Interface and Data Architecture for Query Preview in Networked Information Systems

CATHERINE PLAISANT, BEN SHNEIDERMAN, KHOA DOAN, and TOM BRUNS

Human-Computer Interaction Laboratory
University of Maryland Institute for Advanced Computer Studies

There are numerous problems associated with formulating queries on networked information systems. These include increased data volume and complexity, accompanied by slow network access. This article proposes a new approach to a network query user interfaces that consists of two phases: query preview and query refinement. This new approach is based on the concepts of dynamic queries and query previews, which guides users in rapidly and dynamically eliminating undesired records, reducing the data volume to a manageable size, and refining queries locally before submission over a network. Examples of two applications are given: a Restaurant Finder and a prototype for NASA's Earth Observing Systems Data Information Systems (EOSDIS). Data architecture is discussed, and user feedback is presented.

Categories and Subject Descriptors: H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Query formulation*; H.5.2 [**Information Interfaces and Presentation**]: User Interfaces

General Terms: Design, Human Factors

Additional Key Words and Phrases: Direct manipulation, EOSDIS, dynamic query, graphical user interface, query preview, query refinement, science data

This work is supported in part by NASA (NAG 52895 and NAGW 2777) and by the NSF grants NSF EEC 94-02384 and NSF IRI 96-15534.

Authors' addresses: Catherine Plaisant and B. Shneiderman, Human-Computer Interaction Laboratory, University of Maryland Institute for Advanced Computer Studies, College Park, MD 20742; email: plaisant @cs.umd.edu; ben@cs.umd.edu; http://www.cs.umd.edu/projects/hcil; K. Doan, Raytheon Corporation, 4500 Forbes Boulevard, Lanham, MD 20706; email: kdoan@killians.gsfc.nasa.gov; T. Bruns, Delorme Mapping, P. O. Box 298, Lower Main Street, Freeport, ME 04032; email: tbruns@delorme.com.

1. INTRODUCTION

The exploration of networked information resources becomes increasingly difficult as the volume of data grows. We identified at least the following problems of information retrieval in networked environments:

—*Data Volume:* The amount of data available is rapidly increasing. For example, some sensor data in NASA's Earth Observing Systems is growing at the rate of gigabytes per day. Organizing and indexing the volume of new records is difficult. Since many users seek specific records, a rapid way to focus on information of interest is needed.

—*Data Diversity:* Data come in a variety of forms, such as text, image, audio, video, or combinations of these. Some formats are application specific, making it difficult for search and retrieval tools to identify and categorize them.

—*Slow Network Access:* Slow network access is a well-known problem of information retrieval in networked environments. When network traffic is high, data transmission rates deteriorate. Therefore, user task completion is accelerated if the number of network accesses is reduced.

In this article, we present a user interface to support efficient query formulation for networked information systems using dynamic queries and query previews.

Dynamic queries are an extension of graphical query interfaces based on aggregation/generalization hierarchies [Weiland and Shneiderman 1993; Shneiderman 1994]. Dynamic query user interfaces apply the principles of direct manipulation and imply

—visual representation of the query

—visual representation of the results

—rapid, incremental, and reversible control of the query

—selection by pointing, not typing, and

—immediate and continuous feedback

Dynamic queries involve the interactive control by users of visual query parameters that generate rapid, animated, and visual displays of database search results. As users adjust sliders or buttons, results are updated rapidly (within 100 milliseconds).

The enthusiasm users have for query previews emanates from the sense of control they gain over the query. Empirical results have shown that dynamic queries are effective for novice and expert users to find trends and spot exceptions [Ahlberg et al. 1992; Williamson and Shneiderman 1992; Tanin et al. 1997].

Early implementations of dynamic queries used relatively small files of a few thousand records. They required the data to be stored in memory to guarantee rapid update of the display. We developed algorithms and data

structures that allow larger files to be handled (up to 100,000 records) [Tanin et al. 1996], but slow network performance and limited local memory are obstacles when trying to use dynamic queries for large distributed databases.

Query previews offer a solution to this problem. We describe a simple example of query previews, the Restaurant Finder, to illustrate the basic principles. Then the two-phase query formulation process and a system architecture are presented. A dynamic query user interface prototype for NASA's EOSDIS (Earth Observing Systems Data Information Systems) is used to show how this approach has been applied. Evaluations from expert reviews and a controlled experiment are reported. Finally, related work and conclusions are presented.

2. QUERY PREVIEWS

Traditionally, there are two strategies for information seekers to obtain data from large information systems [Marchionini 1995]. Analytical strategies depend on careful planning, recall of query terms, iterative query formulation, and examination of results. Browsing strategies depend on user recognition of relevant information, and therefore they are heuristic and opportunistic. Analytical strategies require users to have a good knowledge of the application domain and be skillful in reasoning. Browsing strategies require less knowledge, but can be difficult when the volume of data is large.

Keyword-oriented or form-based interfaces are widely used for formulating queries on networked information systems. They often generate zero-hit queries, or query results that contain a large number of results which users must browse. Users can limit how many results a query returns (e.g., 20) to limit the duration of the search, but it is impossible to estimate how much data was not returned, and how representative of the entire search space the results are. Users also often fail to find data if appropriate keywords cannot be guessed.

Query previews combine browsing and querying. Summary data (such as the number of records for each attribute value) guide users to narrow the scope of their queries. The summary data, which varies with the database and application, provides an overview of the database from several perspectives. It is generally orders of magnitude smaller than the database itself, and can be downloaded quickly to drive a dynamic query interface locally on the user's machine. Therefore, query previews support a dynamic query user interface where the visual display of the summary is updated in real time in response to users' selections. Users can rapidly reduce the number of records to a manageable size.

Query previews empower users to perform more complex searches by using visual strategies, and they have many advantages:

—reduce zero-hit queries

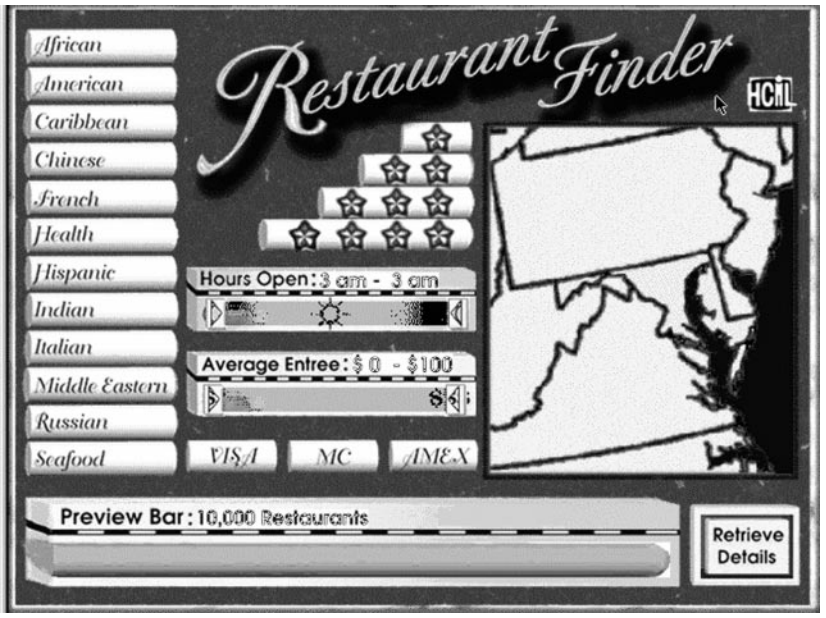

Fig. 1(a). Restaurant Finder. Users can choose an area on the map and make choices with buttons and sliders.

—reduce network activity and browsing effort by preventing the retrieval of undesired records

—represent statistical information of the database visually to aid comprehension and exploration

—support dynamic queries, which aids users to discover database patterns and exceptions

—suitable to novice, intermittent, or frequent users

3. A SIMPLE EXAMPLE OF QUERY PREVIEW: THE RESTAURANT FINDER

The Restaurant Finder (Figures 1(a) and 1(b)) illustrates the concept of visual interaction with summary data, the essence of dynamic query previews. The Restaurant Finder is designed to help users identify restaurants that match certain criteria. Users first specify criteria of the restaurants they want, such as type of food or price range. This reduces the number of selected restaurants to a more manageable size (Figure 1(b)). The request is then submitted to the network, which retrieves more data on the selected restaurants. Users can then continue to refine their queries with additional, more specific criteria.

Consider a database of 10,000 restaurants in the mid-Atlantic region. The Restaurant Finder's user interface provides sliders and buttons for selecting desired cuisine, range of cost, range of hours, geographic regions,

Fig. 1(b). Restaurant Finder. The user has now selected two cuisine types, a price range, and a geographical area, reducing the number of restaurant to review to 95 as shown on the result bar which is updated continuously as users adjust their queries.

rating, and accepted charge cards. As selections are made, the result bar shown at the bottom of the screen changes length proportionally to the number of selected restaurants that satisfy the users' selection (possibly thousands of restaurants). Zero-hit queries are eliminated: users can quickly see if there are any Chinese restaurants open after midnight, and they will rapidly realize that there are no cheap French restaurants in Washington, DC. Database distributions are visible: users may discover that there are more Chinese restaurants than Italian restaurants, but that more Italian restaurants are open after midnight. In the query preview, only summary data are downloaded from the network, allowing real-time interaction and eliminating network delays until a useful subset of the data has been identified. Then more details will be downloaded from the network about this subset (e.g., geographical location indicated on a zoomable local map, data for parking availability, number of seats, or handicapped access) to allow users to refine their query. Finally users can click on individual restaurants and review menus and directions to make the final selection.

4. MAIN EXAMPLE AND PROTOTYPE: THE CASE OF EOSDIS

We use the NASA's Earth Observing System Data Information System (EOSDIS) to illustrate our two-phase query preview approach.

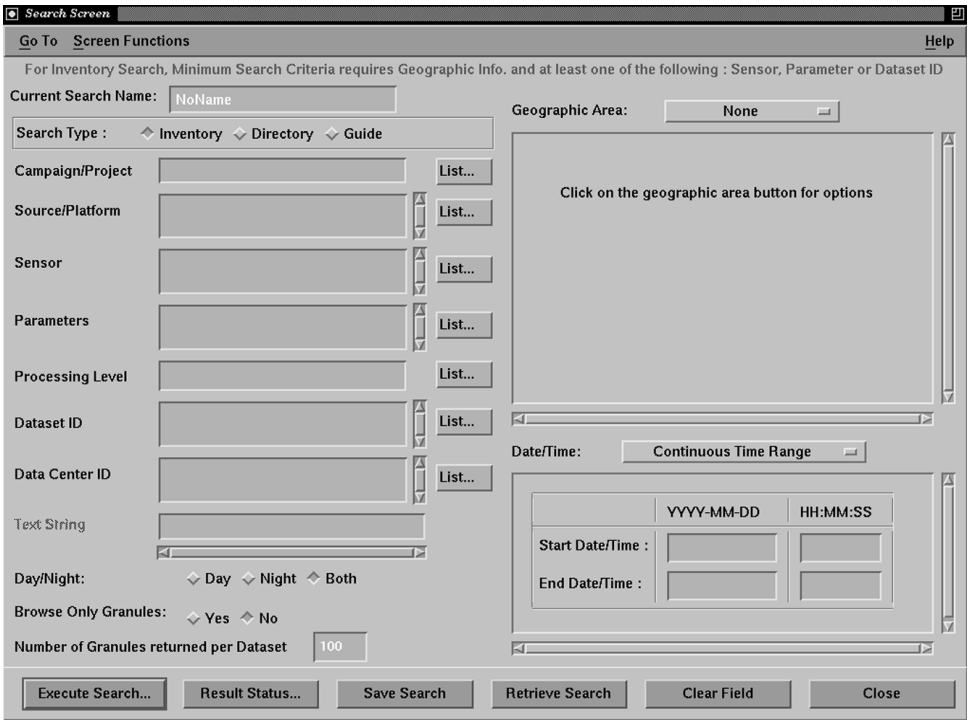

Fig. 2. Classic form fill-in interfaces for EOSDIS (Figure 2) permit searches of the already large holdings, but zero-hit queries are a problem; and it is difficult to estimate how much data are available on a given topic.

4.1 EOSDIS Science Data

Diverse users (scientists, teachers, students, etc.) can retrieve earth science data from hundreds of thousands of datasets. Datasets, named collections of data with authoritative metadata, contain pictures, measurements, or processed data, from nine data centers around the country. Standard EOSDIS metadata include spatial coverage, time coverage, type of data, sensor type, campaign name, level of processing, etc. Classic form fill-in interfaces for EOSDIS (Figure 2) permit searches of the already large holdings, but zero-hit queries are a problem and it is difficult to estimate how much data are available on a given topic and what to do to increase or reduce the result set.

4.2 Prototypes

An early version of our two-phase approach was implemented in Visual Basic [Doan et al. 1996]. Then a more complete prototype was implemented in Tcl/Tk (available in video [Plaisant et al. 1997a]), and more recently a working Java implementation was prepared on the World Wide Web (WWW). The interface consists of two phases: query preview and query refinement.

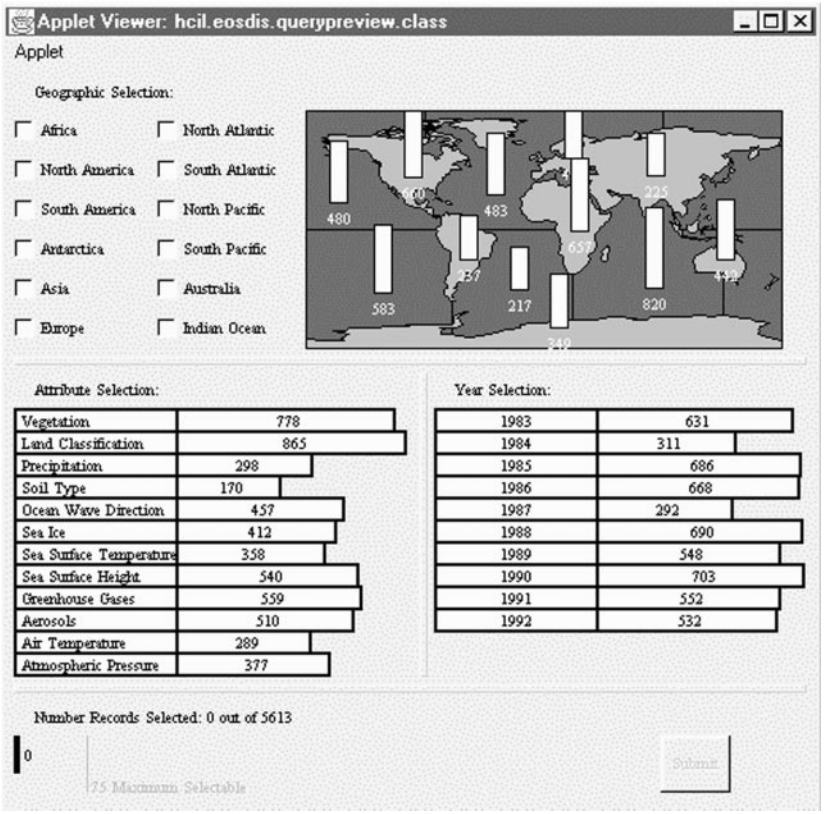

Fig. 3(a). The query preview screen displays summary data on preview bars. Users learn about the holdings of the collection and can make selections over a few parameters (here geographic, environmental parameter, and year).

4.3 EOSDIS Query Preview

In the query preview (Figure 3), users select rough ranges for three attributes: geographical location (a world map with 12 regions is shown at the top of the screen), parameters (a menu list of parameters such as vegetation, land classification, or precipitation), and temporal coverage (in the lower right). The spatial coverage of datasets is generalized into continents and oceans. The temporal coverage is defined by discrete years.

The number of datasets for each parameter, region, and year is shown on *preview bars*. The length of the preview bars is proportional to the number of the datasets containing data corresponding to the attribute value. At a glance users can see that the datasets seem to cover all areas of the globe, but there is more data on North America than South America. Users can also see that parameters and years are covered relatively uniformly in this hypothetical EOSDIS dataset collection. The result preview bar, at the bottom of the interface, displays the total number of datasets.

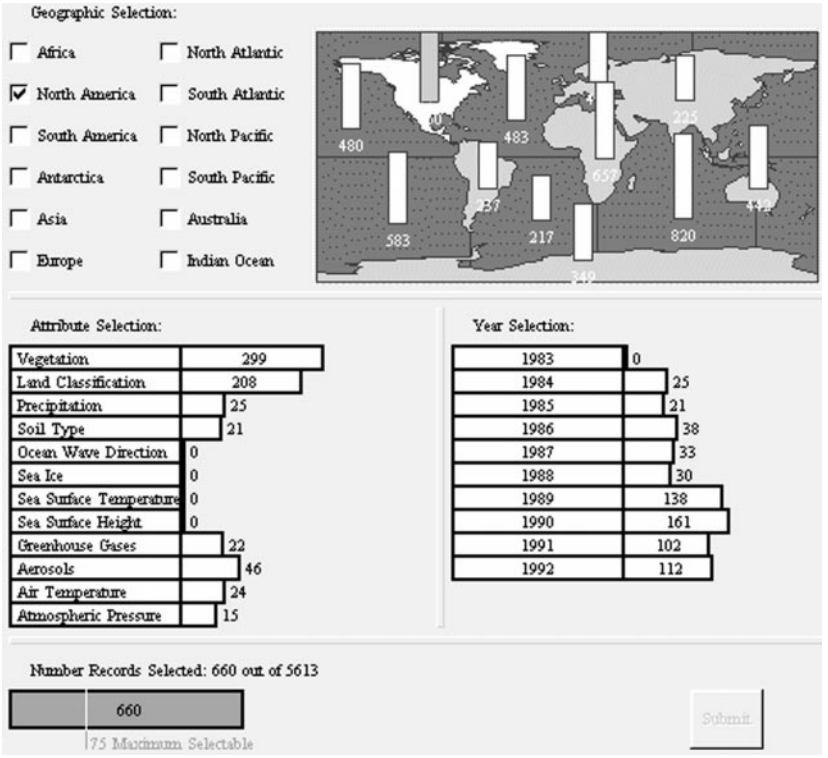

Fig. 3(b). The query preview screen displays summary data on preview bars. Users learn about the holdings of the collection and can make selections over a few parameters (geographic, environmental parameter and year). Here the user has selected North America, and all preview bars are updated (using a logarithmic scale). Selections apprear bright yellow.

Only rough queries are possible since the spatial coverage of datasets is generalized into continents and oceans, while the temporal coverage is defined by discrete years.

A query is formulated by selecting attribute values. As each value is selected, the preview bars in the other attribute groups adjust to reflect the number of datasets available. For example, users might be interested only in datasets that contain data for North America, which are selected by clicking on the North America checkbox (left of the map) or by clicking on the image of North America on the map. All the preview bars change in a fraction of a second (see Figure 3(b)) to reflect the distribution of datasets for North America only. The result preview bar at the bottom changes size to indicate the number of datasets for North America (660 in this example).

Users continue to define a query by selecting from other attribute value groups. In this example, users pick the two largest attribute values for North America, "Vegetation," and "Land Classification" (see Figures 3(b) and 3(c)). The preview bars in the spatial and year attribute value groups adjust to reflect the new query.

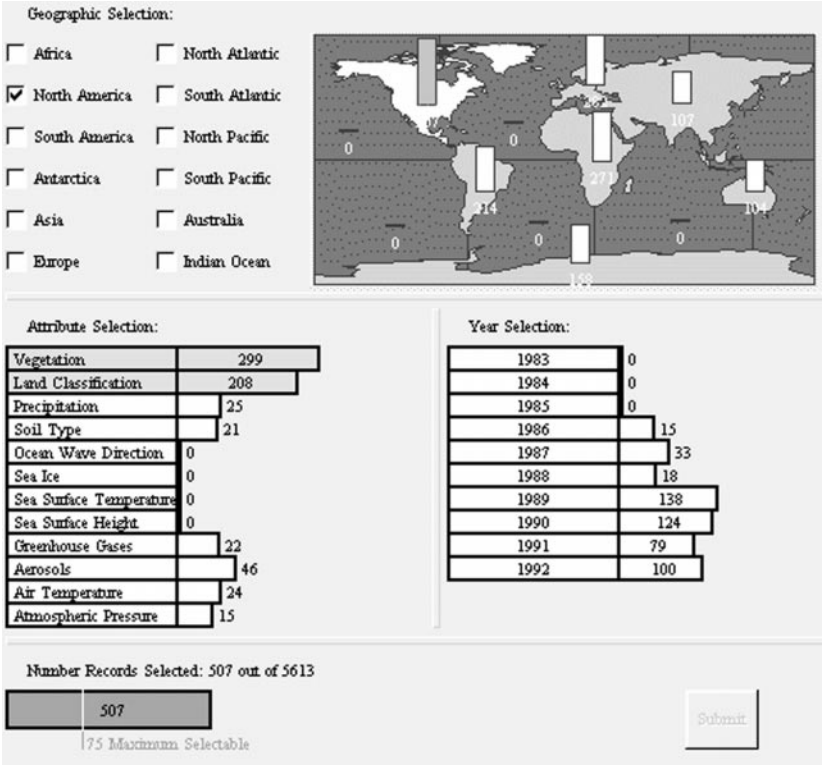

Fig. 3(c). Vegetation and Land Classification are now selected. The preview bars show which years have data.

The OR operation is used within attribute value groups, the AND operation between attribute value groups [Weiland and Shneiderman 1993]. Those AND/OR operations are made visible by the behavior of the bars which become smaller when an attribute value is specified for the first time (e.g., picking the first year) while becoming longer when additional values are added for a given attribute (e.g., when more years are added). This conjunction of disjunctions design handles many queries conveniently and allows rapid exploration that reduces the need for some more complex boolean queries [Weiland and Shneiderman 1993; Young and Shneiderman 1993].

Users further reduce the number of selected datasets by choosing specific years, in the example 1986, 1987, and 1988, three years which have data as shown on the preview bar (Figure 3(d)). These selections change the number of datasets in the other attibute value groups, and the preview bars are updated.

When the "Submit" button is pressed the rough query is submitted to the EOSDIS search engine, and the metadata of the datasets that satisfy the query are downloaded for the query refinement phase. In the example the query preview phase narrowed the search to 66 datasets.

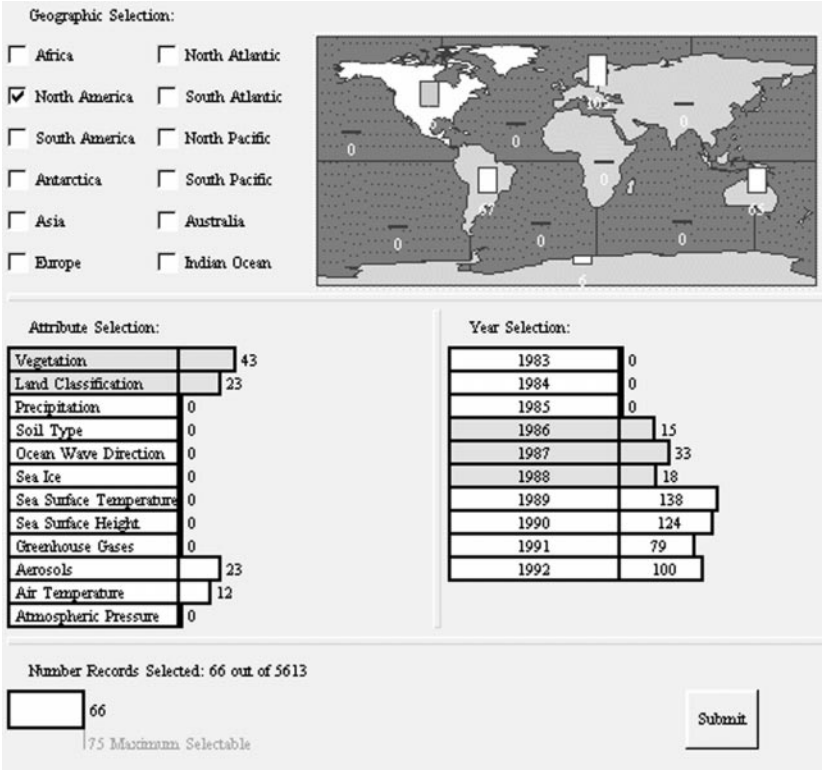

Fig. 3(d). Three years (1986, 1987, and 1988) have been selected. The result bar shows that an estimated 66 datasets will satisfy this query (the scale is logarithmic). The query can now be submitted.

4.4 EOSDIS Query Refinement

The query refinement interface supports dynamic queries over the metadata, i.e., over all the attributes of the datasets. These include the detailed spatial extent and temporal interval, parameters measured in the dataset, the sensor used to generate the dataset, the platform on which the sensor resides, the project with which the platform is associated, the data archive center where the data are stored, and the data-processing level which indicates raw sensor data (level 0) to highly processed data (level 4).

A temporal overview of the datasets is given in the top left (Figure 4(a)). Each dataset is now individually represented by a selectable line. Controls are provided to select values for the common attributes: the data archive center, project, platform, sensor, and data-processing level. Beside those common attributes additional attributes can be included in the metadata, but since the number of attributes may be large, menu access needs to be provided for those less common attributes. At the bottom of the screen a table lists all the datasets and gives exact values for the attributes.

In the refinement phase of the query, users can select precise values for the attributes. The map, already zoomed to the area selected in the query

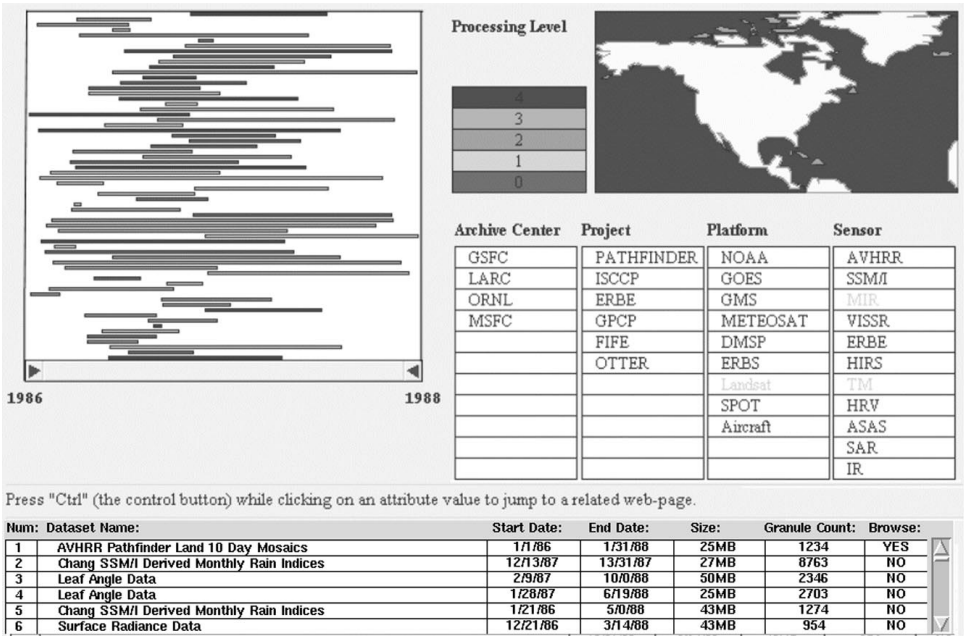

Fig. 4(a). In the query refinement users can browse all the information about individual datasets. The result set can be narrowed again by making more precise selections on more attributes.

preview, should be zoomable to allow precise selection. The time line of the overview, already narrowed to the years selected in the query preview, can be rescaled to specify narrower periods of interest.

In this second dynamic query interface the result of the query is immediately visualized on the overview. As attribute values are selected the number of lines on the overview changes to reflect the query in a few milliseconds, since there is no access to the network.

All controls are tightly coupled to do the following:

—*Describe selected datasets:* When users click on a dataset of the timeline overview, the corresponding attribute values are highlighted on all controls, e.g., the sensor is highlighted, the spatial coverage shown on the map, the row of the dataset table is highlighted and scrolled to the front if needed (Figure 4(b)).

—*Indicate valid values:* Once some attribute values have been selected, controls can reflect the now invalid values by graying them out (e.g., selecting a platform will most likely eliminate some of the sensors which will become grayed out). This can be achieved by analyzing the metadata of the datasets.

In Figure 4(c) the number of datasets was reduced by selecting the processing levels 2 and 3, two archive centers, and three projects. More details about a dataset such as descriptive information and sample data

Archive Center	Project	Platform	Sensor
GSFC	PATHFINDER	NOAA	AVHRR
LARC	ISCCP	GOES	SSM/I
ORNL	ERBE	GMS	MIR
MSFC	GPCP	METEOSAT	VISSR
	FIFE	DMSP	ERBE
	OTTER	ERBS	HIRS
		Landsat	TM
		SPOT	HRV
		Aircraft	ASAS
			SAR
			IR

Fig. 4(b). Partial screen showing highlighted parameter values corresponding to a dataset selected on the timeline overview.

can be retrieved on demand from the network before the decision to download a full dataset is made. The Java implementation also illustrates the benefit of the World Wide Web by allowing interface objects to act as links to relevant WWW information sources. For example, each platform name is linked to a NASA page describing that platform.

5. SYSTEM ARCHITECTURE

The architecture supporting the two-phase query formulation consists of three layers: interface, local storage, and network (Figure 5).

At the interface layer, users formulate and refine queries as described above. The query preview and query refinement interfaces provide a visual representation of the preview statistics, selected datasets, and query parameters.

The local storage layer maintains the data used to drive the dynamic query interfaces of the interface layer. These data consist of a *volume preview table* (summary data that indicate the number of datasets for each attribute value and intersections) for the query preview and dataset metadata for the query refinement. When users initiate a query preview session, the volume preview table is downloaded from the network databases.

The network layer is where the network activities take place. These network activities include updating the volume preview tables, providing the metadata for datasets selected from a query preview, and retrieving the details of a dataset selected in the query refinement.

5.1 Volume Preview Table

The size and dimensionality of the volume preview table is a function of the number of preview attributes and the number of discrete preview values for

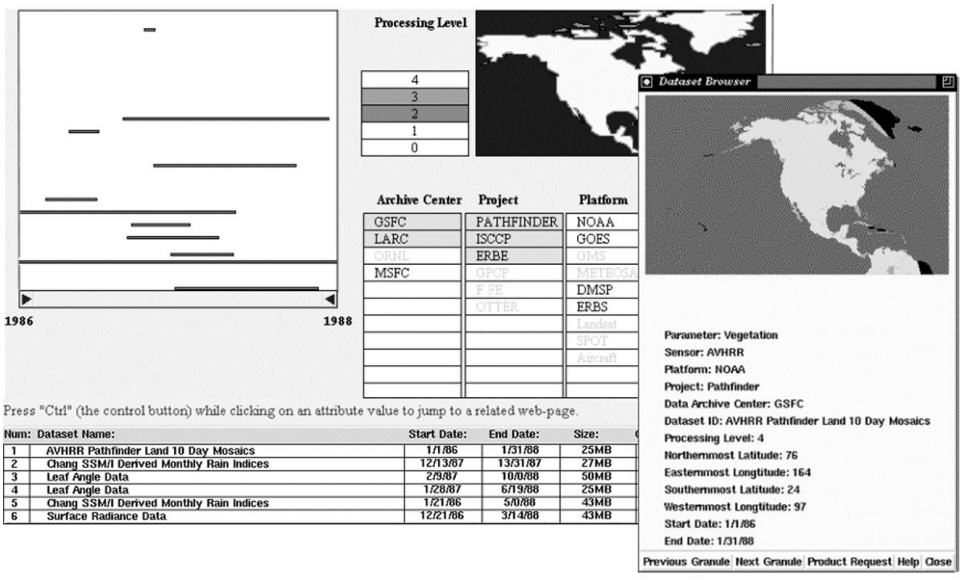

Fig. 4(c). Here the query has been refined by selecting two archive centers, three projects, and two processing levels. More filtering could be done by zooming on the timeline or on the map. The timeline overview and the dataset table reflect the remaining datasets. Details and samples images can be downloaded from the network (window on the right) before the long process of ordering the large datasets.

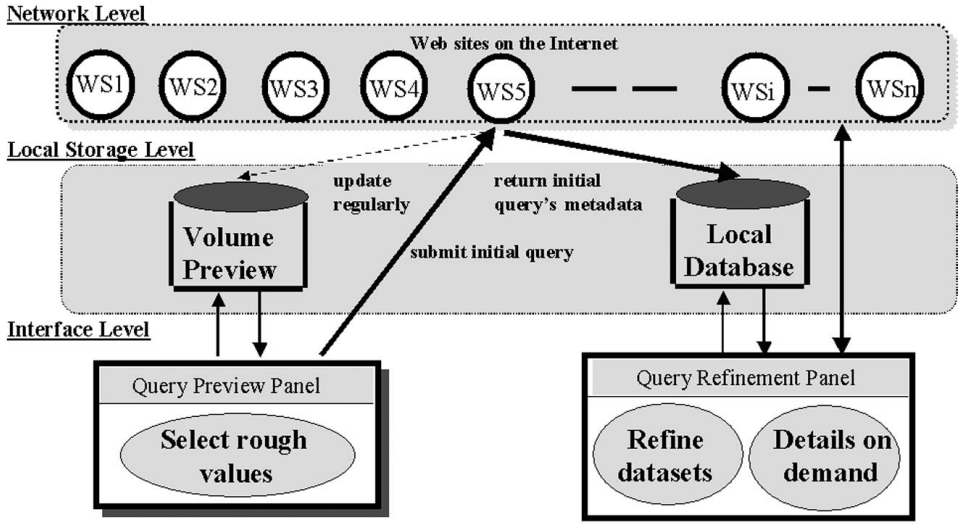

Fig. 5. Architecture of two-phase dynamic query approach for networked information systems.

each attribute. Consider a Restaurant Finder with three preview attributes: cuisine type, rating, and accepted credit cards. Imagine five types

Table I. A Slice of the Volume Preview Table for an Example Restaurant Finder. This 2D table results from specifying one of three preview attributes. In this case, the third attribute, rating, has been specified. This table is used to update preview bars in the query preview interface.

	French	Mexican	American	Indian	Italian
None	18	9	5	6	8
Visa	45	22	40	34	23
MC	12	56	40	23	12
Visa & MC	80	90	120	98	160

of cuisine, four ratings, and two acceptable credit cards. In the simplifying case where each restaurant's attribute can only take a single value the volume preview table would be a five-by-four-by-two table, with a total of 40 combinations. But in our example of the Restaurant Finder, allowable credit cards may be grouped. The cells of the volume preview table must be independent so that there must be cells for each possible combination of credit cards. Two credit cards create four possible combinations (including neither being acceptable), so the volume preview table has five-by-four-by-four or 80 combinations. Each cell in the table (i.e., each attribute value combination) holds an integer representing the number of restaurants in the database for that particular combination. In Table I, corresponding to the "three-star rated" restaurants, the cell for the 3-star Indian restaurants that accept Visa and MasterCard holds the value 98. Such tables are used to update widgets in the query preview interface.

N preview attributes, yield an N-dimensional volume preview table. The total size of the table is many orders of magnitude smaller than the size of the database, or the size of the datasets' metadata. Furthermore, the volume preview table does not change size as the database grows. Even if the database has billions of records, the size of the volume preview table allows it to be loaded into local high-speed storage to support dynamic queries in the query preview phase.

5.2 Controlling the Size of the Table

Nevertheless, the number of attributes and the number of the possible values needs to be carefully chosen if the objects being searched (e.g., restaurants or datasets) can take any combinations of values for their attributes. In the simple case of the Restaurant Finder, each restaurant could have a combination of credit cards. The interface widget only had two buttons for credit cards, but the volume preview table needed four rows to represent the combinations. In the case of EOSDIS a given dataset can contain measurements of several parameters, covering several areas over several years. In the worst case (i.e., if all combinations are possible) the size of the preview table could become $2^{12} \times 2^{12} \times 2^{10}$ (for 12 areas, 12 parameters, and 10 time periods) which would lead to megabytes of data, much too large to load over the network and use in the preview.

A first solution is to ignore in some way the possible combinations and count twice the datasets that have two parameters, once in each cell for

each parameter it contains. This will result in correct individual preview bars (e.g., the preview bar for 1990 really gives the total number of datasets that have any data for that year) but inflate total result preview bar since some datasets are counted multiple times. This might be acceptable if combinations are a small proportion of the data, which is likely to be common because of the high granularity of the selections in the query preview.

A second, more accurate solution to the problem is to analyze the number of combinations, either by looking at the type of attribute (e.g., year combinations are typically year ranges, reducing the number of combinations to 55 instead of 1024 for 10 values), or the distribution of the data itself (e.g., EOSDIS parameters are grouped into only a limited number of compatible combinations).

The first solution has the advantage of keeping the size of the volume preview very small (e.g., $12 \times 12 \times 10$ integers for our EOSDIS prototype, i.e., much smaller than the world map graphic); the second gives a more accurate preview but requires more time and space. In our early prototype we chose to simply duplicate datasets because we did not have access to large amounts of real EOSDIS metadata. The attributes were arbitrarily selected. In our operational prototype for the Global Master Change Directory, we used a hybrid solution where the list of record IDs is kept with the counts, so that all duplicates can be removed to calculate the totals accurately, but this does not scale up very far. Recently, we have begun to deal with the challenge of scaling up the software architecture to accommodate much larger and more varied data collections. In particular, we have implemented several techniques that deal with multivalued attribute data [Plaisant et al. 1999].

To summarize, volume preview tables can become large if combinations are to be previewed accurately or if large numbers of previewing attributes or attribute values are chosen. But the query preview technique can always be tailored by reducing the number of attributes or attribute values in the query preview. The size of the preview table can also be adapted to the users' work environment (network speed, workstation type) or preferences.

5.3 Updating the Volume Preview Table

Since the data of the networked information system changes regularly, volume preview tables have to be updated. Our approach depends on the data providers being willing and able to produce and publish volume preview tables on a regular basis (weekly, daily, or hourly, depending on the application), or on third-party businesses running series of queries to build the tables. Since the preview is only meant to enter rough queries it may be acceptable to use slightly out-of-date volume preview tables. The query preview needs to make clear that the preview bar sizes are an approximation of the real volume and give the "age" of the information used. When the rough query is submitted, the (up-to-date) databases are queried and will return up-to-date data for the query refinement. At this

point the number of datasets returned might be slightly different than predicted by the query preview. This might be a problem when the query preview predicts zero hits while a new dataset that would answer the query has just been added to EOSDIS. This risk has to be evaluated and adequate scheduling of the updates enforced. The Cubetree implementation of datacubes [Roussopolos et al. 1997] seems a promising data structure, as it has efficient query updating.

5.4 Limiting the Download of Metadata

Most users and data center staff will want to limit preview requests to those whose result set is small. The submit button can be disabled when the result set size is above a recommended level (75 in our early prototype).

6. LIMITATIONS OF THE CURRENT EOSDIS PROTOTYPE

The present implementation of the query refinement interface has several limitations. The implementation of the query refinement overview will not scale up well when more than 100 datasets are returned from the query preview. The timeline of intervals will get too tall and occupy too much screen space if intervals are not allowed to overlap. Better methods of handling large numbers of intervals are needed. Possible directions include zooming, optimizing the line packing to make use of screen space, or using line thickness to indicate overlaps. The quantitative and qualitative overview of the large number of datasets is needed to monitor their filtering, but the ability to select individual lines is important when numbers have decreased enough to require browsing of individual datasets.

In our EOSDIS prototype the zooming and panning of the overview have no filtering effect, but we have implemented other examples which demonstrate the benefit of the technique (e.g., for the Library of Congress historical special collections browsing [Plaisant et al. 1997b]). Similarly the filtering by geographical location has not been developed yet in the query refinement. Zooming and selecting rectangular areas is easy, but more sophisticated selection mechanisms used in geographical information systems are probably necessary.

The query preview allows users to specify the most common boolean queries (OR within attributes and AND between attributes). This is appropriate, since the query preview is only meant to be a rough query, but more precise control over the boolean combinations needs to be provided in the query refinement. Our current prototype does not offer such capability. Menu options can be provided to change the "behavior" of widgets, or graphical tools can be provided to allow boolean combination of the widgets [Young and Shneiderman 1993].

7. EVALUATION AND USER FEEDBACK

7.1 Expert User Review

The prototype dynamic query preview interface was presented to subjects as part of a Prototyping Workshop organized by Hughes Applied Informa-

tion Systems (HAIS) in Landover, MD [Poston 1996]. A dozen NASA earth scientists who use EOSDIS to extract data for their research participated in the evaluation and reviewed several querying interfaces during the day.

The hands-on review of our prototype lasted about an hour and a half. Groups were formed with two or three evaluators and an observer/note-taker in each group. They received no training but were given five directions or starting points to explore the prototype. For example, one direction was to "Examine the relationship between the map at the top and the data shown on the bottom half of the window. Try selecting a geographic region and various attributes. How are the data displayed." Evaluators were encouraged to "think aloud" during the session, and their comments and suggestions were recorded.

The 12 professionals reacted positively to the new concepts in the query preview and query refinement interfaces. They agreed that the visual feedback provided in the query preview interface allows users to understand the distribution of datasets. A group of evaluators recommended that it would be an effective tool for subjects who did not know what data were available. Others remarked that some users would not even need to go to the refinement phase, as they would realize immediately that no data were available for them. The query preview interface was said to "allow users to select data, see relationships among data, and explore available resources."

Subjects said that they appreciated the time interval overview concept and liked to be able to select or deselect and see the changes in the overview. Subjects felt that the prototype "led the user" and that it was "an intuitive way to search data." Some users suggested that the map regions and selectable attributes be customizable so that users could interact with information in which they are interested (different specialties may require different query preview attributes). At the time of the test the prototype was set to perform an AND operation within an attribute. This meant that clicking on 1991 just after a click on 1990 would result in all the bars being shorter (since it had restricted to the datasets which had data about 1990 AND 1991). After some confusion, all groups of evaluators were able to figure out that an AND was being performed by seeing the bars grow or shrink. But it was clear that they had expected the interface to perform an OR within an attribute (i.e., retrieving all datasets having data from 1990 or 1991). This was an important change made to the prototype following the evaluation. This anecdote confirms that the visual feedback helped users understand the operations performed by the sytem.

After the evaluation, subjects were given a questionnaire and rated the interface positively. For a complete list of subject comments and questionnaire results, see Poston [1996].

7.2 Controlled Experiment

Twelve computer science students searched a database of films with a form fill-in interface. They were given only 10 minutes of training in the use of the interfaces. The experimental treatments in this counterbalanced with-

in-subjects design were presence or absence of a query preview [Tanin et al. 1997]. The tasks simulated a complex browsing situation such as "Find a PG-13 musical which was produced between 1991 and 1995, if no such film is available, find a war film from the same years with the same rating, if not, try a musical or a war film from 1970–91, and as the last possibility, try a comedy from 1970–95."

The query preview treatment showed whether or not there were any films satisfying the requirements, allowing subjects to rapidly explore alternatives. In the experiment, there were no lengthy network delays, so the time differences would be much larger if there were delays. Subjects using the query preview took an average of only 36.2 seconds while others took 57.5 seconds ($p < 0.05$) for tasks in which the query preview attributes were partially relevant. Stronger results, 24.4 seconds versus 51.2 seconds ($p < 0.05$), were obtained when the tasks closely matched the query preview attributes. This dramatic doubling of speed for query previews is a strong indication of its benefits, which will be even greater in the case of network delays. For tasks in which there was no match with the query preview attributes, there was only a 10% slowdown in performance.

Subjective satisfaction was statistically significantly higher for the query preview users, who rated the query preview interfaces higher on five questions: helpful? faster? enlightening? enjoyable? use it again? Subjects also made useful suggested improvements such as rapid ways to reset the query preview.

8. SUMMARY AND DISCUSSION

The two examples we described illustrate a query formulation process for a networked information system consisting of two phases: query preview and query refinement.

8.1 Query Preview

In the query preview phase, users form a rough query by selecting rough values over a small number of attributes. The scope of the query is large, but the resolution is limited (see Figure 7). Summary data are maintained for each of the query preview attributes and intersections. The total number of items selected by the user's query is visualized on a result preview bar (at the bottom of the screen for both the EOSDIS and restaurant finder examples). Preview sizes can also be rendered on maps or charts, as illustrated in the EOSDIS prototype. These renderings must change within a fraction of a second in response to user input.

Selecting appropriate attribute values or categories rapidly reduces the data volume to a manageable size. Zero-hit queries are eliminated, since users can spot them without issuing a query. Once users are satisfied with the formulated query, it is submitted over the network to the database. More details about individual records are then retrieved to refine the query.

Table II. A Comparison Table of the Two Phases of the Query Formulation Process

	Query Preview	Query Refinement
Number of records	Very large	Manageable (each one is selectable for details-on-demand)
Number of attributes for selection	Few	More or all of the attributes
Selection of attribute values	Rough ranges or metavalues	More precise or exact values

8.2 Query Refinement

In the query refinement phase, users construct detailed queries over all database attributes, which are applied only to those records selected in the query preview. The scope of the query is smaller, but the resolution is finer. The interface provides access to all database attributes and their full range of values.

A characteristic of the query refinement phase is the rendering of each record in a graphical overview. The overview is closely related to the widgets used to refine a query, and reflects the query. By selecting appropriate values of relevant attributes, users continue to reduce the data volume and explore the correlation among the attributes through the visual feedback. Complete details can then be obtained at any time by accessing the database across the network for individual records.

9. RELATED WORK

An early proposal for volume previews in a database search is described in Heppe et al. [1985]. The "Dining out in Carlton" example was provided to illustrate a search technique (for a specific restaurant) based on the volume preview of the number of the available restaurants. However, query previews were not exploited to support dynamic queries and querying in networked information systems.

Retrieval by reformulation is a method that supports incremental query formation by building on query results [Williams 1984]. Each time a user specifies a query, the system responds with query reformulation cues that give users an indication of how the repository is structured and what terms are used to index objects. Users can then incrementally improve a query by critiquing the results of previous queries. Rabbit [Williams 1984] and Helgon [Fisher and Nieper-Lemke 1989] are examples of retrieval systems based on the retrieval-by-reformulation paradigm, which is also the basis of the two-phase query formulation approach.

Harvest [Bowman et al. 1994] was designed and implemented to solve problems common to Internet users. It provides an integrated set of customizable tools for gathering information from diverse repositories, building topic-specific indexes, and searching. Harvest could be used to maintain and update the metadata servers where users can extract information and store it locally in order to support dynamic queries in both the query preview and query refinement phases.

However, Harvest, just like other WWW browsers, still applies the traditional querying technique based on keywords. In order to express a complex query, a more visual query interface may be effective. Marmotta is a form-based tool used within WWW clients to query networked databases [Capobianco et al. 1995]. The ease of use of form-based interfaces is preserved (users need not know the structure of the database). Within Marmotta, icons are used to present the domain of interest and the retrieval requests in a structured form-based interface. Icons are used in Marmotta to formulate a query. The system then translates the query into a syntactically correct format that can be handled by an HTTP server. In order to cope with the increasing data volume, for example in libraries containing millions of documents, it is common to formulate queries on a library catalog. A prototype interface using a ranked output information retrieval system (INQUERY) for a library catalog containing about 300,000 documents has been implemented [Veerasamy and Navathe 1995]. The interface supports a visualization scheme that illustrates how the query results are related to the query words. Visualizing the results of the query keeps users more informed on how the system computed the ranking of documents. Another technique, Tilebars, visualizes term distribution information in each document to supplement result lists in full-text retrieval systems [Hearst 1995].

Butterfly was developed for simultaneously exploring multiple DIALOG bibliographic databases across the Internet using 3D interactive animation techniques [Mackinlay et al. 1995]. The key technique used by Butterfly is to create a virtual environment that grows under user control as asynchronous query processes link bibliographic records to form citation graphs. Asynchronous query processes reduce the overhead associated with accessing networked databases, and automatically formulated link-generating queries reduce the number of queries that must be formulated by users. The Butterfly system provides a visually appealing display. However, it was not designed to support the formulation of complex queries.

10. CONCLUSIONS AND PROJECT UPDATE

In this article, the concepts of query previews and refinement are presented, and two prototypes are described. The evaluation results from a NASA Prototyping Workshop and a controlled experiment confirm the benefits of the query previews. We suggest strategies to control the size of the volume preview table.

An operational query preview system has been implemented for NASA's Global Master Change Directory [Greene et al. 1997]. Consensus has been rapidly reached on attributes and values selection, and performance is satisfactory. Our experience confirmed the importance of metadata accuracy and completeness. The query preview interfaces make visible any problems or holes in the metadata that are not noticeable with classic form fill-in interfaces. This could be seen as a problem, but we think that it will have a long-term beneficial effect on the quality of the metadata, as data

providers will be compelled to produce more complete metadata. Our experience with the Global Master Change Directory demonstrates that the concepts are feasible in a large operational system, such as the EOSDIS directory environment.

This interface was included in 1997 within the operational GCMD where it is offered as an alternative experimental service (http://gcmd.nasa.gov). NASA is monitoring its use. As the number of Java-capable workstations grows and as the capabilities of browsers stabilize, usage is likely to increase. A second implementation using binary query previews (i.e., showing availability of data, but not counts [Plaisant et al. 1999]) is now publicly available for the Global Land Cover Facility (http://glcf.umiacs.umd.edu), a member of the NASA Earth Science Information Partnership (ESIP) Federation.

PROJECT WEBSITE

http://www.cs.umd.edu/hcil/eosdis

ACKNOWLEDGMENTS

We thank Teresa Cronnell for her graphic design of the Restaurant Finder prototype.

REFERENCES

AHLBERG, C., WILLIAMSON, C., AND SHNEIDERMAN, B. 1992. Dynamic queries for information exploration: an implementation and evaluation. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI '92, Monterey, CA, May 3–7), P. Bauersfeld, J. Bennett, and G. Lynch, Eds. ACM Press, New York, NY, 619–626.

BOWMAN, C. M., DANZIG, P. B., HARDY, D. R., MANBER, U., AND SCHWARTZ, M. F. 1994. The Harvest information discovery and access system. In *Proceedings of the 2nd International Conference on World Wide Web*. 763–771.

CAPOBIANCO, F., MOSCONI, M., AND PAGNIN, L. 1995. Progressive HTTP-based querying of remote databases within the Marmotta iconic VQS. In *Proceedings of the IEEE Workshop on Visualization*. IEEE Press, Piscataway, NJ, 122–125.

DOAN, K., PLAISANT, C., AND SHNEIDERMAN, B. 1996. Query previews in networked information systems. In *Proceedings of the Forum on Advances in Digital Libraries*. IEEE Computer Society Press, Los Alamitos, CA, 120–129.

DOAN, K., PLAISANT, C., SHNEIDERMAN, B., AND BRUNS, T. 1997. Query previews in networked information systems: A case study with NASA environment data. *SIGMOD Rec. 26*, 1 (Mar.), 75–81.

FISCHER, AND NIEPER-LEMKE, H. 1989. HELGON: Extending the retrieval by reformulation paradigm. In *Proceedings of the ACM Conference on Human-Computer Interaction* (CHI '89, Austin, TX, Apr. 30–May 4), K. Bice and C. Lewis, Eds. ACM, New York, NY, 333–352.

GREENE, S., TANIN, E., PLAISANT, C., SHNEIDERMAN, B., OLSEN, L., MAJOR, G., AND JOHNS, S. 1997. The end of zero-hit queries: Query previews for NASA's global change master directory. Tech. Rep. CS-TR-3855. Department of Computer Science, University of Maryland, College Park, MD. To appear in the Int. J. of Digital Libraries (1999).

HEARST, M. 1995. Tilebars: Visualization of term distribution information in full text information access. In *Proceedings of the ACM Conference on Human-Computer Interaction* (CHI '95, Denver, CO). ACM, New York, NY, 59–66.

HEPPE, D. L., EDMONDSON, W. H., AND SPENCE, R. 1985. Helping both the novice and advanced user in menu-driven information retrieval systems. In *Proceedings of the British Computer*

Society Conference on Human-Computer Interaction (HCI '85). British Computer Society, Swinton, UK, 92–101.

MACKINLAY, J. D., RAO, R., AND CARD, S. K. 1995. An organic user interface for searching citation links. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI '95, Denver, CO, May 7–11), I. R. Katz, R. Mack, L. Marks, M. B. Rosson, and J. Nielsen, Eds. ACM Press/Addison-Wesley Publ. Co., New York, NY, 67–73.

MARCHIONINI, G. 1995. *Information Seeking in Electronic Environments*. Cambridge Series on Human-Computer Interaction. Cambridge University Press, New York, NY.

PLAISANT, C., BRUNS, T., DOAN, K., AND SHNEIDERMAN, B. 1997a. Query Previews in Networked Information Systems: The case of EOSDIS. In *CHI '97 Technical Video Program*. ACM Press, New York, NY.

PLAISANT, C., MARCHIONINI, G., BRUNS, T., KOMLODI, A., AND CAMPBELL, L. 1997b. Bringing treasures to the surface: iterative design for the Library of Congress National Digital Library Program. In *conference proceedings on Human factors in computing systems* (CHI '97, Atlanta, Georgia, Mar. 22–27, 1997), S. Pemberton, Ed. ACM Press, New York, NY, 518–525.

PLAISANT, C., VENKATRAMAN, M., NGAMKAJORNWIWAT, K., BARTH, R., HARBERTS, B., AND FENG, W. 1999. Refining query preview techniques for data with multivalued attributes. In *Proceedings of the IEEE Forum on Research and Technology Advances in Digital Libraries* (ADL '99). IEEE Computer Society Press, Los Alamitos, CA, 50–59.

POSTON, J. 1996. Prototype workshop 2 (PW2) results report. Tech. Rep. 167-TP-001-001. ECS Development Team, Hughes Applied Information Systems, Landover, MD.

ROUSSOPOLOS, N., KOTIDI, Y., AND ROUSSOPOLOS, M. 1997. Cubetree: Organization of and bulk incremental updates on data cube. In *Proceedings of the ACM Conference on Management of Data* (SIGMOD '97). ACM, New York, NY.

SHNEIDERMAN, B. 1994. Dynamic queries for visual information seeking. *IEEE Softw. 11*, 6, 70–77.

TANIN, E., BEIGEL, R., AND SHNEIDERMAN, B. 1996. Incremental data structures and algorithms for dynamic query interfaces. *SIGMOD Rec. 25*, 4, 21–24.

TANIN, E., LOTEM, A., HADDADIN, I., SHNEIDERMAN, B., SLAUGHTER, L., AND PLAISANT, C. 1997. Evaluation of query previews: User preference and performance. Tech. Rep. CS-TR-3879. Department of Computer Science, University of Maryland, College Park, MD.

VEERASAMY, A. AND NAVATHE, S. 1995. Querying, navigating and visualizing a digital library catalog. In *Proceedings of the 2nd International Conference on the Theory and Practice of Digital Libraries*. http://www.csdl.tamu.edu/DL95/

WEILAND, W. J. AND SCHNEIDERMAN, B. 1993. A graphical query interface based on aggregation/generalization hierarchies. *Inf. Syst. 18*, 4 (June 1993), 215–232.

WILLIAMS, M. D. 1984. What makes RABBIT run?. *Int. J. Man-Mach. Stud. 21*, 5 (Nov. 1984), 333–335.

WILLIAMSON, C. AND SHNEIDERMAN, B. 1992. The dynamic HomeFinder: Evaluating dynamic queries in a real-estate information exploration system. In *Proceedings of the 15th Annual International ACM Conference on Research and Development in Information Retrieval* (SIGIR '92, Copenhagen, Denmark, June 21–24), N. Belkin, P. Ingwersen, A. M. Pejtersen, and E. Fox, Eds. ACM Press, New York, NY, 338–346.

YOUNG, D. AND SHNEIDERMAN, B. 1993. A graphical filter/flow representation of Boolean queries: A prototype implementation and evaluation. *J. Am. Soc. Inf. Sci. 44*, 6 (July 1993), 327–339.

Received: May 1997; revised: December 1997 and May 1998; accepted: August 1998