TECHNICAL
RESEARCH
REPORT

*Institute for
Systems
Research*

Dynamic Queries: A Step Beyond Database Languages

by B. Shneiderman

The Institute for Systems Research is supported by the National Science Foundation Engineering Research Center Program (NSFD CD 8803012), Industry and the University

TR 93-3

CAR-TR-655
CS-TR-3022
SRC-TR-93-3
Jan. 1992

Dynamic Queries: A Step Beyond Database Languages

Ben Shneiderman

Department of Computer Science,
Human-Computer Interaction Laboratory,
and Institute for Systems Research

University of Maryland
College Park, MD 20742

The purpose of computing is insight, not numbers.
Richard Hamming, 1962

Abstract

The capacity to incrementally adjust a query (with sliders, buttons, selections from a set of discrete attribute values, etc.) coupled with a visual display of results that are rapidly updated, dramatically changes the information seeking process. Dynamic queries on the chemical table of elements, computer directories, and a real estate database were built and tested in three separate exploratory experiments. Preliminary results show highly significant performance improvements and user enthusiasm more commonly seen with video games. Widespread application seems possible but research issues abound in the areas of: (1) graphic visualization design, (2) database and display algorithms, and (3) user interface requirements. Challenges include methods for rapidly displaying and changing many points, colors, and areas; multi-dimensional pointing and exploring using 6 degree of freedom input/output devices; incorporation of sound and visual display techniques that increase user comprehension; and integration with existing database systems.

1. INTRODUCTION

The steady growth of online databases and their dramatically improved accessibility through networks has only been partially matched by improvements in methods for browsing, searching, retrieving, filtering, and displaying data. However, certain approaches have captivated users and enabled remarkably powerful control of massive amounts of information. These successful approaches usually depend on powerful filtering tools, continuous visual display of information, rapid, incremental, and reversible control of the query, and pointing rather than typing. These features are interpretations of the direct manipulation principles (described in my *IEEE Computer* article, August 1983), in the database environment. I chose the term *dynamic queries* to emphasize the interactive user control of query parameters that generate an animated display of database results.

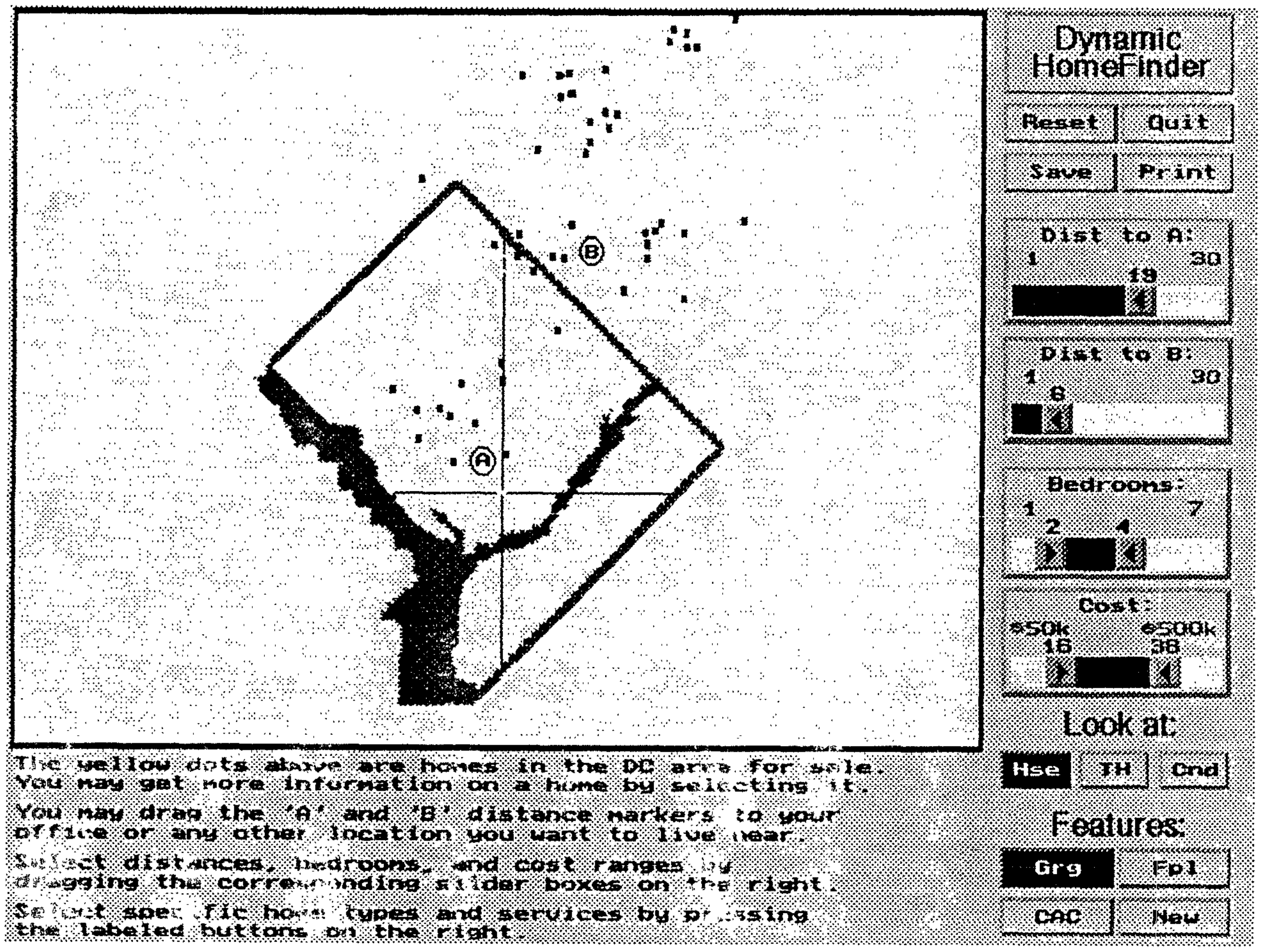
1.1 Dynamic Query examples

Most database queries are specified by typing a command in keyword-oriented language such as SQL, DIALOG, or FOCUS and the result is presented as a tabular list of tuples containing alphanumeric fields. This traditional approach is appropriate in many problem solving tasks, but formulating queries by direct manipulation and displaying the results in a graphical manner has advantages in many situations. For novice users, learning to formulate queries in a command language may take many hours and then they must deal with the high level of errors in syntax and semantics (Welty, 1985). Many projects have demonstrated that graphical queries can be helpful when users formulate queries (Michard, 1982; Wong & Kuo, 1982; Kim, Korth & Silberschatz, 1988). Also, graphical results in context, such as on a map (Egenhofer, 1990) are useful for comprehending the results.

For experts the benefits of graphical interfaces may be still greater since they will be able to formulate more complex queries and interpret intricate results. For example, air traffic control could hardly be imagined without graphical situation displays (looking down on all the planes in a sector), and I believe it could be improved with better query formulation widgets. Network management is an emerging application in which designers recognize the visual displays are potentially a strong benefit because of the extreme complexity in dealing with 10,000 or more nodes and links. Similarly, statisticians, demographers, or sociologists dealing with large multi-dimensional databases will be able to explore and discover relationships more easily.

Geographic applications emerge naturally as candidates for dynamic queries. A system for real estate brokers and their clients was built that located homes by adjusting sliders for the price, number of bedrooms, maintenance costs, quality of schools, etc. (Williamson & Shneiderman,

1992) (Figure 1). Each of the 1100 homes satisfying the query appears as a point of light on a Washington, DC map. Users could explore the database to find neighborhoods with high or low prices by moving a slider and watching where the points of light appeared. Geographic queries were supported by allowing users to mark the locations where they and their spouse work. Then users could adjust the sliders on distance to the work places to give intersecting circles of acceptable homes. An empirical study was conducted with 18 psychology undergraduates to compare the dynamic queries approach to a natural language query facility (Q&A from Semantec), and a 10-page paper listing. The counterbalanced within subjects design found statistically significant speed advantages for the dynamic queries over either alternative for the three most difficult of the five tasks. Subjective satisfaction dramatically favored the dynamic queries. One subject remarked about the dynamic queries treatment: "I don't want to stop, this is fun!"



The yellow dots above are homes in the DC area for sale.
You may get more information on a home by selecting it.

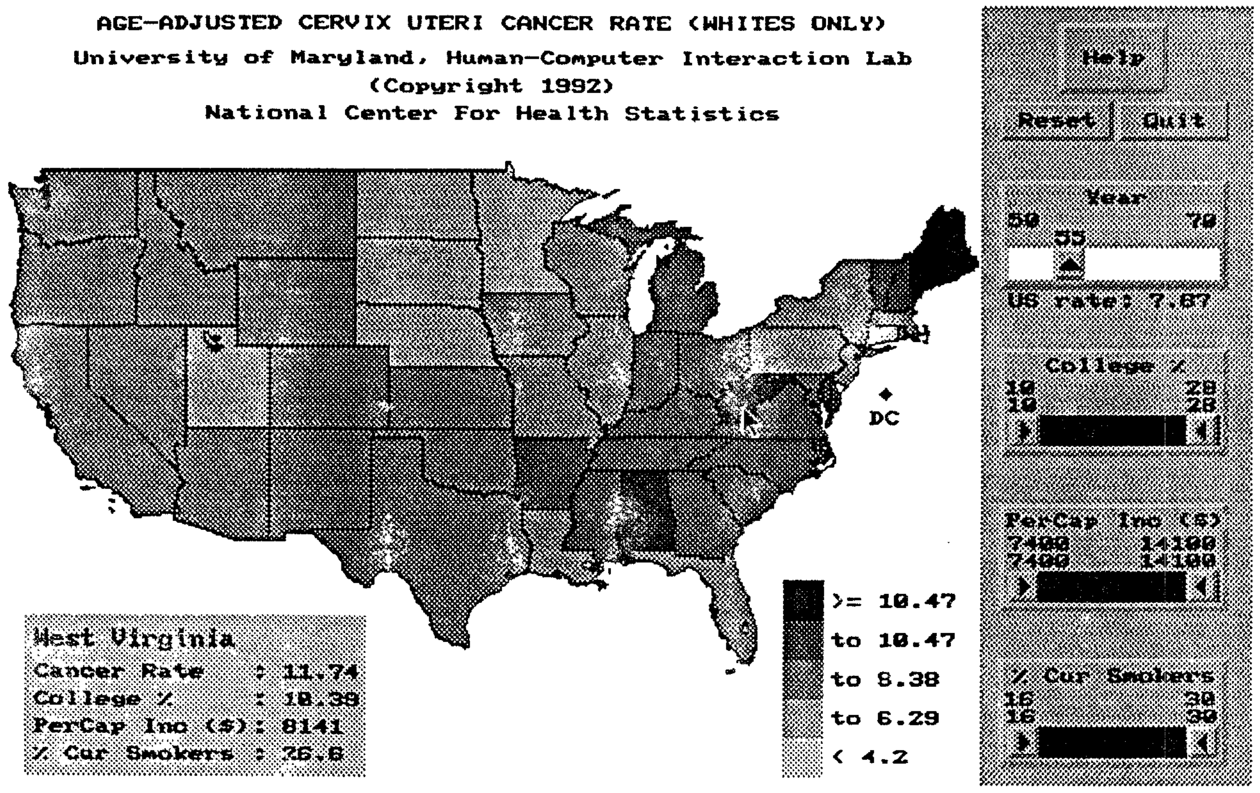
You may drag the 'A' and 'B' distance markers to your office or any other location you want to live near.

Select distances, bedrooms, and cost ranges by dragging the corresponding slider boxes on the right.

Select specific home types and services by pressing the labeled buttons on the right.

Figure 1: The DC Homefinder dynamic query system enables users to adjust the sliders for location, cost, number of bedrooms, home type (house, townhouse, or condominium), and features (Fireplace, Central Air Conditioning, garage, or new construction). The results are shown as points of light which can be selected to generate a detailed description at the bottom of the screen.

Another geographic application we built highlighted entire states of the United States that had cancer rates above a specified value (Plaisant, 1993) (Figure 2). Users could explore the database by looking at different years, or adjusting sliders to select statistics by per capita income, percent college educated, percent smokers, etc. The rapid change in area colors, accomplished with color indexing on the palette, enables users to detect changes in cancer rates over time and correlations with demographic variables.


AGE-ADJUSTED CERVIX UTERI CANCER RATE (WHITES ONLY)
University of Maryland, Human-Computer Interaction Lab
(Copyright 1992)
National Center For Health Statistics

West Virginia
Cancer Rate : 11.74
College % : 10.39
PerCap Inc ($) : 8141
% Cur Smokers : 26.6

Figure 2: This dynamic query shows cervical cancer rates from 1950 to 1970 in each state. Adjustments can be made to the year and state demographic variables such as the percentage of college education, per capita income, and percent smokers.

Applications seem abundant for traditional services that have geographic aspects (travel information systems, hotel or resort selection, choosing a college) as well as scientific or engineering applications (electronic circuits, networks, satellite ground coverage, astronomical star guides). Another likely candidate for output is a calendar or time line in which queries might specify event types (concerts, meetings, conferences) selected by cost, priority rankings, or distance from home.

A dynamic query tool for the chemical table of elements was built with sliders for atomic mass, atomic number, atomic radius, ionic radius, ionization energy, and electronegativity (Ahlberg, Williamson & Shneiderman, 1992) (Figure 3). Appropriate chemicals are highlighted and students can refine their intuitions about the relationships among these properties and the atomic number or position in the table. The dynamic query approach to the chemical table of elements was tested in an empirical comparison with a form fill-in query interface. The counterbalanced ordering within subjects design with 18 chemistry students showed strong advantages for the dynamic queries in terms of faster performance and lower error rates.

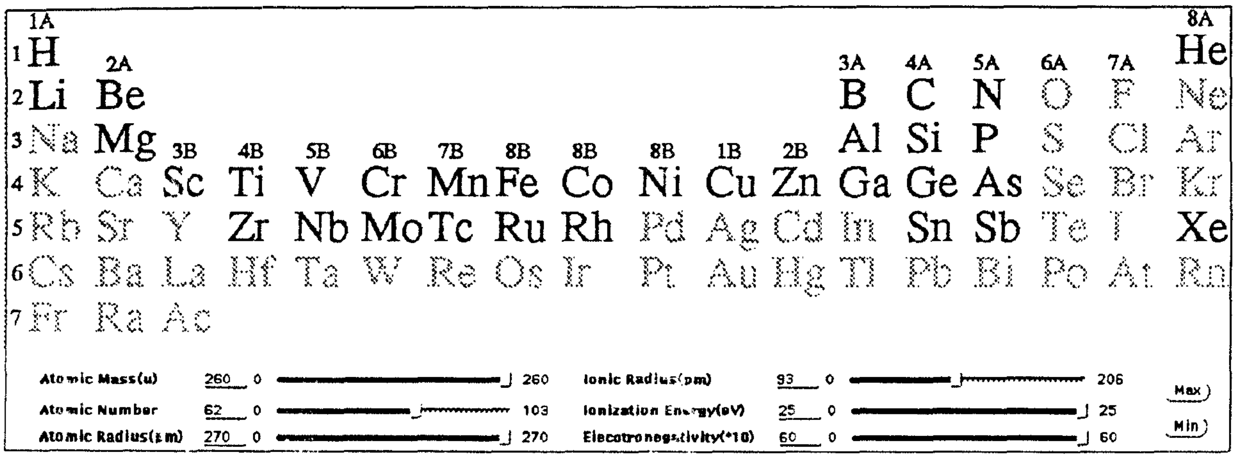
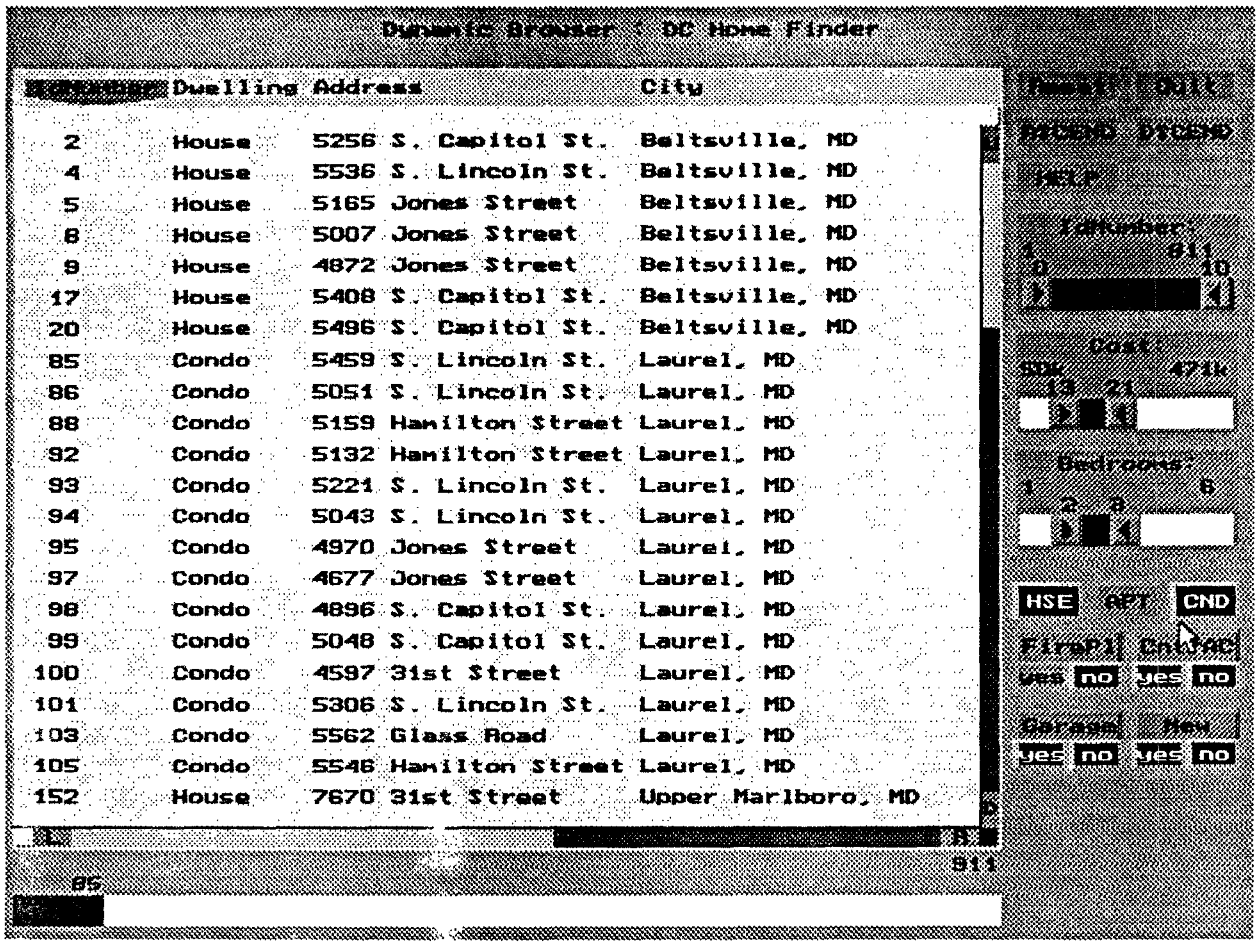

1A
2A
3A 4A 5A 6A 7A 8A
3B 4B 5B 6B 7B 8B 8B 8B 1B 2B
1 H He
2 Li Be B C N O F Ne
3 Na Mg Al Si P S Cl Ar
4 K Ca Sc Ti V Cr Mn Fe Co Ni Cu Zn Ga Ge As Se Br Kr
5 Rb Sr Y Zr Nb Mo Tc Ru Rh Pd Ag Cd In Sn Sb Te I Xe
6 Cs Ba La Hf Ta W Re Os Ir Pt Au Hg Tl Pb Bi Po At Rn
7 Fr Ra Ac
Atomic Mass(u) 260 0 260
Atomic Number 62 0 103
Atomic Radius(pm) 270 0 270
Ionic Radius(pm) 93 0 206
Ionization Energy(eV) 25 0 25
Electronegativity(*10) 60 0 60
Max
Min

Figure 3: The chemical table of elements makes a natural visual display for information on chemical properties. Chemicals matching the query are shown in red. Gaps and jumps are easily found.

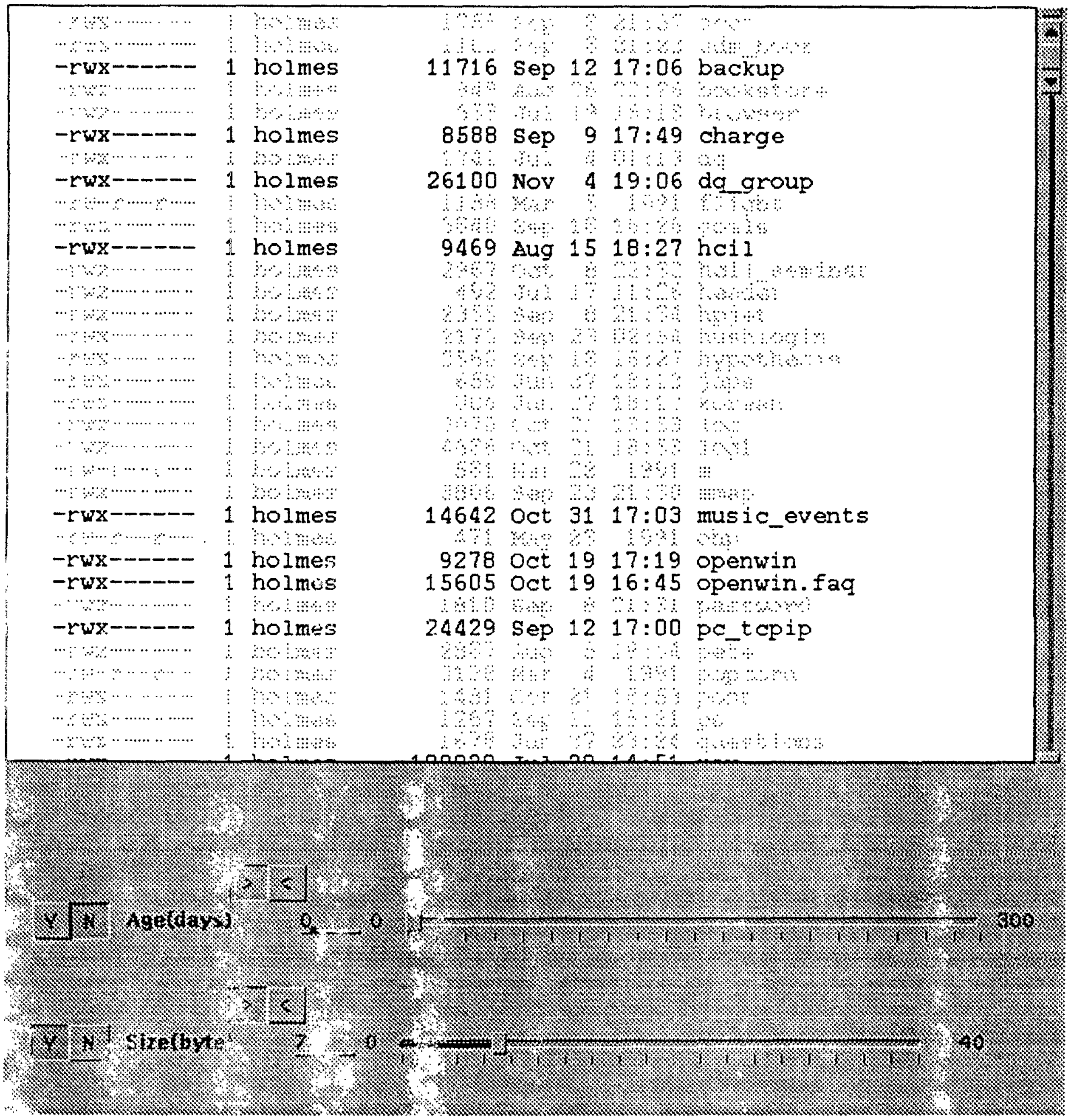
When there are no natural graphical displays for the output, dynamic queries can still be implemented with result set in a traditional alphanumeric tabular display (Figure 4). In our implementation the sliders and buttons are created semi-automatically by the program depending on the values that exist in the imported ASCII database. As the users press down on the mouse and adjust the sliders, the result bar on the bottom changes dynamically to indicate how many items remain in the result set, but the tabular display is updated only when the user releases the mouse button. This policy was adopted to avoid the distraction of the frequent rewriting of the display.



[illegible]	Dwelling	Address	City
2	House	5256 S. Capitol St.	Beltsville, MD
4	House	5536 S. Lincoln St.	Beltsville, MD
5	House	5165 Jones Street	Beltsville, MD
8	House	5007 Jones Street	Beltsville, MD
9	House	4872 Jones Street	Beltsville, MD
17	House	5408 S. Capitol St.	Beltsville, MD
20	House	5496 S. Capitol St.	Beltsville, MD
85	Condo	5459 S. Lincoln St.	Laurel, MD
86	Condo	5051 S. Lincoln St.	Laurel, MD
88	Condo	5159 Hamilton Street	Laurel, MD
92	Condo	5132 Hamilton Street	Laurel, MD
93	Condo	5221 S. Lincoln St.	Laurel, MD
94	Condo	5043 S. Lincoln St.	Laurel, MD
95	Condo	4970 Jones Street	Laurel, MD
97	Condo	4677 Jones Street	Laurel, MD
98	Condo	4896 S. Capitol St.	Laurel, MD
99	Condo	5048 S. Capitol St.	Laurel, MD
100	Condo	4597 31st Street	Laurel, MD
101	Condo	5306 S. Lincoln St.	Laurel, MD
103	Condo	5562 Glass Road	Laurel, MD
105	Condo	5546 Hamilton Street	Laurel, MD
152	House	7670 31st Street	Upper Marlboro, MD

Figure 4: Even when there is no natural graphic framework for a dynamic query display, the method can be used with tabular alphanumeric output. As users adjust the sliders andbuttons for the query, the result bar along the bottom indicates how many items match. When the users stop moving the sliders and let go of the mouse button, the tabular display is re-written.

Another tabular dynamic query was built to allow users to explore UNIX directories (Liao, Osada & Shneiderman, 1992) (Figure 5). Sliders for size (in kilobytes) and age (in days) of files enabled 18 users to answer ten questions such as "How many files are younger than umcp_tai?" The three versions of the program explored approaches to showing the results in standard 'ls -l' format: highlighting matches with color, highlighting matches with asterisks on the same line, and displaying only the matching lines (that is, delete the non-matching files from the display). The latter approach, called Expand/contract, was distracting if updates were made as the slider was being moved, so re-displays occurred when users stopped moving the sliders and let go of the mouse button. In five of the tasks there was a statistically significant speed advantage for the Expand/contract interface. This result occurred only with medium sized directories of approximately 60 entries (two screen pages), and not with a smaller one screen page directory. The benefits of Expand/contract seem likely to grow as the size of the directory increases. These results help us develop guidelines and hopefully theories about how to design displays for dynamic queries.



```
[illegible]
[illegible]
-rwx------   1 holmes      11716 Sep 12 17:06 backup
[illegible]
[illegible]
-rwx------   1 holmes       8588 Sep  9 17:49 charge
[illegible]
-rwx------   1 holmes      26100 Nov  4 19:06 dq_group
[illegible]
[illegible]
-rwx------   1 holmes       9469 Aug 15 18:27 hcil
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
-rwx------   1 holmes      14642 Oct 31 17:03 music_events
[illegible]
-rwx------   1 holmes       9278 Oct 19 17:19 openwin
-rwx------   1 holmes      15605 Oct 19 16:45 openwin.faq
[illegible]
-rwx------   1 holmes      24429 Sep 12 17:00 pc_tcpip
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
```

Figure 5: The standard 'ls-l' tabular display can be the framework for UNIX directory exploration. The sliders, built with Sun DevGuide, allow selections to be made on the age (in days) and size (in kilobytes) of files. Color highlighting and expand/contract methods of display were compared in an exploratory study.

1.2 Advantages

The advantage of dynamic queries seems to be that it allows users to rapidly, safely, and even playfully explore a database. Users may be able to discover quickly which sections of the multi-dimensional search space are densely and sparsely populated, where there are clusters, exceptions, gaps, or outliers, and what trends there are in ordinal data. Such overviews, the ability to explore, and the capacity to rapidly specify known item queries makes dynamic queries an appealing approach for certain problems. Several researchers and developers have recognized the advantages of rapid reformulation of queries even in textual interfaces. The advantages of visual input and output were explored in some statistical display programs, geographic information systems, and computer assisted design packages.

Dynamic queries appear to be effective in both answering specific fact questions and in exploratory browsing of large data sets. Dynamic queries achieve this wide rage of application by applying direct manipulation strategies:

- Graphical presentation of query components
- Graphical presentation of results
- Rapid, incremental and reversible actions
- Selection by pointing (not typing)
- Immediate and continuous feedback

The benefits are the elimination of syntactic errors which encourages exploration. Dynamic queries can reveal global properties as well as assist users in answering specific questions. The disadvantages are that current hardware and software systems are poorly organized to provide the necessary visual input, data structures, and graphical outputs. As the database grows it will be more difficult to update the display fast enough and specialized data structures or parallel computation may be required. Other problems include the need for graphic displays, thereby limiting utility over slow telephone lines, and the difficulty for blind users.

We could analyze the benefits of dynamic queries by analyzing the tasks that could be accomplished by different user communities. For data in which there is a known relationship among variables, the dynamic queries interface is useful for training and education by exploration. For situations in which there are understood correlations, but their complexity makes it difficult for non-experts to follow, dynamic queries can allow a wider range of people to explore the interactions (health and demographic variables, chemical table of elements, economic or market data). Finally, where there is so much data that even experts have not sorted out the correlations, dynamic queries are a unique tool to discover patterns, form and test hypotheses, identify exceptions, segment data, or prepare figures for reports (studying clinical trial data, picking stocks, finding a home).

1.3 Disadvantages

Dynamic queries place high demands on computer systems. We are exploring how to accommodate large datasets and permit rapid access. Rapid and high resolution displays are especially useful for dyanmic queries, but these are not so widely available. Secondly, some application specific work is needed to make the best advantage of dynamic query methods. While we have developed some standardized tools, they still require conversion of data and possibly some programming. Standardized input and output would make dynamic queries easier to integrate into existing database and information systems. Third, our current dynamic queries implement only simple queries that are conjuncts of disjunctions, plus range queries on numeric values. More elaborate queries (group by, set matching, universal quantification, transitive closure, string matching) are contemplated but these are still research and development problems. Fourth, visually handicapped and blind users will have a more difficult time with our widgets and outputs, but we feel that we could accommodate these users as well, and we are exploring audio feedback.

1.4 Goals

> "Visualization is a method of computing. It transforms the symbolic into the geometric, enabling researchers to observe their simulations and computations. Visualization offers a method for seeing the unseen. It enriches the process of scientific discovery and fosters profound and unexpected insights. In many fields it is already revolutionizing the way scientists do science."
> (McCormick, B., DeFanti,T. and Brown, R. et al., 1987. Visualization in scientific computing, Computer Graphics, November 1987 ACM SIGGRAPH)

A vast quantity of information is multi-dimensional and people have great difficulty dealing with such large bodies of information. Alleviating this problem in any significant way can potentially reduce the frustration and enhance the abilities of many people. Effective visualizations of large bodies of multi-dimensional information can help users gain insight into relevant features of their data, construct accurate mental models of the information, and search for regions of particular interest.

Although our initial results are encouraging, there are many unanswered design questions that need exploration, for example how to:

- design widgets to specify ranges of values, such as greater than 212.0 or between 18 and 25,
- enable users to express boolean combinations of slider settings,
- choose among highlighting by color, points or light, regions, blinking , etc.,
- allow varying degrees of intensity in the visual feedback,
- cope with thousands of points or areas by zooming,
- permit weighting of criteria,
- select a set of sliders from a large set of attributes,
- provide 'grand tours' to automatically view all dimensions,
- include sound as redundant or unique coding of one dimension, and
- support multi-dimensional input.

Other issues emerge when no natural two dimensional representation of the data can be identified. Of course a textual representation can always be used (e.g. the list of items highlighted with color, inverse video, font type, and size) and we explored such representations for the dynamic query of directory listings (Figure 3).

The treemap idea is a possible alternative mechanism for visualizing large amounts of hierarchical information on which dynamic queries could be applied (Johnson and Shneiderman, 1991). For example we built a business application allowing the visualization of sales data for a complete product hierarchy, color-coded by profitability and size-coded by revenue. Twelve professional users in our usability study could rapidly determine the state of financial affairs -- large red regions indicate trouble and blue areas signal success. A slider allowed users to quickly observe the changes to the treemap over time and find trends or spot problems.

2. RESEARCH DIRECTIONS

2.1 Graphic visualization and design

As humans we have the ability to recognize the spatial configuration of elements in a picture and notice the relationships among elements quickly. This highly developed visual system allows people to grasp the content of a picture much faster than they can scan and understand text. By shifting some of the cognitive load of information retrieval to the perceptual system designers can capitalize on a well developed human visual processing capability. Appropriate static coding of properties, by size, position, shape, and color, can greatly reduce the need for explicit selection, sorting, and scanning operations.

Graphical display properties such as color (hue, saturation, brightness), texture, shape, border, blinking, etc. are of primary interest. Color is the most important of these visual display properties, and it can be an important aid to fast and accurate decision making. Only visual properties may be statically presented, although there is no need to limit the coding of domain values to strictly visual properties. Auditory properties may be useful in certain circumstances (e.g. lower frequency sounds would be associated with large values and higher frequency sounds with small values), especially as redundant reinforcement feedback.

Although human perceptual skills are impressive, the volume of data required for many tasks is so great that it cannot be displayed on a single screen at once. Interactive control of the drawing is therefore critical because the mapping of content information to the display will vary depending on the information the user requires (see box for details).

Perceptual principles

More effective guidelines for graphic displays are emerging as researchers focus on specific tasks (Tufte, 1983; Foley et al., 1990; Ding and Mateti, 1990; MacDonald, 1990; Marcus, 1992):

* Color coding is most effective when the display does not have a fixed format, the density of information is high, relevant information must be discriminated, and symbol legibility is degraded

* Short term memory is better for color than it is for shape or number of objects and long term memory decays more slowly for a color code than for shape or number (Cavanaugh, 1972).

* Color unifies or clusters elements in a visual field (Moar, 1977; Christ & Corso, 1983).

* Color information is processed earlier than shape or alphabetic information (Lappin, 1967, Newman & Davis, 1962; Stollings, 1984).

* Audio-lization methods may apply sound to help users to understand the distribution of items (Blattner, Greenberg, and Kamegai, 1992).

On the other hand, it is often pointed out that color can be misused, confusing, and distracting. Consequently, we will take caution in the use of color and avoid the following problems reported in the literature: Irrelevant color coding can interfere with the processing of other information that is not color coded. Overuse of color can increase the sense of visual clutter (Teichner, 1979). Too many colors may be confusing. Researchers often suggested that the number of color codes be limited to 7-12 (Narborough-Hall, 1985; Spiker, Rogers & Cicinelli, 1985). However, the number of codes can be increased by varying brightness and saturation as well as hue.

Blattner, M., Greenberg, R. M., and Kamegai, M., 1992. Listening to turbulence: An example of scientific audiolization, In Blattner, M. and Dannenberg, R. B. (Editors), *Interactive Multimedia Computing*, ACM Press, New York, NY.

Cavanaugh, J. P., 1972. Relation between immediate memory span and the memory search rate. *Psychological Review 79*, 525-530.

Christ, K. A. and Corso, G., 1982. The effects of extended practice on the evaluation of visual display codes. *Human Factors 25*, 71-84.

Foley, J. D., Van Dam, A., Feiner, S. K. and Hughes, J. F., 1990. *Computer Graphics: Principles and Practice: Second Edition*, Addison-Wesley Publ. Co., Reading, MA.

Lappin, J. S., 1967. Attention in the identification of stimuli in complex visual displays. *Journal of Experimental Psychology 75*, 321-328.

MacDonald, L. W., 1990. Using color effectively in displays for computer-human interface. *Displays*, July 1990.

Marcus, A., 1991. *Graphic Design for Electronic Documents and User Interfaces*, ACM Press, New York, NY.

Moar, I., 1977. Spatial organization in the free recall of pictorial stimuli. *Quarterly Journal of Experimental Psychology 29*, (4) 699-708.

Narborough-Hall, C. S., 1985. Recommendations for applying color coding to air traffic control displays. *Displays: Technology and Applications*, July 1985, 131-137.

Newman, K. M. and Davis, A. R., 1962. Non-redundant color, brightness, and flashing rate encoding of geometric symbols on a visual display, *Jnl of Engineering Psychology 1*, 47-67.

Stollings, M. N., 1984. Information processing load of graphic versus alphanumeric weapon format displays for advanced fighter cockpits. U.S. Air Force Flight Dynamics Laboratory, Wright-Patterson Air Force Base, OH, Technical Report AFWAL-TR-84-3037.

Teichner, W. H., 1979. Color and visual information coding. *Proceedings of the Society for Information Display 20*, 3-9.

Tufte, E., 1983. *Visual Display of Quantitative Information*, Graphics Press, Cheshire, CT.

2.2 Database and display algorithms

Existing algorithms to store and retrieve multi-dimensional information need refinement to support efficient storage utilization and rapid update of the display (see box).

Specialized data structures to support dynamic queries

For small databases that fit in main memory, we are experimenting with array indexing, inverted files, and k-d trees (Bentley and Friedman, 1979; Samet, 1989).

For larger, disk-based databases, we are examining R-trees (Guttman, 1984, Beckmann, 1990), grid files (Nievergelt, 1984) multiple B-trees and reduced combined indices (Lum, 1970; Shneiderman 1977). When inserts and deletes to the stored information are treated separately, the design of efficient data structures is simplified.

Beckmann, N., Kriegel, H.-P., Schneider, R., and Seeger, B., 1990. The R*-tree: An efficient and robust access method for points and rectangles, *Proc. ACM SIGMOD*, 322-331.

Bentley, J. L. and Friedman, J. H., 1979. Data structures for range searching, *ACM Computing Surveys, 11*, 4, 397-409.

Guttman, A., 1984. R-Trees: A dynamic index structure for spatial searching, *Proc. ACM SIGMOD*, 47-57.

Lomet, D. B. and Salzberg, B., 1990. The hB-tree: A multiattribute indexing method with good guaranteed performance, *ACM Transactions on Database Systems 15*, 4, 625-658.

Lum, V. Y., 1970. Multi-attribute retrieval with combined indexes, *CACM 13*, 11, 660-665.

Nievergelt, J., Hinterberger, H., and Sevcik, K.C., 1984. The Grid File: An adaptable, symmetric multikey file structure, *ACM Trans. on Database Systems 9*, 1, 38-71.

Samet, H., 1989. *Design and Analysis of Spatial Data Structures*, Addison-Wesley Publishing Co., Reading, MA.

Shneiderman, B., 1977. Reduced combined indexes for efficient multiple attribute retrieval, *Information Systems 2*, 4 (1977), 149-154.

Since dynamic query users will work by adjusting the setting on a slider, usage is predictable. Given that we have the results of the previous query, and that the new query has slightly enlarged or contracted one of the ranges of the attributes in the old query, we would like to find the fastest method to update the results. The innovations here stem from the fact that structures must be kept largely in high-speed storage to ensure that the rapid performance demands of dynamic queries are met. We believe that effective strategies will stem from the organization of data along each dimension in buckets adjusted to the granularity of the slider mechanism. For example, if the slider has 100 positions for a field whose range is 1 to 50,000, then the data should be organized into 100 buckets each covering 500 points on the field. Then as the slider is moved to increase the selected set, the buckets can be appended to the selected set. As the slider is moved to decrease the selected set, the buckets can be removed. With three dimensions of 100 buckets each, the database is conveniently broken into 1,000,000 buckets which can be stored and retrieved efficiently.

Data compression methods are important to allow larger databases to fit in 32 megabyte or smaller address spaces. Alternatively, parallel hardware and algorithms that search multiple storage spaces may be needed.

Screen management algorithms also play an important role. When a slider is moved or a button depressed, instead of repainting the entire display it is often more effective to merely repaint the areas or points that have changed. Our early efforts suggest that in some cases manipulation of the palette by color indexing may be effective in providing rapid changes for irregularly shaped regions, even on popular personal computers (Plaisant, 1992). New classes of algorithms for screen management are anticipated as an alternative to more expensive hardware.

2.3 User-Interface Requirements

Our user interface directions are to identify appropriate presentation and interaction methods, and test their effectiveness via usability studies. A central issue is the design of appropriate widgets. Even in our early explorations we were surprised that none of the existing user interface management systems contained a double-boxed slider that would permit specification of a range (e.g. cost of a home is required to be more than $70,000 and less than $130,000). In creating such a slider we discovered how many design decisions and possibilities there were. In addition to dragging the boxes, we had to contend with jumps, limits, display of current values, what to do when the boxes were pushed against each other, choice of colors, possible use of sound, etc.

On the input side we realize that existing widgets are poorly matched with the needs of expert users who are comfortable with multi-dimensional browsing. Two-dimensional input widgets to select two values at once are not part of any standard widget set that we have reviewed, necessitating their creation (Figure 6). The benefits of a single widget are that only one selection is required to set two values and that correct selections can be guaranteed (the black areas indicate impossible selections, for example, the cheapest 7 bedroom house is $310,000).

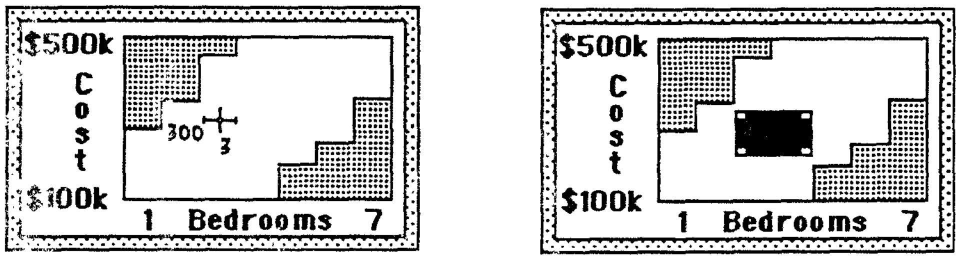


Figure 6: Two prototype two-dimensional widgets. The left one specifies a point indicating the number of bedrooms (3) and cost of a home ($300,000) with a single selection. The right one specifies a range of bedrooms (3 to 4) and cost ($180,000 to $320,000).

Three and higher dimensional input widgets may facilitate exploration of complex relationships. Current approaches for high dimensional input and feedback are clumsy, but research with novel devices such as data gloves (Feiner and Beshers, 1990), Polhemus devices, the SpaceBall, or various 3-D mice may uncover effective methods. With a 3-D mouse users lift the mouse off the desk and move it like a child playing with a toy airplane. The mouse system continuously outputs the six parameters (six degrees of freedom - 6DOF) that define its linear and angular position with respect to a fixed coordinate system in space.

Designers can always decompose the rotation motion of the mouse into the combination of (1) a rotation around the handle of the mouse, (2) a change in the direction where this handle is pointing. When the mouse is held as a pointer, the rotation around the handle is created by a twist of the arm, and it may be therefore natural to users to make the same twisting motion to increase the level of a database parameter as they would to increase the volume of a car radio. Changing the pointing direction of the mouse handle is done by the same wrist flexion that a lecturer would use to change the orientation of a laser pointer to point at another part of the conference screen. It may then also feel natural to users to imagine the planar space of two database parameters as vertical in front of them and point at specific parts by flexing their wrist up, down and sideways.

For example, sophisticated users could perform a dynamic query of the periodic table of elements using the 3D mouse. They would find elements of larger atomic mass by translating the mouse upward; for larger atomic numbers they would move to the right; for larger ionization energies they would move toward the display; for larger atomic radius they would bend their wrist up; for larger ionic radius they would bend their wrist to the right; for larger electronegativity they would twist their arm clockwise. Sliders should probably still be present on the screen, but would move by themselves and give feedback on parameter values.

On the display side many questions have been opened up by our initial efforts. Sometimes points on a map are a natural choice, but non-overlapping areas and overlapping areas seem useful in some applications. Points and areas can be on or off (in which case monochrome displays may be adequate), but we believe that color coding may allow more information to be displayed. Texture and shape coding, plus sound are also appealing directions.

To cope with the larger problem of specifying complex boolean combinations of attribute values Degi Young in our lab implemented the filter/flow model. Figure 7 shows how it might be applied to help students choosing colleges. Users can select from the set of attributes and get an appropriate filter widget (type in for interest areas, sliders for cost, and buttons for scholarships) which is placed on the screen with flow lines showing ANDs (sequential flow) and ORs (parallel flows). The X in each filter widget could be selected to negate the filter values. Clustering of one-in-one-out segments to form a new and save-able filter is possible. This approach was shown to be statistically significantly more effective than SQL for composing and comprehending queries.

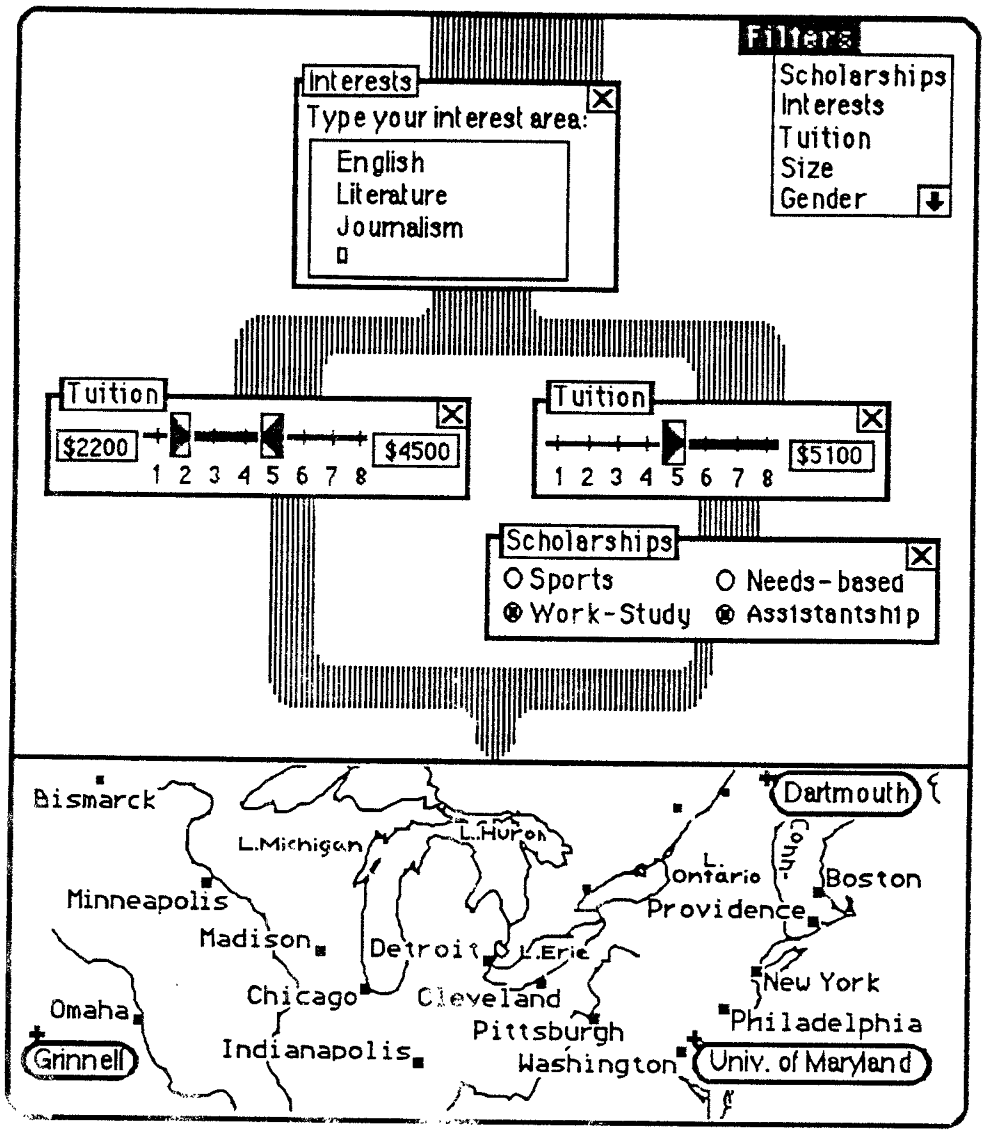


Figure 7: A mockup of a filter/flow Boolean query ((Interests = English or Literature or Journalism) AND ((Tuition greater than or equal to $2200 or less than or equal to $4500) OR ((Tuition greater than or equal to $5100) AND (Scholarships are available by Works-Study or Assistantship)))) combined with map output to show the result (Dartmouth, Grinnell, and the Univ. of Maryland).

3. SUMMARY

Dynamic queries over multi-dimensional information structures will not supersede all previous methods for the handling of information. Usability studies and controlled experiments with a range of tasks and users can confirm or deny a researcher's intuition about when dynamic queries are most effective.

We believe that dynamic queries offer a lively new direction for the application of computers. Many problems that are difficult to deal with using a keyword-oriented command language become tractable when dynamic queries are used. Contemporary computers have become fast enough that this direct manipulation approach can be applied for modest-sized problems. The challenge now is to broaden the spectrum of applications by improved user interface design and by fast database search plus compression methods.

Dynamic queries can be a general approach that could be attached to every database system, spreadsheet, and many stand-alone applications. The benefits accrue both to novice users and to experienced users. Research directions include: (1) graphic visualization design, (2) database and display algorithms, and (3) user interface requirements.

Acknowledgements: Thanks for the creative implementation efforts of Christopher Ahlberg, Christopher Williamson, Holmes Liao, Boon-Teck Kuah, Willaim Weiland, and Vinit Jain for taking these ideas into a reality that was even better than I anticipated. Important contributions to the work and this paper were made by Kent Norman and Catherine Plaisant. We greatly appreciate the support for this research from Johnson Controls, National Center for Health Statistics, NCR Corporation, Sun Microsystems, and Toshiba.

References

Ahlberg, C., Williamson, C., and Shneiderman, B., 1992. Dynamic queries for information exploration: An implementation and evaluation, *Proc. ACM CHI'92 Conference*, 619-626.

Egenhofer, M., 1990. Manipulating the graphical representation of query results in Geographic Information Systems, *1990 IEEE Workshop on Visual Languages*, IEEE Computer Society Press, Los Alamitos, CA, 119-124.

Feiner, S. and Beshers, C., 1990. Worlds within worlds: Metaphors for exploring n-dimensional virtual worlds, *Proc. User Interface Software and Technology '90*, ACM, New York, NY, 76-83.

Liao, H., Osada, M., and Shneiderman, B., 1992. A formative evaluation of three interfaces for browsing directories using dynamic queries, Technical Report Dept. of Computer Science University of Maryland, CS-TR-2841, CAR-TR-605.

Johnson, B. and Shneiderman, B., 1991. Tree-Maps: A space-filling approach to the visualization of hierarchical information structures, *Proc. IEEE Visualization 1991.*

Kim, H. J., Korth, H. F., and Silberschatz, A.,1988. PICASSO: A graphical query language, *Software: Practice and Experience 18*, 3, 169-203.

Plaisant, C., 1993. Dynamic queries on a health statistics atlas, Forthcoming Technical Report, Human-Computer Interaction Laboratory, University of Maryland, College Park, MD.

Welty, C., 1985. Correcting user errors in SQL, *International Journal of Man-Machine Studies* 22, 463-477.

Williamson, C. and Shneiderman, B., 1992. The Dynamic HomeFinder: Evaluating dynamic queries in a real-estate information exploration system, *Proc. ACM SIGIR Conference*, 339-346.

Wong, H.K.T. and I. Kuo, 1982. GUIDE: Graphical User Interface for Database Exploration, *Proceedings of the 8th Very Large Databases Conference*.

Table of Figures