Browsing Local and Global Information

Masum Hasan, Gene Golovchinsky, Emanuel Noik,
Nipon Charoenkitkarn, Mark Chignell,
Alberto Mendelzon, and David Modjeska

Abstract

Current World Wide Web browsers, e.g., Mosaic and Netscape, support users primarily in the task of browsing the Internet. In some situations, users want to explore topics for which relevant information may reside both on a large local database and on the Web. The MultiSurf project seeks to deal with these situations by integrating text browsing of a local database with hypertext browsing of the Web. In the current implementation, local queries are passed to Web index server(s) for simultaneous search on the Internet. An index server matches query terms with remote documents. Local and remote information is then presented to the user in separate windows. The existence of index servers is made transparent to the user. Instead of opening the URL of a server explicitly and filling the form, users click on the keywords of interest in the text. MultiSurf composes these keywords into queries and passes them to the index servers. In addition to (hyper)text browsing, MultiSurf also supports visualization of the conceptual structure of a query session. This paper will describe our earlier work on text browsing and its adaptation to Web browsing. We will also discuss early impressions of the MultiSurf prototype and its functionality. We will comment on how MultiSurf fits into our overall goal of developing large-scale information exploration systems. Finally, we will describe a research strategy for integrating disparate systems through innovative user interfaces.

1 Introduction

World Wide Web (W3) browsers provide a uniform interface to resources throughout the Internet. Each resource on the Web and resource acccessible from Web has a URL (Universal Resource Locator) that specifies its type (e.g., http, ftp, gopher, wais, netnews, etc.), server and location on that server. The resources served by Web servers are hypermedia documents written in the HyperText Markup Language (HTML), identified by the "http" type. HTML documents may contain links that point to other Internet resources. Each link consists of an *anchor* within the source document and the URL of the target document (i.e., the document that will be displayed if the link is selected). The anchor is the portion of the source document that can be activated by the user to jump to the target document. Typically, anchors are highlighted in some way, for example, by colouring the text blue and underlining it.

W3 users can retrieve Web documents by following hypertext links, or they can search for topics of interest by using *index servers* (IS) connected to certain Web servers.

1.1 Problems and Solutions

Current form filling interfaces for searching require a different style of interaction than the point and click method of web browsing. This problem is exacerbated by the fact that different query syntax is used by different index servers. Thus querying is poorly integrated with browsing. Query formulation can be simplified by providing an interface that allows users to mark up text directly. Queries can be expressed by highlighting a passage (e.g., a paragraph) using a drag operation. The browser can then submit content bearing words within the selected text to an index server.

It is easy to get lost in hyperspace [9] while browsing through the Internet. The web space consisting of queries, hit lists (lists of URLs) and documents can be represented as a conceptual graph. Appropriate visualizations of this graph may reduce the likelihood of disorientation problems. These conceptual graphs may be visualized using a spatial metaphor or overview (e.g., the use of maps to organize data [18]). In some cases, a filtering paradigm may be used instead of – or in conjunction with – overview maps. Navigation becomes more difficult as the number of choices increases. As the Web grows and becomes more densely interconnected, the number of sites that are relevant to a given document will increase beyond that which the navigation metaphor can handle. In such cases, querying and filtering methods can supplement or perhaps even replace the navigation approach.

As more organizations add content to the Web, there will be a need to handle proprietary and legacy data in addition to documents on the Web. Currently legacy data can only be integrated with the Web through the use of scripts that emit HTML source. One problem is that not all data lends itself naturally to this treatment. In many cases, HTML cannot support the interaction styles that are useful for exploring legacy databases. For example, object-oriented graphical browsers such as Hy+ [5] require a granularity of interaction that the current HTML standard does not support. In addition, there are privacy and security issues related to storing proprietary data on publicly accessable servers. One solution is to integrate Web and non-Web databases in a single browser. A component of such a browser can provide some form of query translation between databases. For example, a text passage selected from an HTML page could be sent to a web server and to other databases (e.g., Inquery [3] or SMART database [2]), thereby hiding differences in query syntax from the user.

1.2 MultiSurf

We decided to address the problems cited above by augmenting the paradigm of web browsing using two strategies: query-mediated browsing and integration of query results from multiple servers in the interface. We have been experimenting with query-mediated browsing interfaces as an alternative to hypertext [7]. In contrast to hardwired hypertext links, a query-mediated browsing interface constructs links between documents dynamically based on users' queries. It retains some of the flavour of hypertext when the queries are expressed via selection (markup) of existing text.

A number of tools developed at the research laboratories of University of Toronto provide capabilities that we think will be useful for Web exploration. We have integrated these tools with existing Web browsers to overcome some of the problems cited above. The *MultiSurf* tool introduced in this paper integrates a text browsing system, a real-time graph visualization system, and a modified version of Mosaic 2.0.

To overcome the problem of losing context in the hyperspace, the Graphite graph visualization system (discussed in Section 4 of this paper) has been integrated with the MultiSurf system. The browsed portion of the Web space, the query-hitlist graph, and *query graph* (Boolean queries formulated during a session and visualized as graph) are visualized using the Graphite system.

The rest of the paper is organized as follows. In Section 2 we describe a method of query-mediated browsing. Section 3 focuses on the integration of query-mediated browsing with the Web. Visualization aspects of the system are discussed in Section 4, followed by conclusions in Section 5.

2 Query-Mediated Browsing

Query-mediated browsing allows users to formulate queries by selecting arbitrary passages. Some notations such as QRL [7] also allow users to specify Boolean (AND and OR) queries on the selected terms and to construct them

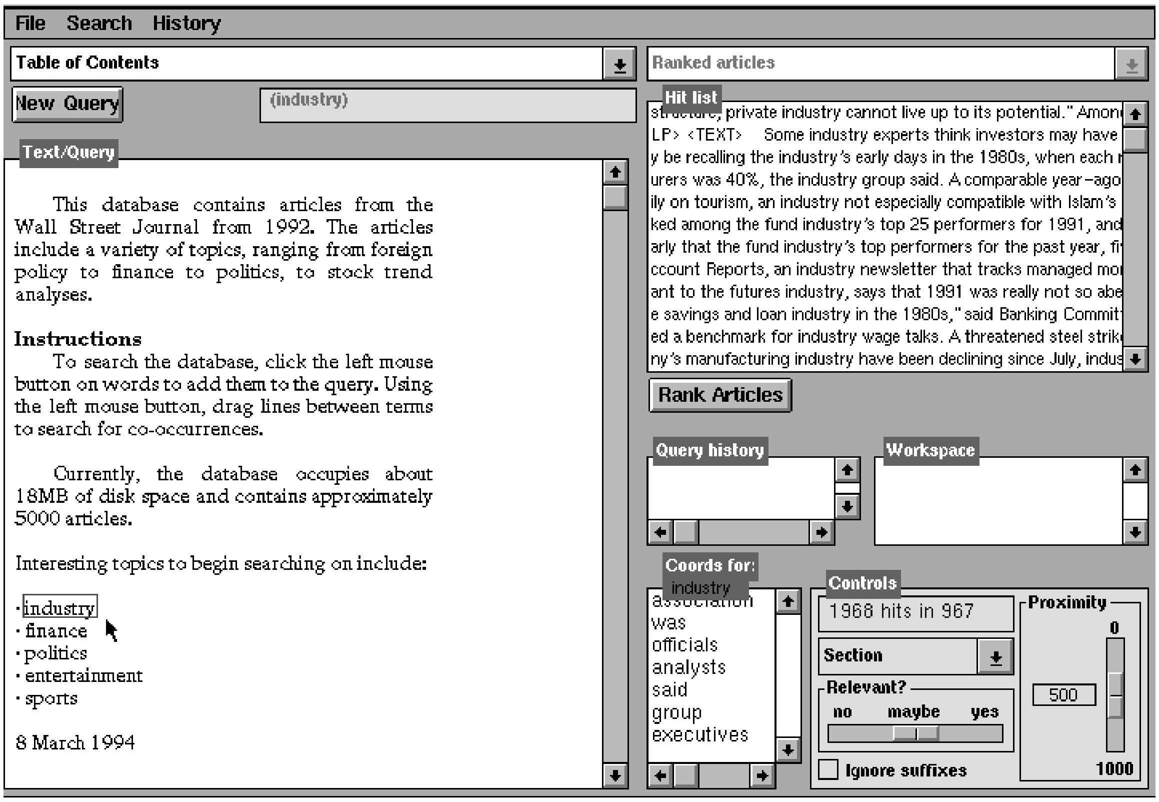

Figure 1: Selection of the term `industry`

graphically on the text to formulate queries. The MultiSurf system grew out of earlier work that we carried out to develop a text browsing system that would allow users to freely intermix browsing and querying styles of interaction during information exploration. The earlier QRL system [7] allowed users to mark up text graphically and inferred the queries from these markups. QRL demonstrated the feasibility of graphical methods for expressing queries. It also showed how browsing could be carried out using interactive queries. This type of interactive querying represents a form of direct manipulation [15] in which the output of one interaction step serves as the input to the next.

The ST-PatTREC system, developed at the University of Toronto, is a text browsing system that supports query-mediated browsing. Figure 1 shows the ST-PatTREC screen after the term `industry` has been selected. The window in the upper right-hand corner of the screen in Figure 1 lists occurrences of the term `industry`. Each hit in this list shows a small amount of the text surrounding the search term. On the lower right of Figure 1 is a window entitled `Coords for:` that shows the coordinate terms for `industry` (e.g., `association`, `officials`, `analysts`, etc.). A small field beside the coordinate terms list shows how many hits have been retrieved in how many nodes (1968 hits in 967 articles). The user has a number of browsing options at this point. An item that mentions the savings and loan industry catches his attention. He selects that item (Figure 2), which is then displayed in the text window. Since the current query is still `industry` (as indicated by the term `industry` in the light gray box above the text window), a box is drawn around the term `industry` within the text window. The user now clicks on the `New Query` button and then clicks on `regulators`, selecting a new topic. The result of this action

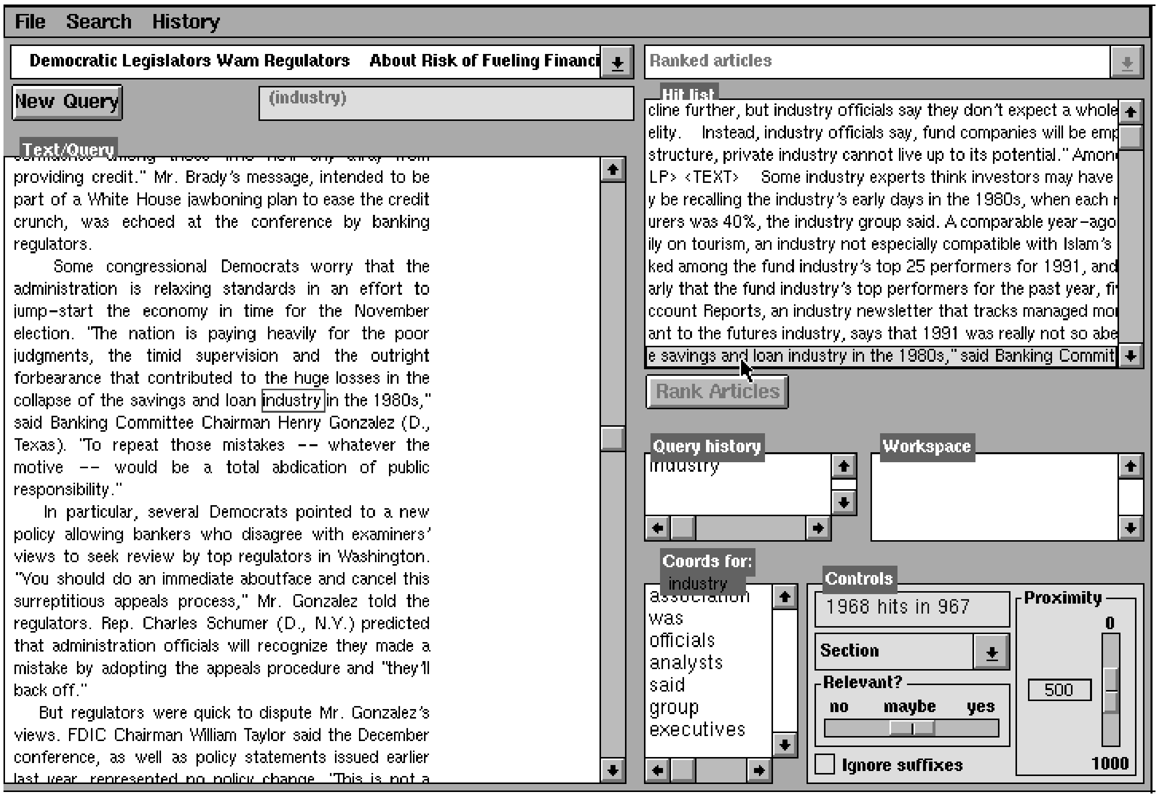

Figure 2: Selection from hit list

is shown in Figure 3. The word `regulators` is selected and the hit list is updated.

This type of interaction enhances the user centeredness of information exploration. While the user cannot choose pre-existing links connected to a node, each word in the text acts as a one-to-many link, and combinations of words can be used to focus or filter the list of links. With this type of query-mediated browsing the user is fully in control of the navigation process and no longer has to rely on the point of view and the thoroughness of anyone but the original authors of the text. Thus the style of interaction described here merges IR and hypertext into a new form of user centered and user controlled information exploration [17], where the user exploits the existing structure of the text in formulating a browsing strategy.

Query-mediated browsing can also be used to browse the web. In addition to following traditional hypertext links, users may browse by selecting arbitrary words or passages from a web page. The advantage of query-mediated browsing for the web is that the location and existence of index servers are transparent to the user. This will be especially useful when a large local database is being explored in concert with the global Internet. Query-mediated browsing on the Internet will be even more useful if autonomously communicating standard index servers (like Internet Domain name servers and similar to what is discussed in [1]) are deployed on the Internet in the future.

One caveat concerning query-mediated browsing on the Internet is that its effectiveness will be constrained by delays caused by bandwidth limitations. This technique works best when it is highly interactive, so that queries on a topic can be launched in quick succession. Lags and delays will tend to interrupt the train of thought and interfere with the fluidity of browsing and exploration. When response times are long, query-mediated browsing may simply provide a convenient way of ex-

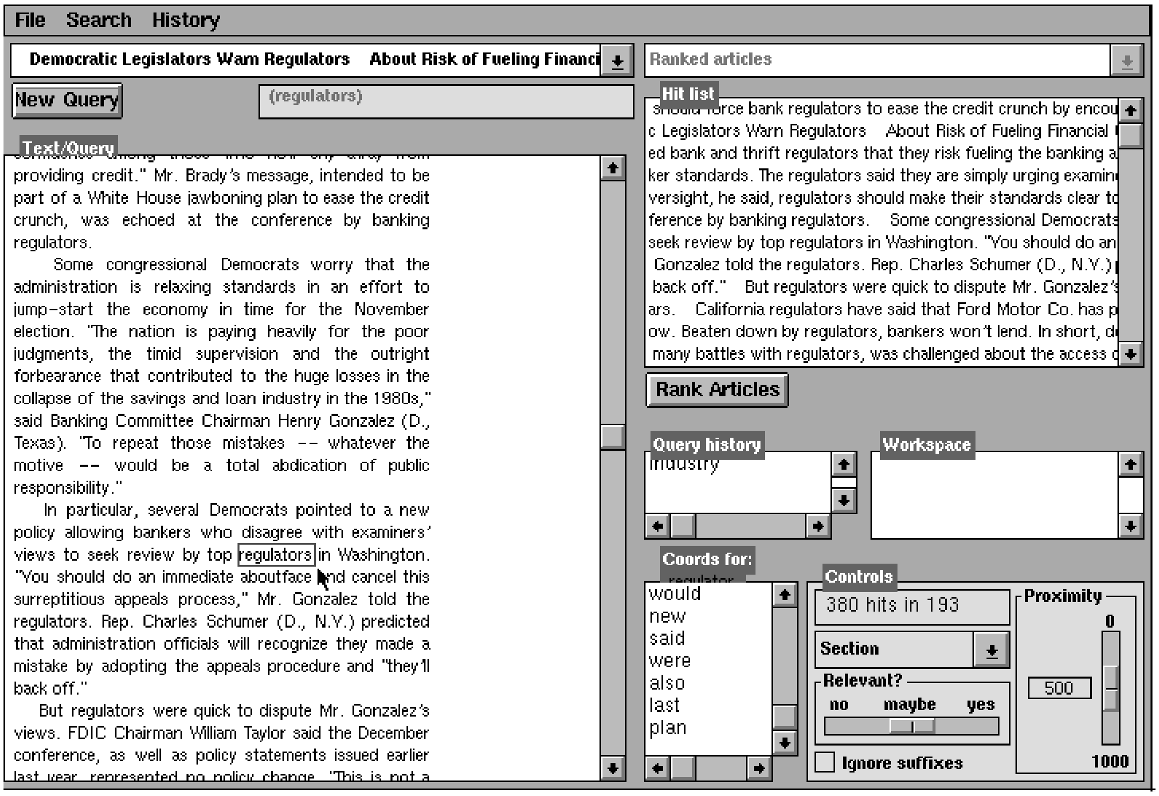

Figure 3: Selection of a new term: `regulators`

pressing queries graphically. However, some of the frustration associated with query-mediated browsing on a network may be alleviated by interleaving fast responses from a local database with the relatively slow feedback from resources across the network. Using selected text passages as queries may also be more intuitive and less error-prone than form-filling interfaces.

3 Web Browsing In MultiSurf

Given the useful properties of query-mediated browsing and the popular success of World Wide Web (W3) browsers, it seems natural to offer the user the best of both worlds. The MultiSurf prototype is being developed to provide this integrated solution. Each query (the keyword(s) selected in the text) submitted by a user of ST-PatTREC is simultaneously sent to one or more W3 *index servers*. When a list of matching documents is retrieved, it is displayed in a control panel, as shown in Figure 4. The control panel has a number of subwindows. The history of queries submitted from ST-PatTREC and the hit lists returned by index servers are displayed in the respective subwindows (Figure 4). Users can select a query from the history list, and then pick a document from the corresponding hit list. The fetched document is shown in a Mosaic window. They may then browse by following the links in the chosen document, or issue another query.

Keywords of an HTML document may be drag-selected in a subwindow of the MultiSurf controller as shown in the right subwindow of Figure 4. This pane shows the HTML-tag-stripped text of the same document as in the Mosaic window. For example, the query `Artificial AND Intelligence` has been formulated by selecting the terms from the text of right subwindow of Figure 4. Work is also in progress to support (HTML) anchored text in

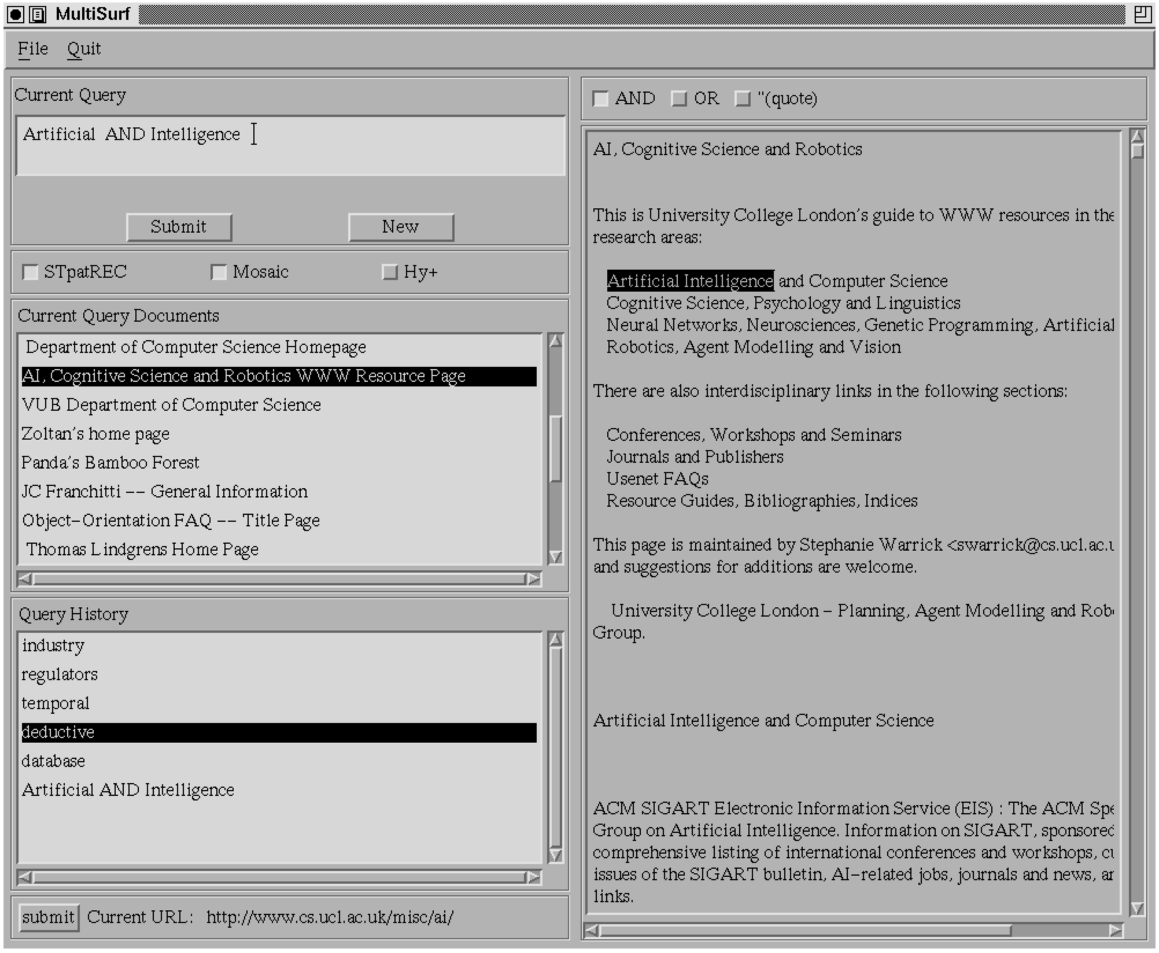

Figure 4: MultiSurf Control Panel

this subwindow.

3.1 Architecture

The architecture of the system is shown in Figure 5. The arrows show directions of control and data flow. The lines labeled `q` in the figure show the flow of the query through the system. A query specified in ST-PatTREC is sent both to the local index search engine, and to the controller application, which (1) shows it in the subwindow `Query History`, (2) sends it to the Graphite real-time graph visualization system, and (3) dispatches it to Mosaic, which sends the query to Internet index servers. The results (hit list) from the Internet index servers (shown by `hl` lines) are displayed in the subwindow `Current Query Documents` of the control panel and in a Graphite window. The various components of the system communicate with each other through Unix sockets.

The following expression is an example of a query sent to an index server named `Harvest`:

```
http://www.town.hall.org/
cgi-bin/BrokerQuery?
host=harvest.cs.colorado.edu:8520&
query=temporal&maxresultflag=50&
descflag=on&verbose=off
```

An alternative design strategy would have been to show the query history and hitlist as HTML documents in Mosaic windows. We decided to use a separate control panel for a number of reasons. First, it is not possible as of this writing to identify a window in any version of Mosaic. We needed this type of window

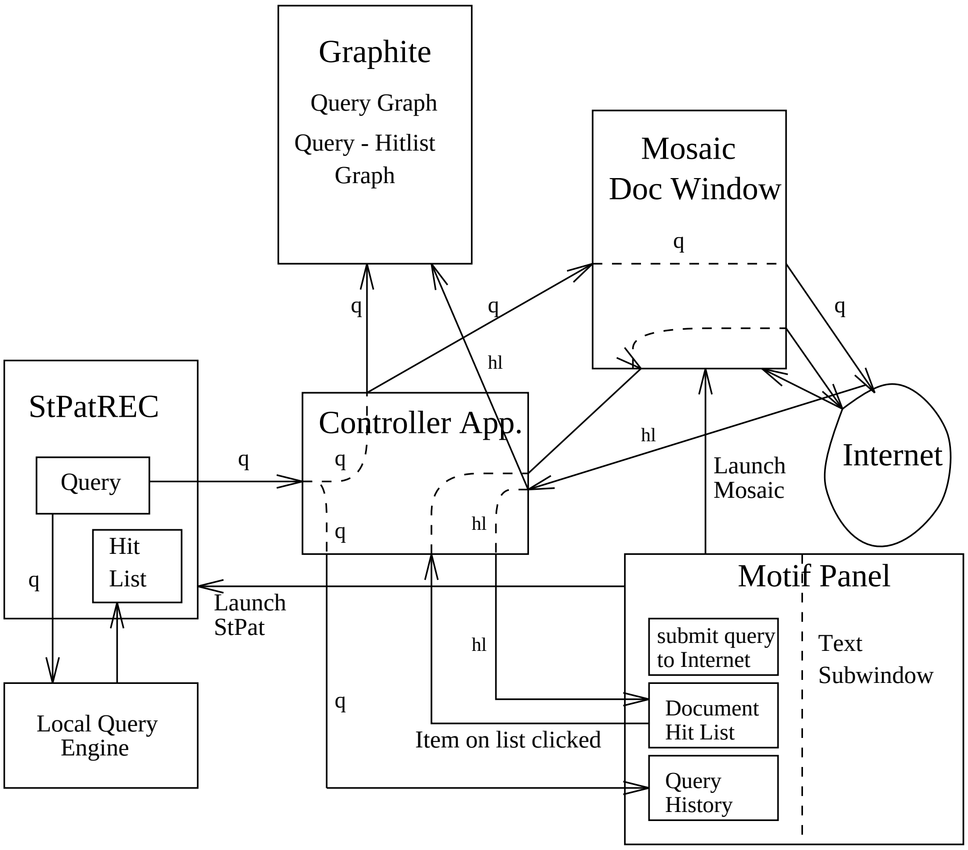

Figure 5: System Architecture of MultiSurf System

identification so that we could show the query history in one window, the hit list in another window, and the documents retrieved in yet another. Windows can be identified in the latest version of Netscape, but the control is one way, from the outside application to Netscape (as of this writing). For example, the click event and the window ID of the window in which the event happened are not reported to outside applications. Second, to incorporate additional features, such as query-mediated browsing, we had to be able to respond to user interactions, such as selections of arbitrary portions of the document text – functionality that is not supported by existing Web browsers.

4 Visualization

The preeceding discussion has focused on the querying and retrieval components of MultiSurf. Visualization is another important component of the system. Views of search topics used to form queries, documents retrieved by queries, and portions of the web browsed by the user can all be visualized as graphs. Graphite is a system used to create general purpose graph visualization utilities, such as graphical editors with integrated automatic layout algorithms. Graphite may be extended to provide application-specific functionality. The Graphite system consists of three main components: the underlying graph layout engine (a C++ library), a graph layout server (an extended Tcl interpreter [13]), and an extensible Graphical User Interface (GUI) graph editor (a Tcl/Tk interactive application [13]). The layout engine can be used generate drawings by employing arbitrary combinations of more than a dozen basic layout algorithms, while the graph layout server and graph editor provide a clean separation between graph layout and graph display and editing, and support two types of clients.

As shown in Figure 6, the first type of client can use the graph layout engine to automatically draw graphs by communicating directly with grtclsh, as done in the Hy+ system [5] where the client manages its own graph browser or editor. The second type of client does not possess its own graph visualization interface, but instead communicates with grwish to display and interact with graphs (in addition to automatic layout); this is the approach taken in the MultiSurf system.

The Graphite layout engine generates drawings of hygraphs (a kind of nested graph formalism) [5]. The engine can compute both two- and three-dimensional layouts; it can draw both nested and flat graphs; and it can create drawings by employing arbitrary combinations of simple layout methods. Furthermore, a novel feature of the engine allows the user to associate ranks or weights with a set of vertices that can then be used to produce a variety of non-geometrical fisheye views [10, 11] and other emphasized views of graphs [12], when computing the layout.

4.1 Query Graph

We are currently exploring the efficacy of a number of different types of visualization in MultiSurf, including query graphs, query hit list graphs and web structure graphs. In a single session a user may specify large numbers of queries. A *query graph* is a 2D graph of the topic terms used in a succession of queries. It provides an overview of the query terms used, and displays them in a manner that suggests the concept space the user is exploring.

In the current implementation of the MultiSurf prototype, the query graph is created in the following way. Each term that appears in a query is placed on the map. When two terms are related by a Boolean AND, a new edge links them in the graph upon which the drawn map is based. For instance, if a query contains the expression "A AND B AND C", then edges are created between each pair of nodes. Terms that appear in multiple queries appear only once in the map, but the edges in which they participate reflect all query expressions containing them. For instance, the query "A AND B AND C", followed by "A AND D", would produce the map shown in Figure 7. The appearance of this graph can then be modified by using different layout algorithms available in Graphite (e.g., a circular or spring layout).

The query graph is intended to provide users with an overview of the topic of interest as indicated in the history of queries that have been used. This can help the user mix and match terms into new combinations, and to get a cumulative picture of how the topic that they are exploring seems to be evolving. Another feature is that with an appropriate layout the query map can show the co-occurrence of search terms. Figure 8 shows an example of query graphs that was created during information exploration.

We recently carried out a study of the usability of query graphs in formal searches on a local database. We found some evidence that they improved the search performance of novice searchers, but they did not significantly improve the performance of experienced online searchers. We are currently studying alternative forms of visualization that may be more useful in supporting information exploration.

4.2 Query Hit List Graph

The hit or document lists corresponding to queries can be visualized either separately or superimposed with the query graph discussed in the previous subsection. This composite graph may then help users to visualize common documents retrieved by different queries, information that is difficult to infer from the lists shown in Figure 4. For example, consider Figure 9, which shows a number of separately submitted queries, and their corresponding hits (matching documents). The visualization shows common documents containing the keywords *temporal*, *deductive*, and *database*. From this display it is apparent that the user need not submit the query *temporal AND deductive AND database*. This not only saves the user time, but also eliminates potentially costly operations, especially if the query is executed over the network.

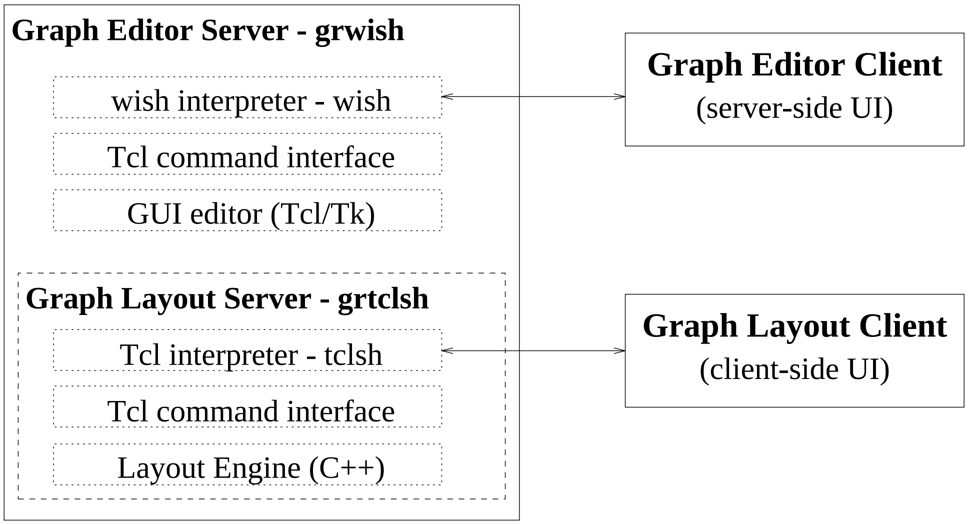

Figure 6: Graphite client-server model

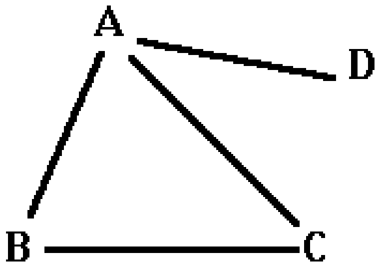

Figure 7: Query graph

4.3 Web Structure Graph

Visualization of web structure may also be useful during web browsing. The web of hypertext documents browsed during a session can be visualized as graphs to help users control their browsing activities. Figure 10 shows a portion of a graph generated during a browsing session. Note that icons are used to represent different types of URLs (e.g., HTML documents, gopher, etc.), and arcs are coloured to show visited and unvisited nodes.

Web structure visualization is currently an active topic, and interested readers are referred to [8] to learn more about web structure visualization and the use of GraphLog [6] queries to manipulate web visualizations.

5 Conclusion

The MultiSurf system described in this paper combines query-mediated browsing, different types of visualization, and the integration of local and Internet databases, thus providing a useful tool for general information exploration. The query-mediated browsing feature in the MultiSurf system allows a user to browse through the web by selecting arbitrary words or passages in the web page, in addition to following *anchored* hypertext links. The location and existence of different index search engines accessed during a query-mediated browsing session are transparent to the user. The visualizations supported by the system attempt to reduce the “lost in hyperspace” and related problems.

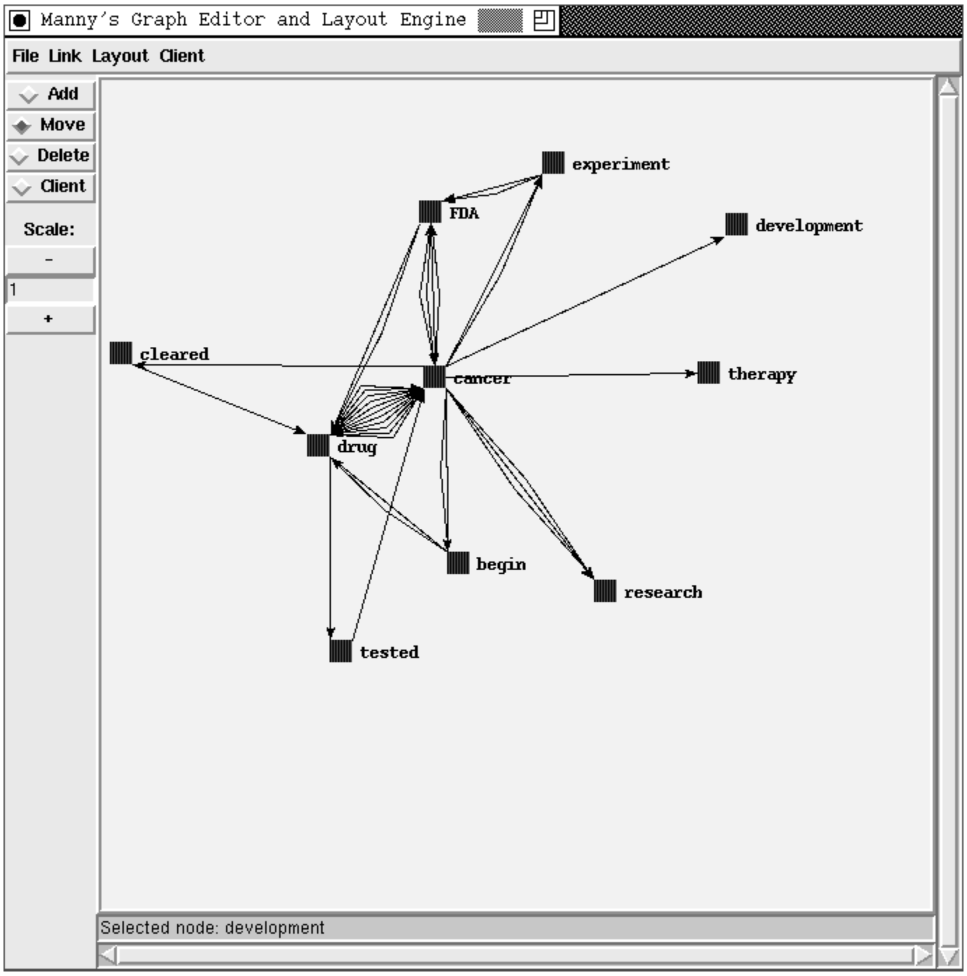

Figure 8: Graphite query graph visualization

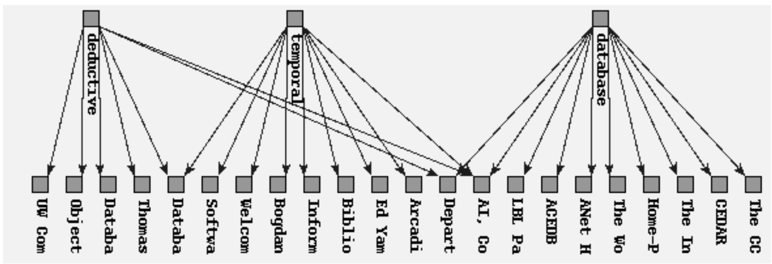

Figure 9: Query hit-list visualization

There are several directions for on-going research related to MultiSurf, including QRL markup of HTML pages, and various issues related to the visual representation of a user's interaction with the system. The current implementation supports one-way communication from the query-mediated browsing system to the Motif-based controller. A complementary connection could pass HTML documents fetched over the Internet to the ST-PaTREC

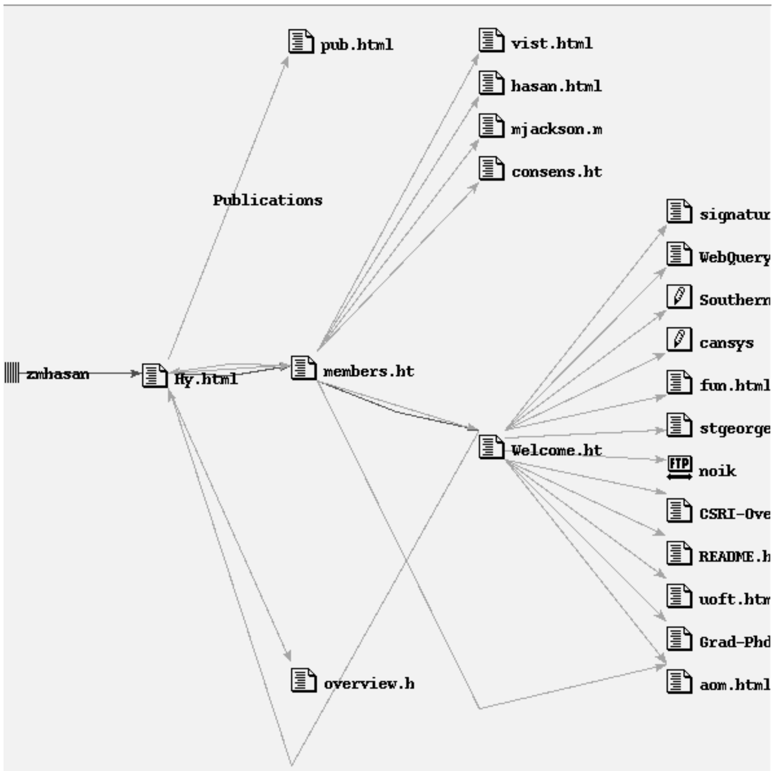

Figure 10: Visualization of a web browsing session

system. Users will then be able to edit the query directly on a retrieved web document. Another alternative, currently under development, is to incorporate the QRL-style query notation directly into a web client. Another possible direction is to investigate the efficacy of the *link-resolving components* approach [16] as an alternative to the current MultiSurf architecture.

Another interesting area for further research is how to use visualization effectively in this query-mediated browsing approach. Clustering may help in visualizing concepts and documents [4]. Scatter/gather (a layout strategy recently developed at Xerox Parc [14]) may also be used to structure visualization maps. Since the Web and associated Internet resources are still evolving, it is unclear how searching and overviewing functions will eventually be integrated into Web exploration. However, MultiSurf represents an interesting alternative to the current methods of web exploration that maintain a sharp separation between querying and browsing interfaces.

References

[1] M. Bowman, P. Danzig, D. Hardy, U. Manber, and M. Schwartz. Harvest: A scalable, customizable discovery and access system. Technical report, University of Colorado - Boulder, August 1994. Report no. CU-CS-732-94.

[2] C. Buckley, J. Allan, and G. Salton. Automatic retrieval with locality information using SMART. *Proceedings of the First Text REtrieval Conference TREC-1*, pages 59–72, 1993.

[3] J.P. Callan, W.B. Croft, and S.M. Harding. The inquery retrieval system. *Proceedings of the Third International Conference on Database and Expert Systems Applications*, pages 78–83, 1992.

[4] M. Chalmers and P. Chitson. Bead: Explorations in information visualization. *Proceedings of SIGIR*, pages 330–337, 1992.

[5] M.P. Consens, F. Ch. Eigler, M.Z. Hasan, A.O. Mendelzon, E.G. Noik, A.G. Ryman, and D. Vista. Architecture and applications of the Hy+ visualization system. *IBM Systems Journal*, 33(3):458–476, August 1994.

[6] M.P. Consens and A.O. Mendelzon. GraphLog: a visual formalism for real life recursion. In *Proceedings of the Ninth ACM SIGACT-SIGMOD Symposium on Principles of Database Systems*, pages 404–416, 1990.

[7] G. Golovchinsky and M.H. Chignell. Queries-r-links: Graphical markup for text navigation. In *InterCHI 93*, Amsterdam, Netherlands, May 1993.

[8] M.Z. Hasan, A.O. Mendelzon, and D. Vista. Visual web surfing with Hy+. 1995. To appear in CASCON 1995.

[9] J. Nielsen. *Multimedia and Hypertext, The Internet and Beyond.* AP Professional, first edition, 1995.

[10] E.G. Noik. Exploring large hyperdocuments: Fisheye views of nested networks. In *ACM Hypertext '93*, pages 192–205, Seattle, WA, November 1993.

[11] E.G. Noik. Layout-independent fisheye views of nested graphs. In *VL '93: IEEE Symposium on Visual Languages*, pages 336–341, Bergen, Norway, August 1993.

[12] E.G. Noik. A space of presentation emphasis techniques for visualizing graphs. In *GI '94: Graphics Interface 1994*, pages 225–234, Banff, AL, Canada, May 1994.

[13] John K. Ousterhout. *Tcl and the Tk Toolkit.* Addison-Wesley, 1994.

[14] R. Rao, J.O. Pedersen, M.A. Hearst, J.D. Mackinlay, S.K. Card, L. Masinter, P-K. Halvorsen, and G.G. Robertson. Rich interaction in the digital library. *Communications of the ACM*, 38(4):29–39, 1995.

[15] B. Shneiderman. *Designing the User Interface.* Addison-Wesley, 1987.

[16] F.Wm. Tompa, G.E. Blake, and D.R. Raymond. Hypertext by link-resolving components. In *ACM Hypertext '93*, pages 118–130, Seattle, WA, November 1993.

[17] J.A. Waterworth and M.H. Chignell. A model of information exploration in a real-estate information exploration system. *Hypermedia*, 3(1):35–58, 1991.

[18] C. Williamson and B. Shneiderman. The dynamic homefinder: Evaluating dynamic queries in a real-estate information exploration system. *Proceedings of SIGIR*, pages 338–346, 1992.

About the Authors

Masum Z. Hasan is a Research Associate at the Computer Systems Research Institute, University of Toronto and a Ph.D candidate in the Department of Computer Science, University of Waterloo. He obtained BEng, MEng in Computer Engineering from the former USSR and MMath in Computer Science from University of Waterloo. His research interests are in active temporal databases, data/telecom network management, networked document browsing/searching, distributed and parallel programming environment, and visualization.

Gene Golovchinsky is a Ph.D. candidate in the Human Factors group of the Industrial Engineering Department at the University of Toronto. He received a M.A.Sc. from the same department, and also holds a BS in Electrical Engineering from UCLA. Gene's research interests include information retrieveal interfaces, information visualization, interfaces for programemrs and interaction styles.

Emanuel G. Noik is currently completing his PhD in the Department of Computer Science at the University of Toronto. He received his BMath from the University of Waterloo in 1988 and his MSc in Computer Science from the University of British Columbia in 1990. His interests include the design and implementation of user interfaces, human-computer interaction, relational data visualization, graph layout, hypermedia, and the World-Wide Web.

Nipon Charoenkitkarn is a Ph.D. candidate in the Human Factors group of the Industrial Engineering Department at the University of Toronto. He received an M.Sc. degree in Engineering Management from The California State University at Northridge, and also holds a B. Eng. in Computer Engineering from King Monkgut's Institute of Technology, Bangkok, Thailand. Nipon's research interests are human-computer interaction, information management and retrieval, hypertext and usability.

Mark Chignell is an associate professor of Industrial Engineering at the University of Toronto. He has a Ph.D in Psychology from the University of Canterbury (New Zealand, 1981) and an MS in Industrial and Systems Engineering from Ohio State University (1984). He taught Psychology at Monash University (1980-82), and was a postdoctoral fellow at the Human Performance Laboratory (Ohio State) in 1982-83. He was an assistant professor of Industrial and Systems Engineering at the University of Southern California (1984-1990) before moving to Toronto. He has co-authored books on expert systems and intelligent databases (published by John Wiley and Sons) and has research interests in the areas of user interface design and information technology. His current research focus is on multimedia information systems and electronic books. His research is presently funded by ITRC, NSERC, Ricoh Corporation, and AECL.

Alberto O. Mendelzon is a Professor in the Department of Computer Science at the University of Toronto. His interests include the theory, practice and applications of databases and knowledge bases, and in particular the data visualization and data manipulation interfaces for them.

David Modjeska is a Ph.D. student in the Department of Computer Science at the University of Toronto. He works in the area of Human-Computer Interaction. Previous degrees include a M.S. in Computer Science from Stanford University and a B.A. in English from Harvard University. His current research interests are hypermedia, information navigation and visualization, and the World Wide Web.

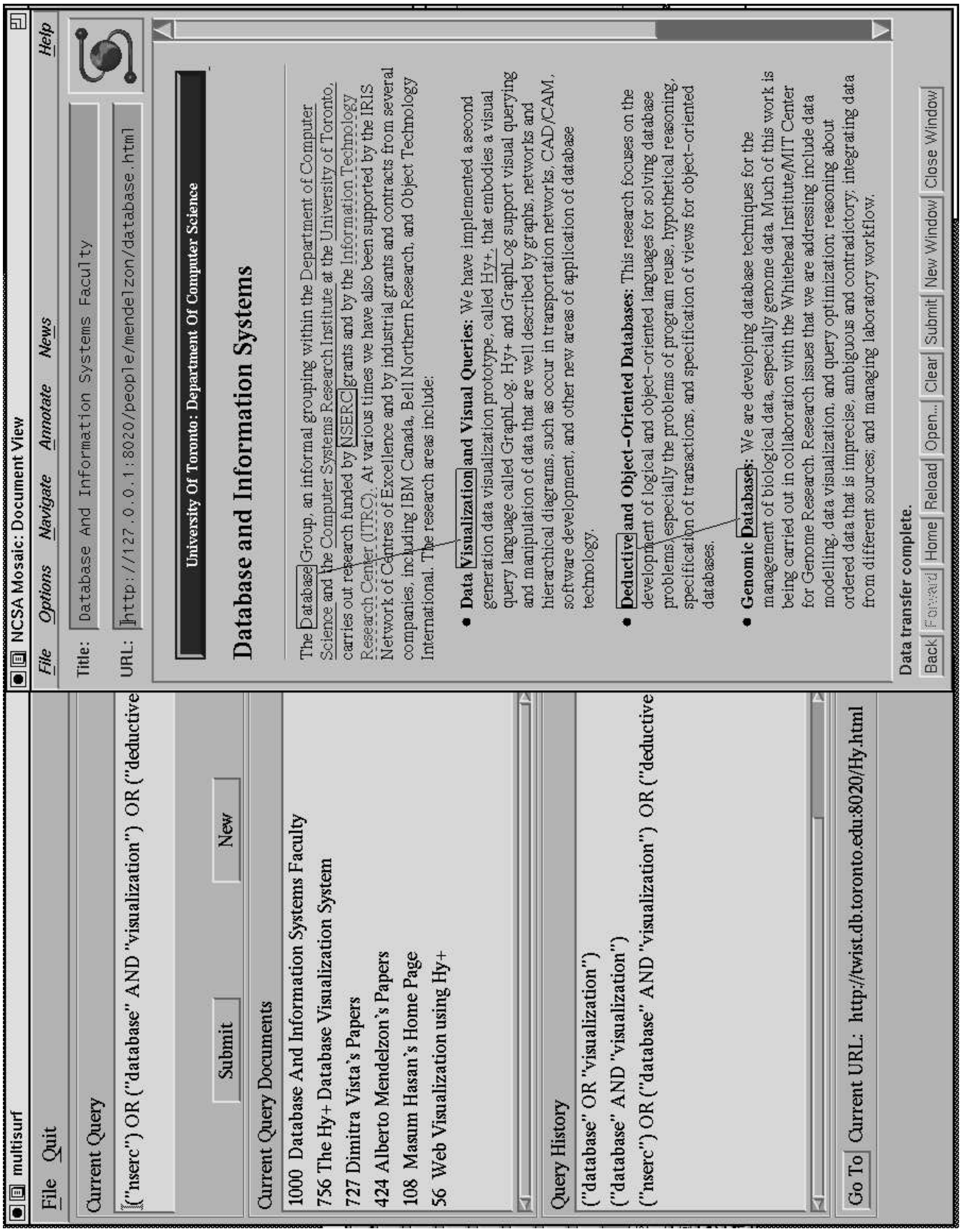

Figure 11: QRL Query Markup in Mosaic 2.6 and MultiSurf Control Panel

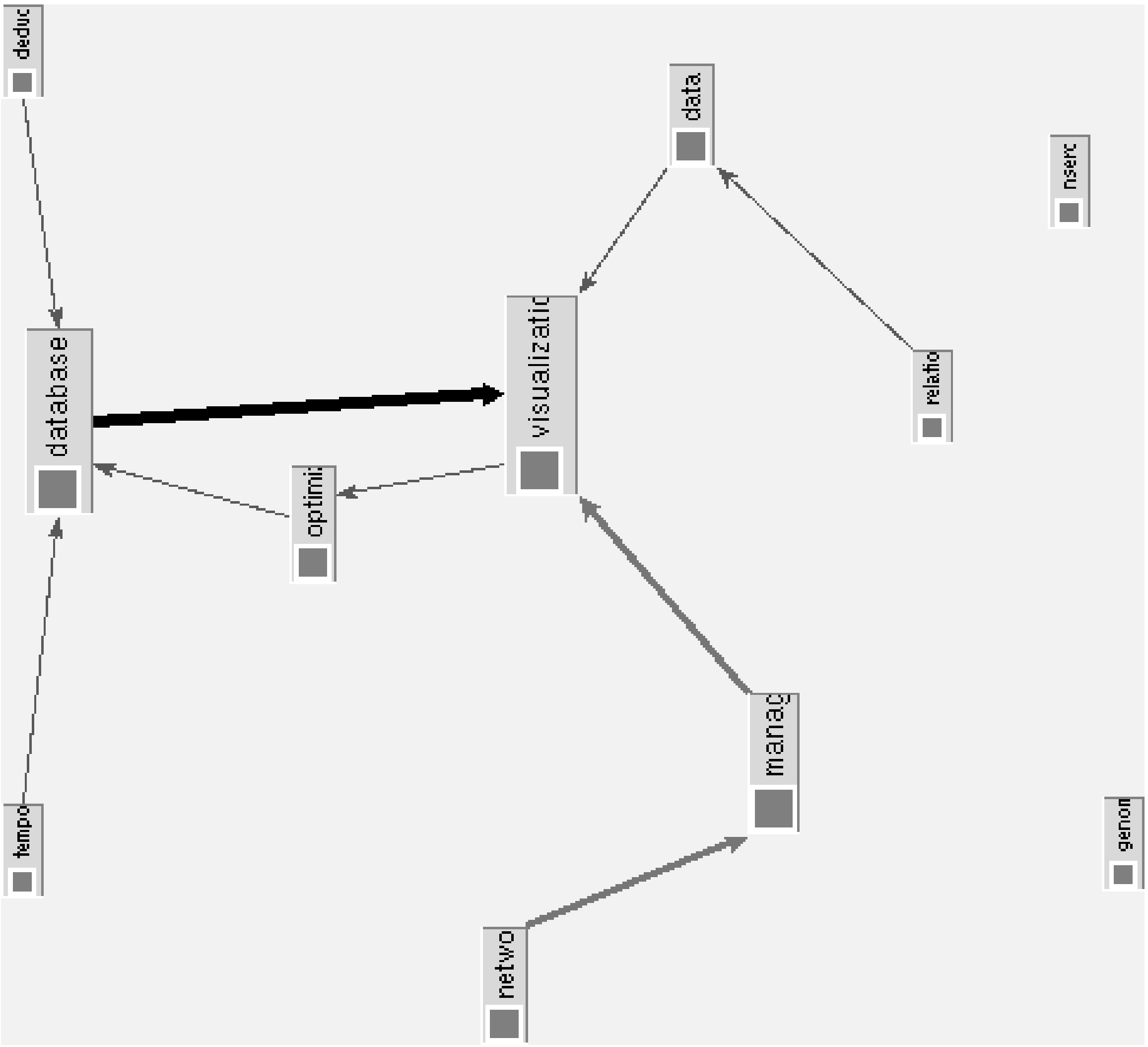

Figure 12: QRL Query Concept Map