Sustaining Interaction in Database Query

R. Inder[a] [*] and J. Stader[b]

[a]Human Communication Research Centre, University of Edinburgh,
2, Buccleuch Place, Edinburgh EH8 9LW Scotland

[b]AI Applications Institute, University of Edinburgh,
80, South Bridge, Edinburgh EH1 1HN Scotland

Current research on database systems breaks the problem into two halves—formulating a query and presenting the results. This paper suggests that there is a third "half" which deserves attention in its own right: sustaining the interaction.

1. Introduction

When computers stored numbers and short strings, which they divulged through printers and VDUs, getting information from them was a matter of specifying reports by typing a string of characters. This is adequate for most uses of computer data, which is for routine purposes: warehousemen load lorries, passengers catch their planes, customers check and pay their bills. Where tasks and data needs are fixed, analysing them and supporting them with appropriate forms or reports is straightforward.

But now, as more and more information is held in networked, multi-media databases, decision makers will increasingly find that relevant information is available on-line. Most will neither have, nor want to acquire, specialised computer skills: they will want interactive tools with sophisticated interfaces to help them access it [1]. Current research focuses on two aspects to getting information from a computer:

Query Languages These let users identify the information they are interested in [2]. This includes helping them determine what data exists and how it can be accessed (e.g. by a graphical or menu-based presentation of the database structure [3]), and letting them specify constraints on the objects of interest.

Information Presentations Apart from producing tables and "reports", there are many ways to present data about a set of objects. "Business Graphics"—pie charts histograms etc.—are now common, and more sophisticated displays are being developed [4].

Work in each of these areas typically excludes the rest of the query process. Thus GQL [5] has sophisticated graphical conventions for indicating the logical structure and quantification of the query, but specifies information presentation with "ticks" against

[*]Support from Grant # 9018050 from the JCI in Cognitive Science and HCI is gratefully acknowledged.

the attributes to appear in a report. This minimalist mechanism is not a shortcoming of the work, but an (implicit) scoping of the problem being addressed which excludes the data presentation. Work on sophisticated data presentations, again perfectly reasonably, makes complementary simplifications by ignoring the selection of the data and attributes to be presented.

Query and display (even with manipulation and filtering) are complementary parts of an overall solution, but these two "halves" together are not enough to give a satisfactory "whole". Creative information workers typically use a computer not for looking up the answer to their problem, but for gathering and presenting information to help them form an opinion. Any piece of information they see can influence their thinking and suggest further aspects of the problem for consideration. Thus they will tend to formulate sequences of linked questions, each being shaped by what has already been presented. Their tools should support this style of use, which has been termed "data archaeology" [6].

2. Flexible, Sustained Interaction

Some of the more sophisticated information presentations allow users to interact with the display and thus discern more of the information in the data [7–9]. For instance, Homefinder [9] presents residential properties for sale as points on a map. It lets users formulate "dynamic queries" using sliders and buttons to restrict price, garden size and so forth: at any instant the map shows only the properties that satisfy the user's constraints, and individual properties appear and disappear as the user adjusts the controls. Users can mouse on properties of interest to get more detailed information on it.

Such intefaces can be both engaging and straightforward to use. But what if, for example, Homefinder's house hunter needs a public room over a certain size, or at least three rooms on the ground floor? The relevant data may well be in an estate agent's database, but a general-purpose interface is unlikely to make it available. Users could be allowed to configure the interface, selecting attributes to be seen and manipulated, but this requires a whole new layer of complexity in the interface, which will have to support browsing the database schema and configuring displays and controls.

Further issues arise when one considers how users can proceed once they have identified one or more items of interest. Homefinder's house hunters get a picture of the property and a description of the accommodation it offers. But what about interior views, or a floor plan, or even its sales history? The system cannot simply fill the screen with the available data about the selected object, even where there is possible, because users will typically not want the whole screen used in this way: they will want to see it in *context*, or to *compare* a number of objects. Thus users need the ability to control what information is presented, how, and where.

Thus selecting an item of interest is, in general, not the end of the query process: but how should an interface be structured so that users can go forward, so they can continue to search on the basis of what they have learnt from such an interactive display?

It should certainly allow results to be used for launching refined or follow-up queries, akin to "piece meal" queries in GUIDE [10] or "Retrieval by Reformulation" in RABBIT [11]. But if the system is to support users who are not trying to extract a single answer, but to form and support an opinion, it must offer more. Users will produce many packages of information on the way, and the system must recognise that they are all potentially relevant. They must be organised so that the user can review them, understand where they came from and reference them in future queries.

Supporting sustained interaction in these ways is, in effect, a third "half" of the database query process.

3. Supporting Sustained Interaction

Every interface aims to be "easy to use". For a database query system, this involves freeing the user from remembering what the database contains and how it is structured. This can even involve hiding that structure and presenting instead some kind of enhanced or idealised domain model, either to support navigation [10–13] or give database independence [6].

Designing a system for use throughout an extended intellectual task increases the importance of minimising the extent to which it imposes a cognitive load which intrudes upon the user's train of thought.

One component of such intrusion is the effort involved in using the query system itself, and thus it is thus important to minimise Norman's Gulfs of Execution and Comprehension [14,13]. Thus the system should provide a query mechanism which weights ease of use particularly heavily in the balance against expressive power (while possibly making more powerful languages available as well). It should also minimise the number of decisions that the user has to take—for instance, by automatically generating multi-media presentations appropriate to the data retrieved [15,16] and consistent with other collections of similar objects.

But a second component of the cognitive load imposed by a query system comes from the effort of understanding the information it is presenting. Within this, two key components can be identified [17]: being flooded by too much or badly structured information, and disorientation caused by loss of context. In one sense, the context of a query is provided by the structure of the data, and navigation tools that make the data model available to the user in some way [3,10,12] can help prevent disorientation. But in another sense the context of a query comes from the user's activities—from the queries that have gone before and the results that they produced. And so, over and above making it easy to ask a question and see the answer, an information system can help the user sustain an interaction by providing an abstract indication of the information retrieved in a way that helps users keep track of the context within which they are working.

Most obviously, the system should ensure that all the sets of results that a user generates during a session can be presented, with information about how they inter-relate, in a form that users can use in structuring and reviewing their session, and in driving it forward.

Moreover, the system should store the basis on which each set of results was generated. This lets it allow users to return to any previous information and be reminded of the "question" that produced it. They can then either modify that question or use it as a context for a follow-up query, working either in the formalism of the original question or in some other query language. In particular, they should be able to "drill down" into the data, not just along paths that have been anticipated and pre-prepared, but in any way they choose.

4. A System that Supports Interaction

The previous section identified several ways that users can be helped to sustain an interaction with the system. This section briefly describes a particular approach to structuring interfaces, known as "Bags and Viewers" [18], which addresses many of them. The approach was developed during the construction of PEXES, a system for providing an interface to geological data [19], which resulted in SDBA, a front-end that exhibits many of its features.

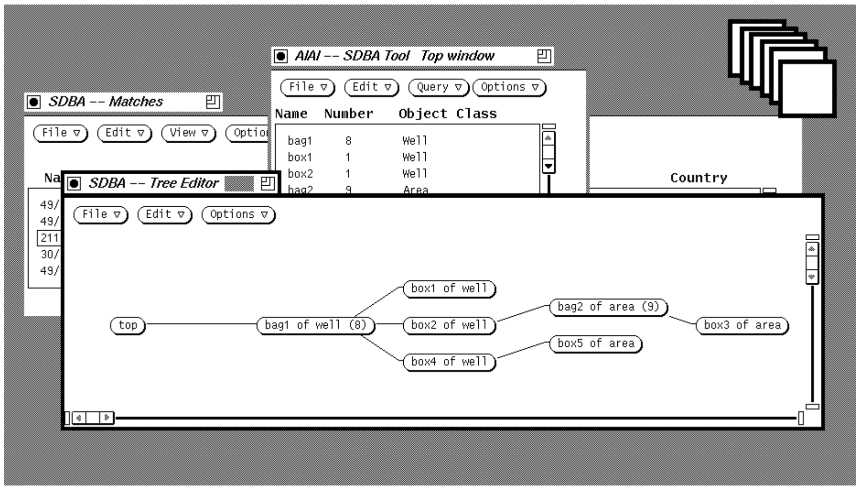
However the data is actually stored, the user of a Bags and Viewers interface sees an "object-oriented" system—i.e. one that contains data about the values of the various attributes of distinct real-world objects. This structure of this model of the world is elicited from domain experts when the interface is being configured for the type of subject matter, typically without reference to the way it is actually stored. The content of the/each database to be accessed is described in terms of this data model, and the interface is responsible for transforming queries and results between the idealised data model and the way the underlying data is actually stored [20].

Bags and Viewers interfaces separate the tasks of specifying the objects of interest and deciding how they should be displayed. Users explicitly identify those objects that are of interest by constraining their attribute values. Any formalism can be used for this, provided it results in some logical combination of constraints on attribute values. The system gathers the objects so specified into a logical *bag*—a *set* in mathematical terminology, but with an indication of the criteria used to determine its contents.

Objects "in" a bag can only be seen if one or more *viewers* are chosen to present information about them. A viewer may be a "form", completed with the values for a collection of attributes, or a report that displays a brief summary of each of a number of objects. But equally it could be a map, or a chart, or any domain-specific display synthesised from data about the relevant object(s)—e.g. a geological cross-section. Different viewers present different meaningful subsets of the information available about the objects, possibly at different levels of abstraction—a menu may explicitly represent every individual in a set in terms of its values for several attributes, whereas a histogram will present the set only as aggregations on one dimension. Thus the choice of viewer controls the flood of information reaching the user, and must ultimately reside with the viewer. However, in order to reduce the number of things that must be specified to launch a query, the system undertakes to select a default viewer for each bag which is created.

Crucially, the bag is to be thought of as containing *domain objects*, and not data about any of their attributes. Thus choosing a viewer to display certain attributes of the objects in a bag in no way restricts the use of another viewer to present those objects completely differently. Each viewer allows additional viewers to be called up to display other aspects of the same objects, and both viewers will be updated if the contents of the bag is changed. However, because very few viewers can meaningfully handle either a single object or several, SDBA also features *boxes*, which are bags that contain just a single object.

Most viewers allow the user not only to call up other views of the objects they are presenting, but also to create bags of objects related to them. Getting information by using this kind of "relative" enquiry feels somewhat like hypertext links [21], and thus



Name	Number	Object Class
bag1	8	Well
box1	1	Well
box2	1	Well
bag2	9	Area

Figure 1. The History Mechanism in SDBA, showing the bags and boxes that have been created during the session and how they relate to one another.

the system can be seen as presenting the database as a hypertext document [22], albeit one where the user can modify, during a session, the links and pages present. As a result, it is not surprising that it is not immune to the navigation problems that are known to beset hypermedia systems [21]. However, the typed nodes and links provide plenty of structure for providing tools to help users keep track of what they have been doing [23]. In particular, SDBA provides users with a history mechanism (see Figure 1) which allows them to immediately access any bag generated earlier in the session.

5. Conclusion

Although SDBA is only a prototype system, but it has served to draw attention to a particular aspect of making an information system "easy to use": helping users sustain an interaction. It has also suggested one way in which this can be done, by providing mechanisms to allow users to see and review the structure of their session and use it for initiating further activities. Finally, it has highlighted an area where there is real scope for cross-fertilisation between work on database query and hypermedia.

REFERENCES

1. P. Sawyer (ed.), *Proc. Interfaces to Database Systems*. Springer Verlag, 1994
2. C. Batini, T. Catarci, M.F. Costabile and S. Levialdi, Visual Query Systems, Research Report, Dipartimento di Informatica e Sistemistica, Universita Degli Studi di Roma "La Sapienza". April 1991.
3. J. Boyle, J. E. Fothergill and P. M. D. Gray, Design of a 3D User Interface to a Database, in [1].

4. G. Robertson, S. Card and J. Mackinlay, Information Visualization using 3D Interactive Animation, *CACM*, **36**(4), 1993.
5. A. Papantonakis and P. King, GQL, a Declarative Graphical Query Language Based on the Functional Data Model, in T. Cararci, F. Costabile, S. Levialdi and G. Sanducci (eds.), *Proc. Workshop on Advanced Visual Interfaces (AVI'94)*, ACM, 1994.
6. R. Brachman, P. Selfridge, L. Terveen, B. Altman, A. Borgida, F. Halpern, T. Kirk, A. Lazar, D. McGuinness and L. Resnick, Integrated Support for Data Archaeology, *Int. Jnl. Intelligent and Cooperative Information Systems* **2**(2), 1983
7. M. Hemmje, C. Kunkelf and A. Willet, LyberWorld - A Visualization User Interface Supporting Fulltext Retrieval. In *Proc. ACM SIGIR '94*, Dublin, 1994.
8. L Tweedie, B. Spence, D. Williams and R. Bhogal, The Attribute Explorero, Video Proceedings CHI'94 Boston, ACM Press, 1994.
9. C. Williamson and B. Shneiderman, The Dynamic Homefinder: Evaluating Dynamic Queries in a Real-Estate Information Exploration System, *Proc. ACM SIGIR Conference on Information Retrieval*, Copenhagen, 1992.
10. H. Wong and I. Kuo, GUIDE: Graphical User Interface for Database Exploration, in *Proc. VLDB 8*, VLDB Endowment, 1982.
11. F. Tou, M. Williams, R. Fikes, A. Henderson and T. Malone, RABBIT: An Intelligent Database Assistant, in *Proc. AAAI 82*. AAAI, 1982.
12. G. Santicci and P. Sottile, Query by Diagram: a Visual Environment for Querying Databases, *Software: Practice and Experience*, **23**(3), 1993.
13. D. Haw, C. Goble and A. Rector, GUIDANCE: Making it Easy for the User to be an Expert. In [1].
14. E. Hutchins, J. Hollan and D. Norman, Direct Manipulation Interfaces, in D. Norman and S. Draper (Eds.) *User Centered Design*, Erlbaum, 1986.
15. J. Mackinlay, Automating the design of graphical presentations of relational information. *ACM Transactions on Graphics, 5* (2) 1986.
16. E. Hovy and Y. Arens, When is a Picture Worth a Thousand Words? — allocation of modalities in multimedia communication. In *AAAI Symposium on HCI*, 1990.
17. H. Hohl, J. Herczeg and M. Ressel, An Interactive Design Environment for Graphical Browsers, in G. Salvendy and M. Smith, *Proc. HCI International '93*, Elsevier, 1993.
18. R. Inder and J. Stader, Bags and Viewers: a metaphor for database access. In [1].
19. R. Inder and B. Wells, PEXES: Combining Knowledge and Data in a Tool for The Explorationist, *Proc. Conf. AI in Petroleum Exploration and Production (*CAIPEP*)*, 1991.
20. J. Stader, R. Inder and P. Chung, Transforming databases for experts. In *Proc. of the Seventh Intl. Conf. on Industrial and Engineering Applications of AI and Expert Systems*, Gordon and Breach, 1994.
21. J. Conklin, Hypertext: An Introduction and Survey, *IEEE Computer* **20**, 1987.
22. Y. Masuda, Y. Ishitobi and M. Ueda, Frame-Axis Model for Automatic Information Organizing and Spatial Navigation, *Proc. European Conference on Hypermedia Technology (*ECHT94*)*, ACM Press, 1994.
23. R. Inder and J. Oberlander, Using Discourse to Aid Hypertext Navigation. This volume.