The Dynamic HomeFinder: Evaluating Dynamic Queries in a Real-Estate Information Exploration System

Christopher Williamson and Ben Shneiderman

Human-Computer Interaction Laboratory
Center for Automation Research and Computer Science Department
University of Maryland
cwilliam@cs.umd.edu ben@cs.umd.edu

Abstract

We designed, implemented, and evaluated a new concept for visualizing and searching databases utilizing direct manipulation called *dynamic queries*. Dynamic queries allow users to formulate queries by adjusting graphical widgets, such as sliders, and see the results immediately. By providing a graphical visualization of the database and search results, users can find trends and exceptions easily. User testing was done with eighteen undergraduate students who performed significantly faster using a dynamic queries interface compared to both a natural language system and paper printouts. The interfaces were used to explore a real-estate database and find homes meeting specific search criteria.

1 Introduction and Background

Most database systems require the user to create and formulate a complex query, which presumes that the user is familiar with the logical structure of the database (Larsson, 1986). The queries on a database are usually expressed in high level query languages such as SQL. This works well for many applications, but it is not a fully satisfying way of finding data. For naïve users these systems are difficult to use and understand, and they require a long training period (Kim, Korth, & Silberschatz, 1988).

Clearly there is a need for easy to use, quick and powerful query methods for database and information retrieval. Direct manipulation has proved to be successful for other applications such as display editors, spreadsheets, computer aided design/manufacturing systems, computer games and graphical environments for operating systems such as the Apple Macintosh (Shneiderman, 1983). Direct manipulation interfaces support:

- Continuous visual representation of objects and actions of interest.
- Physical actions or labeled button presses instead of complex query syntax.
- Rapid, incremental, reversible operations whose results are immediately visible.
- Layered or spiral approaches to learning that permit usage with minimal knowledge.

One of the great advantages of direct manipulation is that it places the task in the center of what users do. Rutowski (1982) describes it as: "The user is able to apply intellect directly to the task; the tool itself seems to disappear." The success of direct manipulation can be understood in the context of the syntactic/semantic model which describes the different levels of understanding that users have. Objects of interest are displayed so that actions are directly in the high level semantic domain. Users do not need to decompose tasks into syntactically complex sequences. Thus each command is a comprehensible action in the problem domain whose effect is immediately visible. The closeness of the command action to the problem domain reduces user problem-solving load and stress.

For information retrieval and database exploration, there have been several attempts to use direct manipulation and escape some of the pitfalls of contemporary boolean systems (Zloof, 1975; Kim, Korth, & Silberchatz, 1988; Williams, 1984). The first two systems do not provide any visual display of actions. Zloof's Query-by-Example relies on users entering values with a keyboard. Even though Kim, et al.'s PICASSO supports input through mouse and menus, it requires users to perform a number of operations in each step. The combination of graphical input/output is not applied in any of these systems.

Other solutions have come about with the invention of natural-language query systems. Some of these systems (Salton and McGill, 1983; Harman and Candela, 1990), usually using statistical ranking and/or relevance feedback, do a fine job retrieving textual information. They are, however, very awkward to apply to graphical information systems and they have other well-documented problems (Helander, 1988). An alternative database interface utilizing dynamic queries (Ahlberg, Williamson & Shneiderman, 1991) would:

- represent the query graphically,
- provide visible limits on the query range,
- provide a graphical representation of the database and the query result,
- give immediate feedback of the result during every query adjustment, and
- allow novice users to begin working with little training, but still provide expert users with powerful features.

With dynamic queries, the query is represented by a number of sliders (Figure 1). Each slider consists of a label, a field indicating its current value, a slider bar with a drag box and a value at each end of the slider bar indicating minimum and maximum values. Sliding the drag box with the mouse changes the value of the slider. Clicking with the mouse on the slider bar increases or decreases the value one step at a time. The database is represented on the screen in graphical form. This paper describes a program dealing with real-estate and, accordingly, the map of the area was chosen as the representation. The result of the query can then be highlighted by coloring, changing points of light, marking of regions, or blinking.

Figure 1. A slider from the Dynamic HomeFinder.

The combination of a graphical query and graphical output fits well into the principles of direct manipulation. The slider serves as a metaphor for the operation of entering a value for a field in the query. Norman (1988) promotes the use of sliders since they also provide a mental model of the range. Changing the value is done by a physical action - sliding the drag box with a mouse - instead of entering the value by keyboard. By sliding the drag box back and forth and getting real-time updates of the query results, it is possible to do dozens of queries in just a few seconds. The operation is incremental and, if the query result is not what users expected, the operation is reversible by just sliding the drag box in the opposite direction. Error messages are not needed - there is no such thing as an 'illegal' operation or a syntax error.

The interaction between the database visualization and the query mechanism is important. The sliders have to be placed close to the visual presentation to reduce eye movement. The highlighting of elements should be in harmony with the coloring scheme of the slider. For example the color of the area to the left of the drag box on the slider bar is the same as the highlighted elements in the visualization, because the values to the left of the drag box are the values that satisfy the query.

2 Dynamic Queries Interface to Real-Estate

The program used for the experiment applied dynamic queries to real-estate. Finding a home is a laborious task for those that have experienced it. The two most common methods currently used are paper and SQL-like database systems. Newspaper and printed listings have survived to still be the most common means for finding a home. These provide little organization, short of being sorted by one field, but are easy-to-use for the novice. In the last few decades, SQL-like systems have appeared to support more complex queries.

Unfortunately, SQL-like systems require training and/or an intermediary; almost none allow the actual homebuyer to perform the search. Further, these systems suffer the problems of most command-line query systems: slow, difficult to use, little feedback, nonreversible, and too many boolean logic errors. In addition, these systems offer no easy way to specify locations. The homebuyer must know neighborhood names, and attempt to figure out if a given neighborhood is near where they wish to live. Finally, these systems suffer from the classic all-or-nothing phenomenon. This commonly occurs when the searcher does not know the contents of the database (as is usually the case in real-estate) and therefore attempts a query that is too general or too restrictive. As a result, the homebuyer alternates between too many and too few results - frantically and laboriously guessing towards the middle to produce a reasonable set of homes. Recently, good natural-language query systems, such as Q&A by Symantec, have been developed in response to complaints about SQL that instead allow queries to be stated in English. This should reduce training with syntax and help novice users, but it does little to correct the other faults of previous query systems.

The dynamic queries interface (Figure 2) provides a visualization of both the query formulation and corresponding results. This application was built using the C programming language. A map of the District of Columbia area is displayed on the left. The homes that fulfill the criteria set by the user's current query are shown as yellow dots on the map. Users perform queries, using the mouse, by setting the values of the sliders and buttons in the control panel to the right. The query result is determined by ANDing all sliders and buttons.

The dynamic homefinder interface is best explained through an example. Take a hypothetical situation where a new professor, Dr. Jones, has just been hired by the University of Maryland. She might encounter this tool in a touchscreen kiosk at a real-estate office or at the student union. She selects the location where she will be working by dragging the 'A' on the map. Next, she selects where her husband will be working, downtown, near the capitol, by dragging the 'B'. Figure 2 shows the interface after Dr. Jones has dragged the 'A' and 'B' indicators to her desired locations (the indicators are more visible in Figure 4).

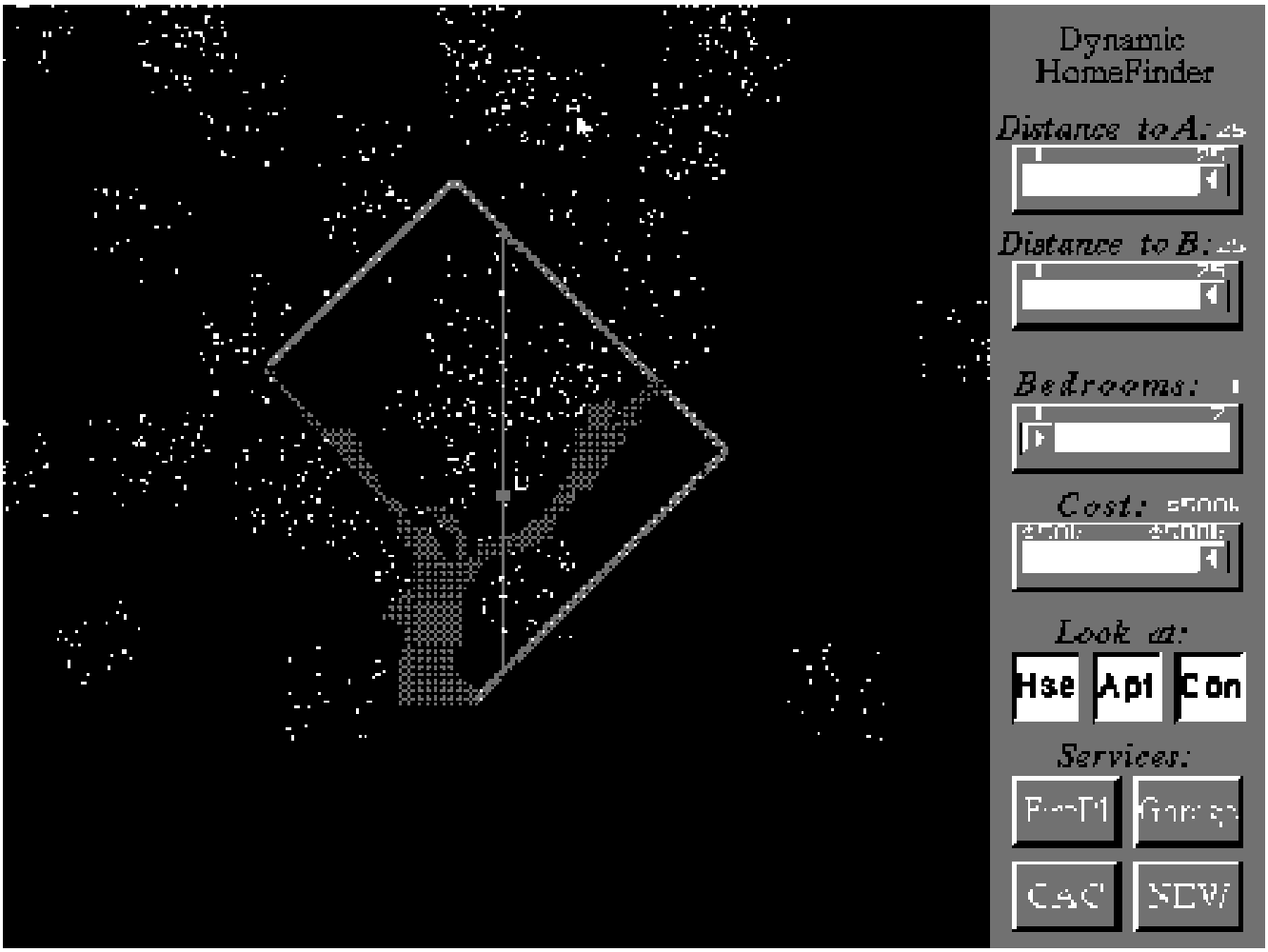


Figure 2. Dynamic HomeFinder (DQ interface) with all homes displayed and A&B markers set. In order to make these snapshots more readable in print, the pallette has been altered; the actual color scheme used was considerably more readable and pleasing than these snapshots depict.

Dr. Jones would like to ride her bicycle to work, if possible, so she sets the 'Distance to A' slider on the right to 5 miles or less. This is indicated by the highlighted region of the slider now indicating from 0-5 miles. Her husband is taking the Metro, so the distance to his office is not very important. Figure 3 shows how the screen looks after she has adjusted the 'Distance to A' slider. Note that this is done instantaneously in a fluid-like manner as she moves which cannot be captured with snapshots, but which enables her to quickly see *how* homes are eliminated as she narrows the distance requirement.

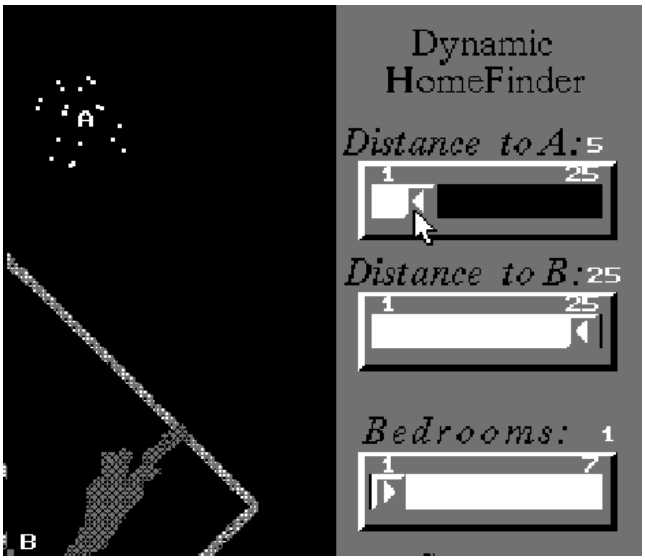


Figure 3. HomeFinder with all homes within a 5-mile radius of the 'A' marker displayed.

Dr. Jones is only interested in houses, not in apartments or condominiums so she toggles those buttons off. Finally, she drags the bedrooms slider down to 4, since she needs at least four bedrooms, she could have more (for a guest room or study for example), again indicated by the highlighting in Figure 4 showing that houses with 4-7 bedrooms are now being displayed. In Figure 4, she also drags the cost slider to $140,000, a modestly-priced home where she used to live. Here we encounter the all-or-nothing phenomena as Dr. Jones has eliminated too many houses with her query. This is easily solved as she realizes that houses must be more expensive in this area. Dr. Jones drags the cost slider up to $220,000 in Figure 5, a price that many more houses in the area fulfill.

Finally, just out of curiousity, Dr. Jones clicks on the 'Garage' button in Figure 6 only to find that few houses have a garage in the price range and area she is looking at. Once she has narrowed her query, it is easy for Dr. Jones to experiment, seeing what services the homes offer, or what is available if she was willing to pay a little more, and so on. In this way the interface encourages exploration and bolsters user confidence.

Although there is no figure depicting it, a mouse click on any of the homes (represented by the yellow dots) brought up a pop-up window with detailed information on that specific home.

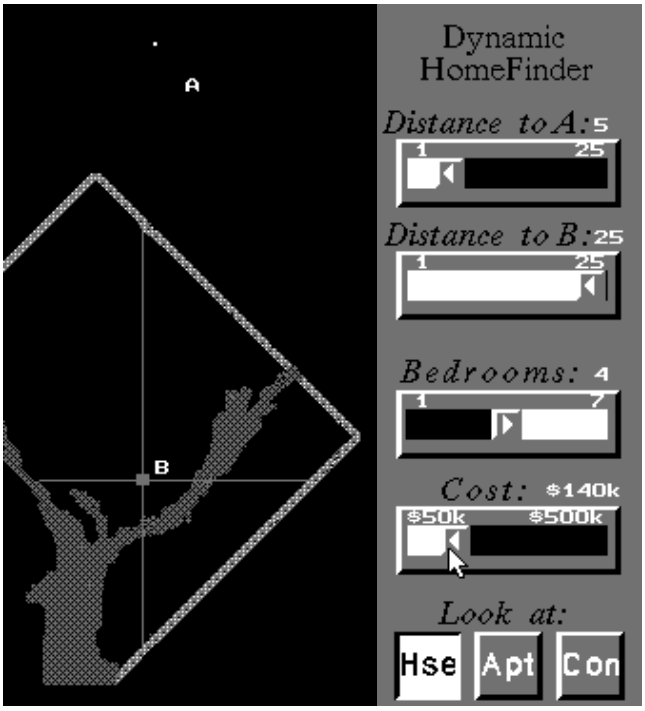


Figure 4. HomeFinder with all houses within a 5-mile radius of the 'A' marker AND have 4 or more bedrooms AND cost less than or equal to $140,000.

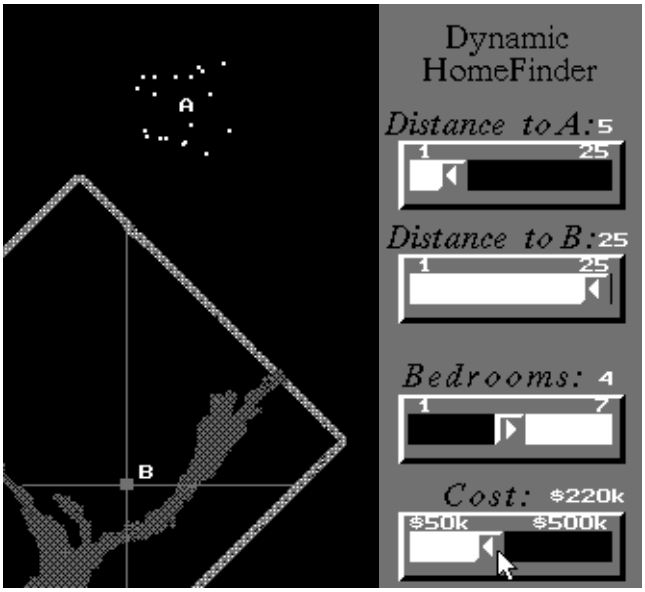


Figure 5. Closeup of Dynamic HomeFinder with all houses within a 5-mile radius of the 'A' marker AND have 4 or more bedrooms and cost less than or equal to $220,000.

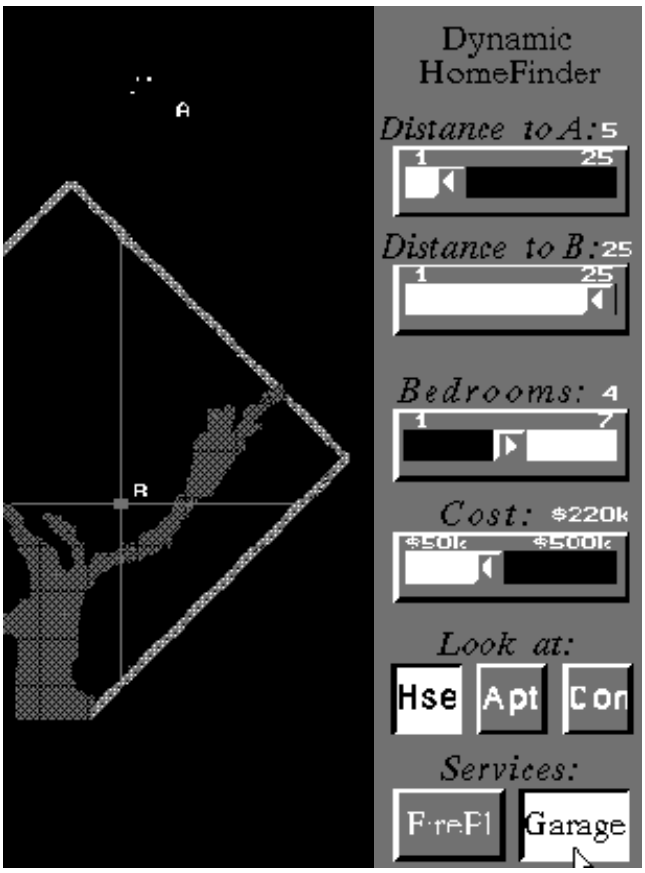


Figure 6. Dynamic HomeFinder with all houses within a 5-mile radius of the 'A' marker AND have 4 or more bedrooms AND cost less than or equal to $220,000 AND have a garage.

3 User Experiment

This experiment compared three different interfaces for database query and visualization: a dynamic queries interface, a natural language query system known as 'Q&A', and a traditional paper listing sorted by several fields. The alternative interfaces were chosen to find out how dynamic queries would fare against the two most common methods currently used to search real-estate databases. Using a within-subjects counter-balanced design, subjects answered a series of five types of questions for each of the three interfaces were presented in random order. The independent variable in the experiment was the type of interface with three different treatments: dynamic queries (DQ), natural language retrieval (QA), and paper listings (Paper). The observed dependent variables were time to find correct answer for each of five questions and the subjective satisfaction with each interface.

The primary hypothesis was that the dynamic queries interface, providing both a graphical query input and a graphical visualization of the search result, would give the best user performance results and would be rated highest in user satisfaction. Performance results were measured as the time until correct answer for each question.

Eighteen undergraduate psychology students, 9 females and 9 males from the University of Maryland subject pool, participated voluntarily in the experiment. Only one subject had previous knowledge of real-estate in the area. Both computer interfaces were run on an IBM PS/2 model 70 (16 MHz 80386) with a 12-inch VGA color monitor and mechanical two-button mouse. All interfaces involved querying information on 944 imaginary homes with varying criteria from real-estate in the Washington D.C. Metropolitan area. Although using real home-seekers would be ideal, a reasonable compromise was made by using novices with varying backgrounds.

3.1 Dynamic Queries interface (DQ)

This interface was already described in detail under section 1.2. As shown previously in Figure 4, it is possible to drag the 'A' icon to a certain location where the homebuyer might work, for example. When the distance slider is moved to 5, all houses within 5 miles of that location are then highlighted. However, these distance sliders and the 'A' and 'B' icons on the map were not used in the experiment, since the other two interfaces could not provide this service.

3.2 Natural language query interface (QA)

The natural language query interface (Figure 7) used was the 'Intelligent Assistant' of a popular commercial package from Symantec, Inc. known as 'Q&A'. This software allows users to pose queries using English. The user types the English query, the system converts it into a logical database query (Figure 7), and then displays the information requested (Figure 8). A 386 machine with hard disk was used so the search time was only a few seconds, fast enough that computation speed was not a significant factor. No graphical output was provided, a textual listing of the homes that satisfied the query were displayed as shown in Figure 8.

```
WHERE ARE THE 3 MOST EXPENSIVE HOUSES WITH A FIREPLACE?
```

```
                    Shall I do the following?

        Create a report showing
          the Address and
          the Neighborhood and
          the Cost and
          the Home Type
        from the forms on which
          the Home Type includes "HOUSE" and
          the Fireplace is YES and
          the Cost is maximum

        Yes - Continue            No - Cancel Request
```

```
DQ.DTF
Esc-Cancel  F10-Continue
```

Figure 7. Natural language system (QA interface) processing and converting English query.

```
WHERE ARE THE 3 MOST EXPENSIVE HOUSES WITH A FIREPLACE?
```

Address	Neighborhood	Cost	Home Type
7924 Jones Street	Chevy Chase, MD	$411,950	House
4719 Dorset Ave.	Chevy Chase, MD	$678,235	House
1287 Highland Ct.	Potomac, MD	$782,125	House

```
DQ.DTF
Esc-Cancel  F10-Continue
```

Figure 8. Q&A displaying results of converted English query.

3.3 Paper interface

There were three sets of laser-printed paper listings in approximately 11-point Courier font, one sorted by cost, one by the number of bedrooms, and one alphabetically by the neighborhood name. Each 10-page listing contained all the information on each of the 944 homes, one-per-line, as shown in Figure 9.

```
[Key: Bed=bedrooms, Fp=fireplace, Gr=garage, Ac=central air, Nw=new]

ID  Type  Address              Neighborhood        Cost     Bed Fp Gr Ac Nw
39  Apt   3792 Campus Drive    Beltsville, MD      $80,950  3  N  N  N  N
54  Apt   4634 Baltimore Blvd. College Park, MD    $90,250  2  N  Y  N  N
230 House 2352 Glass Road      Bladensburg, MD    $100,230  3  Y  Y  N  N
```

Figure 9. Sample lines from paper listing.

3.4 Experiment procedure and tasks

A counter-balanced within-subjects design was used. The question sets were always given in the same order, and the interface order was random. The subjects were given a brief description of the tasks and were asked to sign a consent form. Each session lasted an hour and consisted of four phases for each interface condition:

1. Introduction and training: The experimenter set up the appropriate interface and briefly explained the interface to the subject. The subject was invited to try-out the system and get comfortable with it. Subjects were permitted unlimited time to try-out using the system and were free to ask any questions to the experimenter about the interface. Actual training time varied from two to ten minutes for each subject. The QA interface required significantly more training due to its less obvious querying mechanism. Subjects were informed on the more complex sorting and set operations available in the QA interface.

2. Practice task: A practice task (similar to the complex query question used) was given. During this task, subjects were free to ask questions about both the task and the interface.

3. Timed tasks: Five questions were given on paper. Subjects read each question and were asked if they fully understood it. This was done so as to eliminate variations in subject comprehension speed. When the subject was ready, the experimenter started timing. When subjects found the answer, they verbally expressed it. If it was correct, the experimenter recorded the elapsed time; if not, the experimenter asked the subject to try again. The interface was returned to its initial state between each question.

In deciding the tasks, an informal task analysis was done by asking a local realtor and a few clients what types of questions they would ask a database. The first two are very general, while the final three are probably more representative of 'typical' home-searches:

• *Simple fact* - Find a certain element fulfilling a simple criteria. An actual question of this type is "What is the cost of the cheapest apartment in the database?"

• *Simple neighborhood fact search* - Find the neighborhood (city) fulfilling a simple criteria. An actual question of this type is "What neighborhood has the most expensive houses?"

• *Complex search* - Find one or more elements meeting several criteria. This is the typical type of question a prospective homebuyer would likely ask. An actual question of this type is "What is the address of the cheapest house that has 5 or more bedrooms AND has both a garage and central air conditioning?"

• *Find a trend* - Find the trend for some field. This task requires subjects to create mental picture of how a field changes through the database. An actual question of this type is "Is there a general trend (and if so what is it) of house prices from cheapest to most expensive? (i.e. where are the more expensive houses, and where are the cheapest houses, do they seem to follow any general pattern?) Don't guess, you must find some examples to support your answer."

• *Find exception to trend/Complex search* - Find the exception to a trend in the database. This is, in fact, often what homebuyers are looking for. An actual question of this type is "There is a trend of houses to increase in cost with the number of bedrooms. Find the two bargain homes with the most bedrooms but are still inexpensive."

4. Subjective evaluation: Subjects were asked to fill-out a shortened QUIS (Chin, Diehl, & Norman, 1988) after having completed each interface. Although results for each of the 20 questions (scored on a 7-point scale) were computed, the sum score for each interface (out of a maximum possible of 140) is what is presented in the results section.

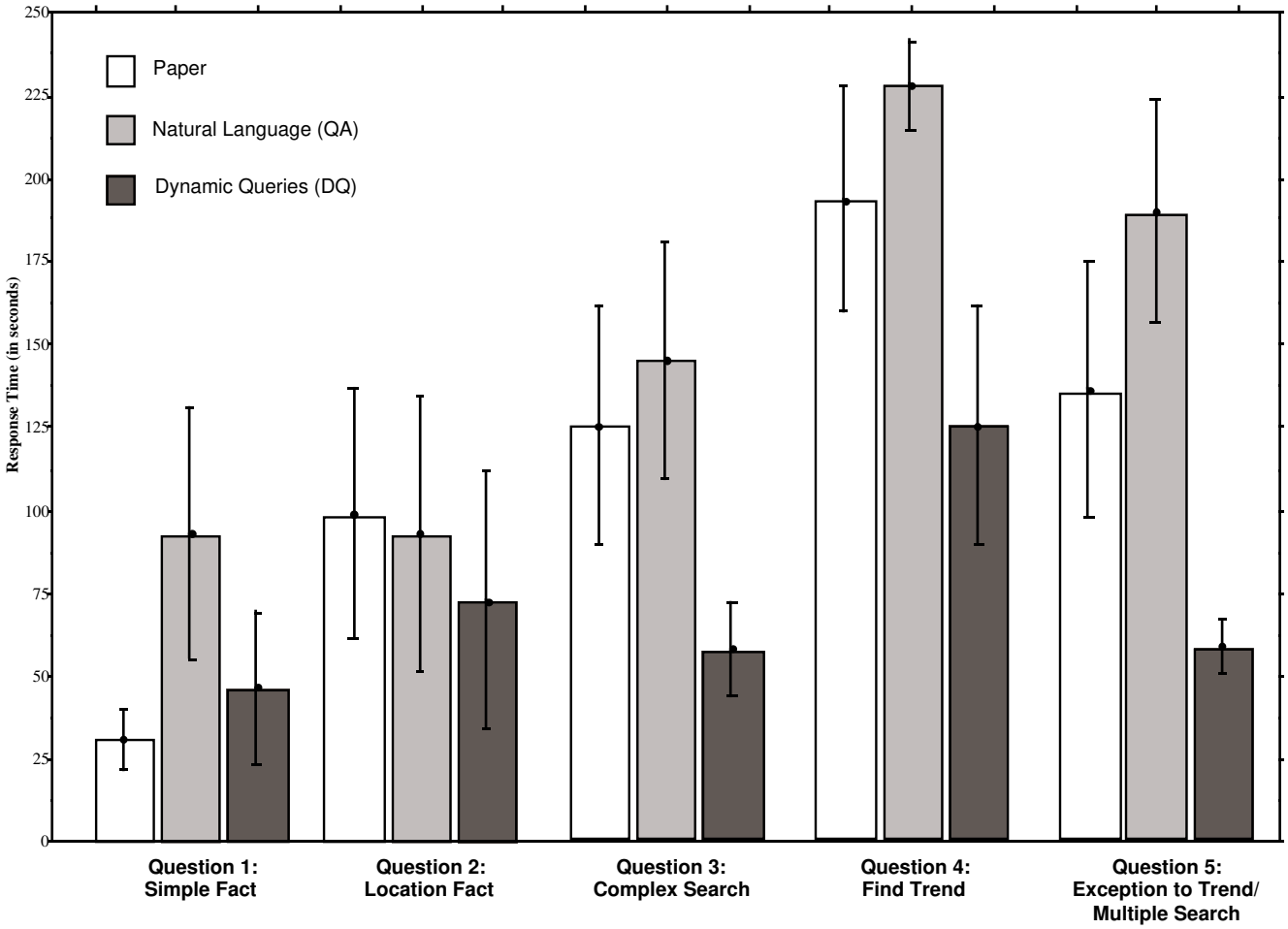
4 Experiment results

The results show a statistically significant difference for the dynamic queries interface over both the other two for all but one task. Analysis of the timed tasks was done using a factorial analysis of variance. An ANOVA showed no significant order or questionnaire difficulty effect (these two were compounded to one factor) with $F(2,51) = 1.17$ ($p > .05$). This, therefore, also shows that there was no significant difference between the three question sets used. Observing the mean times to complete each task shows a significant effect. The mean and standard deviations (in seconds) for each question and interface are shown in Table 1 along with the ANOVA and post-hoc analysis. The results for the QUIS are ratings out of a maximum possible of 140. These means, along with 95% error bars are graphed for comparison in Figure 10.

	QA	DQ	Paper	ANOVA
1	93.0 (76.0) →	46.3 (46.2)	30.6 (17.8)	$F(2,51) = 9.72$ ($p < .005$)
2	92.8 (83.2)	72.5 (78.0)	98.4 (75.3)	$F(2,51) = 0.66$ ($p > .005$)
3	144.8 (71.1) →	57.6 (28.4) ←	125.1 (72.4)	$F(2,51) = 11.6$ ($p < .005$)
4	227.9 (27.0) →	125.3 (72.4) ←	193.4 (68.5)	$F(2,51) = 17.4$ ($p < .005$)
5	189.9 (67.2) →	58.8 (16.3) ←	135.8 (77.0)	$F(2,51) = 29.4$ ($p < .005$)
QUIS	68 (42.3) →	110 (21.5) ←	84 (12.5)	$F(2,51) = 19.6$ ($p < .005$)

→ Scheffe .005 post-hoc showed significant difference favoring indicated interface

Table 1. Means and standard deviations in seconds for each interface and condition for five tasks and subjective satisfaction (out of a maximum of 140)



5 Discussion and conclusions

The primary hypothesis that the dynamic queries interface would give the best user performance results is supported in the results with a significant difference ($p < .005$) in speed, favoring dynamic queries, for all but one task. The dynamic queries interface also scored significantly higher on the satisfaction questionnaire. Surprisingly, for all but one task, paper was also better than the QA interface, although often not by a significant amount. The remarks given by the subjects and comments noted by the experimenter suggest several reasons why the dynamic queries interface faired better.

First, the dynamic queries interface was clearly the most 'fun', most likely due to its animated graphical nature, and this may have produced a motivational factor. The QUIS clearly indicated a preference for DQ (110) over QA (68) and Paper (84), with subjects rating DQ an average of 6.1 on a range from 1 to 7. Many subjects became very frustrated with the QA interface, attempting to figure out how to form a query (this is discussed more later). One subject remarked: "What the hell does this thing want, anyway?" Subjects seemed very comfortable with paper and usually attacked the question in a logical fashion, utilizing the sorted paper listings to their maximum benefit. The DQ interface was clearly preferred by the sighs of relief and the relaxed manner with which they formed queries. One subject said "I don't want to stop, this is fun!"

Second, the well-known issue of text readability may partially explain the success of both the DQ and paper interfaces. Although a VGA monitor was used, several subjects noted that it was easier to search the laser-printed listings than a listing on-screen using the QA interface. Subjects using the DQ interface rarely actually looked at the textual information on a specific home. One subject noted on their evaluation of QA that it "takes a while for your eyes to adjust to the small print."

Third, the subjects clearly had semantic difficulties with the QA interface. Grammatical, spelling, and typographical errors where made in query formulation. The QA's processor often resulted in unexpected queries. One common error was asking: "What is the cheaper house?" when the user meant "cheapest house." QA did not see the singular house and figure out that cheapest was actually what was desired, instead producing a listing of all houses with cost less than the average price. This is even more surprising since 9 out of the 15 task questions could actually be typed in literally, word for word, and QA would produce the correct answer. Clearly subjects spent a lot of mental effort trying to formulate the query, probably more than was necessary.

Fourth, the subjects clearly had problems with the classic boolean query problems. Specifically, they asked for ANDs which QA interpreted as a literal AND when OR was actually what the user wanted. Further, subjects often had problems with inclusion, such as specifying "more than 5 bedrooms" when they actually wanted "5 or more bedrooms."

Fifth, the number of ordinary typos was tremendous. Further, the editor in QA was quite poor, not allowing the user to edit the previous query without retyping it in. This led to much of the time spent typing in queries, which for most non-computer people is a slow process. Occasionally QA would not even report a typo, resulting in a query that failed, not due to the logic, but due to a typo. The subject would therefore think their query had failed and would consequently try another approach to the question.

Sixth, the QA interface often produced the correct answer but the subject missed it. Subjects expressed they were unsure if they had asked the right query to get the result and were unsure if it was therefore the correct answer. A few subjects tried multiple approaches and only answered when they got the same answer using each approach. A possible explanation may be that the subjects get so caught up in the query formulation they forget exactly what they were originally looking for. This frustrating feeling of being lost and unsure was not expressed with either of the other two interfaces.

Finally, the DQ interface's use of highlighting and display on one single screen was clearly a benefit for users. They were able to easily input the query with the sliders. Not one of the subjects showed any difficulty in using the sliders effectively. One subject explained the feeling well: "You can see them right up front...they're right there." The display of the results on a map made task 4 (finding a trend) clearly easier since the relative location could be viewed directly on the map, instead of having to refer to it from the city, state information.

The clear benefit of dynamic queries, particularly for trend and exception to trend questions is clear. The significant differences (F=29.4 and 19.5 respectively) for tasks 4 and 5 are dramatic. The fact that a time limit was placed at four minutes, probably reduced the effect somewhat, since over half the subjects could not answer task 4 using the QA interface in the time allotted. With the DQ interface, however, all but one subject was able to discern the trend in the time allotted.

The scores for task 5 showed dramatic differences, with the DQ interface averaging 58.8 seconds with a standard deviation of only 16.3 seconds, while the QA interface averaged 189.9 seconds, with a standard deviation of 67.2 seconds. In fact, all but one subject answered the question in less than a minute using the DQ interface. The exception to the trend (or multiple search) question required combining the results of two or more searches or finding the outlying point in a set. In this case, the exception was a home that was cheap for what it offered. These types of questions are often what people are searching for in a database. They often wish to find the most stressed part of an airplane, the cheapest car which provides all the options desired, or the deaths that seem to occur not due to cancer or other common environmental factors. Since dynamic queries excel in searching for these outliers, other applications involving this type of search could clearly benefit from its use.

In all fairness, it is difficult to honestly compare the QA, DQ, and paper interfaces since the ways user's use each are so different. QA requires typing with little guidance, while the DQ interface must be 'custom-made' for the fields the user will search on. Therefore, it is difficult to specifically identify the advantages or disadvantages; however, from a usability standpoint, it does seem that the DQ interface does perform better, overall, than QA for some types of questions.

This study and others (Ahlberg, Williamson, & Shneiderman, 1992) have shown dynamic queries to have several benefits over current systems:

- Queries can be made much faster by sliding the sliders and seeing rapid feedback directly on the display.
- Novices can learn to use the system quickly with both query-formulation and result displayed in task domain.
- Intermittent users do not have to remember any syntax. Repeat users quickly remember how to form a query.
- No error messages are needed since sliders restrict ranges. Also encourages user to explore.
- Users can fine-tune their search easily. Simply restricting one field allows tuning until desired number of hits.
- Actions are incremental and reversible.
- Trends and content of the database are easily inferred.
- Well-suited for geographical information systems and public access using a touchscreen.
- Display in task-domain is more useful, less intimidating, and speeds training time.

Although dynamic queries show promise, they are far from perfect. Many drawbacks limit the range of its applicability or pose other problems:

- Data must be ordered in some way, in particular textual fields don't benefit from the dynamic nature of sliders.
- Limited screen real-estate make it difficult to keep both tools and display on-screen.
- Difficult to use when a large number of fields are searchable due to limited screen space and computational issues.
- Data structures to permit more rapid querying (especially for large databases) are clearly necessary.
- Work best with data which can be displayed in some graphical fashion.
- Difficult for dynamic queries to offer more complex boolean queries such as unions and negations.
- Recently we have devloped a range slider with draggable ends, partially solving some of these limitations.
- Smarter ways to utilize the limited screen space without sacrificing simplicity are clearly needed.
- Perhaps the most major drawback, as of now, they require custom programming to create an interface. Even adding additional criteria (such as crime rate or some such) for querying requires significant adjustment. We are currently designing a DQ toolkit to minimize this difficulty.

This initial study needs to be replicated with other subjects, database domains, queries, and with more experienced users to assess the full range of strengths and weaknesses of dynamic queries. Despite the drawbacks, dynamic queries clearly have application in a range of areas. For geographic information systems they show clear promise. For textual purposes, however, the problems outweigh the benefits. Since the results of this study suggest users find dynamic queries more enjoyable to work with and can complete typical tasks in equal or less time than a traditional natural-language query system, it seems clear these principles have a place in the design of future information retrieval systems.

Acknowledgements

We appreciate the support of NCR Corporation and Johnson Controls; as well as the comments from Ben Harper, Christopher Ahlberg, Donna Harman, Kent Norman and other members of the Human-Computer Interaction Laboratory at the University of Maryland.

References

Ahlberg, C., Williamson, C., Shneiderman, B. (1992). *Dynamic Queries for Information Exploration: An Implementation and Evaluation.* Proc. CHI'92: Human Factors in Comp. Systems, ACM Press.

Chin, J., Diehl, V., Norman, K. (1988). Development of an instrument measuring user satisfaction of the human-computer interface. in *Proc. CHI'88 Human Factors in Comp. Systems Conf.,* ACM Press, 213-218.

Fowler, R., Fowler, W., Wilson, B. (1991). Integrating Query, Thesaurus, and Documents through a Common Visual Representation. in *Proc. SIGIR '91*. ACM Press, 142-151.

Harman, D., Candela, G. (1990). Bringing natural language information retrieval out of the closet. *SIGCHI Bulletin* **22**, 42-48.

Kim H., Korth H, Silberschatz A. (1988). PICASSO: A Graphical Query Language, *Software -Practice and Experience*. **18**, 169-203.

Korfhage, Robert R. (1991). To See, or Not to See - Is That the Query? in *Proc. SIGIR '91*. ACM Press, 134-141.

Larsson, James A. (1986). A Visual Approach to Browsing in a Database Environment. *IEEE Computer*, **19**, 62-71.

Norman, Donald A. (1988). *The Psychology of Everyday Things.* Basic Books, Inc., New York.

Helander, M. (1988). *Handbook of Human-Computer I nteraction.* Chapter 13. North-Holland, New York.

Rowe, L.A. (1985). Fill-in-the-Form Programming. in *Proc. 11th International on Very Large Databases*. ACM Press, 394-403.

Rutkowski, Chris. (1982). An Introduction to the Human Applications Standard Computer Interface. *Byte*. **7**, 291-310.

Salton, G. (1983). *Introduction to Modern Information Retrieval*. McGraw-Hill, New York.

Shneiderman, Ben. (1983). Direct Manipulation: A step beyond programming languages, *IEEE Computer*, **16**, 57-69.

Williams, M. (1984). What makes RABBIT run? *Int J. Man--Machine Studies*, **21**, 333-352.

Zloof M. Query-by-Example. (1975). *National Computer Conference*. AFIPS Press, 431-437.