Bags and Viewers: A Metaphor For Intelligent Database Access

Robert Inder [1] and Jussi Stader

AIAI-TR-127

January 1994

submitted to Advanced Visual Interfaces (AVI'94), Bari, Italy

Artificial Intelligence Applications Institute (AIAI)
University of Edinburgh
80 South Bridge
Edinburgh EH1 1HN
United Kingdom

[1] previously at AIAI, now at Human Communication Research Centre (HCRC)
The University of Edinburgh, 2 Buccleuch Place, Edinburgh EH8 9LW

Abstract

We present a way of structuring a database query system to form a bridge between current data handling systems and the data requirements of creative work. The interface is based around specifying the contents of “bags” of objects and inspecting them using “viewers”, which can then be used to launch further queries.

The operation of such an interface is described and illustrated by an example session using a prototype system. A number of features of such an interface are presented, the metaphor is discussed and a number of areas for future work are indicated.

Contents

1 Introduction

In many disciplines, a growing number of experts are carrying out creative work using data held, at least in part, on computers. Accessing this data is an increasingly significant part of their work. The data they require at any moment may depend on their interpretation of the data they have already retrieved, making very flexible but powerful query mechanisms essential.

Formulating queries in most database systems—e.g. using SQL—requires precise knowledge of the language and of the structure of the database. However, many experts who could benefit from using on-line data neither have, nor wish to acquire, substantial computer skills. Interfaces can be built to allow data to be retrieved without knowledge of database structures—e.g. based on "forms"—but only to the extent that the user's data needs can be anticipated, which restricts the user to common paths of enquiry (see Figure 1).

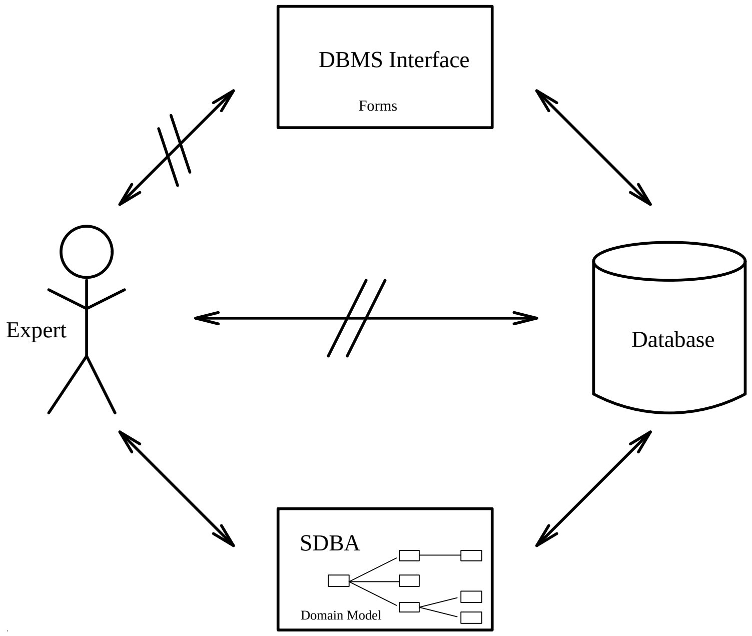

Figure 1: Experts' Routes to Access Data

Work on addressing this problem for a particular class of users—geologists working in petroleum exploration—gave rise to a framework for structuring database access activities. This framework is both intuitive and flexible enough to support creative work with unpredictable requirements. In the next section, we outline this framework, presenting the "*Bags and Viewers*" metaphor as it appears to the user. In Section 3 we illustrate the use of such a system using screen dumps from a prototype system that is under development. Section 4 then presents a number of advantages of the approach, Section 5 discusses some specific points in more detail, and Section 6 identifies some areas where further work is required.

2 Bags and Viewers: The Metaphor

Bags and Viewers describes an approach to structuring the interface to a database query system in such a way that users can incrementally perform complex data retrieval operations without needing to know a query language or the structure of the database being accessed.

Databases must be designed with efficiency and compatibility in mind, which means that their structures do not usually reflect the structures of the real world, or the way experts think about the information the databases hold. Regardless of how the data is actually stored, it is presented to the user as structured in terms of *objects* with *attributes*, and with *relationships* between them. The objects and attributes used to present the data are specified in an explicit model of the field to which the data applies—the *domain*. This domain model is derived from experts in the field, and is quite independent of any particular database and the pragmatic pressures on its design. This idealised structure is mapped to a particular database by means of a database model, a formal description of the contents of the database in terms of the concepts in the domain model.

Both the domain model and the database are described using a conventional "object-oriented" formalism supporting class hierarchies and inheritance. Graphical tools have been developed to assist in the construction of these knowledge bases, which are used to tailor the system's behaviour to the class of the object or attribute involved.

An interaction with the system starts with the user selecting an object type and specifying constraints on the values of its attributes by using *constraint editors*—tools customised for handling and constraining the values of a particular attribute.

At any time the user may ask the system to either count the objects that satisfy the currently specified set of constraints, or to "fetch" them. When the system "fetches" objects, it will by default create a new *bag* to contain them, and present the user with a "tag" for that bag. A bag is a logical container for a collection of objects that satisfy some particular set of constraints, and its tag will indicate the type and number of objects it contains. A tag could be an icon, or possibly an entry in a session-wide "bag" menu, or any other reminder of its existence.

Bags are "opaque", and it is not possible to tell much from a bag itself about its contents—typically only the type and number of objects it contains. To see more, it is necessary to attach one or more *viewers* to the bag, each of which will show some attributes of the objects within. Whenever the system creates a bag, it will also attach a default viewer to it, so that the user can see something of the objects the bag contains. This viewer could display a one-line summary of each object; or, depending on the type of object and the user's preferences, it could be a map, a statistical plot, or any form of data presentation that is suitable for groups of such objects. If the default viewer does not give the required information, the user can re-configure it—for instance, by changing the attributes that form the summary or parameterising a cross-plot in terms of attributes to be used for axes, point shape, colour etc.—or select different or *additional* viewers. Thus the user can decide, in the light of what objects have been selected, how they should be displayed.

Whenever objects (or sets of objects) are fetched from the database, the user can choose whether they should be placed into a new bag, or into an existing one.

Where an existing bag is used, all of the viewers that have been associated with that bag are updated to reflect its new contents. Thus the user can build up an ideal view of a set of objects by associating combinations of viewers with the bag, and then immediately use that view for another set of objects by putting them into the same bag.

Whenever a new bag is used, the system keeps track of how it was created, and can graphically present the complete set of bags created during the session in the form of a *session history* which the user can review easily. As well as containing a set of objects, a bag also retains the constraints that were used to specify that set. These can be used to update the bag, or to re-fill it on subsequent occasions. More importantly the user can return to the constraints at any time and either review them or use them as the basis of another query. This makes it easy to explore alternatives from any point in the session, or refine the query to improve the quality of the results.

Most viewers are interactive, and allow the user to select objects or sets of objects, and thus request more information about them. For example, the one-line object summaries will function as a menu of the entries in the bag. Each selected object is fetched and put into a *box*—a bag for just a single object. Once again, a default viewer will be chosen. This might be a "form"—i.e. a set of attribute names and their values arranged on the screen. However, it could be any display appropriate for the type of object, either pre-stored (such as a photograph) or generated from a collection of data which is retrieved as required. Box viewers, like bag viewers, can be used to seek further details either of the object in the box or of related objects. The user can specify a relationship, and get a bag of the objects to which the one in the box is related in that way. Because viewers are interactive, the interface is very good for generating follow-up queries, and for unconstrained browsing through the data.

A *Bags and Viewers* interface not only hides the structure of the database, but also shields the user from the need to consider the inevitable shortcomings of the data itself, which is often incomplete, sometimes imprecise, and occasionally simply wrong. These factors can be a substantial obstacle for unfamiliar users. However, during the process of mapping the user's query from the domain model to the actual database, the system can ensure that the possibility of missing data is handled appropriately—e.g. by ensuring that missing data cannot cause a match to be rejected. It can also make allowances for uncertainties in the data, and indeed the user's intentions, by allowing constraints to be "fuzzy"—i.e. considering objects which fail to satisfy one or more of the stated constraints, provided that, overall, they are not "too far" away from the stated requirements.

3 The Movie

To give an indication of how the system is used we go through a simple session using a database about geological wells.

When the system is started a top window appears on the screen. A typical session begins with a query.

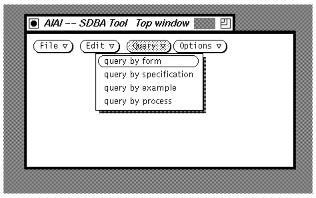

Figure 2: Query Specification

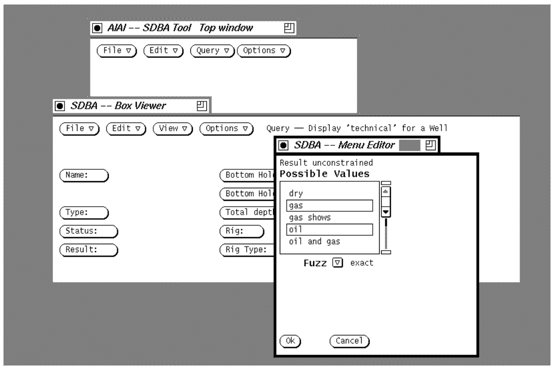

Figure 3: Menu Constraint Specification

Query Specification

As shown in Figure 2 the query menu offers the different query types for selection. We choose query by form which is the simplest way of querying.

We use a form that contains technical information about wells for our query. We are interested in wells that have a result of oil or gas. This will be our first constraint. Clicking on the Result button brings up a constraint editor which is tailored for dealing with the results of wells. As Figure 3 show, it presents us with a list of possible results, from which we select oil and gas , and then OK to dismiss the editor.

Our "oil or gas" constraint appears in the Result button, as shown in Figure 4, as does another constraint, generated in a similar manner, that the Status of the wells of interest must be "completed". Figure 4 also shows a numeric constraint editor being used to constrain the value of the Total Depth to a range of values between 9000 and 15000 feet. When this third constraint has been specified, and its editor dismissed, we can collect the set of wells that satisfy them into a *bag* by selecting the fetch into new bag from the "Options" menu (Figure 5). The form used to generate the query is removed, and replaced (Figure 6) by a viewer onto the new bag, which takes the form of a table showing the eight wells which satisfy the three constraints. In addition, a note of the new bag appears in a list in the query system's top window.

Navigation and Viewing

To find out more about a well, we can select it in the bag viewer's table and select View in detail from the viewer's "View" menu. This causes the selected well to be

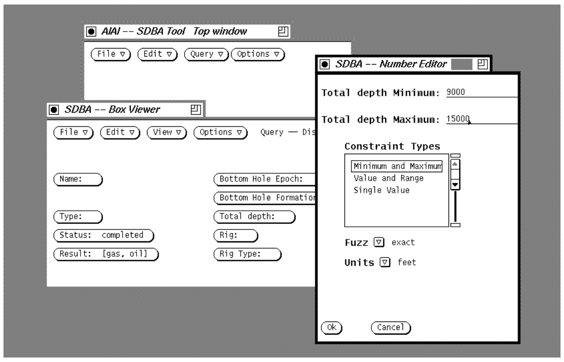

Figure 4: Numeric Constraint Specification

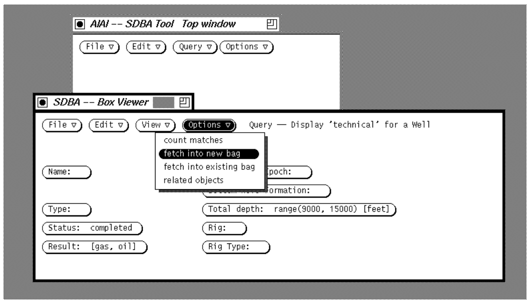

Figure 5: Fetching the matches

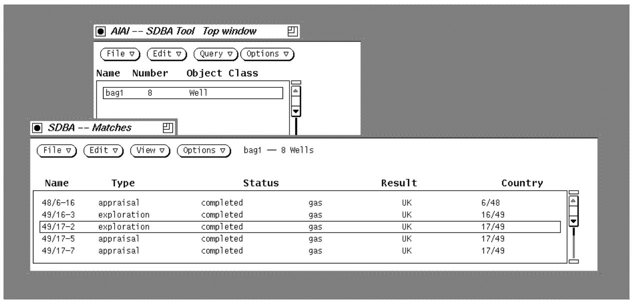

Figure 6: Viewing the Matches

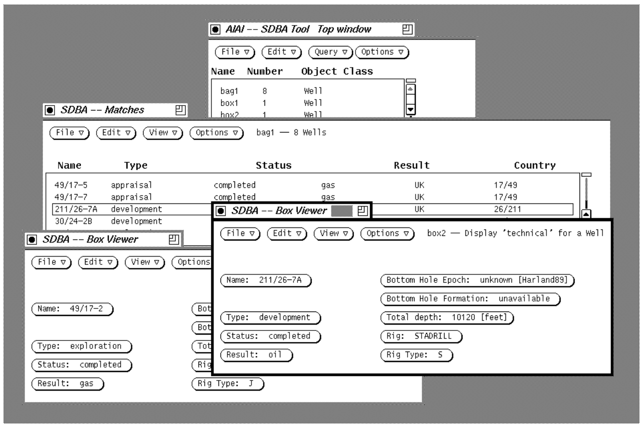

Figure 7: Details

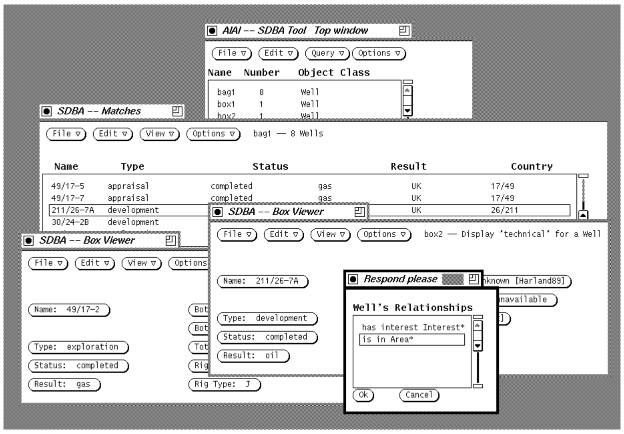

Figure 8: Related Objects

placed in a *box*, and its details to be shown using an appropriate form viewer: Figure 7 shows the result of doing this for two wells.

From the details of a single well, we can easily get information about other objects by selecting related objects from the "options" menu for the form viewer. This brings up a menu (see Figure 8), which shows that there are two relationships between a well and other objects: a well has *interests* and a well *is in* one or more areas. The asterisks indicate that both these are relationships to several other objects. We select is in Area* to find out which areas well '211/26-7A' is in.

We are presented with a viewer (Figure 9) showing summary information for a new bag containing nine areas, all of which are in the Northern North Sea. As before, we can select an area to view in detail, which is duly placed in a box and a form viewer created to display further information about it (see Figure 10).

We now have a quite a number of viewers data on our screen, and it is beginning to look cluttered. Because each viewer is in a separate window, we can use the facilities of our window manager to open and close, re-size and re-organise them on the screen

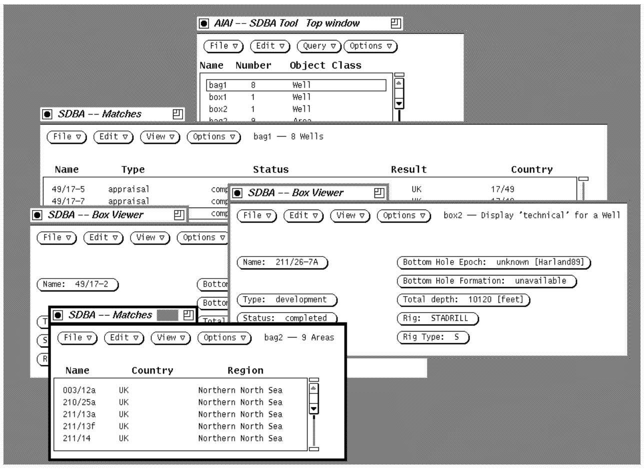

Figure 9: Related Areas

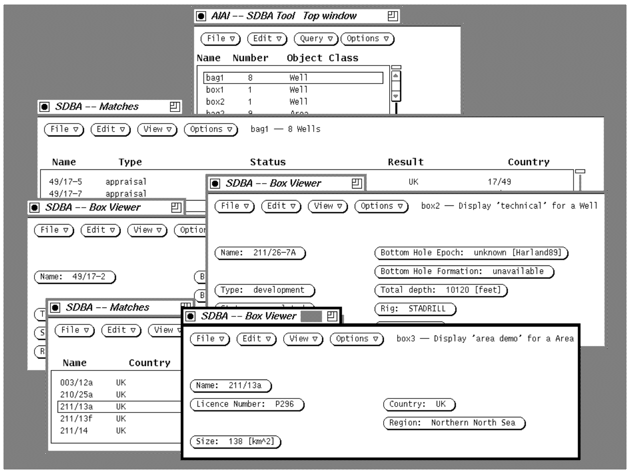

Figure 10: Related Areas

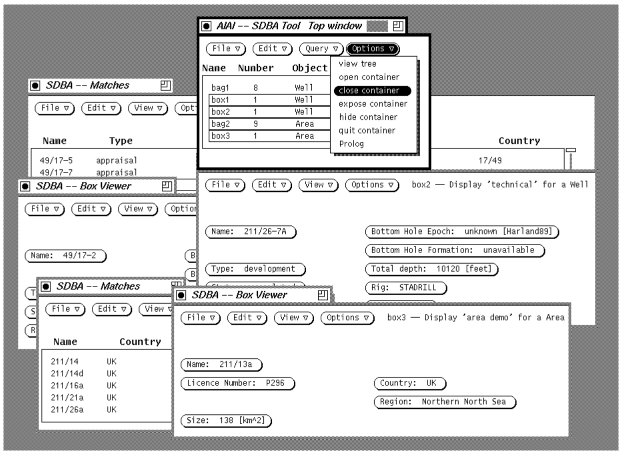

Figure 11: Tidied Screen

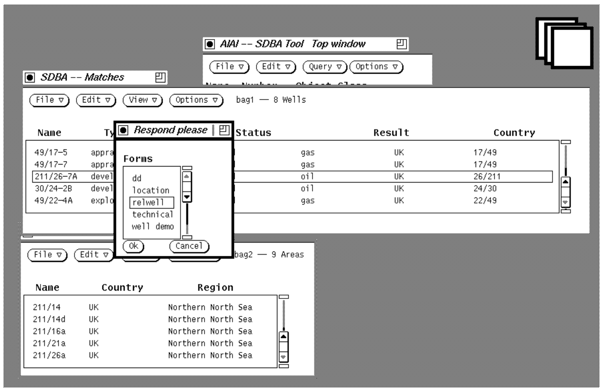

Figure 12: Changing Viewer

however we think best. In addition, we can manipulate them from the top window. By selecting entries in the bag and box menu and then choosing the close container from the Top Window's "options" menu, we can iconify the viewers that are not immediately relevant (see Figure 11). Other entries on the menu let perform related operations on the viewers, such as re-opening or exposing them , or deleting them altogether.

So far, when we requested details of individual objects from a bag, the system chose a default form viewer for us. Figure 12 shows how we can change the details viewer used for displaying information. We use the select details viewer option from the "View" menu of the bag viewer, which presents us with a list of available forms for displaying details about wells. We choose the rel well form. Selecting a well from the table in the viewer will now display different details (see Figure 13). Note that this viewer has been defined so that the form shows an attribute of some related objects—the areas that Well '49/17-2' is in. The names of these areas, which are the same ones we collected in a bag in Figure 10, are mouse sensitive, and can be used to call up further information on the areas, just as the entries in the bag did (see Figure 14).

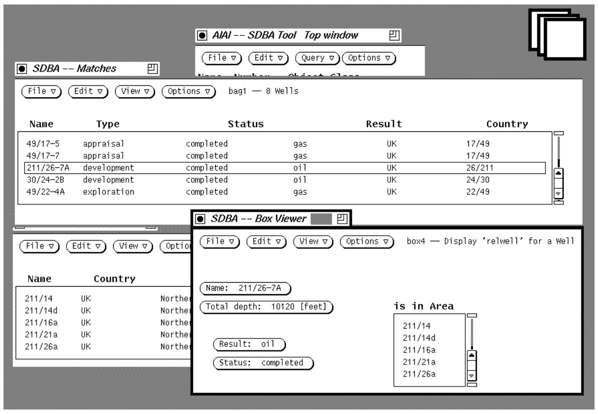

Figure 13: A Well Through Another Viewer

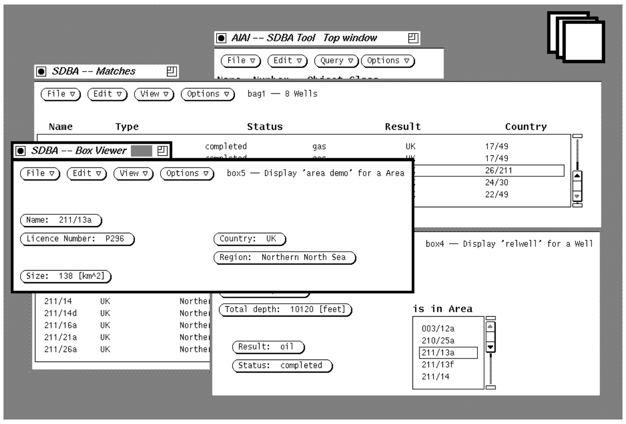

Figure 14: A Shortcut to a Related Object

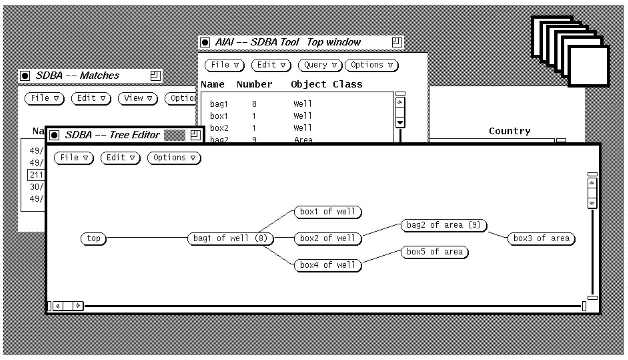

Figure 15: A Session Summary

Session Summary

We have been browsing for some time now, and have many bags and many windows on the screen which contain our (intermediate) results. It is becoming difficult to remember what information is in which window. However, because the system has retained the connections between the bags we have created, we can ask it, by selecting view tree from the "Options" menu in the Top Window, to display a diagrammatic summary of our session so far (See Figure 15). This shows that the session started by generating a bag of eight wells from a query specified from scratch. Viewers onto this bag have been used to create three boxes containing individual wells. One of them was used to obtain a bag of areas, from which we opened a box viewer on an individual area. Another was used directly to open a box viewer on an area.

Like the container list in the top-level window, this summary tree is kept up to date and it can be used to manipulate windows. The tree can also be used to select nodes, or sub-trees of nodes, for the various container-oriented operations—e.g. opening, closing, deleting etc.

Looking Ahead

Everything described so far has been implemented in a prototype system. Because of the limited development effort available, this system has only a limited range of viewers, all of which are text-based. However, one of the advantages of basing an interface on Bags and Viewers is the the ease with which other kinds of data display can be incorporated. It would be straightforward to include other kinds of displays within the overall approach, and Figure 16 illustrates the way that, for instance, map a cross plot viewers could be incorporated within a session.

4 Features of the Approach

The *Bags and Viewers* approach has a number of noteworthy features.

- Data can be accessed without knowledge of query languages or database structure. All interaction is mediated by a domain model, which is elicited from domain ex-

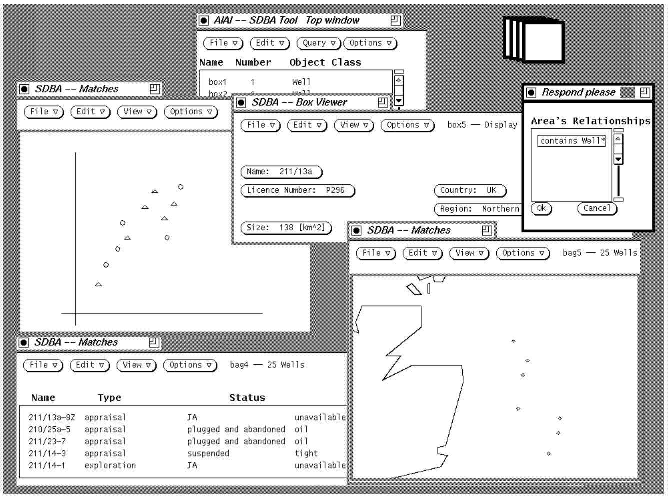

Figure 16: Multi-Media Viewers

perts and can completely ignore all issues of efficiency and consistency—e.g. it can include redundancy, by representing synonyms, derived relationships etc. It is even potentially user extensible, since new attributes can be defined in terms of others, although tools for facilitating this have not yet been developed.

- Because the database structure is hidden from the user, it is possible for multiple databases, possibly with differing structures, to be smoothly integrated. This is especially important in situations involving collaboration or merger between independent database builders or in industries where data is marketed or otherwise distributed.

- Because the domain model is *object-oriented*—that is, it describes the domain in terms of objects positioned within a hierarchical classification scheme—all aspects of the system's behaviour can be tailored to the objects involved. For example, the class of the objects involved in a query is used to determine the appropriate constraint editors and viewers.

- Because the domain model explicitly represents the properties and relationships of the various classes of object that the domain involves, it can be used to help users navigate through the database. Both the properties of a class of object and the relationships in which it can participate can be presented to the user in the form of menus. Because such menus allow the user to navigate through the database on the basis of recognition, rather than recall, using them gives the system the ease of use associated with menu-driven systems in general.

- The constraint editors used in query formulation are chosen and parameterised on the basis of the type of the attribute to be constrained and the object. Some of them can handle very general types of attributes, such as strings, numbers or ranges, and selections from a closed set (menu) of values. Such editors will be tailored to the specific attribute being constrained by giving them information on the possible and normal ranges of values. Other editors can employ notations and embody knowledge specific to that particular attribute: for example, an editor for constraining the age of a rock bed may display a geological time-chart, on which the various periods and sub-periods are mouse sensitive.

Because the editors embody or have access to the acceptable and normal ranges of values, they can draw the user's attention to unlikely choices as soon as they are made.

- Viewers were originally conceived as a means of providing a single access mechanism that would work naturally both for textual interformation and for graphics. These could be either prepared in advance for the system ("canned"), or be generated from retrieved data when required. However, viewers could also be created to present video or audio information, and indeed the overall approach can seamlessly integrate multi-media presentations.

- The system makes it easy to modify both the constraints that define the contents of a bag and the viewers which determine what aspects are presented in what manner, and it can display the effects of these changes immediately. It thus encourages users to build up queries incrementally, refining them in the light of the results being obtained.

- Most information displays can be naturally used to launch follow-up queries by either directly indicating the object or set of objects of interest, or by selecting a relationship which will identify them. These objects will then be fetched into a bag or box as appropriate. Because any viewer is always associated with precisely one bag, and each bag retains its own record of the constraints that determined its contents, such follow-up queries are entirely self-contained, and the user is free to launch them from any viewer at any time. This makes unrestricted browsing through the database as easy as exploring a hypertext document.

- The various bags that the user generates are self-contained—that is, they each retain the set of constraints that defined the objects it contains. This makes it possible to return to any bag at any time, either to change the constraints (and thus its contents) or the way they are displayed, or to use it as the basis for launching further queries. But equally, because viewers are used to request information on other objects, bags can be inter-related. A session with the system has a "discourse structure", with some bags being the "children" of others that provided the context in which they were generated. Because the user is explicitly aware of them, a graphical presentation of the relationships between the bags generated during the session provides a session history that effectively cues recall of their contents and purpose.

- Imprecise queries can be formulated easily using fuzzy matching, with mechanisms which are described in Chung and Inder (1992). Any constraint that the user generates can be tagged as requiring either an "exact" match or a "fuzzy" match, which can be one of three degrees of closeness. The domain model can contain definitions of *goodness of fit* functions (see Chung and Inder (1992)) for each attribute, which indicate how well a given value matches a specified target. The system provides some "common sense" measures of similarity for numeric quantities, but the domain model is expected to contain domain-specific goodness of fit functions. Such a function might indicate the (temporal or geological) closeness between, say, Late Jurassic and Early Devonian, or whether someone asking about Rolls Royces might be interested in Jaguars.

- The underlying fuzzy matching facilities are designed to support a form of true *Query by Example*—that is, queries of the form "find others like this or these". Precisely what constitutes "like" in any particular situation is a matter of domain knowledge, which will in general be applied by an "expert system" embedded in the domain model. The current query system provides a simple mechanism for this expert system to generate fuzzy queries, just by deciding which attributes must be matched precisely, and which "fuzzily".

- The metaphor offers several points in the interaction where limited "intelligence" can easily be built into the software, and thus forms an excellent basis for building an intelligent data access assistant. In addition to the (far from trivial) features already mentioned—hiding the database structures by mapping them onto the domain model, supporting fuzzy matching, flagging unusual constraints and queries, ensuring missing data are handled correctly etc.—the system can also employ "intelligence" in a number of other ways. Most obviously, it can select a suitable display for a set of objects, ideally taking account of the specific objects (both their number and characteristics), the quality and availability of data, and the user's expressed or inferred preferences and objectives. It can also optimise the query, both in isolation (by generating efficient SQL) and overall, by cacheing or pre-fetching either attributes or object keys. Similarly, the system can offer summaries and comparisons of sets, and provide explanations of failed queries—that is, constraint sets that result in empty bags.

5 Discussion

The semantics of the interface is based on treating a bag as a container for a (mathematical) set—i.e. a collections of distinct objects. There is no sense in which the bag contains data, and no distinction between the attributes of the object which are "in" the bag and those which are not: which attributes of the object are visible to the user at any time depends on the set of viewers that are associated with the bag, and of course the data that is present in the database. In addition to the set itself, a bag holds the set of constraints that were used to generate it—in a sense, the "intention" of the set. This can be used to re-generate the contents of the bag, should the database or domain model change, and to form the basis of follow-up queries. Finally, at any moment a bag is associated with a collection of viewers, each displaying some attributes of the objects in the bag in some way.

Despite the familiarity of the concepts involved in the metaphor, their combination within a data-manipulation system in a general manner is novel. It is possible to see other systems as embodying special cases of a *Bags and Viewers* approach. For example, the Macintosh file system allows users to select either a graphical (iconic) or text-based "viewer" to display the set of files in a directory. These can be parameterised to some extent—e.g. by specifying the attribute to be used to determine the order in which files should be listed. Moreover, the elements in the viewers are mouse-sensitive, and can be used to initiate examining specific entities in detail, either by selecting a new viewer on a sub-directory or by invoking the application associated with an ordinary file. Similarly, Nowell and Hicks (1993) describes a bibliographic database retrieval system which separates query generation from display, and although it offers only a single "viewer", that viewer can be parameterised to determine which attributes are displayed and how. Finally, the way the system presents the database can be seen as having much in common with the operation of a hypertext system, with viewers offering "pages" of information, many of which contain links to other, related pages. On such a view, every relationship specified in the domain model defines a potential inter-page link, and the pages themselves are constructed, in line with the user's preferences, when they are needed.

Since a set may contain only a single element, the interface could be cleanly specified without making any distinction between bags and boxes. However, there are reasons for making boxes distinct:

- There are only a few ways of presenting data which are equally applicable to both sets and individual objects. Thus, while a map is sensible for any number of locations, a histogram for a single object is absurd, and there is no obvious way of using a form-like display for a set of objects. This could be dealt with by filtering the range of viewers considered by the system, or indeed offered to the user, according to whether the bag contains more than one item. However, if the number of objects in the bag were subsequently changed, many of the viewers associated with it would become silly or even meaningless. This problem is avoided by using boxes, which are known to always contain precisely one object.

- Whether a query is retrieving a set of objects or examining a specific object in more detail is likely to be one of its most salient features. Distinguishing boxes from bags allows this feature to be reflected in the session history, therefore improving its ability to help the user recall the details of the session.

One of the fundamental aspects of any data or domain model is the ontology it imposes upon the world—i.e. what types of objects are deemed to exist. For greatest generality, everything that could possibly be regarded as an object should be. However, this can easily mean that closely related information is distributed across a large number of objects, which can make formulating informative queries a tedious process. Thus, for instance, seeing which company owns a particular oil well might require fetching the object that is linked to the well by means of an "operated by" relationship. A pragmatic solution to this problem can be found by allowing viewers to include attributes of related objects. This is straightforward for one-to-one relationships. It is more complex for relationships where there may be several related objects, which are currently tackled by displaying the attribute's value as a kind of "sub-viewer", from which the related objects can be selected (See Figure 14).

6 Further Work

There are a number of aspects of a system based on *Bags and Viewers* which still need to be thought through, and in many cases the alternatives need to be empirically evaluated.

- Many decisions about low-level details of the interface have been made the "obvious" way, and alternatives need to be assessed. For instance, it is not clear whether a single click on an item in a viewer should initiate the display of more details about it, or whether it should simply select the item, with the user obliged to then initiate the query from a separate button or pull-down menu. The answer to this question is likely to depend on the range of operations that can be done with a selected item. If the only option open to the user is to ask for more information, having separate selection and query initiation actions is of dubious merit.

- To date, only a small number of general-purpose viewers and constraint editors have been implemented:

  - configurable forms and one-line-per-object summary displays/menu viewers, and
  - constraint editors for numbers (and ranges), strings, set of options and value hierarchies.

  This range needs to be increased by implementing viewers for, say, maps, charts and histograms, cross-plots, photographs etc. In addition, we think that introducing summary attributes for bags, such as the maximum, minimum and average value for an attribute, will increase the range of queries that can naturally be handled. Finally, we have identified the need for a mechanism for working through a (potentially large) set of objects, considering each in turn in some way (e.g. by putting it into a specific box).

- Constraint editors currently work on a single attribute in isolation, without considering the constraints that have been imposed on others. This is a limitation, since the values of different attributes interact strongly: the Gulf of Mexico ceases to be a possible location for a well which has been constrained to be producing from the Forties field in the North Sea, and vice versa. If constraint editors can be made sensitive to such interactions, feedback about queries that will fail can be given to the user much more quickly. It is also still not clear how some types of complex constraints should be expressed and handled by the system—e.g. to find all the installations being operated by more than two companies, or by companies of two or more nationalities.

- The graphical presentation of the session, which shows how bags and boxes were generated from others, offers a powerful mechanism for helping users to keep track of their interaction with the system. However, we believe that there is considerable scope for improvement by, for instance, using the shape and size of the icons in the display to convey information about, for example, the type and number of objects in the bag. In particular, we believe that the usefulness of the device as a trigger for recall will be greatly improved if the user is allowed to arrange, mark and even label the various nodes in a way that is meaningful to them.

- There are still several areas where the system's "common sense" about data is lacking, and where more knowledge could greatly facilitate building domain and database models, and thus the system's ability to present data effectively. For example, the system has no notion of "time", and thus cannot handle databases that contain many values for a single-valued attribute that apply to different times. The system can be used to access such databases, but only by distorting the domain model. Giving it an understanding of temporal variation should both avoid this and allow a more intelligent handling of such data, such as assuming the most recent value is required unless another time is specified, or presenting the way a value of interest varies over time.

- As the metaphor is developed, the possibility of allowing users to alter the contents of a bag—to add or remove objects—must be given further consideration. Objects could be added by allowing the user to merge two bags or, equivalently, to add the result of a query to an existing bag. In this case, the bag's contents would be described by a disjunction of the constraint sets describing the contributing bags. But there is also considerable appeal to allowing users to simply nominate objects to be added to (or removed from) a bag, perhaps by some kind of "direct manipulation"—"put this, this and this to one side for further investigation", "ignore these two" etc. While this may be very natural, it has the drawback that the system would have no basis for associating a constraint set with the resulting collection of objects, and thus it would be in some sense less than a real bag—a "baguette". It is not clear how baguettes can be handled, and the matter of whether the additional interface complexity of doing so is justified by their usefulness needs to be investigated.

Finally, one of the objectives of the work that led to the formulation of the *Bags and Viewers* approach was to support combining multiple sources of data. This can be supported very naturally within the interface paradigm, since users do not explicitly initiate queries and the underlying database structures are hidden. The encapsulation of database-specific information within the database description makes it straightforward to enable the system to query a new database in isolation. However, there are many problems associated with attempting to combine information from different databases being accessed at the same time. At its simplest, answering a query on the basis of several databases involves querying one database to extract attributes which are missing from another. Doing this effectively obviously depends on cross-referencing between the databases—identifying the same object in two databases which have not been designed or maintained to ensure agreement. However, making the situation only slightly more general involves many more complex issues.

In many cases, there will be overlap between databases available to the system—indeed, this *must* be the case for at least some attributes, to allow entries to be cross-referenced. As a result there will, in general, be more than one source of any piece of information. It is straightforward to use multiple sources of an attribute value within a single database, and to use information about the database to indicate how to obtain the best value. For example, a database may define a field to represent a particular attribute, but may only actually have data present for some objects. In such situations, the system can be given not only the details of how to locate the explicit data, but also a formula for deriving a value from other attributes. The system can then is able to make sensible use of both in answering a question.

Such mechanisms could probably be extended to deal with separate database systems, with the required data being assembled from the results of multiple queries. More awkwardly, if the set of objects of interest may be determined by attributes drawn from multiple databases, then either "join" operations must be carried out outside the databases, or appropriate object-identifying information must be translated and transported between them. In both cases, though, the interface must become involved with determining, rather than just presenting, query results. This will require efficient but general-purpose means of sifting and combining potentially large volumes of partial results to be devised. More subtly, not all databases are created equal: the data they contain can differ not only in range, but also in quality and completeness. Not only must these variations be taken into account when formulating a query, they also mean that the user must be able to see, and indeed constrain, the provenance of the information being displayed. Making this kind of information available to users without overwhelming the data itself will be a significant task.

7 Conclusion

Bags and Viewers provides a framework for structuring interfaces to systems which involve accessing collections of data. Such systems should be very flexible—they are capable of generating arbitrary queries—yet should also be very simple to use, in that they involve only a few operations and concepts, and these are reasonably familiar. A prototype system has been developed to illustrate these ideas, and has been ported to three different databases in quite distinct domains. Work on the system is continuing, and we are looking for other domains to which it can be ported and evaluated.

References

Nowell, L, and Hix, D. (1993) Visualizing Search Results: User Interface Development for the Project Envision Database of Computer Science Literature. In Salvendy, G. and Smith, M. (eds) *Human-Computer Interaction: Software and Hardware Interfaces*, Elsevier, Amsterdam.

Chung, P. and Inder, R. (1992) Handling Uncertainty in Petroleum Exploration Data. *Revue de L'Institut Francais de Petrole*, Vol 47, No. 3, May/June 1992. Also available as AIAI-TR-104, AI Applications Institute, University of Edinburgh.