Graphical interfaces for binary relationship databases

F N Teskey, N Dixon and S C Holden

The increasing desire for information systems usable by nonprogrammers leads to the need for powerful database interfaces acting at a level well above that of traditional database systems. Such users may wish to interrogate the database either by use of formal queries employing minimal syntax, or by browsing through the database in a directed manner. Reducing hardware costs, and the widening availability of high resolution video displays, suggests that graphics-based interfaces, with their high information content, can provide a solution to these requirements.

This paper describes the design of one such graphical interface, intended to act as a front-end to a binary relationship model of information. Classes and relationships within a particular model are represented by nodes and arcs which are manipulated by a visual interface system. The interface supports multiple independent user views of the database and supports both the formation of formal queries and the ability to browse through class entities and relationship instances in a nonprocedural manner. A further function of the interface is to allow a simple and flexible means of inserting and deleting classes and relationships, class entities and relationship instances.

One of the applications of this interface is in a research project aimed at integrating the storage and manipulation of diverse types of information. The problems of integration are compounded by the different representations required for differing uses of the same information. Hopefully a unification of the method of storage of textual, graphical, and other types of information will ease the problems of communication among the various functional modules of the system. This should allow modules to coordinate their display activities and pass relevant parameters of their subject information to other modules which cannot 'understand' the different representations.

Keywords: databases, graphical interfaces, graphic representation, binary relationship databases

INTRODUCTION

In recent years computer hardware technology has evolved rapidly, leading to a proliferation of computing facilities. In the near future these facilities will become ever more accessible to a widening spectrum of potential users with greatly varying applications. One application certain to be of common relevance is the storage and organization of information. This information will, of necessity, be stored in a database. Such information systems will be used as powerful tools by those having expertise in fields other than computer-oriented subjects, and having neither the time nor motivation to become computer programmers.

It is commonly observed that the evolution of computer science tends to be one of small steps in discrete subject areas, with only occasional quantum leaps in the level of technology. When these incremental advances are combined in a single system, however, the results can appear revolutionary by allowing and even encouraging users to think in new ways about their problems and how they might be solved. The design of database interfaces is an example of such evolution.

In order that a nonprogrammer may organize, manipulate and inspect his own database, while remaining unaware of its internal structure, there is a need for interfaces that lie between database and end-user that are simple and receptive, yet powerful and flexible enough to satisfy the majority of needs. This paper describes the design of an interface that attempts to satisfy these requirements.

Department of Computer Science, University of Manchester, UK
Received 14 March 1984, revised 2 April 1984

A very important factor influencing the design of a database interface is the model of data used by the host database system (at least at the conceptual level). If the user is to organize his own data, then this database model should appear as natural as possible. Of the available choices of model it is felt that the relational approach comes closest to the human perception of reality, and in particular, a binary relational model (BRM) provides the best model on which to build a database interface. Research into the development of such models is progressing at several establishments[1-5]. This research indicates that databases based on such models may be implemented efficiently and at a reasonable cost, through the use of novel hardware.

The use of graphical displays to illustrate the content and structure of the database is equally important. There is nothing magical about computer graphics, which is literally the production of visible images from information held in a computer memory. The interesting feature of graphics software is that, within the limits of computational power and computability, the represented images may bear any desired relationship to the visible appearance of the structure of the modelled object. This gives the systems designer considerable freedom in the use of metaphor, since he may choose any criteria whatsoever for image production.

As an example, we shall use Date's Suppliers-and-Parts database[6], as summarized in Table 1.

DATABASE INTERFACES

Conventionally, access to a database has required the user to have a working knowledge of either a text-based linear query language (e.g., SQL, QUEL, etc) or, where the range of possible queries is limited, a language specially tailored to the user's application (e.g., the filling in of forms displayed on a VDU by means of menu selection). For a nonspecialized user to become familiar with a query language necessitates an initial period of training in both the language and in the particular user's view of the database. A specially tailored language presents problems if the database is to be altered, as the interface will also have to be changed to account for the presence of new attributes and classes.

Research in recent years has investigated the problem of providing an interface for naive users that is easily learnt and that provides these users with the ability to formulate complex queries in a flexible manner, and to manipulate the database with relative ease[7-12]. Such interfaces may also include the ability to perform data-definition tasks in a better way than the conventional use of a data definition language (DDL), to allow an end-user to create/modify his own database structure.

Graphical interfaces

This research has led to the production of a number of database interfaces for use by nonprogrammers. A substantial number of these, and also the most successful, have been those that have exploited the use of computer graphics and the 2-dimensional capabilities of a VDU screen. These graphical interfaces have two different approaches to solving the problem: query language interfaces and spatial data management systems.

Table 1. Suppliers-parts database

Suppliers

S no.	Sname	Status	City
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

Parts

P no.	Pname	Colour	Weight	City
P1	Nut	Red	12	London
P2	Bolt	Green	17	Paris
P3	Screw	Blue	17	Rome
P4	Screw	Red	14	London
P5	Cam	Blue	12	Paris
P6	Cog	Red	19	London

Shipments

S no.	P no.	Quantity
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

Query language interface

Certain interfaces, such as Query-by-Example (QBE)[11], FORAL LP[9], CUPID[7, pp 75-76], GOING[10], allow the user to form the equivalent of text-based linear language queries through the manipulation of graphical symbols. QBE uses a single graphical symbol, the skeleton table, which is filled in by the user to give an example of the response expected. CUPID forms queries from a collection of rectangles, hexagons and circles. In GOING queries are expressed in terms of ellipses representing domains, and arcs representing connections between domains and connections to conditions. Many examples comparing QBE, CUPID and GOING can be found in Udagawa and Ohsuga[10].

In FORAL LP, the graphical display consists of a diagrammatic representation of the database structure. Queries are formed by the user selecting nodes and arcs on this network, representing classes and relationships respectively, and also by selection from a menu of keywords and commands.

There is still an element of programming involved in these interfaces, since a strict syntax must be adhered to, although less is required than in a conventional query language.

Spatial data management systems

A new generation of information systems has emerged in recent years that has dispensed with the text-based query entirely, and turned to using a very large amount of visual content to allow naive users to navigate through and interrogate certain types of database. The user is able to 'browse' through the database searching for items of interest. Among such systems are the very ambitious Dataland[7, pp 77–83], and the more down-to-earth SDMS[12] and VIEW[7, pp 83–87]. Such systems have virtually no syntactic content, but are of little use for the tasks that are the forte of conventional systems, that is extracting logical and quantitative data and collating it into some tabular form.

Natural languages

An alternative approach to graphical interfaces would be the use of natural languages for access to a database. Early attempts at this approach[8] led to only limited query processing owing to difficulties of syntactic and semantic analysis and a lack of portability to new applications due to the use of domain-dependent linguistic analysis. More recent research in this field[13] uses a general framework[14] to separate the domain-dependent from the domain-independent parts of the linguistic analysis to improve analysis.

Some criteria for the design of the interface

Given the problems of natural language front-ends, the promise shown by earlier graphical systems, and the increasing availability of high quality graphics implementations, it was decided that graphical methods would provide the best avenue of research into a powerful application-independent database interface for naive users.

The requirement for such an interface leads to the following criteria in its design:

- The syntax of queries should be minimal. In particular the amount of typing required should be minimal and the user should be guided in the use of the interface as much as possible. To this end, query formation should proceed by the use of some type of pointing device (and keyboard) to allow the user to interact with the graphical diagrams and menus.
- The classes and relationships within the database that are of interest to any particular user should be readily apparent. Therefore the user should be presented with his own independent view of the database, thus removing any confusion due to the clutter of information required by other users and applications.
- The user should be able to formulate complex queries by a process of modification of much simpler queries. The user may wish to perform a certain amount of browsing through the database at any stage in the formulation of a query. Such browsing may be combined with the selection of class entities and relationship instances together with conditional operators to qualify the query being made.

GRAPHICAL DATA

In any information system, visible images are useful for two reasons. First, they can be used to show the relationships between the conceptual entities whose details are stored in the database. These might be called 'structural pictograms'; for example, Figure 1 displays the relationships between entities in a semantic network. Second, visible images can be used to communicate nonstructural information to the user. The most common visible images used for this purpose are textual images; the major reasons for this have been more to do with the economics of terminal production than with the desirability of text as a communications medium. We refer to these two types as iconic and analogic respectively.

This distinction between the structural and the nonstructural is important for users without extensive experience of the particular data model they are working with. Structural information such as 'rooms have temperatures' allows the user to discover the nonstructural fact that 'the temperature of the bedroom is 68° Fahrenheit'.

Our attitude is not that graphical interfaces are in principle very different from purely textual ones, it is that the nature of the interaction between the human user and the computer system need no longer be constrained by the limitations of having to present information as text. Presumably this argument may apply to other sense-extensions to the range of output devices; sound effects are currently a major selling point for hobby computers. This is also illustrated by the arcade games that have become so popular in the last five years. Here the user interacts with a very complex and fast-changing data model. That such high-bandwidth interaction can seem so natural is presumably a tribute either to the genius of the games designers, who need to think about these matters[15], or to the adaptability of the human being.

Iconic data

An icon is a prototypical image which is meant to be suggestive of some property of the data object that it is used to represent. Such representations are used, for example, to denote a set of processing facilities in the Apple Lisa system; the user selects a particular system function by pointing at the appropriate icon. This approach has been taken much further in an experimental system by Cattel[16], which shows that icons can be used both to represent relationships between individual items of data (concrete structure), and also the relations amongst entity classes (abstract structure).

In the data model discussed in later sections, icons could be used to represent the IS—A relationships that are used to express semantic content. Thus, Figure 2 might be used to compress the representation of the relationships between the individual data shown in Figure 1.

Text is another good example of the use of iconic representation. There is a basic icon for each member of the character set, if emphases such as italics, emboldening and character size are taken into account, then even the representation of a single character depends on information that may turn out to be contextual.

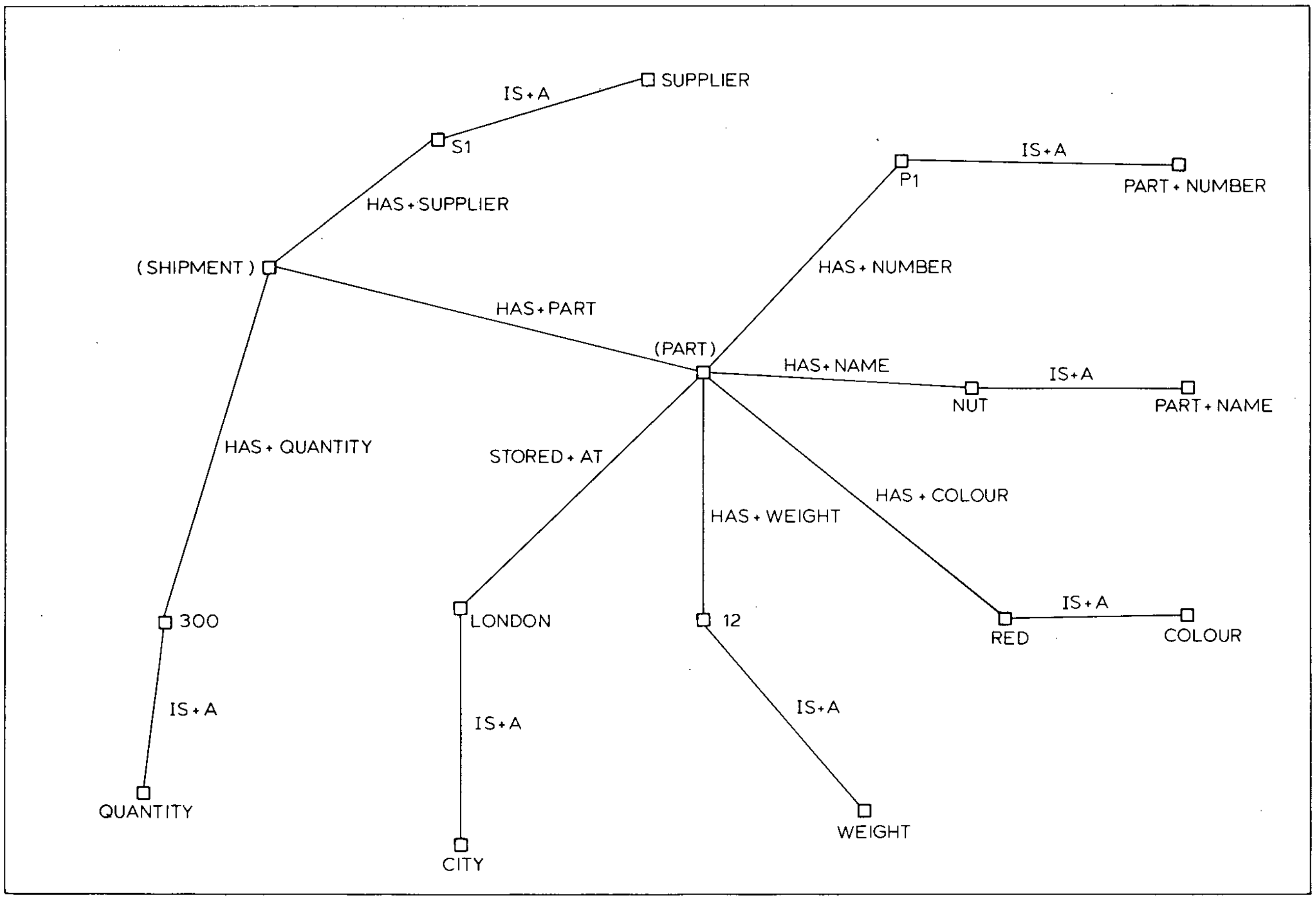

Figure 1. Binary relationship information for part P1

When using bit-mapped displays it would seem natural to store the icons as the desired raster pattern for direct transmission to the display hardware.

Analogic representation

An analogic representation is used for data objects that have no iconic representation. In a general analogic representation, the number of possible instances of an entity type is so large that it is impossible to have individual icons for each possible instance. In this case, an analogue is created either by juxtaposing a number of iconic images, or by algorithmically modifying an archetypical 'base icon'. Thus each word does not have its own iconic representation in textual display; rather, an algorithmic process is used to send a sequence of intermingled control and data characters to an alphanumeric display terminal. Similarly, floating point numbers are displayed by passing them to a routine which computes a string of characters which the human user will recognize as representing a number.

Note that the base icon does not need to be available to the system for representing any particular object, although it would be a useful icon to denote all objects of the same semantic class. A pair of orthogonal axes would, for example, be highly suggestive of some graphical mapping of numerical data.

Towards unified representations

As the example of emphasized text shows, even such a simple data item as a single character may, in the context of a particular application, have many different visible representations. If all possible representations of a data object are iconographic we face the question of which basic operations we wish to be able to perform on all icons. Rotation, translation and scaling are the three standard requirements. There are difficult but soluble problems associated with performing these operations on digitized images. Until these are solved it may be necessary to store the icons as groups of lines which could more easily be scaled, rotated and so on.

Some objects may be represented part of the time iconographically and part of the time analogically, depending on the view of the object the user wishes to adopt. Thus a particular data object representing a map might sometimes be represented as a document icon, sometimes as a map icon, and at still other times as an analogic display of digitized cartographic information. In order to be able to present the appropriate view at any time there needs to be some interaction between the data objects to be viewed and the user's viewpoint. The change from map icon to detailed map display might, for example, be the result of a scaling operation, which certainly represents a change in the point of view.

Some complex objects would have exclusively analogic representation. Such objects might be combinations of lines, for example, in a datastructure which detailed some complex geometrical structure. If some universal icon were adopted to represent all data objects at the least discriminating viewpoint, it would naturally apply.

Since the graphical data would represent many different conceptual objects, it seems natural to expect that different types of operation, both computational and

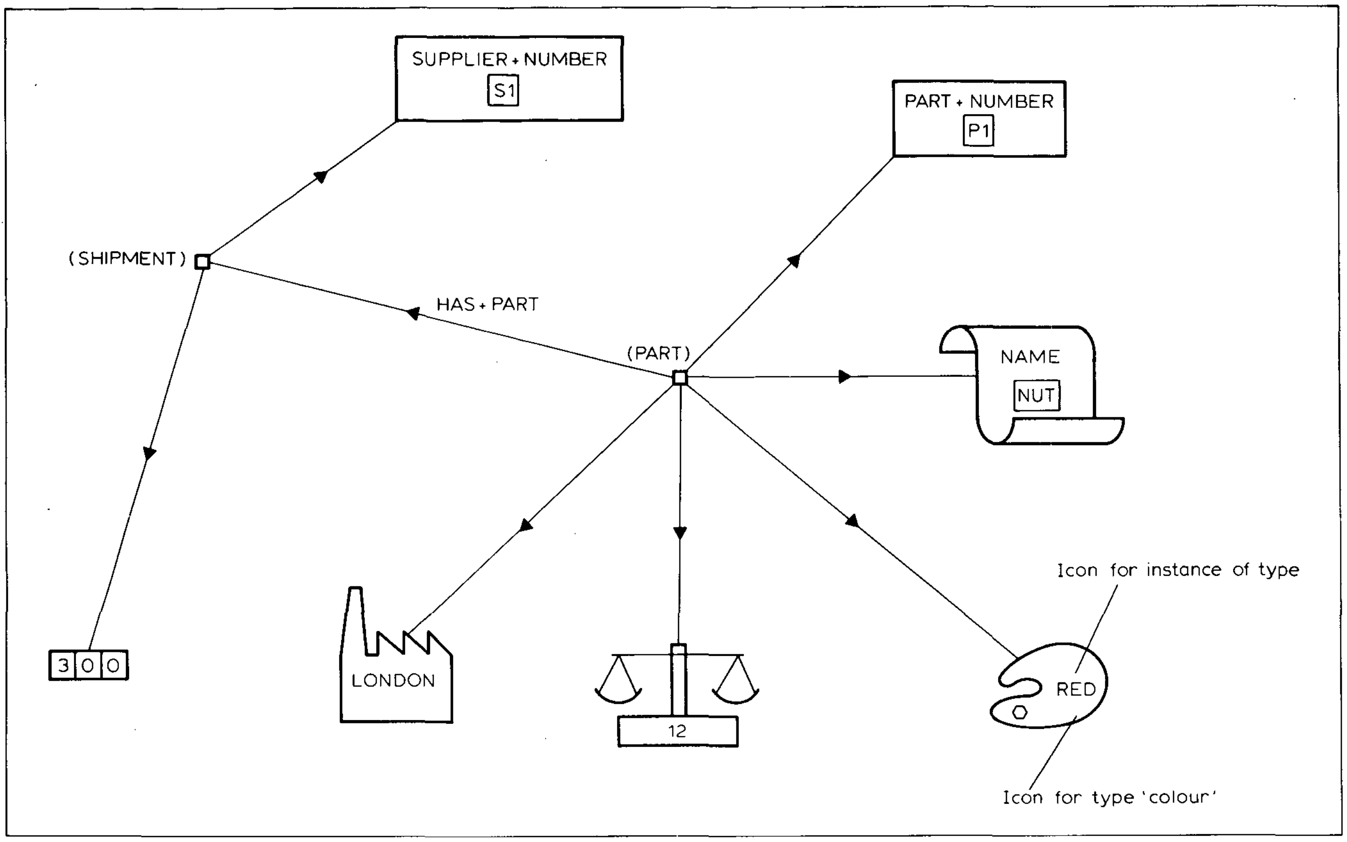

Figure 2. Iconic representation of binary relationships

graphical, could be performed on them. An object-oriented paradigm such as is used in the smalltalk-80[17] system would allow common specification of algorithms common to classes of object, with 'behavioural subclasses' defined by implementing the same operation in different ways. This would need a system that was capable of binding procedure names at run time, which may not turn out to be straightforward. The property-inheritance characteristics of object-oriented systems are a natural way to specify datastructures in a manner that allows easy refinement of behaviour.

THE BINARY RELATIONAL MODEL

The BRM is an information model consisting of two basic components, entities and binary relationships between entities. Entities may have some real world representation or value, or may not, in which case they are merely internal identifiers within the database. The entities within the database may be grouped together into different, possibly overlapping classes.

Several implementations of BRMs exist. The interface system described in this paper is based on a BRM simulation developed by the Intelligent File Store (IFS) group in the authors' department and uses the interface defined in Azmoodeh[1]. This is a subset of the semantic binary relational model (BRM) developed by the IFS group[18]. In describing the general properties of the BRM, we shall use the conventions of that implementation[2, 3, 19].

The membership of a class may be defined by an instance of a special unary relationship, 'IS_A', which has its own predefined semantics. For example, Figure 1 describes the information that would be required for storing facts about the part P1 in Date's Suppliers-and-Parts database. Nodes represent entities and arcs represent binary relationship instances between entities. Certain entities have a value (e.g., 'NUT', 'PART_NAME', etc.) while others do not (e.g., the real world representation of the shipment is its values for supplier, part and quantity; similarly part is represented by its values of name, colour, weight and location). Classes may consist of entities with a value (lexical values) or without (nonlexical values) but not a mixture of both.

The entities and relationships within the database are categorized broadly into information and meta-information. Meta-information consists of information defining the structure of classes and relationships within the database, as well as describing their properties. Figure 3 describes part of the meta-information used by the IFS implementation. A class is represented by an entity that is a member of the class 'CLASS', and similarly for relationships. Both 'CLASS' and 'RELATIONSHIP' are nonlexical classes, but the names of classes and relationships are defined by connections to members of the set 'NAME'. 'CLASS' and 'RELATIONSHIP' and the relationships between them represent the structure of the database, while 'CLASS_STATUS' and 'COMPLEXITY' represent the properties of the information model.

VISUAL INTERFACE SYSTEM

Conventionally, input of the BRM schema is by use of a standard data definition language. Since this schema may be represented diagrammatically, it is possible to input this schema directly in graphical mode, using suitable symbols to represent classes and relationships. The information structure stored in a BRM is represented in

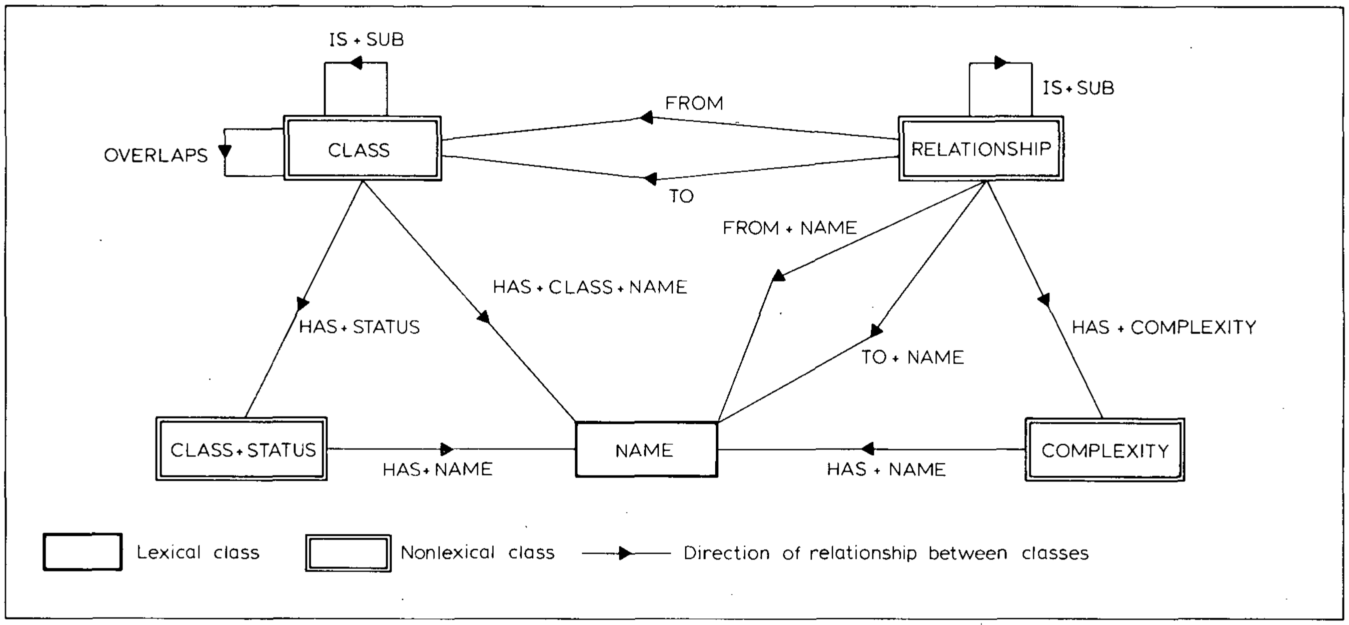

Figure 3. BSN for metadata of IFS-BRM

the graphical interface by nodes and arcs, an approach used by Senko in his FORAL LP interface[16] as an interface to a data manipulation language (DML). It is our intention to use VIS as a method not only for retrieval, but also for input, update and deletion. Nodes (drawn as boxes) represent classes while arcs represent relationships from one class to another, the direction being indicated by an arrow. Lexical classes are represented by a single-walled box, while nonlexical classes are represented by a double-walled box. Where there is more than one relationship between any pair of classes, a unique arc must exist for each relationship that is of interest to the particular user. Figure 4 shows a BRM representing the suppliers-and-parts database.

A problem of the FORAL LP implementation was the difficulty of representing large database structures on a single screen. This problem is overcome in VIS by the use of three devices:

- The BRM is inscribed in a world coordinate system over which the user may roam by shifting his viewport, and zooming in or out to inspect the required part of the BRM.
- All the classes and relationships within an application need not be displayed at once. Multiple representations are allowed by use of different nodes collected together to form networks. For example, Figure 5 is as valid a representation of the suppliers-and-parts database as Figure 4. This ability to break complex models down into a collection of simpler models has the advantage of clarifying the contents of the database by breaking it down into more easily assimilated units.
- Not all classes and relationships within a particular application need be displayed.

VIS Metadata

As we have seen, the structure and properties of information within the BRM may be defined by its metadata, and this may be treated in the same fashion as any other data. In a similar fashion the graphical interface also stores its own metadata, concerning user information and graphical display data, within the BRM. Figure 6 is a BRM representation of this metadata.

The structure of the metadata concerns users and networks. Each user has a user_name, a password, and an access_status, as well as a collection of networks. Each network has a name, and consists of nodes and arcs. Each node has an X_position and a Y_position (world coordinates), and represents a class. Each arc represents a relationship, and lies from one node to another. In addition, an arc may pass through more than one vertex which also has coordinates, these being ordered by their vertex_list_position. This allows greater freedom in drawing arcs. For example, in Figure 6 the relation HAS_TO_NODE, from node to arc, is represented by an arc which has a single vertex.

Independent user views

Since a network need not contain all the classes and relationships within a given database, the VIS can be tailored to a particular user's requirements. The user does not see information he is not interested in, nor information that the database administrator does not wish him to see. A user need not have only one network for each application, but may have many, depending on the type of query he is interested in. Thus, the interface supports multiple independent views of the database.

DATABASE MANIPULATION

The graphical interface allows queries to be formed and data-definition tasks to be accomplished, through the use of a pointing device (e.g., bit pad and puck, mouse, lightpen, etc.) and commands selected from multiple menus. The menus list commands that may be performed within each state of the interface. The pointing device is used to select from menus, and also to select classes and relationships. A class may be selected by pointing at the corresponding node of the BRM, while a relationship is selected by pointing at a vertex of the arc representing that relationship, or, if no such vertex exists, by selection of the FROM and TO classes of the relationship.

BRM creation

The structure of the database can be defined or modified by the use of the editor module to create a network containing the new classes and relationships. Such networks may also be created to present independent user views of the database. Networks are created by the placing of nodes representing classes and by the drawing of arcs representing the relationships. Having placed each node or arc, the user is prompted for any additional information concerning the class or relationship (e.g., in the IFS-BRM the user must supply name, type and status for classes; from-name, to-name and complexity for relationships). Classes and relationships not already present in the database will be created and information concerning classes and relationships will be updated. The ability of the user to perform these data-definition tasks is dependent upon the user having a sufficient access-status. The facility also exists for the user to remove nodes and arcs from the network, but he is not allowed to delete classes and relationships unless they are empty. Additional commands also exist to move nodes and vertices, in order to create clear and tidy networks.

Data retrieval

Data-retrieval operations take place through the use of a sophisticated browsing module. The instrument of retrieval is the browsing window, a window that acts as a viewport onto the domain of a lexical class, displaying entities within a particular class that satisfy the query being made. The contents of these windows is, by default, the entire domain, but may be restricted to particular sets of items. The outline of a browsing window is shown in Figure 7.

Simple browsing

The most primitive retrieval operation is the ability to open a window on a particular lexical class and inspect its contents. For instance, Figure 8 shows windows opened on classes NAME and CITY. Notice that there is no distinction on whether these entities are related to PART or SUPPLIER. Lexical entities that are not in the window may be viewed by pointing at the scroll bar, and moving to

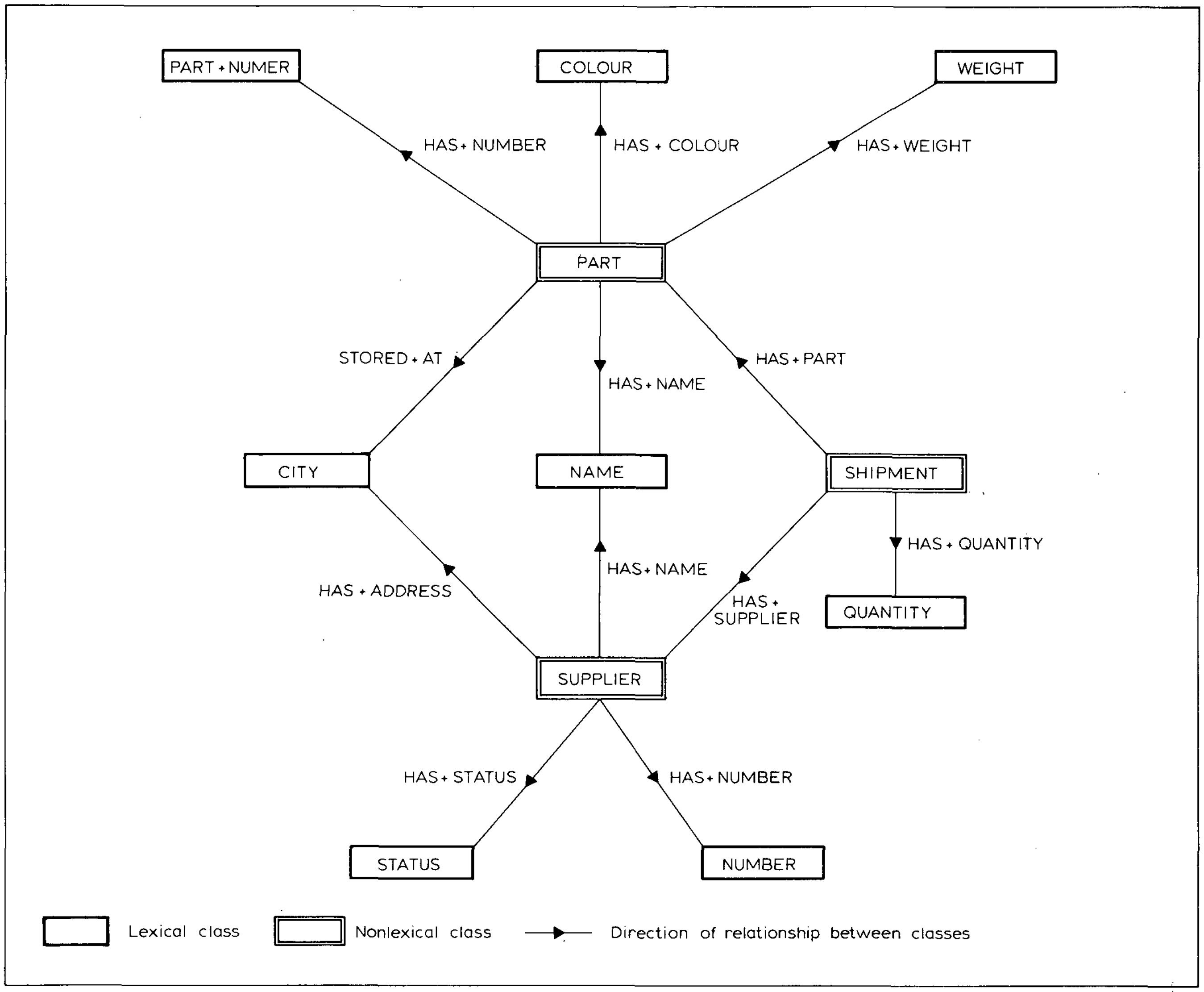

Figure 4. BSN for suppliers–parts–shipments database

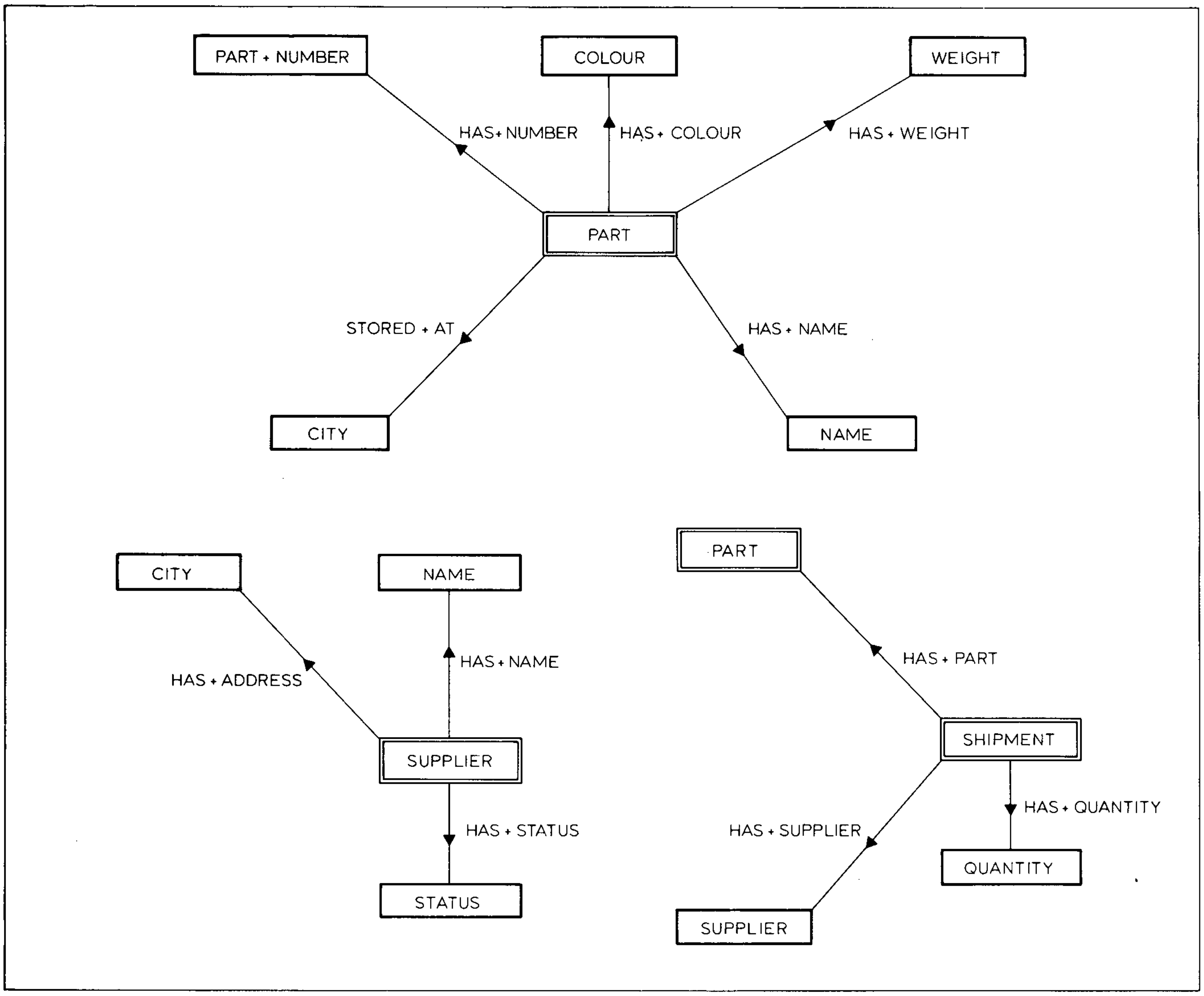

Figure 5. Suppliers–parts–shipments database represented by three BSNs

left or right to scroll up or down respectively. To aid clarity, when a window is first opened on a class, it can then be dragged by the pointing device into its required location.

Relational browsing

Queries are made by the selection of relationships which are then linked together to form the query. This query formation is performed incrementally by selecting one relationship at a time. Selecting a relationship causes a window to be opened for each lexical class connected to that relationship. If a window already exists for a particular class, then the existing window is used. The exception to this is where a relationship is selected which would form a closed loop with the relationships already existant, e.g., selection of (PART) HAS NAME (NAME) and (SUPPLIER) HAS NAME (NAME) would cause two windows to be opened for NAME, one for each relationship. As each new relationship is selected, the window contents are set to the current range of the selected relationship. Since a query may involve the use of multiple windows on a particular class, the query made so far is displayed by having the relevant windows and arcs displayed in a special query window, with nonlexical classes being represented by their class names (*see* Figure 9). The query window may be moved over the screen so as not to hide items of interest. The selected relationships are highlighted by thickening of the relevant arcs.

Conditions may be placed on the contents of a particular window by pointing at the conditional area (indicated by a 'C') at the foot of the window, followed by entering a conditional expression, either directly from the keyboard, or by selection from a menu of conditional operators. The contents of the window are then restricted to those values satisfying the given condition. In addition, the contents of the window may be defined by selecting particular elements as in simple browsing.

As an example, consider the following query:

'How many screws have been shipped by Smith?'.

This query could be posed by the following series of actions. Figure 9 shows the step-by-step development of the query in the query window.

- Select (SUPPLIER) HAS NAME (NAME), Figure 9(a).
- Select (PART) HAS NAME (NAME), Figure 9(b).
- Select (SHIPMENT) HAS SUPPLIER (SUPPLIER), Figure 9(c).
- Select (SHIPMENT) HAS PART (PART), Figure 9(d).

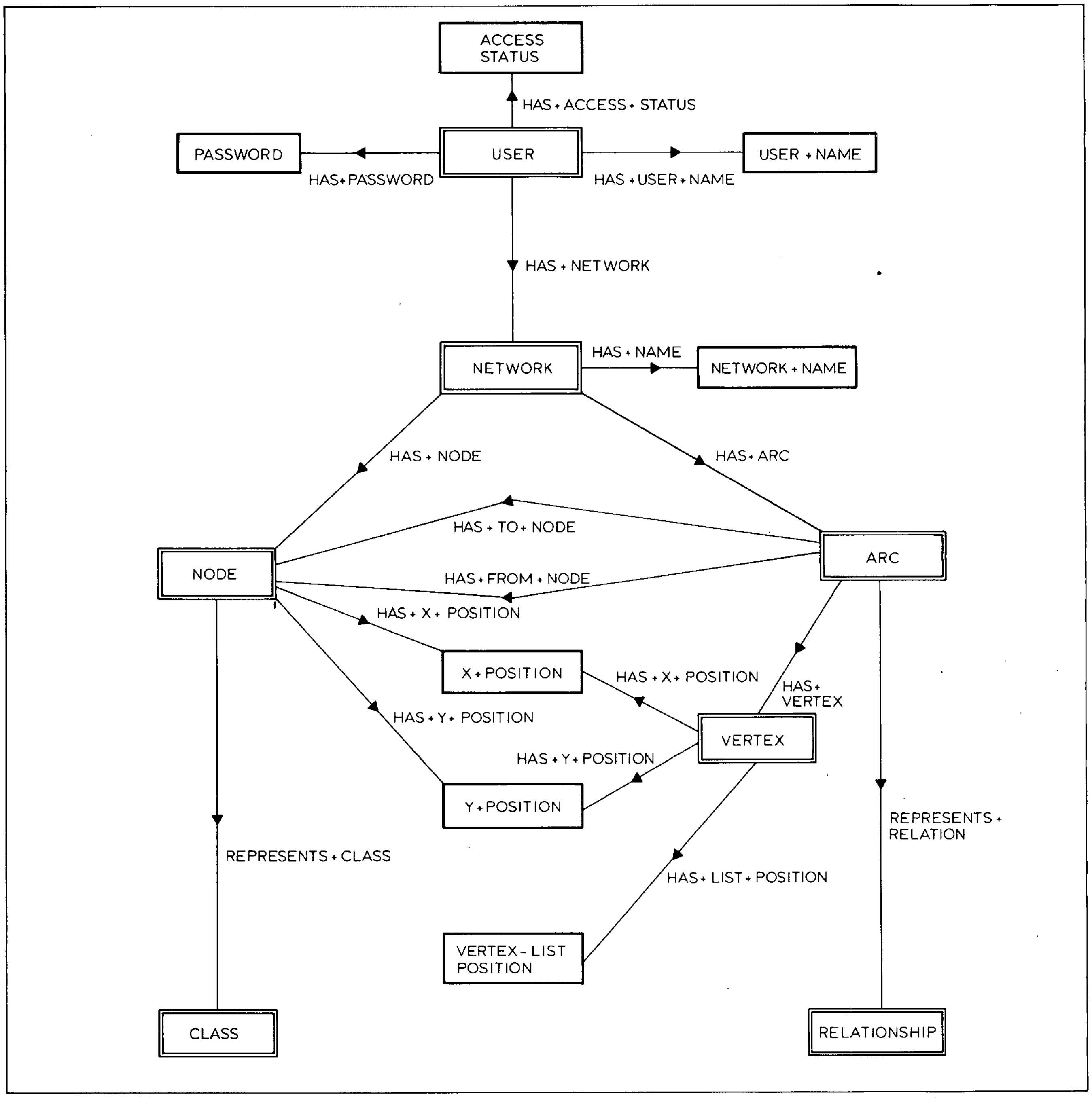

Figure 6. BSN for graphical interface metadata

- Select (SHIPMENT) HAS QUANTITY (QUANTITY), Figure 9(e).

By browsing through the relationship instances retrieved so far we could see that Smith has supplied two shipments of screws, one of 400 and another of 200. However, for a more complicated database, the output could be clarified by supplying the conditions (PART) HAS NAME 'SCREW' and (SUPPLIER) HAS NAME 'SMITH'. The first condition may be supplied by pointing at the condition box of the part name window and typing 'SCREW', or by selecting SCREW from this particular window (note that the conditional operator '=' is assumed unless some other is supplied). The supplier name can be specified in a similar manner. This results in windows appearing as in Figure 9(f). Finally, since the total number of screws may be required, this could be done by selecting the function TOTAL from a menu of conditional operators and functions for the QUANTITY window Figure 9(g).

Note that the order of selecting relationships or placing of conditions was unimportant. For the simple database, it was not necessary to complete the full query to get an answer, since browsing allowed the user to see the result at an early stage. The query could be formed with minimal

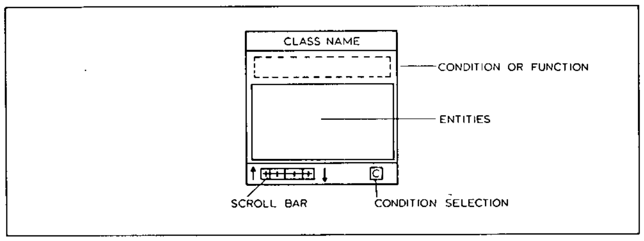

Figure 7. Browsing window

use of the keyboard, since lexical values could be supplied by selection from a browsing window. It remains to be seen whether all queries can be formulated in such a simple manner.

Data input/update/deletion

The graphical interface does not seem appropriate for bulk data update. It is felt that this may be better accomplished by the use of offline methods. However, it may be that the browsing windows can offer a possible method for inserting small amounts of data, and for updating and deleting data in a similar fashion to QBE, which uses the same syntax for update as for retrieval. Further research is being undertaken to determine the most appropriate methods for updating both large and small volumes of data.

CONCLUSIONS

We believe that in any information system the user should be able to comprehend fully the underlying knowledge representation. The BRM provides a representation that is conceptually simple and yet very powerful. We have used the BRM as the basis for a graphical database system that offers the nonprogrammer the opportunity to perform a wide range of information-retrieval functions. The value of a relational database from the point of view of the graphics programmer is the potential ability to provide large dynamic object stores. In return, graphics integrated into a uniform representational paradigm offer the database user a multiplicity of potential viewpoints, and the ability to compose images from multiple diverse views of sets of related objects. The work is still at an early stage; the editor and simple browsing have been implemented on an ICL PERQ under both POS and PNX, and multiple views and relational browsing are now being implemented. Further research is necessary before we can demonstrate a prototype system.

ACKNOWLEDGEMENTS

The work described in this paper was supported by a research grant (GR/C/51295) from SERC and a research studentship from GEC. We are very grateful to the IFS group in the Department of Computer Science, University of Manchester for use of their BRM simulation. This paper was originally presented at the BCS Conference on the Storage and Retrieval of Integrated Graphics and Text held in London on 22–23 November 1983.

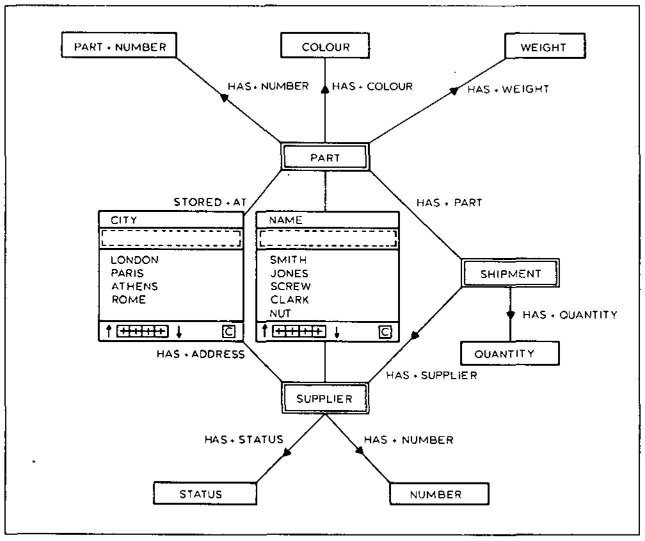

Figure 8. Windows opened on CITY and NAME

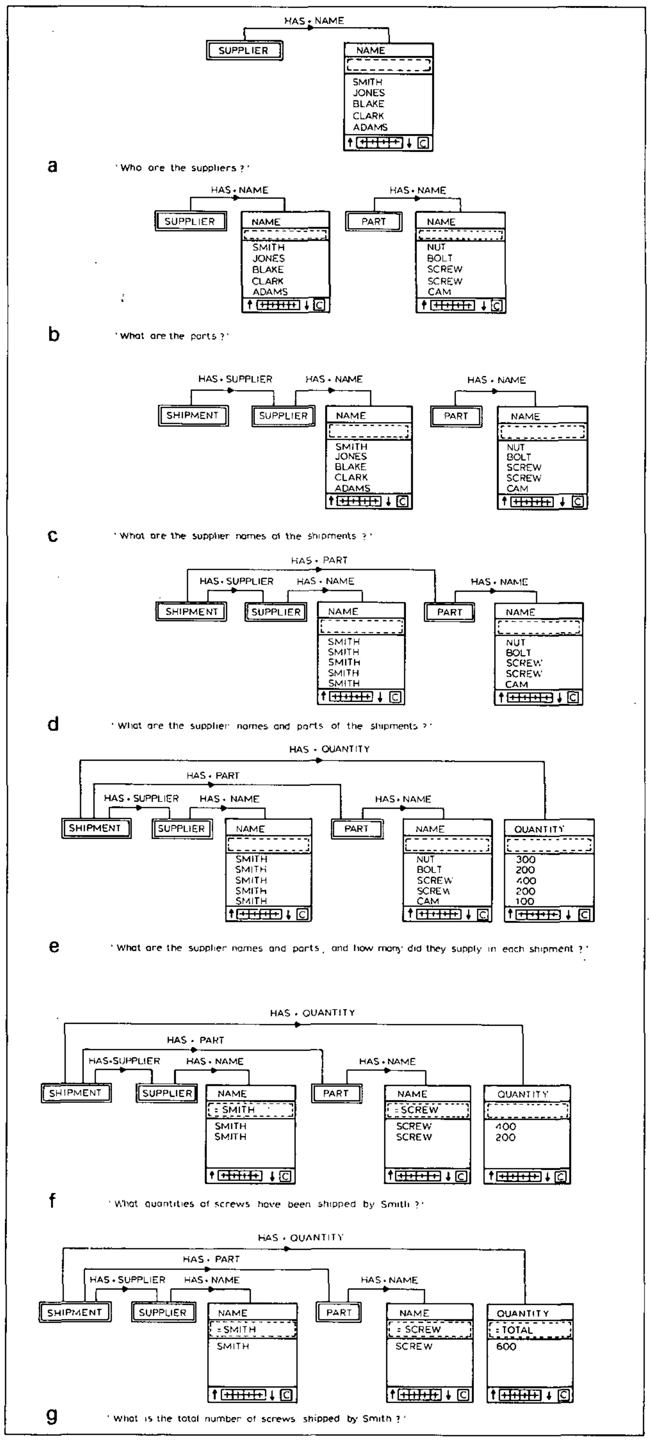
a
HAS • NAME
SUPPLIER
NAME
SMITH
JONES
BLAKE
CLARK
ADAMS
'Who are the suppliers ?'
b
HAS • NAME
PART
NUT
BOLT
SCREW
SCREW
CAM
'What are the parts ?'
c
HAS • SUPPLIER
SHIPMENT
'What are the supplier names of the shipments ?'
d
HAS • PART
'What are the supplier names and parts of the shipments ?'
e
HAS • QUANTITY
QUANTITY
300
200
400
200
100
'What are the supplier names and parts, and how many did they supply in each shipment ?'
f
= SMITH
= SCREW
400
200
'What quantities of screws have been shipped by Smith ?'
g
= TOTAL
600
'What is the total number of screws shipped by Smith ?'

Figure 9. Stages in relational browsing

REFERENCES

1 **Azmoodeh, M** 'A BRM machine and its interface procedures' Internal Report IFS/3/83, Department of Computer Science, University of Manchester (Feb 1983)

2 **Lavington, S H and Azmoodeh, M** 'IFS — a proposal for a data base machine' *Proc. 2nd Brit. Nat. Conf. Data Bases* British Computer Society (July 1982)

3 **Lavington, S H** 'Intelligent file store: project overview No 4' Internal Report IFS/5/82, Department of Computer Science, University of Manchester (Oct 1983)

4 **McGregor, D R and Malone, T W** 'The fact database: a system based on inferential methods' in *Information retrieval research* **Oddy** *et al.* **(Ed.)** Butterworths, UK (1981) pp 203–217

5 **Winterbottom, N and Sharman, G C H** 'NDB: non-programmer data base facility' Technical Report TR.12.179, IBM United Kingdom Laboratories Limited (Sept 1979)

6 **Date, C J** *An introduction to database systems* 3rd Edn, Addison-Wesley, UK (1975)

7 'Database', in *Infotech State of the Art Report, Series 9* **M Atkinson** (Ed.) Pergamon Infotech Ltd. (1981)

8 **Hendrix, D G, Sacerdoti, E D, Sagalowicz, D and Slocum, J** 'Developing a natural language interface to complex data' *ACM Trans. Database Syst.* Vol 3 No 2 (June 1978) pp 105–147

9 **Senko, M E** 'FORAL LP — making pointed queries with a light pen' *Inf. Process. 77* IFIP, North Holland, The Netherlands (1977) pp 635–640

10 **Udagawa, Y and Ohsuga, S** 'Novel techniques to interact with relational databases by using a graphics display' *J. Inf. Process.* Vol 5 No 4 (1982) pp 256–264

11 **Zloof, M M** 'Query-by-example: a data base language' *IBM Syst. J.* Vol 16 (1977) pp 324–343

12 **Herot, C F**, 'Spatial management of data' *ACM Trans. Database Syst.* Vol 5 (1980) pp 493–514

13 **Boguraev, B K and Sparck Jones, K** 'How to drive a database front-end using general semantic information' Technical Report No 32, Computer Laboratory, University of Cambridge

14 **Konolige, K** 'A framework for a portable natural language interface to large databases' Technical Note, Artificial Intelligence Center, SRI International (1979)

15 **Malone, T W** 'What makes computer games fun?' *Byte* Vol 6 No 12 (December 1981) p 258

16 **Cattel, R G G** *Design and implementation of a relationship-entity-datum data model,* Palo Alto Research Centre, Xerox

17 **Goldberg, A and Robson, D** *Smalltalk-80: the language and its implementation* Addison-Wesley, UK (1983)

18 **Azmoodeh, M, Lavington, S H and Standring, M** 'The semantic binary relational model of information' to be presented at *3rd Joint BCS ACM Symp. Res. & Dev. in Inf. Retr., Cambridge, July 2nd–6th 1984*

19 **Azmoodeh, M** 'A scheme for representing information and its implication for storage technology' Internal Report IFS/2/82, Department of Computer Science, University of Manchester (Sept 1982)