CAR-TR-655
CS-TR-3022
SRC-TR-93-3

Sept. 1993
Revised Jan. 1994

Dynamic Queries for Visual Information Seeking

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

Keywords: dynamic queries, database search, information retrieval, direct manipulation, user interface, human-computer interaction, visual languages, visual information seeking

Abstract

Dynamic queries are a novel approach to information seeking that may enable users to cope with information overload. They allow users to see an overview of the database, rapidly (100 msec updates) explore and conveniently filter out unwanted information. Users fly through information spaces by incrementally adjusting a query (with sliders, buttons, and other filters) while continuously viewing the changing results. Dynamic queries on the chemical table of elements, computer directories, and a real estate database were built and tested in three separate exploratory experiments. These results show statistically significant performance improvements and user enthusiasm more commonly seen with video games. Widespread application seems possible but research issues remain in database and display algorithms, and user interface design. Challenges include methods for rapidly displaying and changing many points, colors, and areas; multi-dimensional pointing; incorporation of sound and visual display techniques that increase user comprehension; and integration with existing database systems.

1. INTRODUCTION

Some innovations restructure the way people think and work. Our experiences with *dynamic queries* interfaces suggest that they offer a dramatic change from existing methods for querying databases. While languages such as SQL have become standardized and form fill-in interfaces have become widespread, now dynamic queries empower users to perform far more complex searches by applying visual information seeking strategies. As users adjust sliders or select buttons, the results are continuously updated within 100 milliseconds, enabling them to answer simple fact questions and find patterns or exceptions.

Dynamic queries are an application of the direct manipulation principles in the database environment (Shneiderman, 1983). They depend on presenting a visual overview, powerful filtering tools, continuous visual display of information, pointing rather than typing, and rapid, incremental, and reversible control of the query.

> Definition: *dynamic queries* describes the interactive user control of visual query parameters that generate a rapid (100 msec update) animated visual display of database search results[1] .

1.1 Dynamic queries examples

The enthusiasm of users for dynamic queries appears to emanate from the sense of control they gain over the database. They quickly perceive patterns in the data, fly through the data by adjusting sliders, and generate new queries in 100 msecs based on what they discover through incidental learning. By contrast, most database queries are specified by typing a command in keyword-oriented language such as SQL, DIALOG, or FOCUS and the result is 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 graphically has advantages in many situations.

For novices, learning to formulate queries in a command language may take several hours and then they must deal with the high level of errors in syntax and semantics (Welty, 1985; Borgman, 1986). Many projects have demonstrated that visual information seeking methods can be helpful in formulating queries (Michard, 1982; Wong & Kuo, 1982; Kim, Korth & Silberschatz, 1988) and graphical results in context, such as on a map (Egenhofer, 1990) or a wall (Robertson, Card & Mackinlay, 1993), aid comprehension.

For experts, the benefits of visual 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

[1] This definition differs from the use of the term dynamic queries to describe SQL queries that are posed at run time instead of compile time.

be imagined without graphical situation displays (looking down on all the planes in a sector). Network management is an emerging application for which visual displays are a benefit because of the extreme complexity in dealing with 10,000 or more nodes and links (Consens, Cruz & Mendelzon, 1992). Similarly, statisticians, demographers, or sociologists dealing with large multi-dimensional databases will be able to explore and discover relationships more easily using dynamic queries (Becker & Cleveland, 1987).

Geographic applications emerge naturally as candidates for dynamic queries. We built a system for real estate brokers and their clients that allowed them to locate homes by adjusting sliders for the price, number of bedrooms, distance from work, etc. (Williamson & Shneiderman, 1992) (Figure 1). Each of the 1100 homes satisfying the query appeared 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

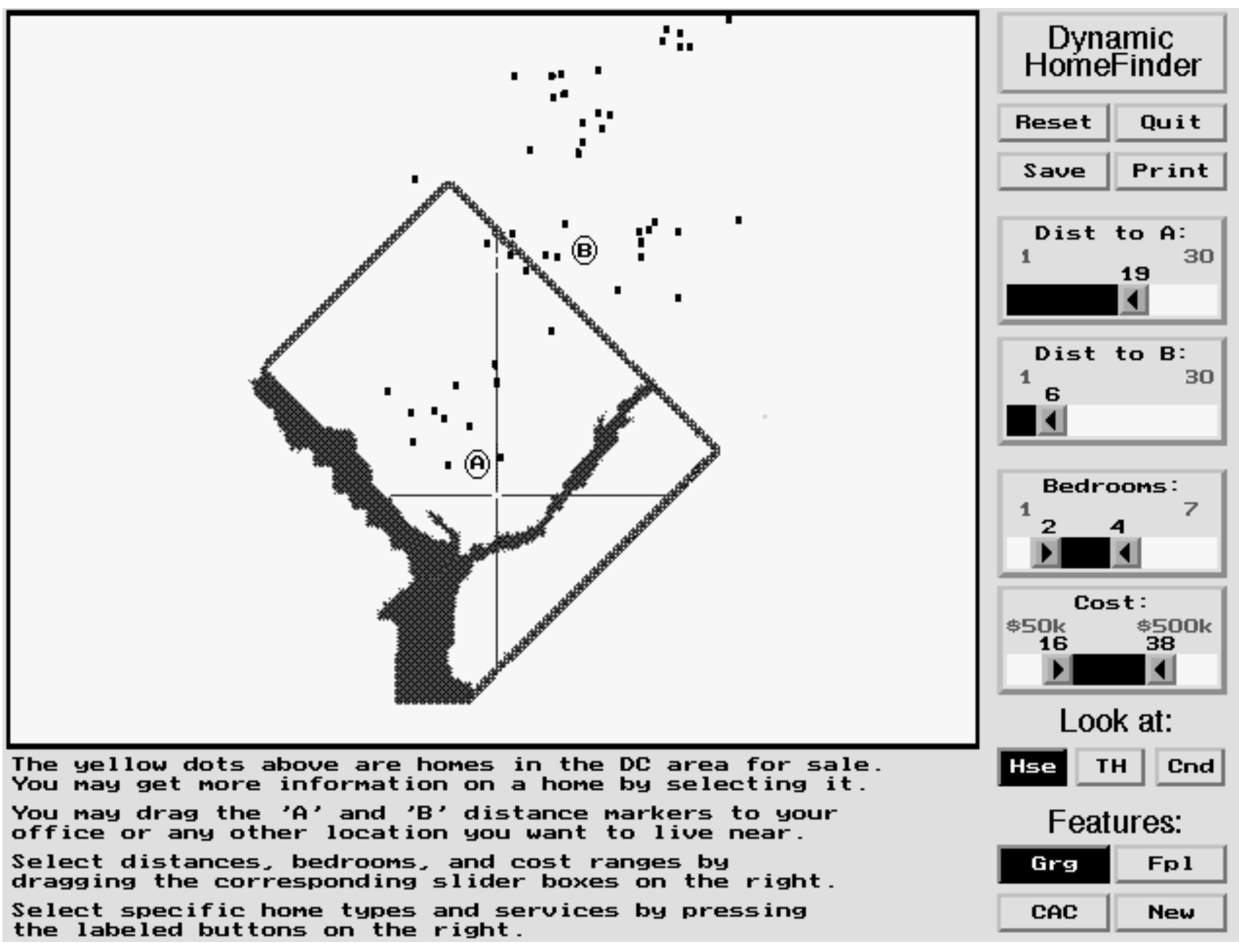

Figure 1: The DC Homefinder dynamic query system enables users to adjust the sliders for location, cost, number of bedrooms, home type (house, townhouse, or condominium), and features (Garage, Fireplace, Central Air Conditioning, or New construction). The results are shown as points of light which can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

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 Symantec), 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 (Figure 2). Subjective satisfaction dramatically favored the dynamic queries. One subject remarked about the dynamic queries treatment: “I don’t want to stop, this is fun!”

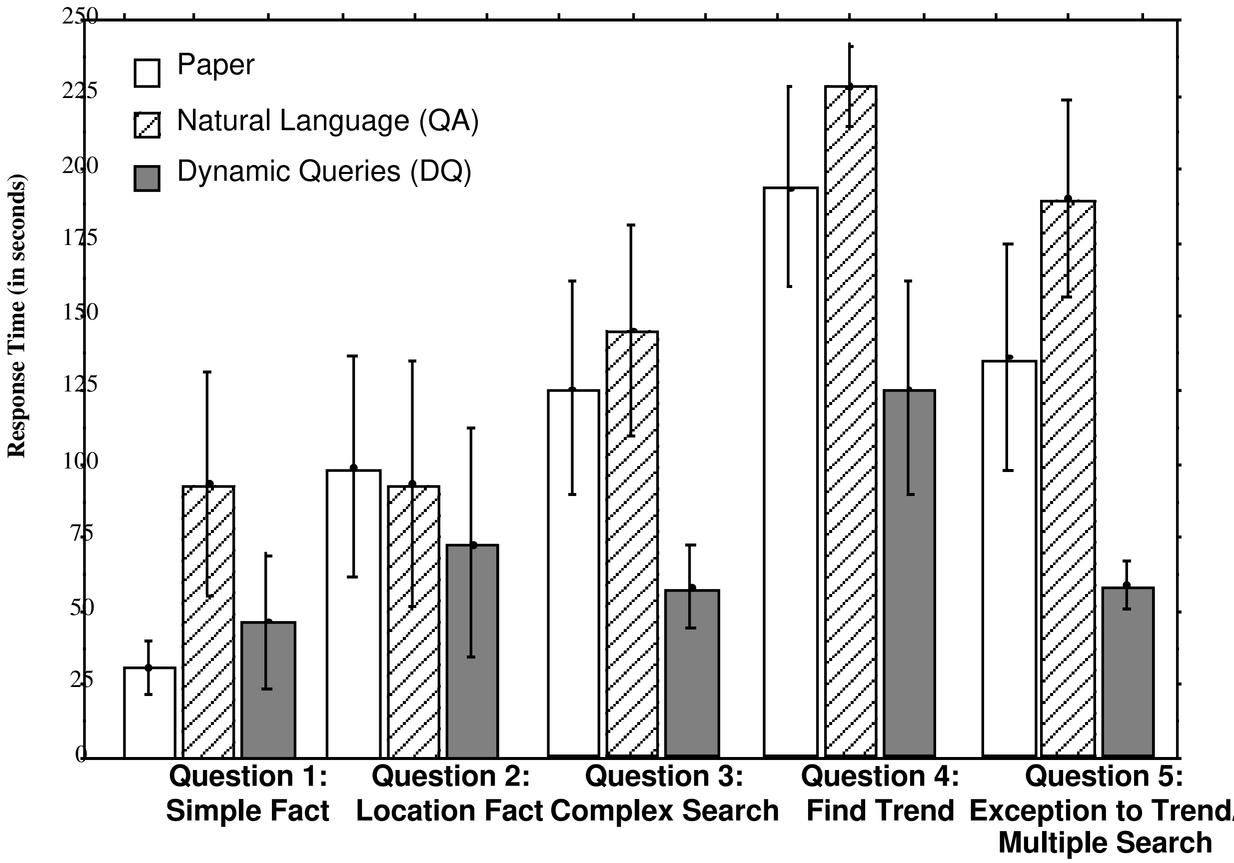

Figure 2: Means and standard deviations for response times of 18 subjects to five queries with paper, natural language, and dynamic queries interfaces. The results show an advantage for the dynamic queries interface as query complexity increased (Williamson & Shneiderman, 1992).

Another geographic application we built highlighted entire states of the United States that had cancer rates above a specified value (Plaisant, 1993) (Figure 3). 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 colors, accomplished with color indexing on the palette, enabled users to detect changes in cancer rates over time and correlations with demographic variables. An extended system built in our lab is being distributed to statisticians by the National Center for Health Statistics.

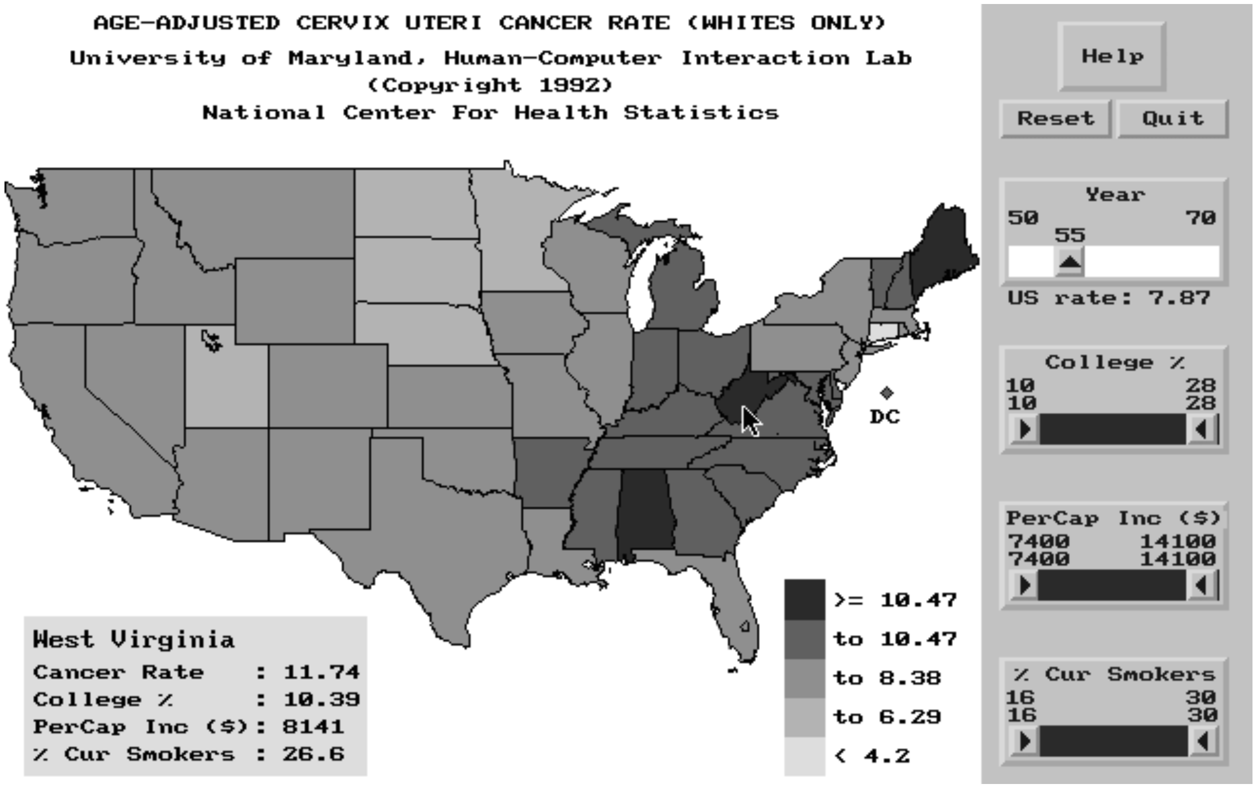

Figure 3: This dynamic query shows cervical cancer rates from 1950 to 1970 in each state. Adjustments can be made to the year and state demographic variables such as the percentage of college education, per capita income, and percent smokers.

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 queries 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 4). 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 queries approach to the chemical table of elements was tested in an empirical comparison with a form fill-in query interface (FG) and a textual output interface (FT). 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 (Figure 5).

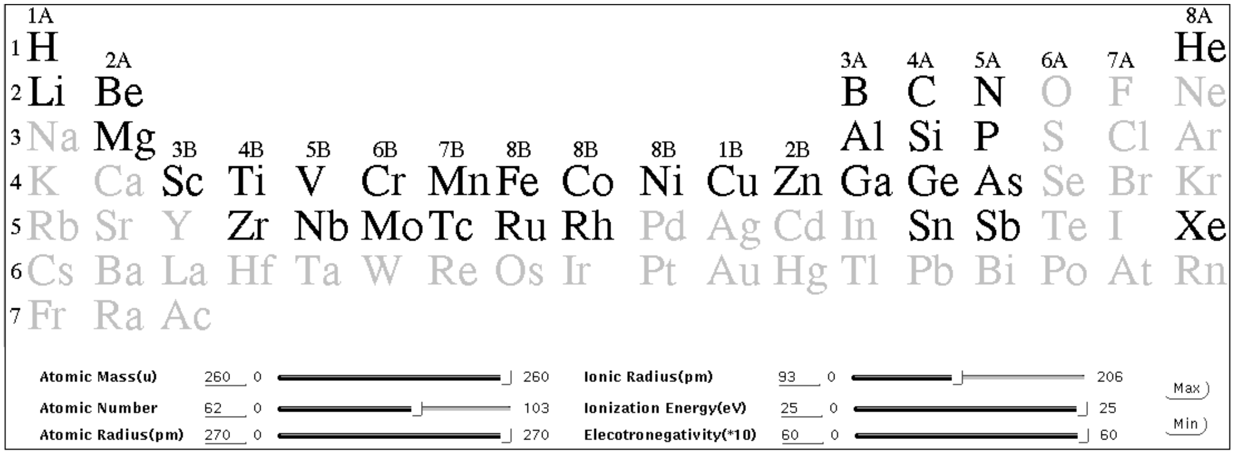

Figure 4: The chemical table of elements makes a natural visual display for information on chemical properties. Chemical matching the query are shown in red. Gaps and jumps are easily found.

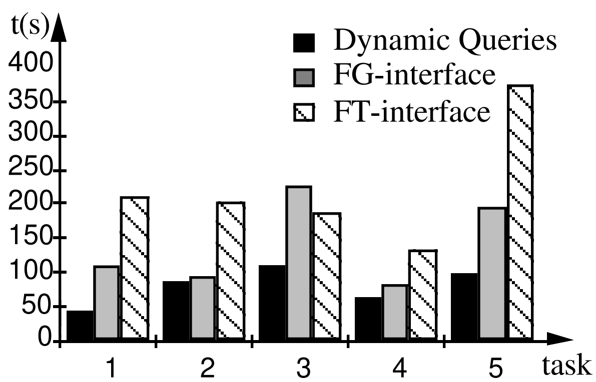

Figure 5: Means response times for 18 subjects performing five tasks with the dynamic queries, form fill-in plus graphic output (FG), and form fill-in with tabular output (FT). The results strongly favor the dynamic queries interface (Ahlberg, Williamson & Shneiderman, 1992).

When there are no natural graphical displays for the output, dynamic queries can still be implemented with result sets shown in a traditional alphanumeric tabular display (Figure 6). 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.

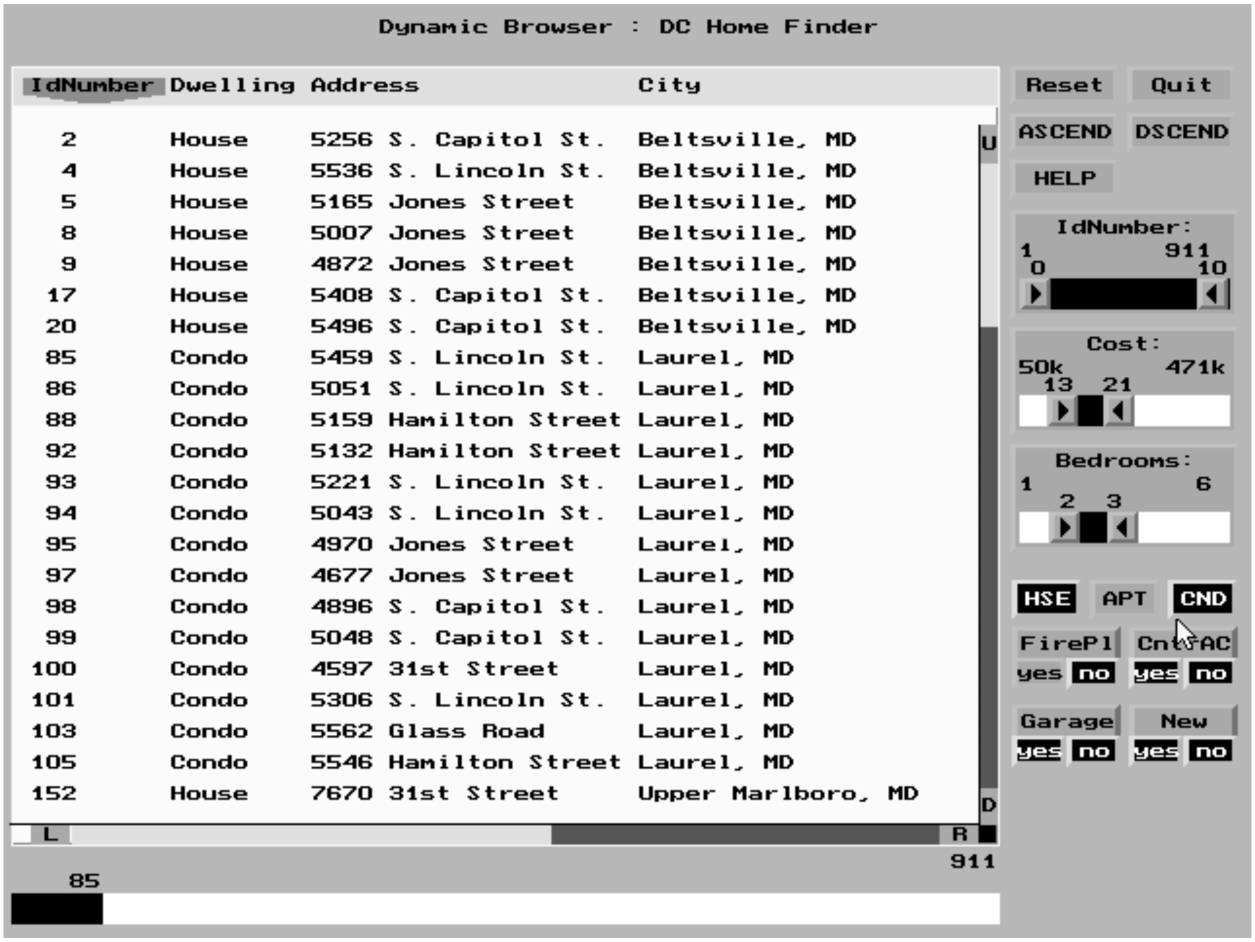

Figure 6: Even when there is no natural graphic framework for a dynamic query display, the method can be used with tabular alphanumeric output. As users adjust the sliders and buttons for the query, the result bar along the bottom indicates how many items match. When the users stop moving the sliders 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 7). 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 directory size increases. These results help us develop guidelines and theories about how to design displays for dynamic queries.

Figure 7: 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. The advantages of rapid reformulation of queries was hinted at in the early work on textual interfaces in Rabbit (Williams, 1984). The advantages of visual input and output was explored in some statistical display programs (Buja, McDonald, Michalak, & Stuetzle, 1991). Dynamic queries achieve these advantages by applying direct manipulation strategies (Shneiderman 1992):

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

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 may help users 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

The disadvantages of dynamic queries stem largely from their poor match with current hardware and software systems. First, the requirement for rapid performance in search algorithms and display strategies cannot be easily satisfied with current database management tools. Therefore, we are exploring which data structures and algorithms accommodate large datasets and permit rapid access (Jain & Shneiderman, 1993). Rapid graphical display methods are especially useful for dynamic queries, but these are not widely available.

Second, application specific programming is needed to take 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 plus software toolkits would make dynamic queries easier to integrate into existing database and information systems. Alternatively, dynamic queries could be generated by User Interface Builders or User Interface Management Systems.

Third, our current dynamic queries implement only simple queries that are conjunctions of disjunctions, plus range queries on numeric values. Our filter/flow metaphor offers one approach to providing full boolean functionality (see Figure 9), but this prototype must still be refined and implemented within database management systems (Young & Shneiderman, 1993). 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.

2. RESEARCH DIRECTIONS

> "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 et al., 1987)

While our initial implementations of dynamic queries have generated enthusiasm, we are more aware of challenges than successes. Research opportunities appear rich in database and display algorithms and user interface design.

2.1 Database and display algorithms

Since rapid update of the display is essential, existing algorithms to store and retrieve multi-dimensional information need refinement (Bentley & Friedman, 1979; Samet, 1989). For small databases that fit in main memory, we experimented with array indexing, grid structures, quad trees, and k-d trees. Linear array structures with pointers were effective with small databases, but their inefficient use of storage limited the size of the databases that could be handled. With uniform distributions grid file structures were efficient while the quad and k-d trees became more attractive as the distributions became more skewed (Jain & Shneiderman, 1993)

For larger, disk-based databases, alternatives include 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.

In dynamic queries, there is always a current query result on display and each new query is a slightly enlarged or contracted version of the current query. It seems that special data structures and algorithms might allow rapid update of 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 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.

2.2 User-Interface Design

As humans we have the ability to recognize the spatial configuration of elements in a picture and notice the relationships among elements quickly (Tufte, 1983; Foley et al., 1990). 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. However, our understanding of when and how to apply these methods is poor, and basic research is needed.

Graphical display properties such as color (hue, saturation, brightness), texture, shape, border, blinking, etc. are of primary interest. Color is the most effective of these visual display properties, and it can be an important aid to fast and accurate decision making (Ding & Mateti, 1990; MacDonald, 1990; Marcus, 1992). 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 (Blattner, Greenberg &

Kamegai, 1992).

Basic research on color, sound, size and shape coding, etc. needs to be reexamined in the context of dynamic queries. Rapid and smooth screen changes are understood to be essential for perception of patterns, but we would like to develop more precise requirements to guide designers. In our experience, delays of more than 2-3 tenths of a second are distracting, but precise requirements with a range of situations and users would be helpful.

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

- design widgets to specify multiple ranges of values, such as between 14 and 16 or between 21 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 7). A natural possibility is to create a two dimensional space such as a scattergram. Instead of showing homes as points of light on a city map, they could be points of light on a graph whose axes were the age of the house and its price. Sliders could still be used for number of bedrooms, quality of schools, real estate taxes, etc.

Treemaps are another 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 observe quickly the changes to the treemap over time and find trends or spot problems.

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 8). 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 dotted areas indicate impossible selections, for example, the cheapest 7 bedroom house is $310,000).

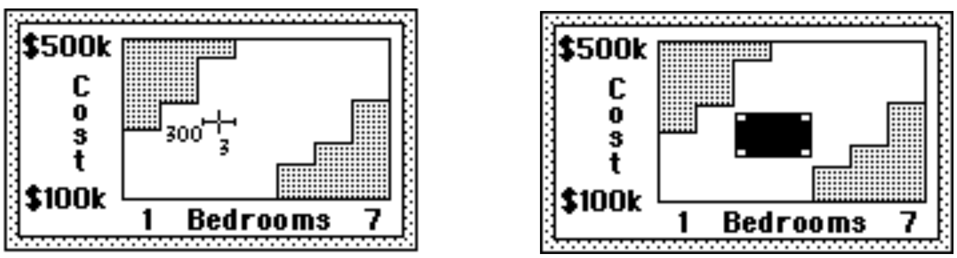

Figure 8: Two prototype two-dimensional widgets. The left one specifies a point indicating the number of bedrooms (3) and cost of a home ($300,000) with a single selection. The right one specifies a range of bedrooms (3 to 4) and cost ($180,000 to $320,000).

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 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 3-D 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.

Another input issue is the ways of specifying alphanumeric fields. While a simple type-in dialog box is possible, more fluid ways of roaming through the range of values is helpful. To this end we began to develop an 'alphaslider' to allow users to quickly sweep through a set of items that might be the days of the week or the 6000 names of actors in a database of movies (Ahlberg & Shneiderman, 1994).

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 we have developed the filter/flow model (Young & Shneiderman, 1993). 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.

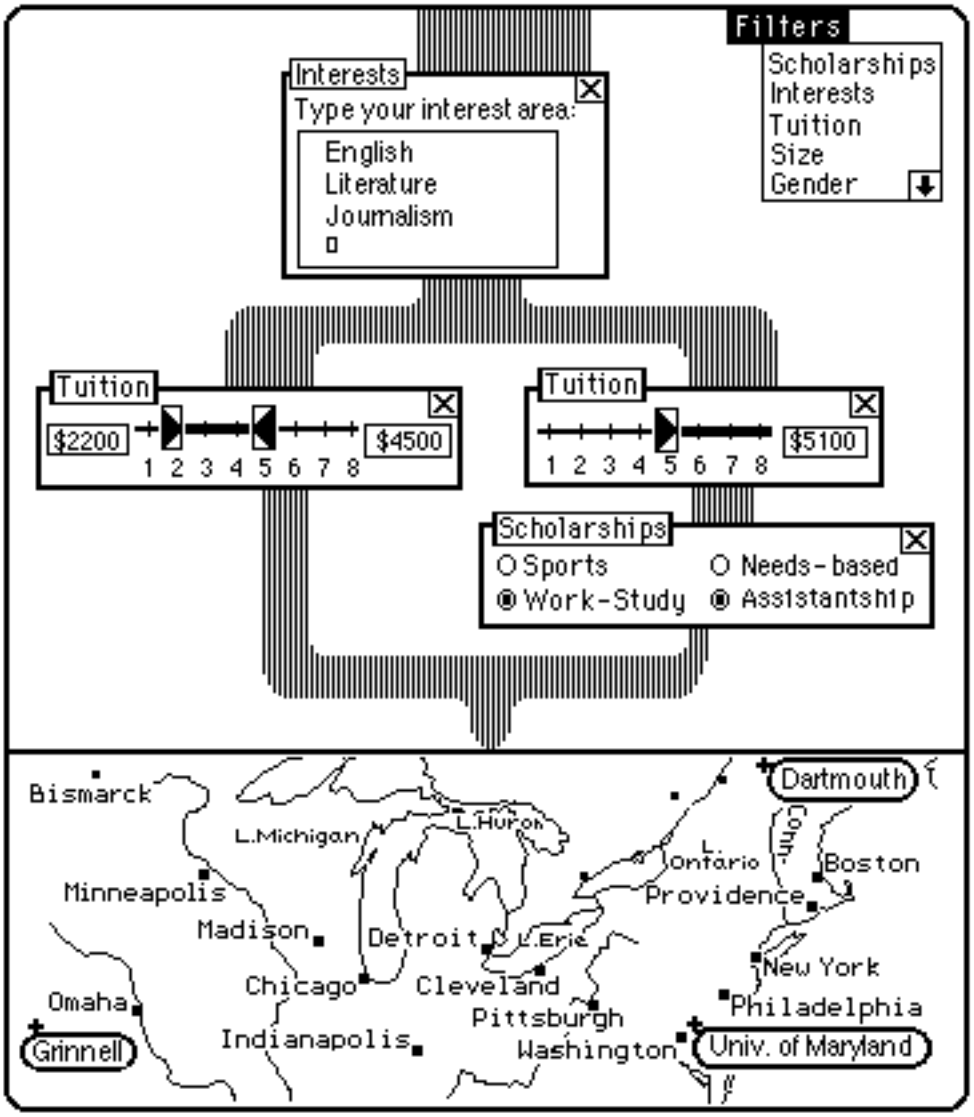

Figure 9: A mockup of a filter/flow boolean query ((Interests = English or Literature or Journalism) AND ((Tuition greater than or equal to $2200 or less than or equal to $4500) OR ((Tuition greater than or equal to $5100) AND (Scholarships are available by Work-Study or Assistantship)))) combined with map output to show the result (Dartmouth, Grinnell, and the Univ. of Maryland).

3. SUMMARY

Dynamic queries offer a lively new direction for database query. Many problems that are difficult to deal with using a keyword-oriented command language become tractable with dynamic queries. Contemporary computers have become fast enough that this direct manipulation approach can be applied for modest-sized problems and still ensure an update time of under 100 msec. 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 become a general approach that is attached to every database system, spreadsheet, and many stand-alone applications. The benefits accrue both to novice and expert users. Research directions include: (1) database and display algorithms, and (2) user interface design.

Acknowledgements: Thanks for the creative implementation efforts of Christopher Ahlberg, Christopher Williamson, Holmes Liao, Boon-Teck Kuah, 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. and Shneiderman, B., 1994. Alphaslider: a rapid and compact selector, *Proc. ACM CHI'94 Conference* (to appear).

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

Beard, D. V. and Walker II, J. Q., 1990. Navigational techniques to improve the display of large two-dimensional spaces. *Behavior & Information Technology 9*, 6, 451-466.

Becker, R. A. and Cleveland, W. S., 1987. Brushing scatterplots, *Technometrics 29*, 2, 127-142.

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.

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.

Borgman, C. L., 1986. Why are online catalogs hard to use? Lessons learned from information-retrieval studies, *Journal of the American Society for Information Science 37*, 6, 387-400.

Buja, A., McDonald, J. A., Michalak, J., and Stuetzle, W., 1991. Interactive data visualization using focusing and linking, *Proc. IEEE Visualization '91*, 156-163.

Consens, M. P., Cruz, I. F., and Mendelzon, A. O., 1992. Visualizing queries and querying visualizations, *ACM SIGMOD Record 21*, 1 (March 1992).

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.

Eick, S., 1993. Data visualization sliders, AT&T Bell Laboratories Report, Napervillie, IL.

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.

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.

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

Hartson, H. R., Siochi, Antonio C., and Hix, D., 1990. The UAN: User-oriented representation for direct manipulation interface designs, *ACM Trans. on Information Systems 8*, 3, 181-203.

Jain, V. and Shneiderman, B., 1993. Data structures for dynamic queries: An analytical and empirical evaluation, University of Maryland, Department of Computer Science Technical Report #3133.

Johnson, B. and Shneiderman, B., 1991. Tree-Maps: A space-filling approach to the visualization of hierarchical information structures, *Proc. IEEE Visualization 1991*.
Johnson, B. and Turo, D., 1992. Improving the visualization of hierarchies with treemaps: Design issues and experimentation *Proc. IEEE Visualization 1992.*
Kim, H. J., Korth, H. F., and Silberschatz, A.,1988. PICASSO: A graphical query language, *Software: Practice and Experience 18*, 3, 169-203.
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.
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, *Communications of the ACM 13*, 11, 660-665.
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.
McCormick, B., DeFanti,T. and Brown, R. et al., 1987. Visualization in scientific computing, Computer Graphics, November 1987 ACM SIGGRAPH.
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.
Norman, K., 1991. *The Psychology of Menu Selection: Designing Cognitive Control at the Human/Computer Interface*, Ablex Publishing Corp., Norwood, NJ, 350 pages.
Plaisant, C., 1992. Dynamic queries on a health statistics atlas, Forthcoming Technical Report, Human-Computer Interaction Laboratory, University of Maryland, College Park, MD.
Robertson, G. G., Card, S. K., and Mackinlay, J. D., 1993. Information visualization using 3-D interactive animation, *Communications of the ACM 36*, 4, 56-71.
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, 149-154.
Shneiderman, B., 1983. Direct manipulation: A step beyond programming languages, *IEEE Computer 16*, 8, (August 1983), 57-69.
Tufte, E., 1983. *Visual Display of Quantitative Information*, Graphics Press, Cheshire, CT.
Welty, C., 1985. Correcting user errors in SQL, *International Journal of Man-Machine Studies 22*, 463-477.
Williams, M., 1984. What make RABBIT run? *International Journal of Man-Machine Studies 21*, 333-352.
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*.
Young, D. and Shneiderman, B., 1993. A graphical filter/flow model for boolean queries: An implementation and experiment, *Journal of the American Society for Information Science 44*, 4 (July 1993), 327-339.

Table of Figures