensional 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 users. Yet, large online data cannot be immediately and continuously accessed.

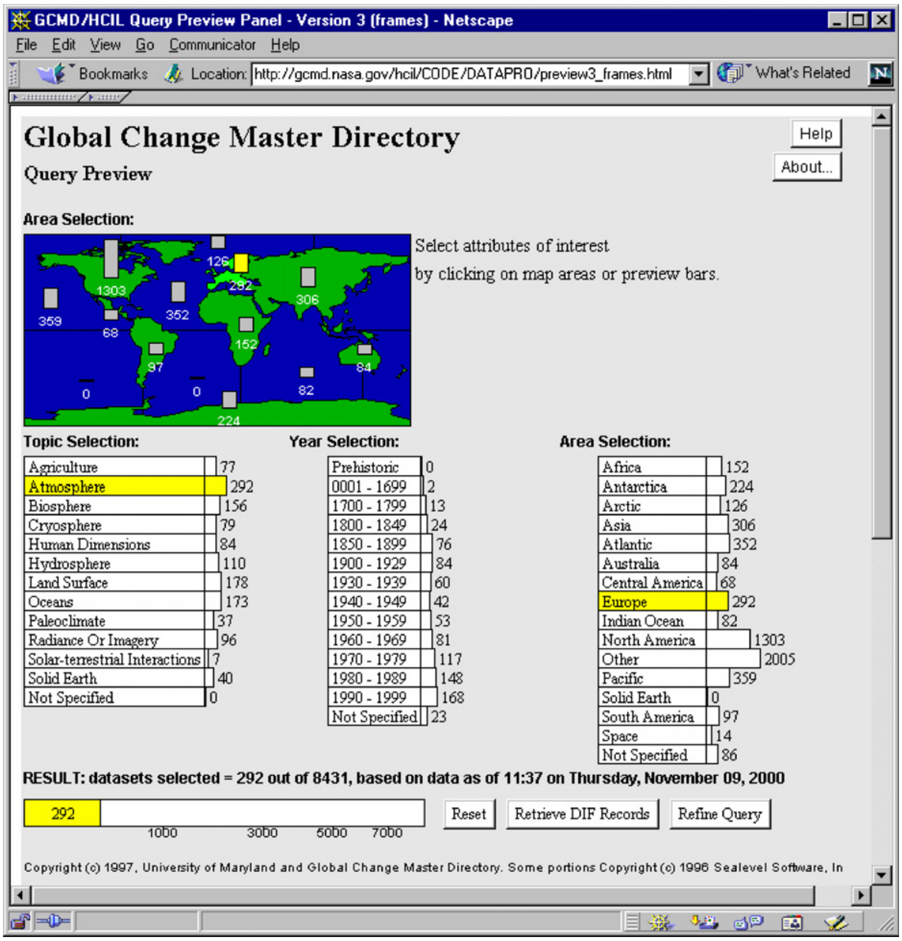
Later we worked on NASA's large online data collections. These collections are stored in vast distributed data archive centers and contain various types of data (e.g., documents, images, numerical values, etc.). The attempt to apply dynamic queries to these collections required us to look for solutions to make dynamic queries work with large online data. Using overviews and previews, we made dynamic queries applicable to large online data and allowed users to efficiently prune irrelevant data. This NASA project led to the first ideas on 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 pre-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 to continue the query process. 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 [6–8]. This strategy was created as an experimental interface [9] for the Global Change Master Directory and was the basis for the Global Land Cover Facility interfaces, all part of NASA. This paper creates a general and more transparent concept from the initial query previews idea by helping users work on any of the data attributes for online data browsing.

Query previews show a few discriminating (i.e., capable of selecting a few items from a large set) attributes of the data so that the users' selection on these attributes would immediately lead to a smaller subset of the data. A few commonly used attributes of the data are pre-selected to form a query preview by the software designer. 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 using histograms. 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 because aggregate data size remains fixed even as the number of records in a database table grows. Fig. 2 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 preview phase is followed by a refinement phase.

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 many 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 (details available from Tanin et al. [10]).

In this first user study, we identified the clearly specified tasks as tasks having a straightforward and an accurate definition, i.e., known-item searches. For example, “List all the State of Maryland


GCMD/HCIL Query Preview Panel - Version 3 (frames) - Netscape
File Edit View Go Communicator Help
Bookmarks
Location: http://gcmd.nasa.gov/hcil/CODE/DATAPRO/preview3_frames.html
What's Related
Help
About...
126
1303
292
306
359
352
68
152
97
84
0
0
82
224
Reset
Retrieve DIF Records
Refine Query
1000
3000
5000
7000

Global Change Master Directory

Query Preview

Area Selection:

Select attributes of interest by clicking on map areas or preview bars.

Topic Selection:

Agriculture	77
Atmosphere	292
Biosphere	156
Cryosphere	79
Human Dimensions	84
Hydrosphere	110
Land Surface	178
Oceans	173
Paleoclimate	37
Radiance Or Imagery	96
Solar-terrestrial Interactions	7
Solid Earth	40
Not Specified	0

Year Selection:

Prehistoric	0
0001 - 1699	2
1700 - 1799	13
1800 - 1849	24
1850 - 1899	76
1900 - 1929	84
1930 - 1939	60
1940 - 1949	42
1950 - 1959	53
1960 - 1969	81
1970 - 1979	117
1980 - 1989	148
1990 - 1999	168
Not Specified	23

Area Selection:

Africa	152
Antarctica	224
Arctic	126
Asia	306
Atlantic	352
Australia	84
Central America	68
Europe	292
Indian Ocean	82
North America	1303
Other	2005
Pacific	359
Solid Earth	0
South America	97
Space	14
Not Specified	86

RESULT: datasets selected = 292 out of 8431, based on data as of 11:37 on Thursday, November 09, 2000

292

Copyright (c) 1997, University of Maryland and Global Change Master Directory. Some portions Copyright (c) 1996 Sealevel Software, In

Fig. 2. 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.

employees from the employee database" is a clearly specified task. 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 such tasks, the relevance of the query preview attributes to the query is not an influential factor as users are best served by directly going to the refinement phase, e.g., the form fillin interface (tasks of this worst case scenario are called T1).

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, DC 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 counterbalanced design with 12 subjects. All the subjects were new to the data set used in the study yet they were well trained in using computers and web search engines. 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 interfaces with query previews.

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 T2 and T3 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.

Also, as expected, users of the form fillin only 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 the 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 let them concentrate on a much smaller set of possible queries in the refinement phase.

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 absolutely 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 small, the time for thinking and 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 an average of 5.2 for the interface without it on a scale of 1–9 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.

3. Generalized query previews

With this paper we define a general user interface architecture for efficient browsing of large online data. The generalized query previews user interface architecture has the following three components (Fig. 3):

- *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, i.e., in the form of histograms.
- *Raw data component*: displays and/or used for the analysis of the retrieved records.

Generalized query previews allow 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

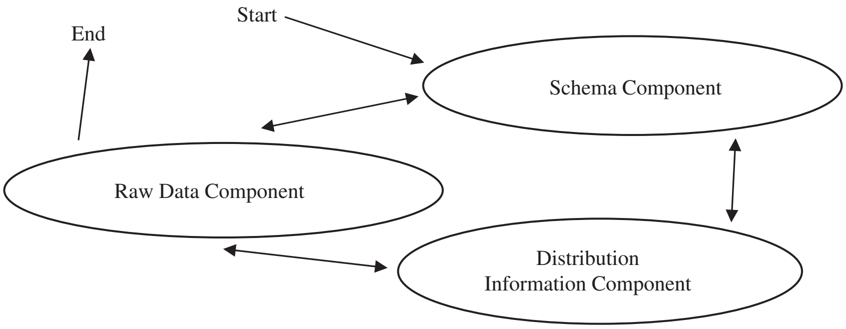


Fig. 3. Interactions between the three generalized query previews components.

desired, forward it to another program for further analysis. This pattern can be repeated, if needed, with many different attributes. Hence, we are now not confined to the limitations of using only a few pre-selected attributes of query previews.

To demonstrate and later to experiment with the generalized query previews user interface architecture, we implemented prototype systems. First, we implemented a system using 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 10 attributes and approximately 3000 entries. The data are 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. Second, we implemented a system using a sample data set that we generated from the Internet Movie Database (www.imdb.com). These data contain information about various films. It has again about 10 attributes but this time it contains 10,000 entries. These film data are stored in a single relation. In the remainder of this section, we analyze the generalized query previews user interface architecture using a prototype that we implemented with the Census data. We refer to this prototype as the ExpO system, and we will go through the components of ExpO to give the details on our generalized query previews user interface architecture.

3.1. 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 Fig. 4, 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 these data, 'loc', 'payroll', 'sale', and 'size' are the tables that share a common identifier. The second level shows 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 can become annoying to use a complex visualization. However, large databases may need scalable approaches. Also, in some applications, only a few tables sharing a set of common attributes may be of interest to users. In others, many clusters containing such attributes may exist.

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 we do not have a simple universal relation, selections of two attributes from two tables without any common attributes should be prevented. Otherwise, this can cause conflicts in implicit join operations in 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 (Fig. 5). The panel on the right depicts a few 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

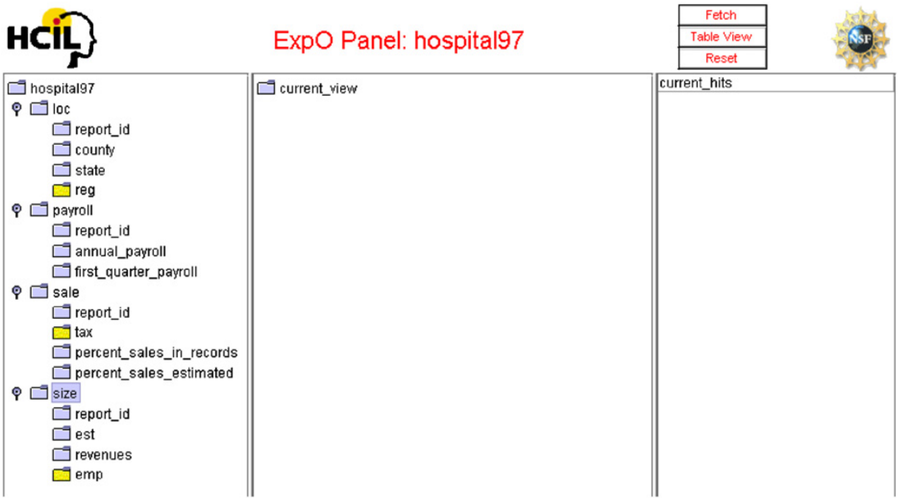


Fig. 4. An example generalized query preview interface, ExpO, the left panel displays the table and attribute names of a relational database as an implementation of the schema component.

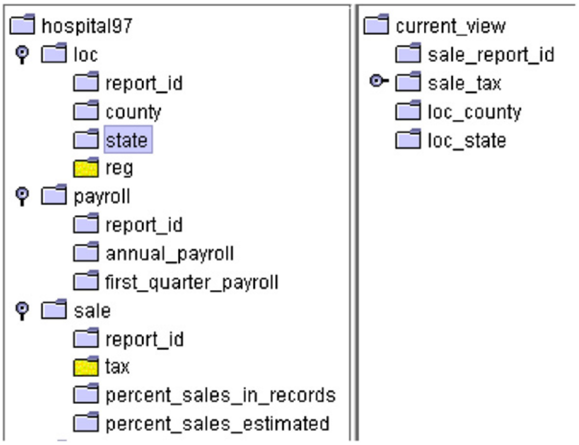


Fig. 5. ExpO with a user-defined view after four attribute selections. The user-defined view is also a hierarchical browser.

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 also be pre-defined views from the data, and need not directly map to the relations of the database.

3.2. 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 Fig. 5, 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.

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 (Figs. 6 and 7). Forming these buckets is the duty of the system designer, who should gather information about the details of the data and users’ needs. In some cases, e.g., gender of a person, the formation of the buckets may be easy. 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.

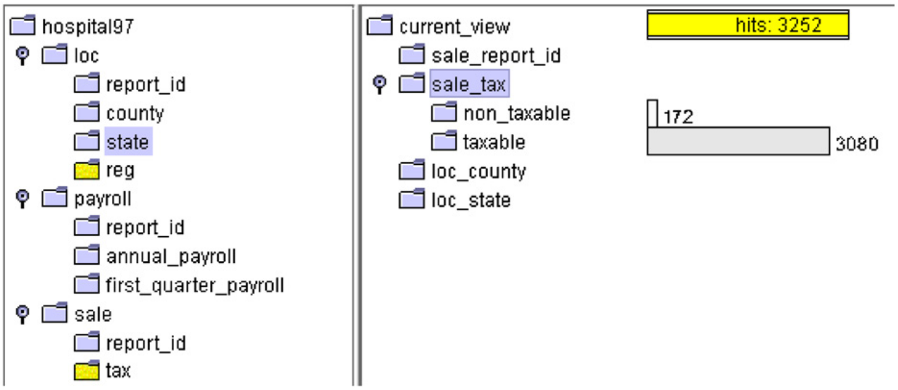


Fig. 6. ExpO with the data distribution information attached to the user-defined view.

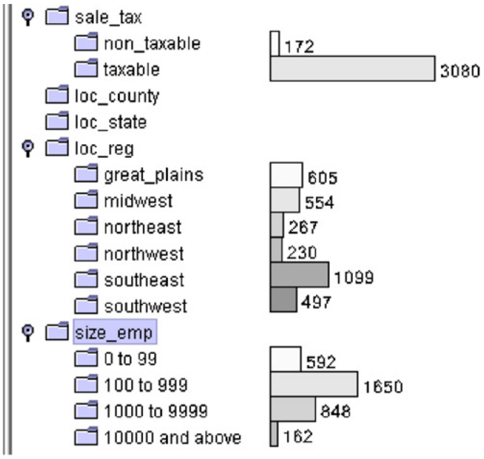


Fig. 7. Information attached to the buckets of three attributes in the user-defined view.

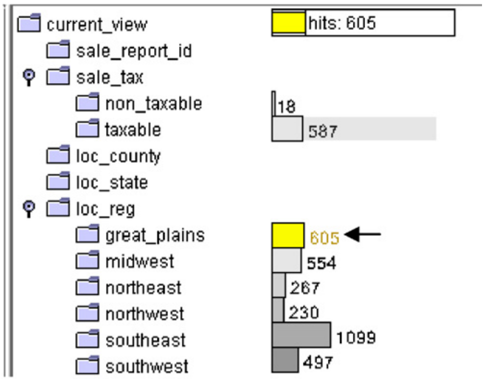


Fig. 8. A selection is made on one of the buckets, 'great_plains'.

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 Fig. 6). Hence, users will be aware of the consequences of their query submissions, i.e., whether they are submitting mega-hit or zero-hit queries.

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 (Fig. 8). 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. 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 running into problems with network or server loads. 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 (Fig. 9). 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

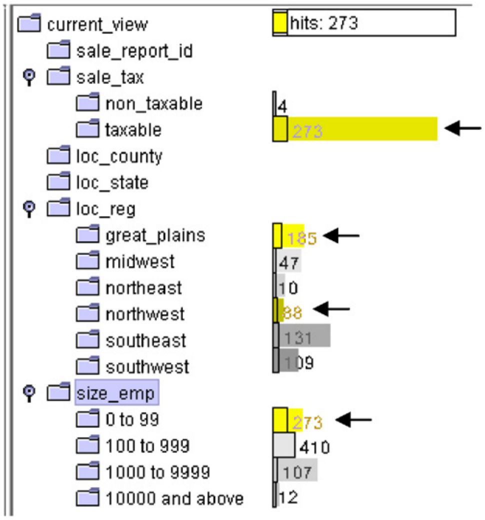


Fig. 9. Multiple selections are made, 'taxable', 'great_plains', 'northwest', and '0 to 99'.

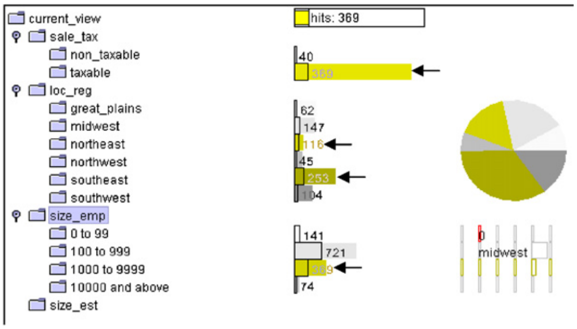


Fig. 10. Other visual aids, such as a pie chart, may also be available to users.

experiences show that the current ExpO implementation is one of the most intuitive implementations.

It is possible to display the distribution information on different types of visual aids. Fig. 10 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. Representations, such as a color-coded stack of bars instead of a single bar, can easily be utilized.

3.3. *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- nor mega-hit result sets is an issue. The benefit to users is in rapid and meaningful exploration of data, and the benefit to system maintainers is the reduced network and server load. For some cases, the system designer can take drastic measures such as preventing query submissions for zero-/mega-hit queries. Fig. 11 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.

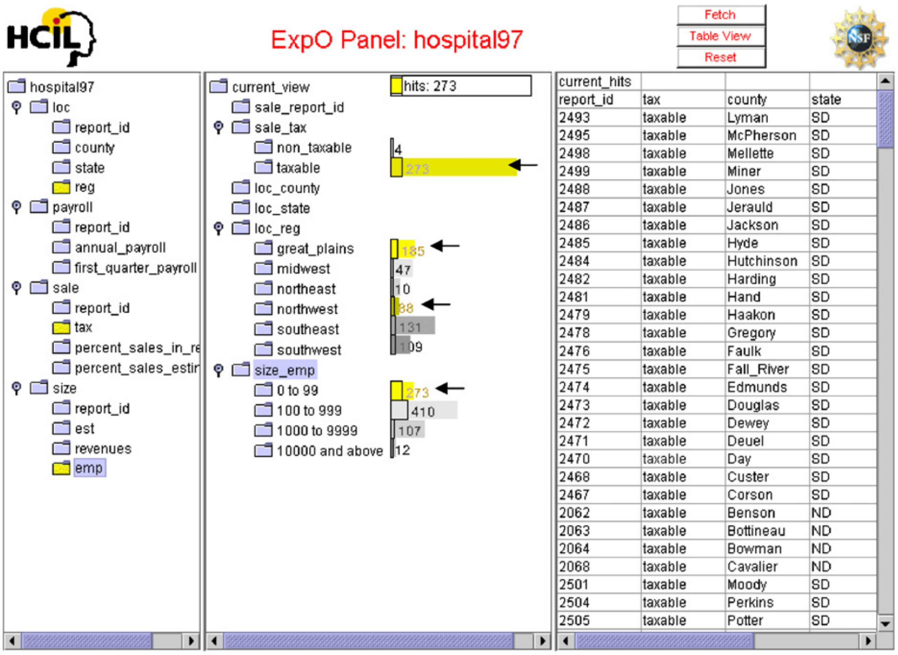
4. Second study: generalized query previews with computer experts

Generalized query previews are a more mature idea than the query previews concept. Although they are general, they also introduce some overheads such as defining a user-defined view and possibly confusing expansions and contractions of charts. Hence, there is a possibility that user performance and preferences could be disturbed, and users may get confused and annoyed by the generalizations. Therefore, there was a need for new user studies to verify and quantify the benefits of generalized query previews and measure subjective user preferences. This section focuses on our study with computer experts on generalized query previews. This study uses our ExpO system with Census data and focuses on a setting where people with strong computer science background, i.e., a degree in computer science, performing as our subjects. The third and final study focuses on a very commonly used data set, the Internet Movie Database, with specific interest on only novice computer users, i.e., users who are not well-versed on topics such as databases and networking.

4.1. *Hypothesis*

In this second study, we identified the task types that would put generalized query previews into their best and worst situations. Clearly specified tasks and unclearly specified tasks were again used. However, for generalized query previews, 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,



current_hits			
report_id	tax	county	state
2493	taxable	Lyman	SD
2495	taxable	McPherson	SD
2498	taxable	Mellette	SD
2499	taxable	Miner	SD
2488	taxable	Jones	SD
2487	taxable	Jerauld	SD
2486	taxable	Jackson	SD
2485	taxable	Hyde	SD
2484	taxable	Hutchinson	SD
2482	taxable	Harding	SD
2481	taxable	Hand	SD
2479	taxable	Haakon	SD
2478	taxable	Gregory	SD
2476	taxable	Faulk	SD
2475	taxable	Fall_River	SD
2474	taxable	Edmunds	SD
2473	taxable	Douglas	SD
2472	taxable	Dewey	SD
2471	taxable	Deuel	SD
2470	taxable	Day	SD
2468	taxable	Custer	SD
2467	taxable	Corson	SD
2062	taxable	Benson	ND
2063	taxable	Bottineau	ND
2064	taxable	Bowman	ND
2068	taxable	Cavalier	ND
2501	taxable	Moody	SD
2504	taxable	Perkins	SD
2505	taxable	Potter	SD

Fig. 11. ExpO with results to a query displayed in a panel on the right, 273 hits are listed.

and (3) users will always prefer generalized query previews.

4.2. Independent and dependent variables

The independent variable is the user interface type and the treatments are: (1) A form fillin interface (using the same (re)naming convention with the new task types, abbreviated as 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.

4.3. Subjects and materials

Sixteen computer science graduate students volunteered as subjects. All of them use computers every day and have at least 5 years of experience with computers. 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 (Fig. 12). Another set of eight subjects worked in reverse order.

The materials include a form fillin interface 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.

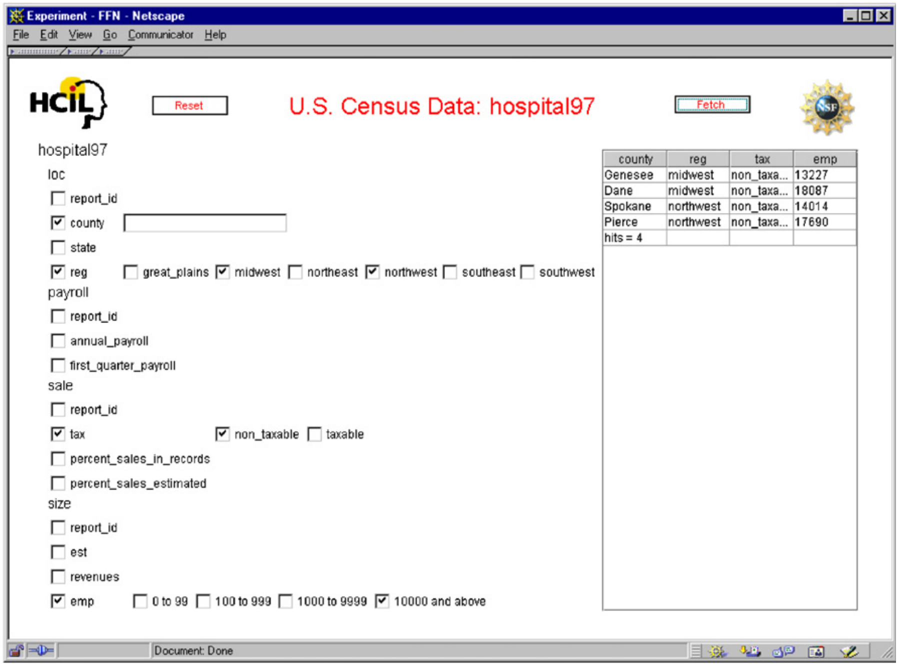
The form fillin interface in Fig. 12 was used to perform queries on a United States Census Bureau data set where attributes in this sample data set were shown with a simple hierarchical browser. Attributes were selected by marking the toggles near them. This action also triggers the display of editable fields attached to these attributes. The final 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.

4.4. 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′: Clearly specified tasks, e.g., “Please get a list of all the counties that are in the northwest


Experiment - FFN - Netscape
File Edit View Go Communicator Help
HCIL
Reset
U.S. Census Data: hospital97
Fetch
NSF
Document: Done

hospital97

loc

☐ report_id

☑ county

☐ state

☑ reg ☐ great_plains ☑ midwest ☐ northeast ☑ northwest ☐ southeast ☐ southwest

payroll

☐ report_id

☐ annual_payroll

☐ first_quarter_payroll

sale

☐ report_id

☑ tax ☑ non_taxable ☐ taxable

☐ percent_sales_in_records

☐ percent_sales_estimated

size

☐ report_id

☐ est

☐ revenues

☑ emp ☐ 0 to 99 ☐ 100 to 999 ☐ 1000 to 9999 ☑ 10000 and above

county	reg	tax	emp
Genesee	midwest	non_taxa...	13227
Dane	midwest	non_taxa...	18087
Spokane	northwest	non_taxa...	14014
Pierce	northwest	non_taxa...	17690
hits = 4			

Fig. 12. The form fillin interface (FFN′) used in the study. The rectangle on the right is used for displaying the results to a query (four hits for this query).

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 second study used a within subject counter-balanced design with 16 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 sample 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 16 different combinations. Subjects received only introductory level training (not to exceed 10 min in total). They performed only two training tasks on each of the user interfaces.

4.5. Results

Fig. 13 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 1–9 scale (with higher numbers indicating stronger preferences). The first question addressed the general preference of subjects for using either of the interfaces (Fig. 14). The results show a statistically significant preference ($t(15) = 6.37$, $p < 0.05$) for the ExpO system over the form fillin interface, FFN′.

The rest of the questions asked what the subjects thought about the user interfaces. The results (average scores, standard deviations, minimums,

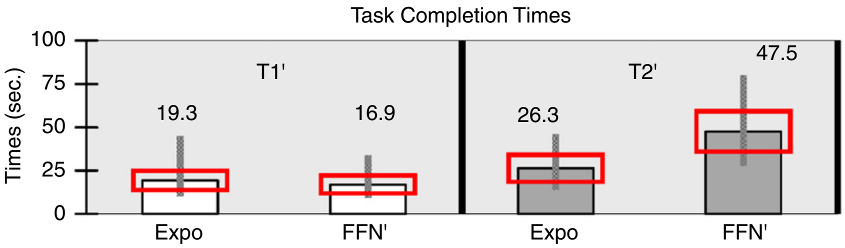


Fig. 13. FFN′ stands for the form fillin interface. Bars indicate the average values, rectangles indicate the standard deviations, and lines indicate the range from the minimums to the maximums.

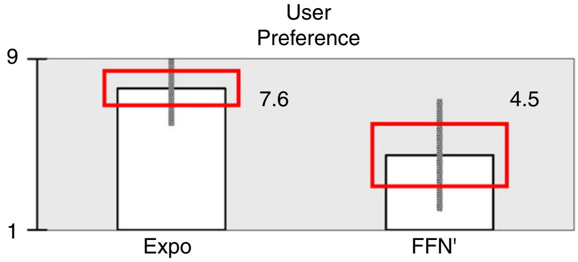


Fig. 14. User preference for 16 users (higher numbers indicate higher satisfaction).

and maximums) appear in detail in Fig. 15. 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, $p<0.05$). An important piece of data that was collected with this study was the number of queries submitted for each task (Fig. 16). 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.

4.6. Discussion of the results of the second study

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. Again 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 users to see which of the possible queries should be used. On the other hand, in the form fillin interface, 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 relatively immediate, the time for thinking and filling in the specifications of each query caused significant differences in performance. In many working systems, network and database delays could substantially increase the advantage of the generalized query previews.

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 understanding and control in performing the tasks. Yet, many users experienced some problems understanding the concept of a view and adopting to the bar expansions and contractions when they first started

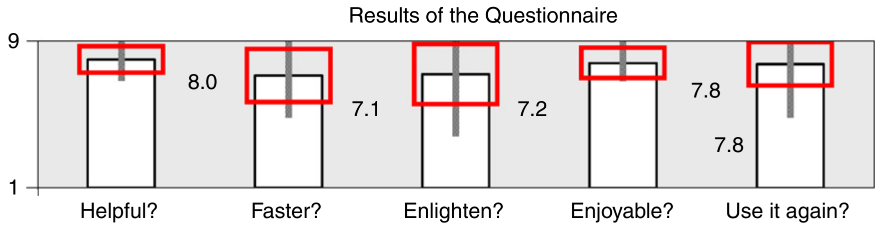


Fig. 15. Subject questionnaire results (number of users is 16). Higher numbers indicate higher satisfaction for using the ExpO system.

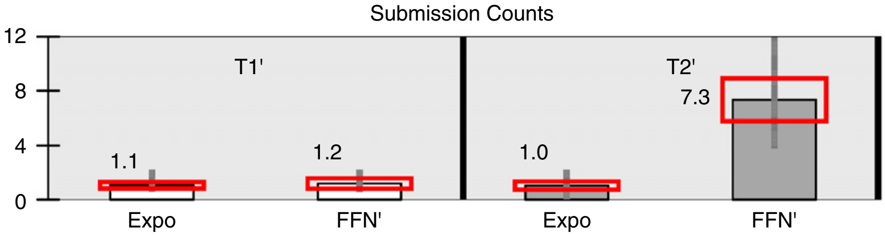


Fig. 16. Number of queries submitted with the form fillin (FFN′) increases to 7.3 for T2′ tasks (unclearly specified) while remaining small with the ExpO system.

using the ExpO system. These problems seem to diminish quickly with user experience.

This second study supports the claim that benefits of generalized query previews exceed the overhead of the generalizations. The benefits can 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 second user study due to the generalizations. This issue will be clearer with our third study in the following section.

5. Third study: generalized query previews with novice computer users

This section focuses on our third user study with generalized query previews. This study uses a new version of the ExpO system where we can browse approximately 10,000 films that were obtained from the Internet Movie Database. We also implemented another system that is very similar to the ExpO system but without the bars to help users. This interface can be viewed as an intermediate interface between the form fillin approach and the generalized query previews. After the second study, our aim was also to see whether the benefits of generalized query previews were due to the existence of the distribution information or just due to the fact that users can easily focus on a few attributes with the help of the schema component. More importantly, we need to see whether novice computer users can adapt to the generalizations as well as the experts. In both of these studies, it is important to note that the users were inexperienced with the particular database that was being used for that study. We continue to use the sample prototype system name ExpO and also name the altered version, i.e., without bars, as ExpO–WoB. The form fillin interface (FFN′) for films was again available but slightly adapted to the new domain of movies.

5.1. Hypothesis

Clearly specified tasks and unclearly specified tasks were again used. The hypothesis is also similar to the second study: For clearly specified tasks (T1′), generalized query previews will lead to slower task performance than other interfaces; the number of queries submitted to the server will be the same in all three interfaces. For unclearly specified tasks (T2′), the generalized query previews will lead to faster performance than others; the number of queries submitted will also be lower than others. Users will prefer generalized query previews to others in general.

5.2. Independent and dependent variables

The independent variable is again the user interface type and the treatments are: (1) A form fillin interface (FFN′), (2) a generalized query preview interface (ExpO) and (3) a generalized query preview interface but without the bars (ExpO–WoB). 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.

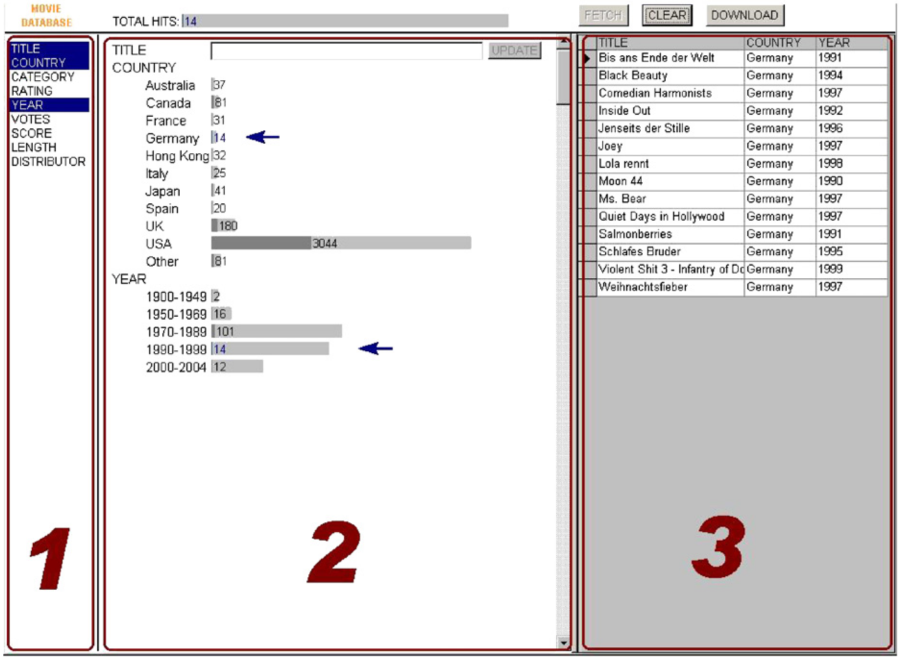
5.3. Subjects and materials

Forty-eight subjects were involved in this study. None of them were experts on computers, i.e., databases, networking, etc. All of them had basic knowledge on how to use a computer and browse the web.

The materials include the FFN′ interface for querying our movie database, the ExpO–WoB interface, and the ExpO interface for the same database. We prepared several tasks (training and real tasks) to be performed by the subjects, a subject background survey, and a subjective preference questionnaire (all similar to the second user study).

The survey includes questions that will ascertain the experience level of the subjects with computers and with search engines and the web. The subjective preference questionnaire includes questions that aim to find out which of the interfaces the subjects preferred.

The new ExpO system with the movie database is shown in Fig. 17 and was utilized in this study. The ExpO–WoB system is very similar to this interface but it does not have the bars. The new fillin interface, retains the abbreviated name of FFN′, is similar to the one in the second user study (with minor changes to the way attributes are displayed, i.e., to fit into the desired look and feel of the movie database domain). It basically contains an exhaustive list of all attributes with a similar design to the distribution information component of the new ExpO interface and also does not have the bars. For this study, as we had an immediate real-life implementation of the form fillin interface available over the Internet (i.e., www.imdb.com, advanced search), we collected information on average query processing/network-transfer time. This value (5 s to return the results) was used in all the interfaces in the third study to simulate the result-set downloads and users were not



TOTAL HITS: 14

TITLE	
COUNTRY	
Australia	37
Canada	81
France	31
Germany	14
Hong Kong	32
Italy	25
Japan	41
Spain	20
UK	180
USA	3044
Other	81
YEAR	
1900-1949	2
1950-1969	16
1970-1989	101
1990-1999	14
2000-2004	12

TITLE	COUNTRY	YEAR
Bis ans Ende der Welt	Germany	1991
Black Beauty	Germany	1994
Comedian Harmonists	Germany	1997
Inside Out	Germany	1992
Jenseits der Stille	Germany	1996
Joey	Germany	1997
Lola rennt	Germany	1998
Moon 44	Germany	1990
Ms. Bear	Germany	1997
Quiet Days in Hollywood	Germany	1997
Salmonberries	Germany	1991
Schlafes Bruder	Germany	1995
Violent Shit 3 - Infantry of Dc	Germany	1999
Weihnachtsfieber	Germany	1997

Fig. 17. The new version of the ExpO interface used in the third study that uses a movie database (the rectangle labeled with number 1 maps to the schema component, the rectangle labeled with number 2 maps to the distribution information component, and the rectangle labeled with number 3 maps to the raw data component).

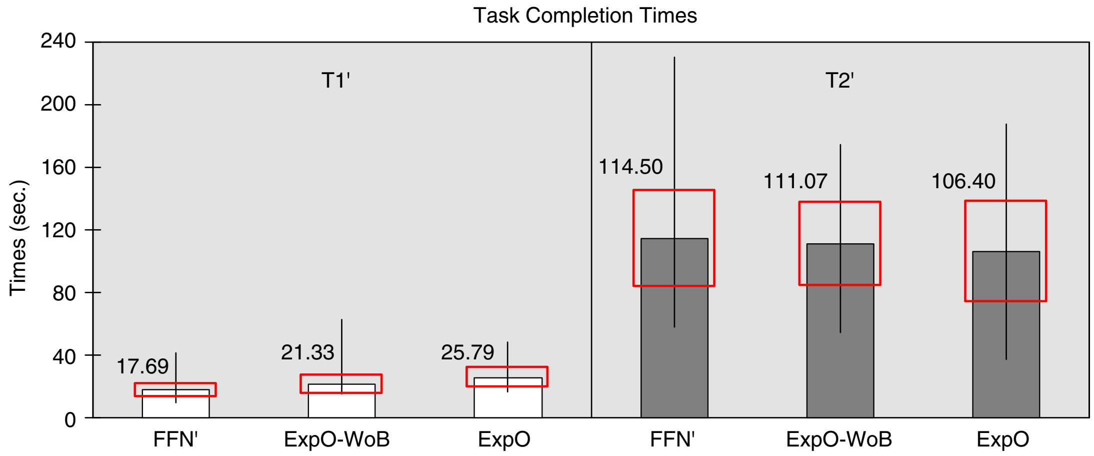


Fig. 18. Task completion times in seconds for the task types T1′ and T2′.

informed that this was a simulation. Although this value is much less than the ones we observed with very large data collections that are currently available from the US Census Bureau, it still brings more realism to the third study (the query processing time of the first two studies was in average 2 s).

5.4. Tasks and structure

The third study uses a within subject counter-balanced design with 48 subjects. Each subject was tested on all three interfaces, but the order of the interfaces was different for each subject. Three parallel but different sets of tasks were used with each of the three interfaces to reduce the chance of performance improvement. The application order of the parallel sets of tasks were different for each subject (in average two were randomly chosen out of six possible orders). Each set of tasks includes the two tasks types (T1′, T2′), with two sample tasks for each of these types. The order of the task types within a task set was reversed. The order of the tasks within each task type was also reversed. So, we had a total of (6 × 2 × 2 × 2) 48 subjects. The statistical analysis use two-tailed paired two-sample t-test for means. Each task is considered separately leading to a degrees of freedom of 95.

The tasks given to the subjects were similar to the ones with the second and first studies and they ask subjects to find a single or a list of film(s) from the movie database satisfying certain constraints and preferences.

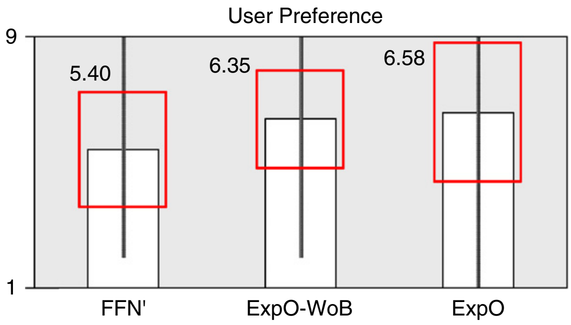


Fig. 19. User preferences for 48 users.

5.5. Results

Fig. 18 summarizes the times for completing each of the task types for our subjects for each of the user interfaces. As hypothesized, for T1′ tasks, the ExpO system yielded slower performance than the form fillin interface, FFN′, and the ExpO–WoB system ($t(95) = 11.24$ and $t(95) = 6.06$, respectively for $p < 0.05$). For T2′, the ExpO system yielded slightly faster performance than the form fillin interface ($t(95) = 2.21$, $p < 0.05$) but we could not detect a significant difference between the ExpO and ExpO–WoB systems. The subjects again answered six questions about their preferences on a 1–9 scale. The first question addressed the general preference of subjects for using either of the interfaces (Fig. 19). The results show a preference ($t(47) = 2.42$, $p < 0.05$) for the ExpO system over the FFN′ but we could not detect a significant difference between the ExpO and ExpO–WoB interfaces.

The rest of the questions asked what the subjects thought about the user interfaces. The results

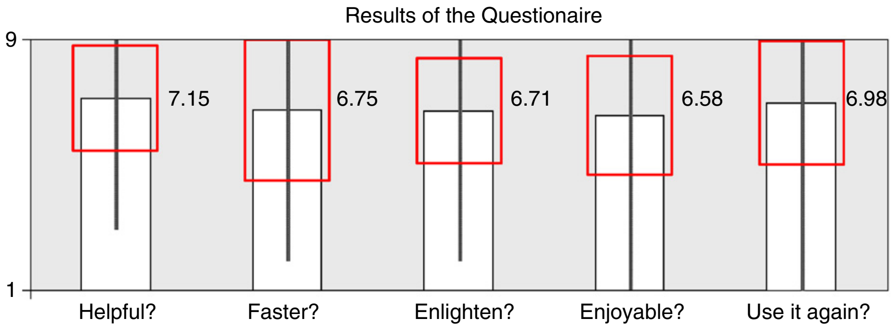


Fig. 20. Questionnaire results. Higher numbers indicate higher satisfaction with the ExpO.

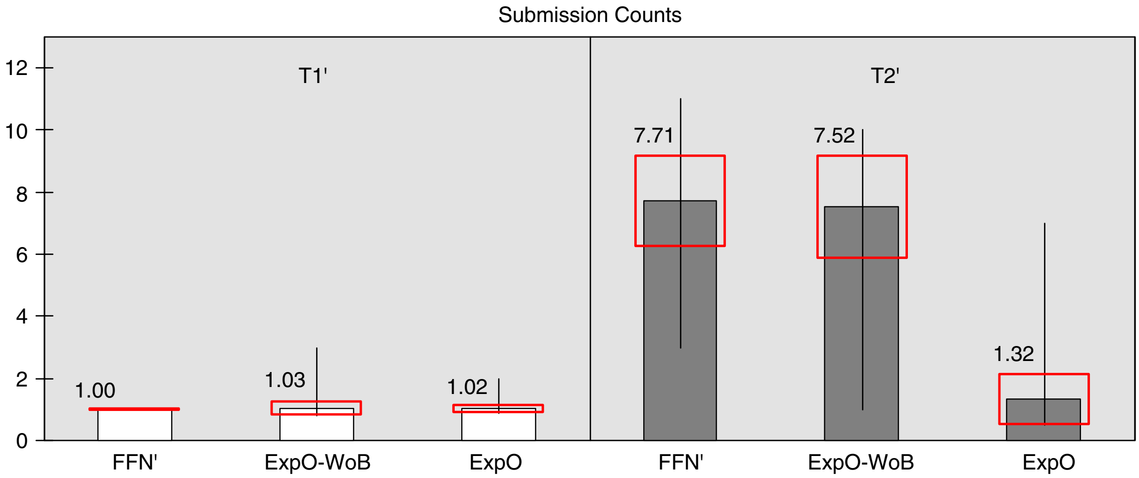


Fig. 21. Number of queries submitted with the form fillin (FFN′) increases to 7.71 for T2′ tasks (unclearly specified) while remaining small with the ExpO system.

(average scores, standard deviations, minimums, and maximums) appear in detail in Fig. 20. The scores for all of the questions were statistically significantly above the mid-point scale value of five ($t(47) = 8.81, 5.38, 7.01, 5.74$, and 6.87, respectively, $p<0.05$). An important piece of data collected in this study was the number of queries submitted for each task (Fig. 21). The results show a statistically significant difference ($t(95) = 37.20$, $t(95) = 34.68$, for FFN′ and ExpO–WoB at $p<0.05$, respectively) for the ExpO system with the T2′ tasks. For T1′, the difference was not significant.

5.6. Discussion of results of the third study

Our findings confirmed some of our previous results, however, there was an important change with the novice users. For unclearly specified tasks (T2′), the generalized query previews yield faster performance times than the form fillin interface but only slightly faster in comparison to the second study. Yet, the number of queries, a measure of network and server load, was still dramatically lower (Fig. 21). This is a compelling advantage, not only for reduced network and server loads, but because it demonstrates that users understood the query process and got to results with just a few steps (approximately 1/6 of the number of queries). The ExpO–WoB had a high query submission rate; in general, it did not behave much differently than the FFN′.

For the clearly specified tasks (T1′), as expected, the generalized query previews produced slightly slower performance times, but no statistically significant difference was observed for the submission counts.

Users still prefer generalized query previews but this preference seems to be weaker in comparison to the expert users. These results can be explained after also looking at the user comments and our observations of the novice users. Many users stated that the generalized query previews are efficient but they had a hard time adapting and understanding the bars and the possible uses of distribution information. Understandably, some novice users could not adapt to the bars as quickly as the computer experts did. Hence, these novice users spent more time performing the queries. This reduced their performance with the ExpO system and decreased their preferences towards using it. On the other hand, the informed query formulation and submission process led to a significant reduction in query submissions.

This study continued to support our claim that benefits of generalized query previews can still exceed the overhead of the generalizations. The novice users spend more time on analyzing the distribution information, but as a result they make more informed choices during the query process, resulting in a drastic decrease in number of queries submitted to the system. They appreciate the benefits of the additional information in the bars and maintain their positive attitude towards generalized query previews.

However, this study also confirms our beliefs that the implementers of generalized query previews should be cautious about their application and user domain. Although the number of query submissions is significantly lower, the time to analyze a set of bars should be considered carefully before implementing an application. Generalized query previews can therefore be recommended for situations where users have to explore data and try to understand relationships, i.e., when the task is unclearly specified. Experience of users with computers should be taken into account. The bars that show the result set sizes are useful and facilitate efficient querying, even though it presents novice users a challenge for absorbing the distribution information. It is also important to note that, the generalized query previews approach was not meant to attack the issues that may appear in known-item searches, efficient formulation of general SQL queries, or improve the performance of the data experts who can query the data without any exploration.

6. Implementation issues

The generalized query previews user interface architecture utilizes a client–server approach for storing, computing, and transferring data (Fig. 22). 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 users' selection. 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 users' 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.

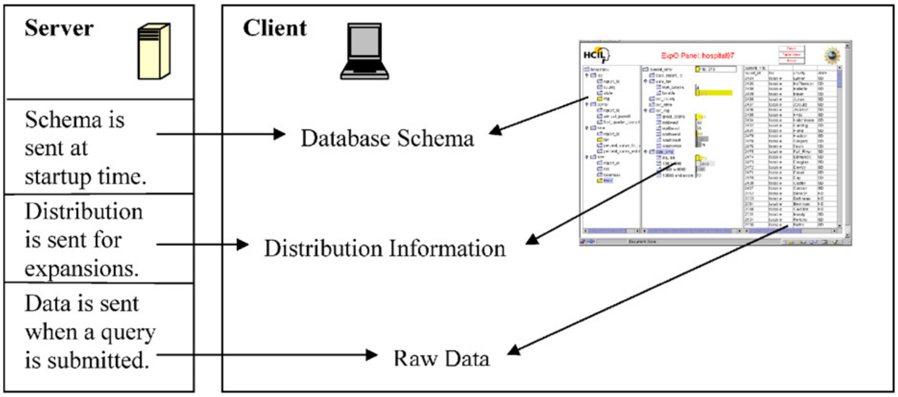

Server
Client
Schema is sent at startup time.
Database Schema
Distribution is sent for expansions.
Distribution Information
Data is sent when a query is submitted.
Raw Data
HCIL
ExpO Panel: hospital97

Fig. 22. The architecture for transferring data in generalized query previews, shown on the ExpO system.

The most critical piece of data that is transferred is the data distribution information, usually stored in multi-dimensional 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 the 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 lead to reduced freshness of the distribution information. If we want to keep only one multi-dimensional 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 are 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 could be needed. Keeping these efficient access paths up to date may gain importance, but may also be difficult. Many of these issues have been investigated by the database researchers in the realm of data warehouses and online analytical processing (OLAP) [11,12]. Finally, as a well-known issue in databases, joining multiple tables at runtime for answering a query with raw data gets difficult with the increasing number of tables, attributes, and selections.

Formally, lets 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 multi-dimensional 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 $i = 1, \ldots, l$ denoting the number of current attributes manipulated by the user. There will be a $\mathit{Combination}(k, l) + \mathit{Combination}(k, l-1) + \cdots + \mathit{Combination}(k, 1)$ number of these arrays where the combination operation is calculated as $\mathit{Combination}(k, l) = (k(k-1)\cdots 1)/((l(l-1)\cdots 1)((k-l)(k-l-1)\cdots 1))$.

There are two more challenges of the client–server architecture that require a 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 also without contradicting our initial design paradigm of not downloading/querying large 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 main 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 multi-dimensional 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 multi-dimensional 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 multi-dimensional 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 itself.

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 pair (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 for the histograms. Therefore, to accommodate multi-valued data types and hence attack the representation challenges, the multi-dimensional arrays must be enhanced. We introduced some approaches to address these challenges by creating various new data structures during our previous work [13]. Extensions to this work can also be found in [14,15].

7. Related work

Visual data mining and information visualization researchers have been working on effective visual methods for browsing and manipulating abstract information spaces [16–19]. Most of these methods rely on bar charts, scatter plots, and other means of similar visual explanations [20–23]. 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 and challenges of contemporary data sets and support exploration of large online data tables stored in relational databases.

Heppe et al. [24] created one of the first menu-driven information retrieval systems. They used volume previews and progressive query formulation to help users. Spence continued this line of work with his Attribute and Influence Explorers [25,26] 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 [27] 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 has become popular in online e-commerce applications (i.e., product selection) [28]. 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. A recent approach that is closely related to our previous work, query previews, is RB++ [29]. They use histograms in a similar fashion to our previous work to see an overview of the contents of a website. Flamenco is another recent system where users can smoothly move in a large information space over the web using faceted metadata [30]. Our work parallels many of 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 relational databases.

Table Lens [31] 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 [32] 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 [33] took a different approach to visualizing multi-dimensional tabular data. 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 on individual data rows, columns, and individual items of tabular data, making them less applicable to large online data.

A comprehensive approach, SDM [34], 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 [35] 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 [36]. Views are linked to each other and updates occur simultaneously. VisDB [37] 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 [38]. 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 [39] 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 users in a more compact fashion. They also did not extend to large online databases due to their close links to individual data items.

Another source of related work is from the research area of databases. For example, DataSplash [40] (formerly Tioga-2) is a database visualization environment 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 compact way to see previews. SeeData, a system for displaying the relational schema of a database [41], uses 2D bar charts to show relationships between thousands of relations. Polaris [42] 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. Hellerstein et al. [43] focuses on iterative methods like progressive sampling and targets the problem of visualizing large data tables. Marmotta, a querying system for networked databases [44], 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 are 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 [45]. More recent applications such as the Eureka database exploration system helps users continuously browse an Internet-based database [46]. 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.

8. Conclusions

Generalized query previews form a new user interface architecture for browsing large online databases. Using metadata (e.g., the distribution of data) for browsing and pruning raw data is an intriguing idea. Especially, when the data are 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 the two strategies introduced for defining (and confining) queries. With this work, we saw that using overviews and previews can enable efficient and intuitive browsing of large online data and can be a faster alternative to traditional approaches for accessing such types of data.

Our controlled experiments analyzed the application and user domains for generalized query previews, which proved the approach to be especially useful when users need to probe the data. We observed that there are significant advantages of our work for expert computer users while novice users may find it challenging to immediately absorb the distribution information. It would be interesting for future studies to analyze novice user behavior over time. Another interesting topic for future study is to compare metadata data preparation costs to the gains we observed on query submission counts.

For practitioners, implications of user experience with computers should be considered before deployment, especially when designing generalized query preview systems with many relations and attributes. Our work continues to disseminate academically as well as commercially (e.g., www.endeca.com). Data about these deployments would further clarify the application and user domains, i.e., when/where previewing-based systems can be successful.

We also list other interesting future directions for improving generalized query previews. 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, we also envision users exploring 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 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 [47] 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 pre-defined ones may form a promising idea. Users may want to set the border values for the buckets rather than using the pre-defined 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 web, with its large collection of 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 web search engines.

Acknowledgments

This work is supported in part by the United States Census Bureau, NSF, and NASA. We thank Dr. Catherine Plaisant for her contributions throughout this project.

References

[1] B. Shneiderman, D. Byrd, W.B. Croft, Clarifying search: a user-interface framework for text searches, D-Lib Magazine, http://www.dlib.org/dlib/january97/retrieval, 1997.

[2] C. Williamson, B. Shneiderman, The dynamic home finder: evaluating dynamic queries in a real-estate information exploration system, Proceedings of the ACM SIGIR'92 Conference, 1992, pp. 338–346.

[3] C. Ahlberg, B. Shneiderman, Visual information seeking: tight coupling of dynamic query filters with starfield displays, Proceedings of the ACM CHI'94 Conference, 1994, pp. 313–317.

[4] C. Ahlberg, E. Wistrand, IVEE: an information visualization and exploration environment, Proceedings of the IEEE Information Visualization Symposium, 1995, pp. 66–73.

[5] C. Ahlberg, C. Williamson, B. Shneiderman, Dynamic queries for information exploration: an implementation and evaluation, Proceedings of the ACM CHI'92 Conference, 1992, pp. 619–626.

[6] K. Doan, C. Plaisant, B. Shneiderman, Query previews in networked information systems, Proceedings of the Forum on Advances in Digital Libraries, IEEE Society Press, Silver Spring, MD, 1996, pp. 120–129.

[7] K. Doan, C. Plaisant, B. Shneiderman, T. Bruns, Query previews in networked information systems: a case study with NASA environmental data, ACM SIGMOD Record 26 (1) (1997) 75–81.

[8] C. Plaisant, T. Bruns, K. Doan, B. Shneiderman, Interface and data architecture for query previews in networked information systems, ACM Trans. Inform. Systems 17 (3) (1999) 320–341.

[9] S. Greene, E. Tanin, C. Plaisant, B. Shneiderman, L. Olsen, G. Major, S. Johns, The end of zero-hit queries: query previews for NASA's global change master directory, Int. J. Digital Libraries 2 (2) (1999) 79–90.

[10] E. Tanin, A. Lotem, I. Haddadin, B. Shneiderman, C. Plaisant, L. Slaughter, Facilitating data exploration with

query previews: a study of user performance and preference, Behav. Inform. Technol. 19 (6) (2000) 393–403.

[11] S. Chaudhuri, U. Dayal, An overview of data warehousing and OLAP technology, ACM SIGMOD Record 26 (1) (1997).

[12] J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, H. Pirahesh, Data cube: a relational aggregation operator generalizing group-by, cross-tab, and sub totals, J. Data Mining Knowledge Discovery 1 (1) (1997) 29–53.

[13] R. Beigel, E. Tanin, The geometry of browsing, Proceedings of the Third Latin American Conference on Theoretical Informatics, 1998, pp. 331–340.

[14] X. Lin, Q. Liu, Y. Yuan, X. Zhou, Multiscale histograms: summarizing topological relations in large spatial datasets, Proceedings of the 29th International Conference on Very Large Data Bases, 2003, pp. 814–825.

[15] C. Sun, D. Agrawal, A. El Abbadi, Exploring spatial datasets with histograms, Proceedings of the International Conference on Data Engineering, 2002, pp. 93–102.

[16] S. Card, J.D. Mackinlay, B. Shneiderman, Readings in Information Visualization Using Vision to Think, Morgan Kaufmann, San Francisco, CA, 1999.

[17] N. Gershon, S.G. Eick, Information visualization, IEEE Comput. Graphics Appl. 1997.

[18] M. Hearst, User interfaces and visualization, in: B.-Y. Ricardo, R.-N. Berthier (Eds.), Modern Information Retrieval, ACM Press, New York, 1999, pp. 257–323.

[19] B. Shneiderman, C. Plaisant, Designing the User Interface: Strategies for Effective Human–Computer Interaction, Fourth ed, Addison Wesley, Reading, MA, 2004.

[20] J. Bertin, Semiology of Graphics, University of Wisconsin Press, Madison, Wisconsin, 1983.

[21] E. Tufte, The Visual Display of Quantitative Information, Graphics Press, Cheshire, Connecticut, 1983.

[22] E. Tufte, Envisioning Information, Graphics Press, Cheshire, Connecticut, 1990.

[23] E. Tufte, Visual Explanations: Images and Quantities, Evidence and Narrative, Graphics Press, Cheshire, Connecticut, 1997.

[24] D. Heppe, W.S. Edmondson, R. Spence, Helping both the novice and advanced user in menu-driven information retrieval systems, in: Proceedings of the HCI '85 Conference, British Computer Society Press, 1985, pp. 92–101.

[25] L. Tweedie, R. Spence, D. Williams, R. Bhogal, The attribute explorer, Video Proceedings of ACM CHI'94 Conference, 1994, pp. 435–436.

[26] L. Tweedie, R. Spence, H. Dawkes, H. Su, Externalising abstract mathematical models, Proceedings of the ACM CHI'96 Conference, 1996, pp. 406–412.

[27] M.D. Williams, What makes RABBIT run?, Int. J. Man Machine Stud. 21 (1984) 333–335.

[28] K. Wittenburg, T. Lanning, M. Heinrichs, M. Stanton, Parallel bargrams for consumer-based information exploration and choice, Proceedings of the ACM Symposium on User Interface Systems and Technology, 2001.

[29] G. Marchionini, B. Brunk, Towards a general relation browser: a GUI for information architects, J. Digital Inform. 4(1) (2004) http://jodi.ecs.soton.ac.uk/Articles/v04/i01/Marchionini/

[30] M. Hearst, J. English, R. Sinha, K. Swearingen, P. Yee, Finding the flow in web site search, Commun. ACM 45 (9) (2002) 42–49.

[31] R. Rao, S. Card, The table lens: merging graphical and symbolic representations in an interactive focus + context visualization for tabular information, Proceedings of the ACM CHI'94 Conference, 1994, pp. 318–322.

[32] C. Beilken, M. Spenke, Visual, interactive data mining with infozoom—the medical data set, in: Proceedings of the Workshop on Discovery Challenge in Third European Conference on Principles and Practice of Knowledge Discovery in Databases, Prague, 1999, pp. 49–54.

[33] A. Inselberg, B. Dimsdale, Visualizing multi-variate relations with parallel coordinates, Proceedings of the Third International Conference on Human–Computer Interaction, 1989, pp. 460–467.

[34] M.C. Chuah, S.F. Roth, J. Mattis, J. Kolojejchick, SDM: selective dynamic manipulation of visualizations, Proceedings of the ACM UIST'95 Conference, 1995, pp. 61–70.

[35] S.F. Roth, P. Lucas, J.A. Senn, C.C. Gomberg, M.B. Burks, P.J. Stroffolino, J.A. Kolojejchick, C. Dunmire, Visage: a user interface environment for exploring information, Proceedings of the IEEE Information Visualization Symposium, 1996, pp. 3–12.

[36] M. Livny, R. Ramakrishnan, K. Beyer, G. Chen, D. Donjerkovic, S. Lawande, J. Myllymaki, K. Wenger, DEVise: integrated querying and visual exploration of large datasets, Proceedings of the ACM SIGMOD'97, 1997.

[37] D.A. Keim, H.P. Kriegel, VisDB: database explorations using multidimensional visualization, IEEE Computer Graphics Appl. 1994, pp. 40–49.

[38] B.G. Becker, Volume rendering for relational data, Proceedings of the IEEE Information Visualization Symposium, 1997, pp. 87–90.

[39] J. Goldstein, S.F. Roth, Using aggregation and dynamic queries for exploring large data sets, Proceedings of the ACM CHI'94 Conference, 1994, pp. 23–29.

[40] A. Woodruff, C. Olston, A. Aiken, M. Chu, V. Ercegovac, M. Lin, M. Spalding, M. Stonebraker, DataSplash: a direct manipulation environment for programming semantic zoom visualizations of tabular data, J. Visual Languages Comput. 12 (5) (2001) 551–571.

[41] J. Antis, S.G. Eick, J. Pyrce, Visualizing the structure of relational databases, IEEE Software 13 (1) (1996) 72–79.

[42] C. Stolte, P. Hanrahan, Polaris: a system for query, analysis, and visualization of multi-dimensional relational databases, Proceedings of the IEEE Information Visualization Symposium, 2000.

[43] J.M. Hellerstein, R. Avnur, A. Chou, C. Hidber, C. Olston, V. Raman, T. Roth, Interactive data analysis: the control project, IEEE Comput. 32 (8) (1999) 51–59.

[44] F. Capobianco, M. Mosconi, L. Pagnin, Progressive HTTP-based querying of remote databases within the Marmotta Iconic VQS, Proceedings of the IEEE Information Visualization Symposium, 1995, pp. 122–125.

[45] S. Pollitt, Interactive information retrieval based on faceted classification using views, in: Proceedings of the Sixth International Study Conference on Classification, University College, London, 1997.

[46] J. Shafer, R. Agrawal, Continuous querying in database-centric web applications, in: Proceedings of the 9th International World Wide Web Conference, Amsterdam, May 2000.

[47] B. Shneiderman, Dynamic queries for visual information seeking, IEEE Software 11 (6) (1994) 70–77.