Palo Alto Research Center

An Entity-Based Database Interface

By R. G. G. Cattell

XEROX

An Entity-Based Database Interface

by R. G. G. Cattell[1]

CSL-79-9 August 1979

Abstract: A user interface to a database designed for casual, interactive use is presented. The system is *entity-based:* the data display to the user is based upon entities (e.g., persons, documents, organizations) that participate in relationships, rather than upon relations alone as in the relational data model (Codd[1970]). Examples from an implementation of the system are shown, for a prototype personal database (PDB), developed in connection with the ZOG system at Carnegie-Mellon University (Robertson et al[1977]). Some details of the interface and associated issues concerning relational normal forms, views, and knowledge-based assistance are presented. Experience with the prototype system suggests that the entity-based presentation is appropriate for types of casual interactive use that existing database interfaces do not address, such as browsing. It is proposed that such an interface could be used to supplement a query language or other interface to allow users both kinds of views of the data.

[1]This work was primarily performed while the author was employed at Carnegie-Mellon University and was sponsored by the Office of Naval Research under contract N00014-76-0874. This paper was revised for publication as this report at the author's present address, Xerox Palo Alto Research Center. Further work based on the entity-based idea is now in progress. Comments on the ideas expressed herein are solicited.

CR Categories: 3.79, 4.33

Key words and phrases: database interface, man-machine interface, semantic data model, entity-relationship data model

XEROX
PALO ALTO RESEARCH CENTER
3333 Coyote Hill Road / Palo Alto / California 94304

1. Introduction

Motivation and Goals

This paper presents a human interface to a database system with some desirable properties for interactive use. The central idea of the interface is based on viewing data as a graph network. The graph used is analogous to some recent database models (e.g., Chen[1976]), as opposed to the strictly tabular form of the relational data model (Codd[1970]). The nodes in the graph are objects in the data base (referred to as items, or entities) such as a person, place, company, department, or document. The edges in the graph are relationships (tuples) between these entities. (Unlike ordinary graphs, the edges may relate more than two nodes, but the reader may ignore this for now.) At any time, a particular entity and all the relationships (edges) in which it participates are displayed on a user's CRT screen. The user may move to a related entity (adjacent node) in the graph by selecting one of the relation instances shown on the screen in this menu. The related entity then becomes the current one, with the relationships in which *it* participates again displayed as a menu. This simple mechanism is the basic idea of the entity-based interface.

This mode of interaction is promising for a casual interface to a conventional data base because completely naive users very quickly become accustomed to making selections to obtain the information they want. This ease of use was observed in the PROMIS Problem-Oriented Medical Information System (Schultz et al[1971]), that helped motivate the ZOG and PDB work. It is not necessary to learn any sort of data base interaction language; the user is simply making choices, whether browsing through the data base or looking for a specific piece of information. This interface is a competitor for natural language interfaces to data bases, for some applications.

The entity-based mode of interaction is promising for a personal data base for more subtle reasons. The structure of the data base is very similar to that of the semantic net model of human memory, in which long-term memory consists of a set of entities and relationships between them. The experiences reported here suggest that a "mirror" of the user's memory in this form has some desirable properties. For example, the user can very quickly find information in the data base, because it is "indexed" in essentially the same way the user has assimilated the relationships. The data base can thus serve as a particularly efficient External Memory (Newell & Simon[1972]) for the user. The prototype implementation of the entity-based data base was in fact termed PDB (Personal Data Base) because of its envisioned usefulness to a student, scientist, manager, etc., for this purpose.

Developers of similar information systems for personal data base use have observed that systems of this sort can be very useful if a wide range of kinds of information people use daily can be represented and manipulated easily. Cashin et al[1973] and Reitman et al[1969] built systems with

similar goals and mechanisms to this work. However, here a structured database model is used as the basis (rather than pure text plus cross-references), without sacrificing this utility for "miscellaneous" information.

Others have also developed interfaces to relational database systems taking advantage of the two-dimensional display and/or making selections on the screen (Zloof[1975], Senko[1977b]). These interfaces are similar to the one presented here in that they take advantage of the fact that recognition is easier than recall for the user: the system presents the choices which the user chooses or fills in. These interfaces *differ* from the one presented here in that they present the user with the *data schema* (the types of data and relation definitions) rather than the data themselves: the user composes queries by selecting relations and typing in particular desired values rather than using selection to move between particular data. The two kinds of interfaces are appropriate for mutually exclusive kinds of use. Working at the level of the data themselves is useful for browsing through information or answering simple single-tuple queries. Working at the higher level allows manipulation of sets of items as in a query language but gives the user little assistance in exploring the data directly. It is possible to allow both levels of interface in one system by representing the high-level database schema as any other data in the database, as will be shown shortly. A database system should of course provide various kinds of interfaces to the same data to best support a variety of applications.

Relations and Entities

Each CRT display page, corresponding to an entity in the data base graph, is referred to as a frame. The title of a frame for an entity is a print name for the concept it represents, and the relationships (relation instances) in which it takes part are displayed as selections on the display. The user may follow one of these relation-links (which in the simplest case are binary relationships between concepts) to the other item(s) involved by making that selection.

Example

For example, a typical frame might be displayed as follows:

Don Knuth
A. type: person
B. member-of: Stanford CS Dept.
C. spouse: Jill Knuth
D. phone: 936-1212
E. author-of: Fundamental Algorithms
F. author-of: Structured Programming with Go tos

The selections labelled A through F represent relationships in which the entity "Don Knuth"

participates. The frame for "Stanford CS Dept." would be displayed if the user made selection B:

Stanford CS Dept.
A. type: organization
B. psub-of: Stanford
C. phone: 497-2273
D. city: Palo Alto
E. member: Don Knuth
F. member: . . .

The actual mechanism for making selections differs according to the hardware interface available. Some terminals have touch-sensitive screens, and the user simply touches the text of the selection on the screen. On conventional terminals, selections can be made by typing the character preceding the selection.

In addition to simply moving to adjacent nodes in the data base, there are various commands which the user may invoke. For example, the user may create and delete relationships, go directly to a frame with a given name, or return to a previously displayed frame. The commands are listed in the appendix.

Database Scope

It is difficult to communicate without on-line demonstration the "feel" of the entity-based interface and the types of data to which it affords itself. In a typical interactive session with the PDB prototype, for example, a user might arrive at a frame representing the subject area of Database Models by wandering down a subject hierarchy network. Displayed on the screen now would be other related subject areas as well as papers on that subject, projects in that area, etc. The user could select a particular article, at which time its author, publisher, etc., would become visible. If notes on the article have been entered, then one or more of the selections would be a *notes* relationship, and these can be viewed. Furthermore, the notes can be broken up into individual comments on the article, so that further comments and questions that arise (possibly entered by different users) can be linked together in a network of related ideas. When examining any one idea, the related ideas are immediately available selections. The user can back up to previously viewed entities, for example to look at other articles by the same author or to view organization(s) to which the author belongs and examine other work at the same location. The total input from the user to examine all of these data might be as little as a dozen single-fingerstroke selections.

In the prototype data base there are frames representing persons, places, projects, institutions, documents, journals, subject areas, statements, questions, propositions, and special data (e.g., dates and phone numbers). The relationships between these are used to represent a wide range of kinds of information. These can include the kinds of information one might find in an address book, on library index cards, employee records, inventories, and in general in any systematic set of data for

which databases are conventionally used. In addition, relations were also used to embody less structured information. Discussion of issues can be decomposed into questions and statements through relations such as *answer* (to a question), *evidence* (for a statement), *counterevidence, implication, suggestion, proposition,* and so on. Arbitrary relations can be invented by the user as needed to encode information, e.g., "Joe Smith is a *friend* of George Jones who *attended* conference XYZ *in* 1978". The relation *psub* is used to indicate physical subparts of geographical entities (e.g., USA and Massachusetts). The relation *ssub* is used similarly in a lattice of subject areas (e.g., Computer Science and Data Bases), for classification of articles, projects, and so on. Other relations such as *osub* (organizational subpart) also create hierarchies. All relations are superimposed in the user's window into the database as opposed to consulting a different source for each index or type of information.

2. The Extended System

The next few subsections discuss some issues which arose in the implementation of an entity-based database interface.

Entity Display

The basic display of entity as frames in the prototype system has already been described: the print name of the item is given, followed by all the relationships in which it participates. The relationships are displayed as lines of text, which can be defined in BNF as <selection> ::=

<selection letter>. <relation><inverse indicator>: <other item>

where <inverse indicator> is either null or "-of" depending on the directionality of the relationship. For example, note in the previous example that for Don Knuth we display

... member-of: Stanford CS Dept ...

while for the Stanford CS Dept frame we see

... member: Don Knuth ...

A more sophisticated convention for display of relationships is desirable, but this simple mechanism worked adequately for the PDB application. We will discuss some more general mechanisms later.

Note that certain nodes in the data base graph are used only as text strings, and would normally have only one edge emanating from them. For example, the phone number 936-1212 would have the one selection "phone-of: Don Knuth". Such an item is referred to as a *value* item, as opposed to an *entity* item which corresponds to something with independent existence in the real world (see Wiederhold[1977] or Hall et al[1976] for a good discussion of this distinction). Value items need

not be treated as full-fledged items, to improve time or space costs in the implementation. For example, the values might be stored directly for value items, but a pointer of some form is desired for entity items. There need only be displayable frames for entity items (thus the term "entity-based interface"). We will use the term *item* to refer to both entity and value items, and *frame* to refer to their display on the user's screen.

If an entity is involved in many relationships, there may be too many selections to display on the CRT screen. In this case the frame is broken into multiple pages, or *subframes*, with a mechanism to move forward/backward through the pages (see appendix).

Another feature provided in the prototype implementation proved valuable for some uses: it is possible to associate an arbitrary piece of data (in this case, text) with an entity in addition to its print name. The text, if it exists, is displayed when the entity is displayed. This might be used, for example, for a further description of an entity, or for small documents (such as notes on an article that the data base references). It should be noted here that print names and links take relatively little space in storage, so that thousands of items can typically be cached in the physical primary memory. In contrast, the text fields must normally be stored in secondary memory.

The BNF for a frame is now

<frame> ::= <print name of item>
<supplementary text>
<selection>*

where <selection>* is 0 or more <selection>s as defined above (we are ignoring multiple pages here).

Note that the print names of items must be unique, since these are the sole way to refer to a frame. If two objects in the real world actually have the same name, they are made unique by appending a generated number to the name. For casual use, some sort of string search mechanism is useful for referring to items. For example, a wild card "?" may be used when an item is desired with a given substring in its print name.

Item types

Items in the database belong to certain domains, and each relation specifies the domains (types) it requires of its attributes (arguments). For example, the "author" relation requires a *document* as its first attribute, and a *person* as its second attribute. The system uses this information not only to check that new relationships are valid, but also to automatically derive the types of items on the basis of relationships that are input.

The type mechanism used is the conventional specification of relation attribute domains in the relational database model (Codd [1970]). Relations are items, too: relations and the relationships in which they participate are stored and manipulated just as any other item. For example, the frame

for *spouse* looks like this:

spouse
A. type: relation
B. prop: symmetric
C. dimension: 2
D. attr[1]: person
E. attr[2]: person

The frame indicates that "spouse" is a relation (this is required by the system for use as such), that it is symmetric (this suppresses the "-of" extension for the inverse relation), and that it takes two items as attributes, both of type person. Thus the schema of the data base, that defines and controls the data base system, is represented in the data base itself, and thus can be examined and manipulated just as any other data.

It is possible to go still one step further, and allow a *hierarchy* of types. For example, a subtype of type "animal" might be type "person", "elephant", "cat", etc. Type "person" might be subdivided into "male person" and "female person", as well as in other dimensions ("faculty", "staff", "student"). What this hierarchy of types gains us is a more precise specification of relation domains. For example, the second attribute of the "father" relation must be a "male person", not just a "person". The idea of a hierarchy of types is not new; these have been used extensively in Artificial Intelligence (Quillian[1970], Fahlman[1978]). Other ideas are also suggested by the analogy of the data base to semantic nets, such as the inheritance of properties to subtypes (this will be discussed shortly).

N-ary relations and Normal Forms

Thus far, we have dealt only with binary relations. We now introduce the more general mechanism.

Relations with one attribute are termed *properties.* "Symmetric" in the frame for "spouse" is an example of a property.

Relations with more than two attributes are represented in the current implementation by *contexts.* Two attributes of the relation must be thought of as distinguished from the others; the remaining attributes comprise the context. Syntactically, contexts are displayed in brackets. For example,

member[professor]: John Smith

indicates that John Smith is a member of an organization, and in particular that John Smith is a professor there. A special mechanism to make selections is of course necessary now: for a touch screen, the fields of the selection must be independently selectable; for conventional terminals, the typed selection letter must be preceded by a digit indicating the attribute position desired (this defaults to the other primary attribute if no digit is given, giving the convenient effect we had

previously for binary relations). A more general mechanism to represent relationships with more than two attributes is desirable, but again, this was found to be adequate in the prototype.

Relations of order higher than three may similarly be expressed, with multiple attributes as contexts. Real-world relationships which require many attributes have been difficult to find in this work, however. Relations whose third or fourth attribute is not semantically subordinate to the other two (e.g., the "role" of the membership, "professor", in the above example) are even more unusual; the author has not encountered a realistic one.

In contrast to this, examples in the relational data base literature are frequently relations of higher order. A typical example is one whose primary key is some entity domain such as persons in some role (e.g., employees). A series of successively more restrictive normal forms are introduced (1NF, 2NF, 3NF by Codd[1970], 4NF by Fagin[1976]) to avoid semantic update anomalies which essentially are the result of too many "real-world" relations represented by a single database relation. These real-world simplest form relations are termed functional dependencies (FDs). The normalizations reduce the order of relations by introducing new relations to separate these undesired FDs.

The rule used in the PDB data base is relations of *minimum* possible order. Specifically, the data base is in what we will term *Functionally Decomposed Normal Form (FDNF)*, in which the relations are exactly the FDs defined over the real world represented. FDNF is more restrictive than 4NF: not only do update anomalies due to non-orthogonality not occur, but in fact *no further* normalization could possibly be performed in the sense of breaking out implicit dependencies.

A further interesting difference of FDNF over the conventional relational representation is that it is *canonical*: there is *only one* fully decomposed form to represent any given set of facts in a model of the real world, while there are many equivalent but different sets of relations in 4NF. There is some discussion of FDNF by Swenson & Schmid[1975], who refer to FDs between an entity and value as *characteristics*, and between entities and entities as as *associations* (they apparently overlooked FDs of order greater than two). As Senko[1977a] puts it,

> "If followed to its apparent conclusion, the [current trend of] work will result in the definition of a basic data structure component for representing a single fact in the real world, rather than a complex structure [hierarchy or n-ary relation] containing many facts."

Hall et al [1976], who refer to entities as *surrogates*, recommend the equivalent of FDNF as more understandable but also for better properties with respect to update anomalies on primary keys. Recently, a large family of semantic data models have been proposed (e.g., Chen[1976], Benci et al[1976], Smith & Smith[1977], Kerschberg et al[1976], Hammer & McLeod [1978], Rissanen[1977]) which now recognize entities and irreducible relational form (FDNF) as desirable extensions to the relational model. A discussion of these data models is beyond the scope of this paper.

Keep in mind that the logical model which the user sees is independent of the physical model

(which could join several functional dependencies into a single relation); most of the arguments mentioned in the literature are concerned with the logical model.

Standard Database Functions

As mentioned previously, the entity-based interface is an extension rather than replacement for other interfaces. Searches and modifications of the data base can be made either through a programming language or a command/query language (these were provided only in a rudimentary form in the current implementation). The significance of the system described here is the human interface rather than the underlying storage mechanism: the system would probably best be implemented on top of a standard relational or other data base system. (However, reasonably fast access to all relationships in which an entity participates is needed in the underlying system to do this.)

The example of a *personal* data base has been used in this work. However, there is considerable profit in a shared data base when a large network of items is commonly used by multiple persons. The same issues arise here as in any system with multiple users modifying a data base, of course. Locks must be made upon items/relations modified, records kept of originator and time for changes. When it is intended that the system be used as a shared data base as well as a personal one, we need also consider:

1. The user may wish to mark certain items as private, i.e., not visible to other users (this is done with the relation "private").

2. The user may wish to have views of the data base other than the common view. This question is addressed in the next section.

Extensions to the Interface

A few extensions of the interface thus far described should be mentioned here, although only briefly since these are hypothetical and have not yet been implemented.

One valuable extension would be *views*, i.e., showing the user relations other than those explicitly stored in the data base. Relations may be added or deleted from what the user sees, respectively, through:

1. *Inferred Relations:* A relation may be displayed which is not explicitly in the data base, but redundant and derivable from the relations present. For example, if X is the area code of Y, and Z is a geographical subset of Y, then X is the area code of Z. Thus an area code can be stored once but be displayed for all cities, companies, and persons in the area implicitly. The inheritance-of-properties inference used in semantic nets suggests defining defaults through inferred relations. A set of rules would be used to define inferred

relations in terms of existing ones. Inferred relations may either be stored or computed at display time. Some research work has been done on dealing with views and inferred relations (e.g., Minker[1976]).

2. *Filters:* If the user is interested in only one dimension of the data base, or some subset of dimensions, then display can be limited to a specific set of relations. This can be done on a permanent or temporary basis, and can be used in conjunction with a type hierarchy to view entities in particular roles (e.g., as a "professor" as opposed to the more general category "person").

Given a CRT display with graphics capability, another extension of the interface should be considered. The entity-based interface is based upon giving the user a window into the database through which a single graph node and relations emanating therefrom may be viewed. A natural extension of this is *multiple* windows into the database, so that many nodes may simultaneously be viewed. This changes the character of the interface for output to the user, but also for input: a user may point to an existing entity rather than type its name in creating new relationships.

Another area for future work is in encoding procedural knowledge to produce an "intelligent" user interface (Goldstein[1979]). As discussed earlier, a window must be truncated upon display if there are too many relationships to display. Knowledge about the semantics of the data, the user's interests, and the "conversational" context (previous nodes viewed) could be used to decide which relationships to display first. The form in which the relationships are displayed may also be varied according to this knowledge, for example by automatically abbreviating names of well-known items.

3. Conclusions

A psychological evaluation of the relative effectiveness of the entity-based interface is beyond the scope of this paper. However, some statistics on the use of the prototype system should be useful in getting an idea of the properties of the interface. Also, the reader can find a more involved discussion of the issues in a psychological study of the entity-based interface, Mantei & Cattell[1979].

Some Results

The idea of the entity-based interface is a marriage of the ideas of selection-based interface in the ZOG system and the structure provided by a more sophisticated data model. The PDB prototype system was implemented using the LEAP facilities of the SAIL programming language (Feldman[1969], Reiser[1976]) on the DEC PDP-10.

For a rough idea of the use of PDB, consider the author's current personal data base. The data base consists of 1700 items, consisting of approximately 1300 entities and 400 values. The data base grew over a period of about 6 months, in sessions ranging from 20 seconds (typical for a single

retrieval or entry) to an hour (for extensive browsing and updates). About an hour a week is spent using the system. Browsing is not useful until the data base becomes large enough so that the user cannot keep the vast majority of the data in his own long term memory; there is of course no fixed point at which this occurs, but the current size is sufficient. A simple one writer / multiple reader locking mechanism allowed some limited experience with multiple users of the system; browsing was found to be especially useful in examining information entered by other users.

Summary

The entity-based approach looks more attractive than was originally anticipated. It is mechanically simple and consequently easy to use, yet fast for experts because of the small number of keystrokes or selections required to reach a particular piece of data. It provides a view of data qualitatively different from conventional interfaces in that relationships can be viewed and followed directly. The prototype implementation provides a demonstration that the entity-based database interface can be used for a personal data archive. Some extensions and simplifications of the interface presented here look promising for future research, and the author is now engaged in some further experiments in this regard.

Bibliography

Benci, E., Bodart, F., Bogaert, H, & Cabanes, A. Concepts for the Design of a Conceptual Schema, *Proceedings of the IFIP TC-2 Working Conference on Modelling in Data Base Management Systems*, 1976.

Cashin, P., Robinson, M., Yates, D. Experience with SCRAPBOOK, A Non-formatted Data Base System, *Proceedings IFIPS Congress*, 1973.

Chen, P. The Entity-Relationship Model - Towards a Unified View of Data, *ACM Transactions on Data Base Systems* 1,1 (1976).

Codd, E. F. A Relational Model of Data for Large Shared Data Banks, *CACM* 13, 6 (June 1970).

Codd, E. F. Seven Steps to Rendezvous with the Casual User, in Data Base Management, J. W. Klimbie and K. L. Koffeman, (Eds.), North-Holland Publishing Co., Amsterdam, 1974.

Fagin, R. Multivalued Dependencies and a New Normal Form for Relational Databases, *ACM TODS* 2,3 (September) 1973.

Fahlman, S. A System for Representing and Using Real-World Knowledge, Ph.D. thesis, MIT AI lab, 1977.

Feldman, J., & Rovner, P. An Algol-based Associative Language, *CACM* 12, 8 (August) 1969.

Goldstein, I. Personal Communication, 1979.

Hall, P., Oulett, J., & Todd, S. Relations and Entities, *Proceedings of the IFIP TC-2 Working Conference on Modelling in Data Base Management Systems*, 1976.

Kerschberg, L., Klug, A., Tsichritzis, D. A Taxonomy of Data Models, *Systems for Large Data Bases*, P. Lockemann & F. Neuhold (eds), North-Holland, 1976.

Hammer, M. & McLeod, D. The Semantic Data Model: A Modelling Mechanism for Data Base Applications, *Proceedings ACM SIGMOD* 1978.

Hill, R. LUSH Resolution and its Completeness, (CS memo), University of Edinburgh.

Mantei, M., and Cattell, R. A Study of Entity-based Database Interfaces, unpublished experimental summary, 1979.

Minker, J. Search Strategy and Selection Function for an Inferential Relational System, *Transactions on Data Base Systems* 3, 1.

Newell, A., and Simon, H. *Human Problem Solving*, Prentice-Hall, Englewood Cliffs, New Jersey, 1972.

Pirotte, A. "The Entity-Property-Association Model: An Information-Oriented Data Base Model", MBLE Report R343, (March) 1977.

Quillian, M. R. Semantic Memory, in M. Minsky, (ed.), *Semantic Information Processing*, MIT Press, 1968.

Reiser, J. (Ed.), SAIL, Stanford CS Report, (August) 1976.

Reitman, W., Roberts, R., Sauvain, R., Wheeler, D., & Lynn, W. AUTONOTE - A Personal

Information Storage and Retrieval System, *Proceedings of the 24th National Conference of the ACM* (1969).

Rissanen, J. Independent Components of Relations, *ACM TODS* 2, 4 (December) 1977.

Robertson, G., Newell, A., Ramakrishna, K. ZOG: A Man-machine Communication Philosophy, Carnegie-Mellon CS technical report (1977).

Schultz, J., Cantrill, S., and Morgan, K. An Initial Operational Problem Oriented Medical Record System – for Storage, Manipulation, and Retrieval of Medical Data, *AFIPS Proceedings*, vol. 38, 1971, pp. 239-264.

Senko, M.E. Data Structures and Data Accessing in Data Base Systems Past, Present, Future, IBM Systems Journal **16**, 3 (1977a).

Senko, M. E. FORAL LP - Making Pointed Queries with a Light Pen, *Proceedings of IFIPS Congress*, Toronto, 1977b.

Smith, J.M., and Smith, D.C. Data Base Abstractions: Aggregation and Generalization, *ACM TODS* 2, 2 (June) 1977.

Swenson, J. R. On the Semantics of the Relational Data Model, in King, F. (ed.), *Proceedings ACM SIGMOD Conference*, San Jose, 1975.

Waltz, D. L. Natural Language Interfaces, SIGART newsletter #61, February 1977.

Wiederhold, G. Data Base Design, McGraw-Hill, New York, 1977.

Zloof, M. M. Query by Example, *Proceedings AFIPS 1975 NCC*, Vol. 44, pp. 431-437.

Appendix: PDB System Commands

"B" Go Back and redisplay the last frame (item) displayed.

"D" Re-Display the current frame. Useful when modifications have destroyed or invalidated the current screen copy.

"E" Erases the current frame and all relations between it and other frames. Asks "Are you sure?".

"F" Executes special Function. Prompts for the function, e.g., "I" to print isolated frames (no relations), "P" to print relations satisfying a given triple, "M" to modify relations satisfying a given triple.

"G" Prompts for the name of a frame and Goes there (i.e., displays the frame and makes it the current frame).

"I" Prompts for the name of a text file and Inputs frames from it. Update switch after file name causes the addition of relations (and related frames) only for frames which already exist. Several different input formats are permitted, with a default standard.

"L" Prompts for a string and Lists the names of all frames whose name contains the string as a substring.

"N" Prompts for a new Name for the current frame. Note that all references to this frame in relations are thereby changed.

"O" Prompts for a text file name and Outputs all frames and relations to that file. Several output formats are available.

"P" Causes terminal output to be Printed on a file, as well as the terminal (useful for recording interaction). Typing "P" a second time turns printing off again.

"R" Creates a new Relationship involving the current frame. Prompts for the relation name (CRLF causes last relation created to be used again) and the object of the relation.

"S" Saves the core image, which can later be restarted (faster than using text files.)

"T" Prompts for Text to be associated with the current frame. This is printed out after the title whenever the frame is displayed.

"U" Undoes a relationhip involving this frame; prompts for selection character of relationhip to be deleted.

"X" Exits to the monitor (doesn't do a Save).

LF (Line Feed) Causes the next subframe of this frame to be displayed (when there are too many relations for one screenful).

ESC (Escape) Causes the previous subframe of this frame to be displayed.

"?" Gives a list of commands

Type-in not preceded by "/" is interpreted as a selection, i.e., the other item related through the relation labelled by the typed character will become the current frame and be displayed. If the selection letter is preceded by a digit then: (1) if the digit is 0, the relation itself becomes the current frame, (2) if the digit is 1 or 2, the first or second attribute (argument) of the relation becomes the current frame, respectively, (3) if the digit is in the range 3-9 then the corresponding attribute (in the context brackets [...]) becomes the current frame.

Frame Names:

On all commands, the name of an item may be preceded by "?" and the item with the given name as a substring of its name will be used (gives error if ambiguous or no match). Alternate names for items may be established using the "alias" relation. On all commands except Input, the system prompts before creating a new frame when given a new item name. Type checking is performed on creation of new relations; the conventions for this, and the relation-naming conventions ("was-" prefix, "-of" postfix, and N-ary relations) are not described here. Frame names may not contain the character "+"; this is used as a delimiter in the Ascii files. All other non-control characters are legal and significant, including blanks, except "?" can't be used as the first or last character (due to above feature) and ":" can't be used in the names of relations.

XEROX

XEROX

An Entity - Based Database Interface

By R. G. G. Cattell

Xerox Corporation
Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, California 94304

 Printed in U.S.A.

CSL-79-9