A NOVEL USER INTERFACE FOR INFORMATION EXPLORATION AND VISUALIZATION

Max M. North, Sarah M. North, and Nazir A. Warsi
Human-Computer Interaction Group
Clark Atlanta University, Atlanta, Ga 30314, (404) 880-6942
max@acm.org, snorth@cc.gatech.edu, warsi@cau.auc.edu

ABSTRACT

The primary objective of this study is to investigate the design, implementation, and evaluation of a graphical user interface prototype, called InfoVis, for direct manipulation of databases. This prototype information exploration and visualization interface allows users to explore a database with graphical widgets such as a novel *dynamic slider*. Specifically, the InfoVis interface enables the user to search a database without the need to create or formulate complex syntactical query statements. The *dynamic slider* and other widgets are utilized to assist users in mental visualization and representation of objects and actions. A pilot empirical study was conducted to assess the effectiveness of the InfoVis interface prototype. The primary results suggest that subjects who used the InfoVis interface performed at a higher level than their counterparts who did not use InfoVis interface.

INTRODUCTION

Generally, information exploration of databases is performed using query languages. Most query languages such as Structured Query Language (SQL), are based on relational algebra or calculus. While relational algebra and calculus provide a powerful means to formulate and specify queries, their usage is an extremely tedious and complex task (Cha 1990), (Mayhew 1992), (North et al. 1993), (Nowell et al 1993), (Zloof 1975) for computer users, especially for *naive* users. In addition, query languages are redundant in the sense that the same query may be expressed in many different ways (Kim et al 1988). In fact, empirical research indicates a wide variation in response times in the implementation of different query languages. This research argues that providing a graphical visualization of the databases, queries, and search results will empower users with the complex task of information exploration (Ahlberg et al 1992), (Cha 1990), (Johnson 1992), (North et al 1993), (Nowell et al 1993), (Sarker et al 1992). To provide and enhance graphical visualization, a new multiple attribute presentation widget called *dynamic slider* is introduced (North et al 1993). The *dynamic slider* enables the user to present multiple value ranges rather than a single or anchored value to minimum or maximum points.

DYNAMIC SLIDER AS A NOVEL MULTI ATTRIBUTES WIDGET

Traditionally, sliders have been used as a metaphor to assist the user in entering either a single value or a single range value anchored to minimum or maximum points of a field. The concept of dynamic queries (Ahlberg et al 1992) has been extended so that the users can implicitly construct complex queries by utilizing visual and graphical techniques and tools such as *dynamic sliders.* By utilizing *dynamic sliders,* InfoVis interface is able to represent dynamic range(s) rather than a single or anchored value to provide the mechanism for assisting the user in formulating more complex queries. The use of novel *dynamic sliders* widget distinguishes the InfoVis interface from other database interfaces.

In InfoVis, the query is represented primary by number of *dynamic sliders* widgets and other widgets. A *dynamic slider* consist of a name, size, type, global relationship, minimum and maximum values, a slider with two drag boxes, and ranges. Figure 1 denotes only a simple *dynamic slider.* A *dynamic slider* (Figure 1-a) is capable of mimicking the simple or traditional slider by just manipulating (clicking on the slider bar) one of the drag boxes at a time. For instance, by manipulating the top drag box values from the minimum point to the desired point is represented (Figure 1-b). By manipulating the bottom drag box values from the maximum point to the desired point is represented (Figure 1-c). Representing range of values that is not anchored to minimum or maximum points could be accomplished by manipulating both drag boxes to desired points. Figure 1-d demonstrates this representation. A *dynamic slider* was built with the user's ease of use in mind. Although, it may be simply understood and operated by the user, it is a very powerful widget. Because of its property of being a basic block of widget, it may be included in many interfaces to revise the static nature of interface to a more dynamic nature thus provide more powerful and efficient interface.

DYNAMIC SLIDERS WIDGET WITHIN FRAMES AND CLUSTERS

Frames were introduced in the mid-1970's by Minskey to present knowledge. Since then, the Minskey's frame concept have been extensively utilized by other applications, such as databases. Simply, frames are structured forms of attributes (Gupta et al 1991), (Leong et al 1989). One of the advantages of the frame is that users do not need

prior knowledge of the attributes for an object (Gupta et al 1991), (Koh et al 1989). In essence, frames reduce the burden of typing and eliminate need for user knowledge of acceptable attribute values. Frames also provide several other advantages. The default values of attributes may be explicitly stated (Koh et al 1989); however it may be possible to set a default for other query interface (e.g., SQL) but a user has to memorize the specific of the default. Frames also provide for consistent placement of attributes. This visual consistency should ease the task of understanding sets of attributes. On the contrary, SQL or natural language does not enforce a consistent ordering of

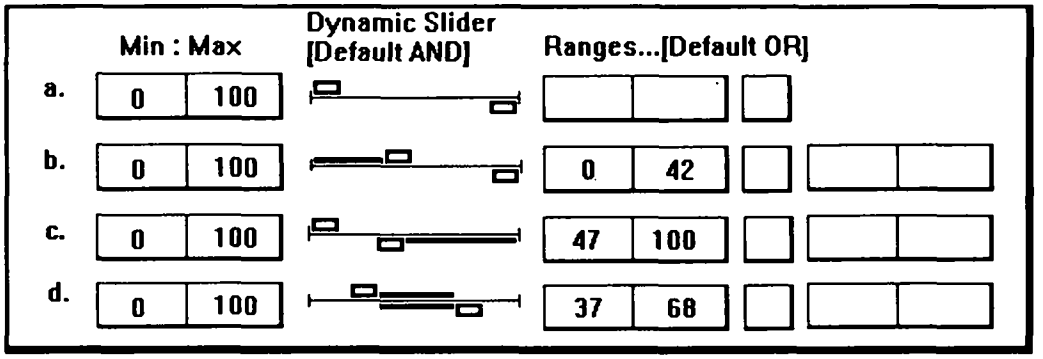


Figure 1. A simple Dynamic Slider.

attributes. Finally, frames can be graphically and visually extended to allow for building more complex queries. Frames with all of the advantages appears to be static and bounded to the specified attributes values thus allowing the user to set attributes that are totally depended on the widgets capabilities used within a frame. The *dynamic slider* utilized within the frame extends the power and characteristics of the frame from a static state to a dynamic state. This new dynamic state of the frame, called *dynamic frame* allows for more efficient use of the frames and possibly combines several static frames to one dynamic and more general specifications and constrains of the attribute values.
Clusters have been introduced and successfully utilized in VIMSYS query interface for image databases (Klinger et al 1993). Klinger, et al, define the clusters as follows: "Clusters are abstract query structures which encapsulate any combination of query objects. A cluster is simply a box with a user-provided natural language label." In general, clusters possess the following characteristics and can be: (i) linked to other clusters, or frames; (ii) expanded to reveal their contents; (iii) modified for customization; (iv) moved around the screen; and (v) retrieved, saved, grouped, renamed, or removed. Clusters reduce the burden on the user for manipulating groups of attributes. Specifically, clusters reduce any group of attributes into a single unit with a customized natural language label, such as: "Aircraft query base on range of external fuel"; or "Aircraft' performance at high altitude."
Since the label is customized as chosen by the user, it will provide a better cue for recalling and constructing more complex query rather than remembering and interacting with several frames and set of attributes at a time. Clusters also provide a mechanism for the user to graphically manipulate while forming queries. Moreover, the small size of the clusters allow more manageable and meaningful information such as frames and other clusters to be displayed on the same screen (Klinger 1993). Clusters contribute extensively to the domain of module reusability. Clusters may be retrieved and combined together to form a new query. These characteristics of clusters increase the efficiency of formulating especially complex queries. Finally, clusters are linked via a specific attribute. This limitation will enhance the user's understanding by simplicity of the link element. For more detail information of clusters the reader may refer to the work of Klinger, et al at UIST '1993 (Klinger et al 1993).
Clusters power also may be extended by the dynamic nature of the dynamic frames it represents. In other words, clusters inherent the properties of the frame or frames that are within them, called *dynamic clusters*, and in turn frames inherent the properties of *dynamic sliders* within themselves. This hybrid architecture gives birth to the new interface for exploration and visualization of information. The basic block of the *dynamic sliders* empowers the frames and clusters which empowers the user for facing the challenges of graphically forming complex queries with a very low cognitive load.

THE INFOVIS CAPABILITIES AND CHARACTERISTICS

Several attempts to use direct manipulation for databases have been made. Examples include the user friendly query language PICASSO (Kim et al 1988) and another product, Query-by-Example (Zloof 1975). Although these approaches are powerful, they do not provide visual presentation of information and actions. The InfoVis interface provides a more powerful interface by implementing the following guidelines: (i) continuous graphical representation of database, query, and information exploration outcome; (ii) visible range of the object by utilizing

dynamic sliders and other widgets (Ahlberg et al 1992); and (iii) immediate feedback as the user physically manipulates the sliders, selection buttons, etc.

Specifically, the InfoVis interface possesses the following capabilities and characteristics: (i) objects and graphical widgets of interest are continuously visible to the user (Beard et al 1991), (Cha 1990); (ii) the outcome of the information exploration is produced by physical actions such as manipulation of the *dynamic sliders* and other widgets rather than utilizing the complex query syntax (Ahlberg et al 1992); (iii) the provision for incremental, rapid, and reversible actions that will be immediately displayed for users; and (iv) a minimal learning curve to aid *naive* as well as expert users in the exploration of information. In essence, the InfoVis interface enables the users to visualize and explore information similar to human cognitive information processing using minimal cognitive load (Chimera 1992). In addition, the InfoVis interface enables the user to see in one view objects, actions, and results, and InfoVis interface assists the user in extracting the meaning and relationship of objects (Sarkar et al 1992).

The InfoVis interface capabilities include the following (Ahlberg et al 1992):

I. **Basic data manipulation** - Consists of the *select, update, delete,* and *insert* operations.
   - *Select* operation - the data retrieval operation that specifies the method of selecting tuples of relation(s) and constrained by predicates.
   - *Update* operation - the operation modifies one or more records in a specified relation and consideration of predicates.
   - *Delete* operation - this operation provides the deletion of one or more records from relation constrained by predicates.
   - *Insert* operation - the insert operation allows the user to insert a new tuple into a specified relation.

II. **Basic data retrieval** - Provides three basic operations, *selection, join,* and *projection* of multiple tuples queries.

III. **Condition specification** - Supports the following Boolean and comparison operators utilizing the *dynamic sliders* widget: *and, or, not,* =, not =, >, ³ , <, and £ to allow the formulation of more complex predicates.

IV. **Arithmetic and aggregate operators** - Support the following arithmetic operators and functions: average, minimum, maximum, sum, and occurrence count.

Furthermore, the following general aspects are included:

I. Multiple constraints upon single attributes such as the predicate value between *a* and *b*.

II. Implicit conjunctive form of predicates. By default, the specified predicates are assumed to be of a conjunctive nature *(and)* unless otherwise indicated by the user such as disjunctive connection *(or)* (Kim et al 1988), (Nowell et al 1993).

III. Set Operations such as *in, contains, set difference, set intersection,* and *set union.*

THE INFOVIS INTERFACE AT WORK

The user is presented by the graphical meta structure of the database. Figure 2 denotes an example of *Aircraft* database which is comprised of six databases in this particular case. The graphical meta structure at this time is defined by the database designer and an administrator.

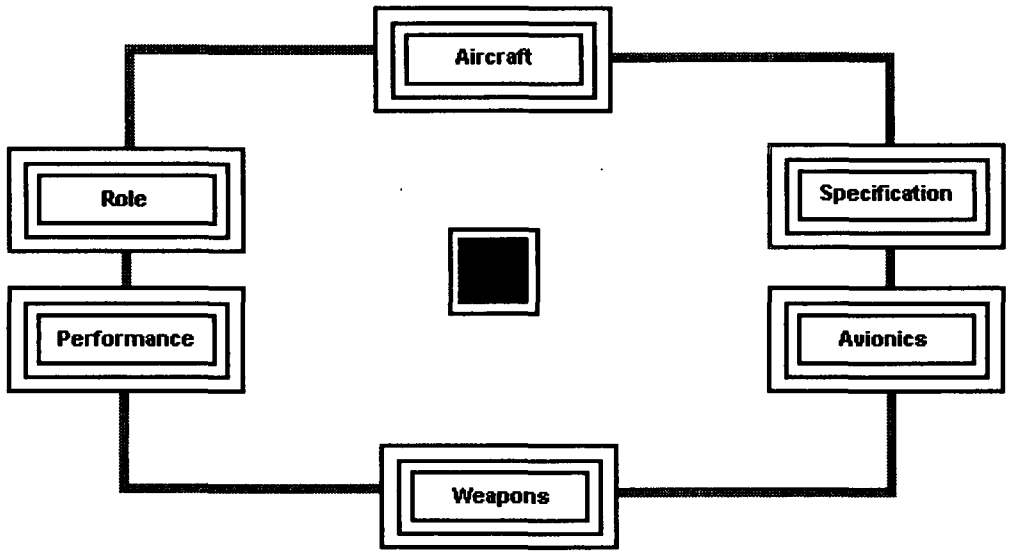


Figure 2. Initial graphical meta structure of the Aircraft database.

By clicking on the middle point, shaded square, InfoVis interface will reveal the basic structures and an example of each database. The databases are linked via one or more attributes. If the user is interested in only individual or several databases then the user may click on any desired box on the screen to activate and reveal the structure and sample data. This mechanism also allows the user to select the desired database or databases for direct manipulation. This global view of the database reduces the mental load on the user and allows the user to view the entire databases and their relationships in a general form first and gradually move to more detail as desired. At this point, each database or combination of databases may be selected (see Figure 3).

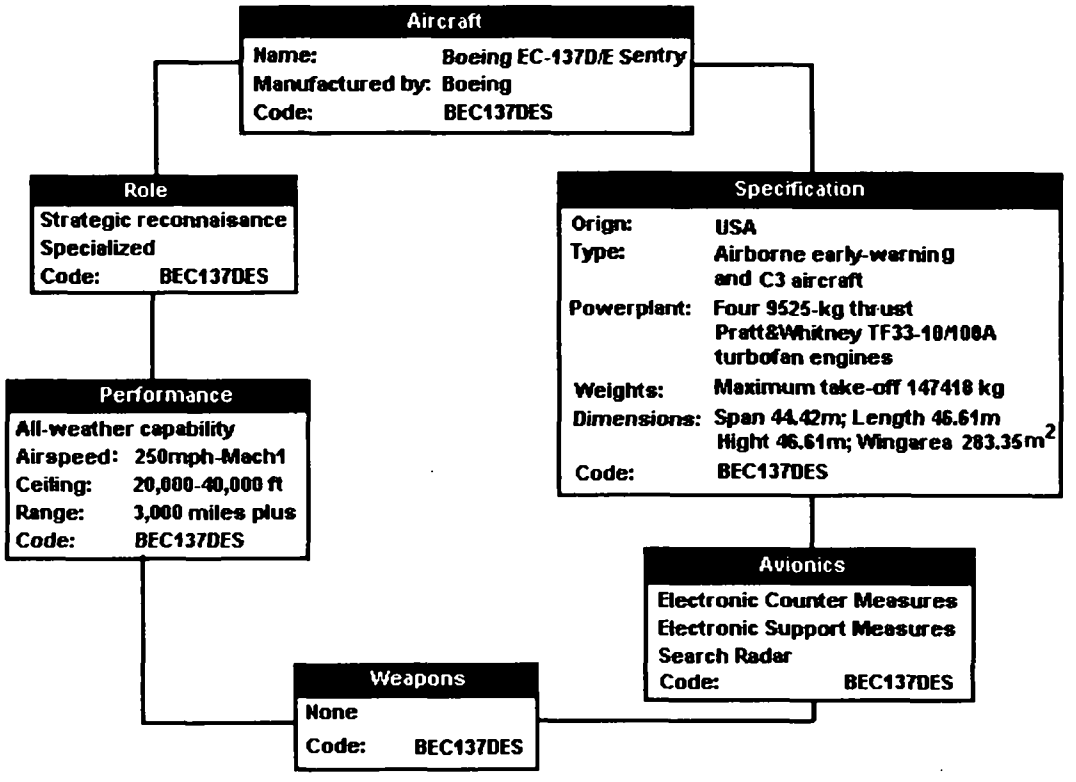


Figure 3. The global view of the Aircraft database structure and sample data.

For simplicity, the query part of the InfoVis interface is introduced through examples on the following tuple:

- *Aircraft (service ceiling, take-off run, speed at high altitude, range of external fuel, rate of climb at sea level).*

Where the *Aircraft* tuple specifies the *service ceiling, take-off run, speed at high altitude, range of external fuel, and rate of climb at sea level* (Table 1).

The query frame (see Figure 4) consist of structure of the database, such as name, size, type, etc. Each attribute is

Aircraft	Service Ceiling [ft]	Take-off Run [ft]	Speed at High Altitude [Mach]	Range/External Fuel [km]	Rate of Climb Sea Level [ft/min]
F/A-18 Hornet	50,000	1,400	1.8	3,706	40,000
F-14 Tomcat	50,000	1,300	2.34	3,220	30,000
F-15 Eagle	60,000	900	2.5	4,500	50,000
F-16 Fighting Falcon	50,000	1,200	2.0	3,800	40,000
F-4B Phantom	62,000	4,390	2.25	3,700	28,000
Mig-23 Flogger	61,000	2,950	2.35	2,500	30,000
Mig-25 FoxBat-A	80,000	4,525	2.83	3,000	40,950
Saab JA37 Viggen	60,000	1,310	2.1	3,000	40,000
Tornado F_MK2	50,000	2,500	2.16	4,800	30,000

Table 1. Aircraft Tuple.

linked with a *dynamic slider* which specifies single range, dynamic range, or ranges in general. The sliders are placed at the same line with attributes to minimize the users visual tracking. By clicking on the attribute select box the user may choose (i) the attribute(s) selected for imposing constraints; and (ii) the attribute(s) selected for outcome output of the query window. The other novel idea in InfoVis interface project was to allow the user to specify the order of the relationship by simply entering a numerical ordering number in the global relation box and to select the kind of relationship of the attributes as desired. By default the attributes of the databases are connected together by conjunctive predicate *(and)*. If the query needs to be connected by disjunctive predicate *(or)*, the user may click on the default box to change the *and* to *or*. The user is provided Minimum and Maximum values of each attribute. Minimum and Maximum values allow the user to extract meaning from the collective data in the database and assists the user in building queries. It also reduces the cognitive load of the user by providing ranges.
At this point the user may manipulate the *dynamic sliders* to set and lock desired range or ranges of values of each attributes. After the range or ranges are selected it may be locked and then the predicates *"and"* or *"or"* (user's perspective) to the next set of ranges as desired by the user. This dynamic mechanism enables the users to build a more complex query without being worried about SQL or other query languages. Figure 4 shows an example of

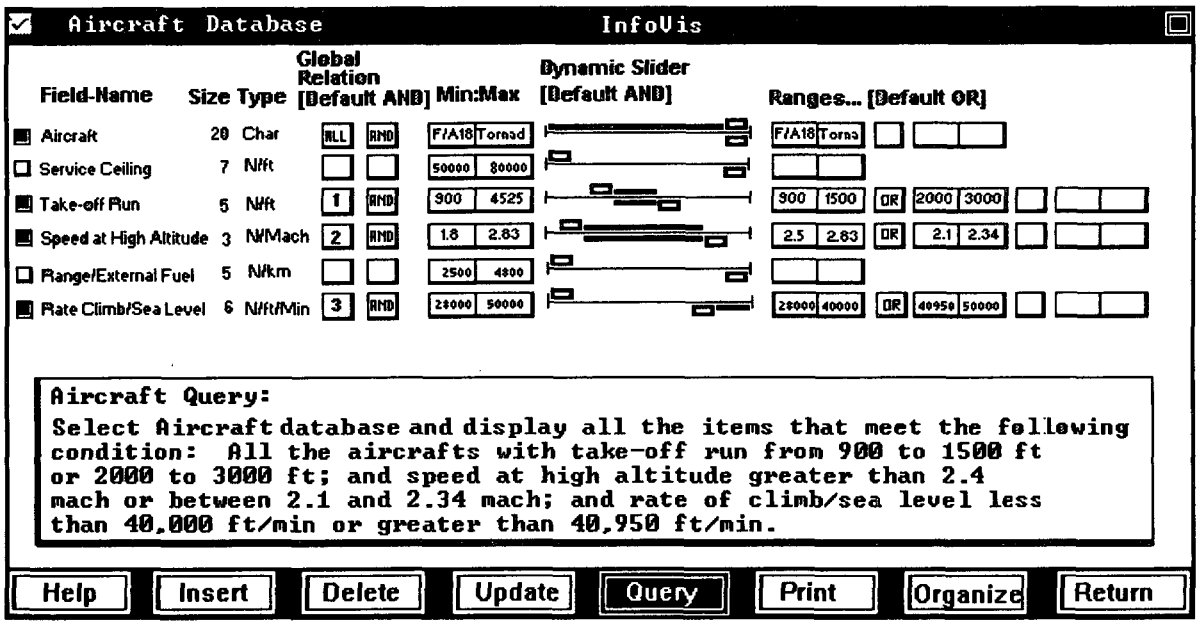


Figure 4. The Query dynamic frame of the Aircraft database.

direct manipulation of *Aircraft* database that responds to the following query:

> *Select aircraft database and display all the items that meet the following condition: All the aircraft with take-off run from 900 to 1500 feet or 2000 to 3000 feet; and speed at high altitude greater than 2.4 mach or between 2.1 and 2.34 mach; and rate of climb at sea level less than 40,000 feet/minute or greater than 40,950 feet/minute.*

THE EXPERIMENT

This experiment compared two different interface for query building for databases. The first interface was InfoVis that was just briefly described above. The other interface was SQL, the traditional query building interface. The major hypotheses was that, there would be a significant difference between the performance of subjects who use InfoVis interface and the subjects who use the traditional SQL interface. The independent variables in this experiment were the interfaces that subjects utilized to build queries: (i) InfoVis interface; and (ii) SQL interface. The dependent variables were: (i) time to build query in order to get the desired output; and (ii) usefulness, effectiveness, learnability, and attitude (Mayhew 1992). The prior data collected from the users revealed that users'

skills vary from novice to expert with novice users dominating the population. The experimental subjects were twenty individuals (n=20) who were randomly selected for this study. Ten subjects were placed in the experimental group and ten subjects were placed in the control group. A typical subject was 18-23 year-old (76%), and female (72%).

Initially, a prototype InfoVis Interface was designed, developed, tested, and implemented on a Sparc SunWorkstation™. Each group was given an overview of experimental procedures and a short instructional period, primary to acquaint users with the domain subject and interfaces. The subjects from the experimental and control groups were presented with ten different queries to perform on *Aircraft* database. These queries started with simple query and gradually became more complex. For instance, subjects were given the following queries:

- *Find aircraft(s) that have service ceiling of 60,000 feet and above.*
- *Find aircraft(s) that have speed at high altitude of above mach 2.16 or equal; and take off run of 1,300 feet; or rate of climb at sea level above 40,000 feet.*
- *Find aircraft(s) that have range of the external fuel between 3700 km and 4,800 km.*
- *Find the performance of MIG-25 - FOXBAT-A.*

PRELIMINARY EVALUATION

Evaluation of the prototype system is critical to ensure the quality of the final product. To determine the desirable characteristics and features of the InfoVis prototype, a comprehensive survey was conducted. Prototype development and its evaluation were tightly interleaved. Each prototype was evaluated in realistic settings, and the outcomes was accommodated in the design of the next prototype, thus ensuring continuous quality improvement during each stage of system development. In summary, to ensure the usefulness, effectiveness, learnability, and attitude of the InfoVis interface prototype, several usability tests for each intermediate prototype design was conducted (Mayhew 1992).

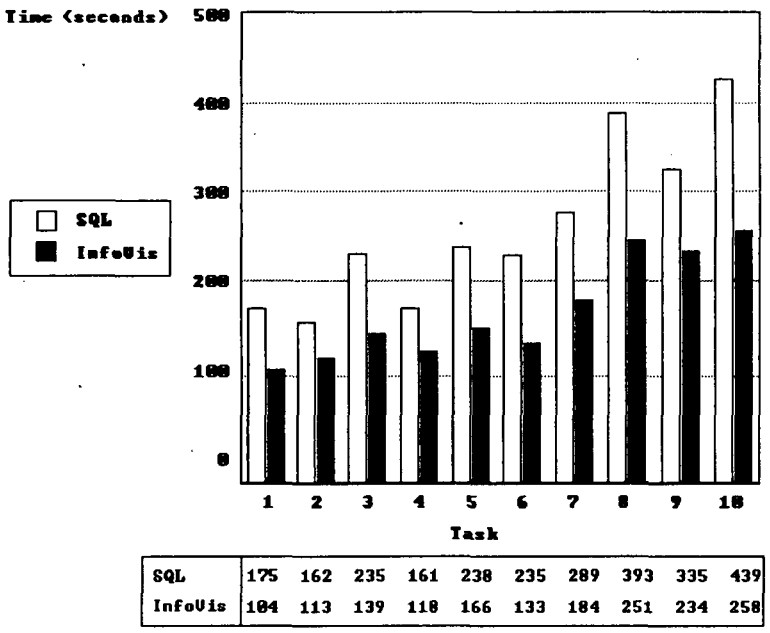
PRELIMINARY RESULTS

The data were subjected to appropriate statistical procedures. These procedures included a measure of central tendency, and the t-test. The result of the t-test indicated that there exists a significant difference between the performance of the experimental group and the control group ($t=2.69$, $df=18$, $p<0.05$), (Table 2).

	Control Group (SQL)	Experimental Group (InfoVis)	t	df
n	10	10		
Mean	266.20	170.01	2.69	18
S.D.	96.71	58.93		

Table 2. Means, Variance, Standard Deviations, Degree of Freedom, and t-test of subjects in experimental and control group.

Although the preliminary results of this research indicated that the subjects in experimental group performed at higher levels than subjects in the control group, a larger sample is recommended for future research. Figure 5 shows the mean time to complete each task utilizing SQL and InfoVis interfaces. In general, Figure 5 shows the superior performance of the InfoVis interface over SQL interface for each task that was performed by experimental and control groups.



SQL	175	162	235	161	238	235	289	393	335	439
InfoVis	104	113	139	118	166	133	184	251	234	258

Figure 5. Mean time to complete each task utilizing SQL and InfoVis interfaces.

CONCLUSION

Although we have built several prototypes to test out the ideas, the work is still in its early stages. InfoVis interface is an ongoing project with Clark Atlanta University's Human-Computer Interaction Group. The preliminary results of the pilot study ($t=2.69$, $df=18$, $p<0.05$) so far are very encouraging, and users' feedback is positive.
Using twenty subjects, several testings of the interface have been performed, and the InfoVis interface is now being tested with larger populations and more complex queries. Further prototyping and testing will improve the overall performance of InfoVis interface. Finally, the InfoVis interface is an operational prototype though functionally quite a simple system. In other words, the preliminary results so far are very encouraging, and users' feedback has been positive. In essence, the empirical results demonstrate that InfoVis interface appears to be a promising approach.

DISCUSSION AND FUTURE RESEARCH

The hypotheses that there would be a significant difference between the InfoVis interface and the traditional query procedure was confirmed. Thus, it appears that InfoVis interface provides much better and faster learning and building queries for information exploration. Of course, this is only a prototype of InfoVis interface, and many significant problems must be resolved. For instance, graphical meta structures provided to the user at this time is build by database designer and administrator, further work must be undertaken to allow the user to build this initial stage of the interface. Also provisions must be provided for additional customization capabilities for the user. At this point the number of links between the databases is limited to one link. Additional investigation is needed to provide the user with the mechanism for multiple linkages. In addition, as the system is used over time, many *dynamic frames* and *dynamic clusters* will be stored in the library. Libraries may be tagged private or public depending on the person who creates it. Eventually, the number of *dynamic frames* and *dynamic clusters* in libraries either private or public will increase and re-using these stored *dynamic frames* and *dynamic clusters* will become increasingly difficult (Klinger et al 1993). Also, each dynamic frames or dynamic clusters may have different versions with different attribute sets that may add to the problem at hand. An Object (refers here to *dynamic frames* and *dynamic clusters*) Management Interface will be needed to assist users in manipulation of objects in the libraries. The *dynamic slider* concepts have been applied to the relational database at this time. Further research is necessary to find the effect of *dynamic frames* and *dynamic clusters* on other type of databases, such as hierarchy and network databases. Finally, it is desirable to compare the InfoVis interface with two or more graphical interfaces to find out if there are any significant differences among their performance.

ACKNOWLEDGEMENT

This project is sponsored by Boeing Computer Services and partially supported by the U.S. Army Center of Excellence in Information Science (Contract Number DAAL03-92-6-0377) and David and Lucile Packard Foundation. The views contained in this document are those of the authors and should not be interpreted as representing the official policies of the U.S. Government, either expressed or implied.

REFERENCES

Ahlberg, C., Williamson, C. and Shneiderman, B. (1992) "Dynamic Queries for Information Exploration: An Implementation and Evaluation", **Proceedings of ACM CHI'92 Human Factors in Computing Systems Conference**, pp 619-626.

Beard, D. and Walker, J. (1991) "Navigational Techniques to Improve the Display of Large Two-dimensional Spaces", **Behaviour & Information Technology**, Vol 9, pp 451-466.

Cha, S. (1990) "Kaleidoscope: A Cooperative Menu-Guided Query Interface (SQL Version)", **Proceedings of IEEE Artificial Intelligence Applications**, pp 304-310.

Chimera, R. (1992) "Value Bars: An Information Visualization and Navigation Tool for Multi-attribute Listings", **Proceedings of ACM CHI'92 Human Factors in Computing Systems Conference**, pp 293-294.

Gupta, A., Weymouth, T. and Jain, R. (1991) "Semantic Queries with Pictures: the VIMSYS Model", **Proceedings of the 17th International Conference on Very Large Data Bases**.

Johnson, B. (1992) "TreeViz: Treemap Visualization of Hierarchically Structured Information", **Proceedings of ACM CHI'92 Human Factors in Computing Systems Conference**, pp 369-370.

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

Klinger, J., Swanberg, D. and Jain, R. (1993) "Concept Clustering in a Query Interface to an Image Database", **proceedings of UIST '93**, pp 11-21.

Koh, T. and Chua, T. (1989) "On the Design of a Frame-based Hypermedia System", **Hypertext II.**

Leong, S., Sam, S. and Narasimhalu, D. (1989) "Towards a Visual Language for an Object-Oriented Multi-Media Database System", **Visual Database Systems**, North Holland Publishing.

Mayhew, D. J. (1992) **Principles and Guidelines in Software User Interface Design**, New Jersey: Prentic-Hall, Inc., A Simon & Schuster Company, Englewood Cliffs.

Minsky, (1988) **The Society of Mind**, Simon & Schuster.

North, M. M. and North, S. M. (1993) *"Sarah:* An Information Exploration and Visualization Interface for Direct Manipulation of Databases", **Proceedings of Graphics Interface '93 Conference**, pp 163.

Nowell, L. T., Hix, D. and Labow, E. D. (1993) "Query Composition: Why Does It Have to Be So Hard? ", **Proceedings of East-West International Conference on Human Computer Interaction**, pp 226-241.

Sarkar, M. and Brown, M. (1992) "Graphical Fisheye Views Of Graphs", **Proceedings of ACM-SIGCHI'92 Human Factors in Computing Systems Conference**, pp 83-91.

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