

Copyright © 1974, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

CUPID -- The Friendly Query Language

by

Nancy McDonald and Michael Stonebraker

Department of Electrical Engineering and Computer Sciences
and the Electronics Research Laboratory,
University of California, Berkeley, California 94720

ABSTRACT

CUPID (Casual User Pictorial Interface Design) is a facility designed to support non-programmer interactions with a data base system. As such it contains both a data sublanguage and a definition capability. The data sublanguage is complete, high level and picture oriented, depending almost entirely on "menu-move" operations on a CRT terminal. Hence, users do not have the ability to type extensive English text and difficult natural language processing can be completely avoided. The definition feature serves two purposes: 1. to resolve ambiguous interactions, and 2. to allow users to define individual terms, such as "East Coast", "large", etc., for local and global purposes. CUPID is being implemented on top of a general purpose relational data base management system, INGRES, running on a PDP-11/40 computer system at the University of California, Berkeley.

Research sponsored by the National Science Foundation Grant GK-43024X.

I. INTRODUCTION

If the computer is to become an everyday tool of the nonprofessional, the needs and desires of the casual user must be considered. In the past "... narrow technical considerations and immediate cost constraints dominated computer technology..., in large part at the expense of human ease, convenience and social effectiveness" [5]. With the cost of instruction-execution decreasing and the cost of programming steadily increasing, more effort should be expended in reducing the human labor needed to interact with the machine.

Information retrieval and data base management are areas which might benefit from easy human-machine interactions. We feel there are two approaches to enhance such interactions within this context:

1. artificial intelligence methods for natural language processing
2. picture modelling

The difficulties of the first approach are well summarized by one of its proponents, E. F. Codd: "It is very unlikely that any two English-speaking persons understand precisely the same English" [2, p.1]. Since users have a prolific capacity for generating infinite sets of well and ill-formed utterances, the processing of arbitrary English textual input is extremely difficult. Some specific problems are:

- a. typographic and spelling mistakes may be present
- b. English is a language full of ambiguity
- c. a large vocabulary is involved
- d. syntactic analysis is difficult

We feel that there is great promise in the second approach -- picture modelling. The graphics version of CSMP (Continuous System Modeling Program) [12] is a successful example of this approach applied to simulation

problems. The only effort, to our knowledge, in this direction in the data base context is the work of Zloof [16]. However, as with natural language processing, such difficulties as 1. complex syntax allowing arbitrary amounts of information within a 'box', 2. unlimited text strings, and 3. typographic and spelling error possibilities present problems for implementation of his proposal. The system described here has points in common with that in [16]. However, the above drawbacks are avoided.

This paper will describe CUPID (Casual User Pictorial Interface Design), a high level interface facility for a relational data base system. It is the front-end user interface for the relational data base system, INGRES [6,10,11] and compiles "pictures" into the query language QUEL [6,11]. CUPID is being implemented on a PDP-11/40 hardware configuration at the University of California, Berkeley. In CUPID, a query is diagrammed by the user with "menu=move" operations on a cathode ray tube device. Thus, typographic and spelling errors are largely eliminated. Secondly, only constants can be typed from the keyboard; thus most of the complexities and ambiguities of natural language processing are avoided. Moreover, in the structured framework present in CUPID, the resolution of remaining ambiguities is made relatively easy. The vocabulary of the system is limited to user supplied constants and system defined terms. This is a tiny subset of English. Lastly, the processing of CUPID's graphic diagrams is straightforward.

We shall present here the syntax and definition capability of CUPID through annotated examples to illustrate the advantages of naturalness and ease of use.

II. SYNTAX

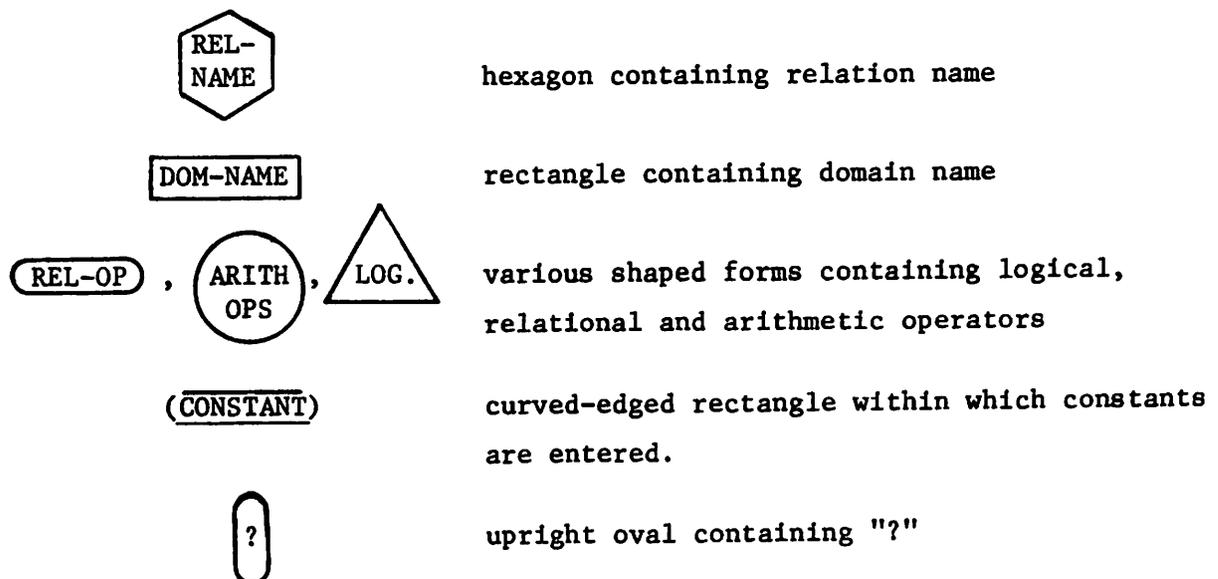
In designing a pictorial interface syntax, we set the goals of:

1. simplicity
2. naturalness
3. ease of use

The visual interpretation illustrated in this section is but one possibility. Due to the powerful graphics modelling system, PICASSO [1], on which the picture processing portion of CUPID resides, we can experiment easily with other possible picture representations.

We recognize the importance of the screen configuration in meeting the design goals of the syntax. Although we have selected one of several possible configurations, we intend to perform human-factors testing on it and other possibilities at a later date. Figure 1 gives information from the chosen layout relevant to this paper. The top of the screen is the modelling space into which CUPID symbols (from the bottom right) are moved. The bottom left portion of the screen contains preselected relation names and domain information used to express the query in the modelling space.

The major components of a picture query in the present representation are:



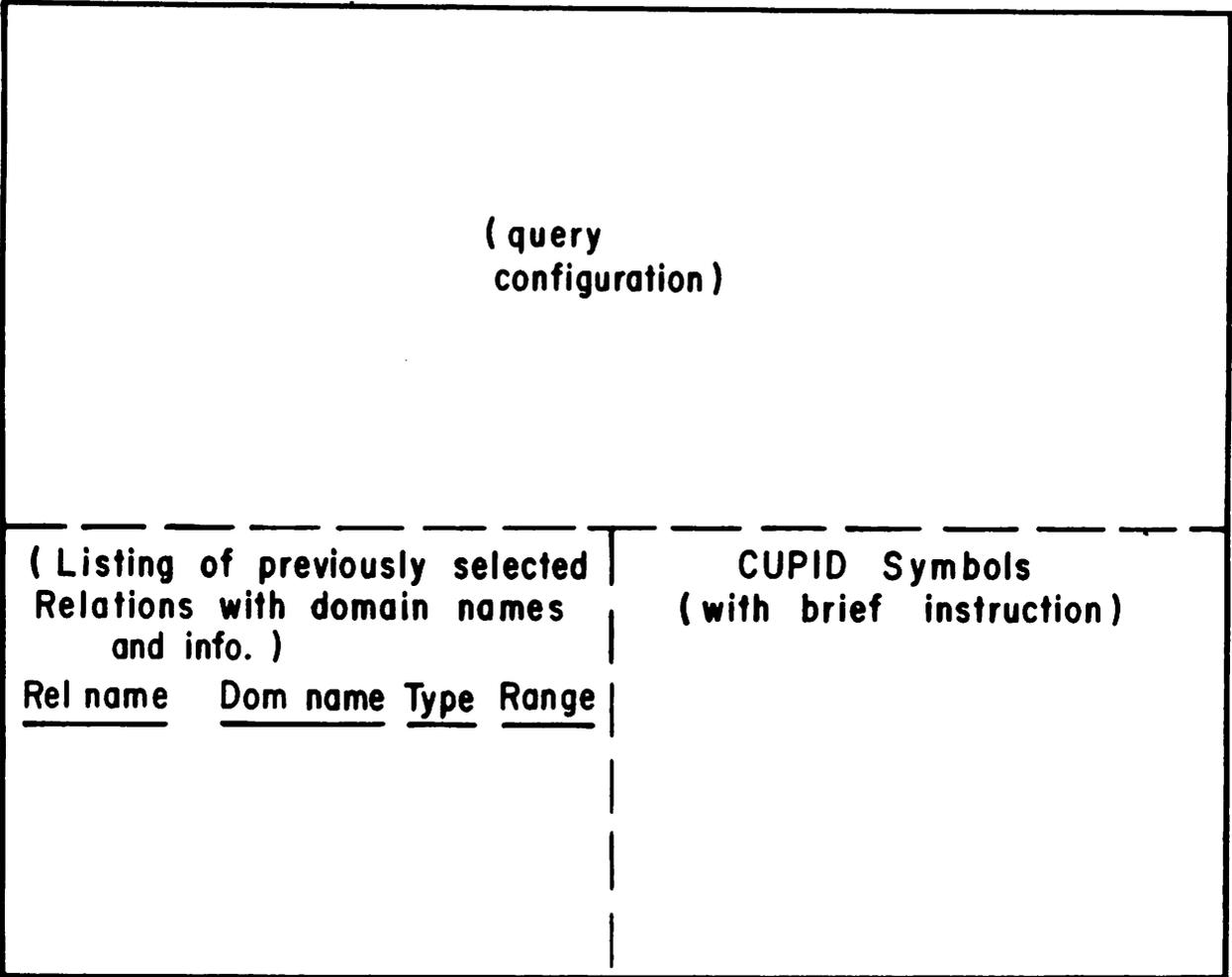
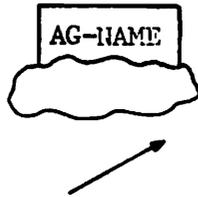


Figure 1 Screen Configuration



wiggly circle with a box on top containing the set or aggregate name arrow to designate connection "from-to"

A formal syntax of CUPID is provided in the appendix.

The following examples illustrate the retrieval mode of the language in a sequence of increasingly complex queries. Although updates involve only simple modification to retrieval operations, space does not permit their inclusion. Please note the format of the examples emphasizes the difference between English language queries and their CUPID equivalents as they appear in the modelling space.

The data base referenced in all examples is a sample warehouse system. PARTS are incoming from SUPPLIERS and go out to customers via ORDERS. The relations involved are:

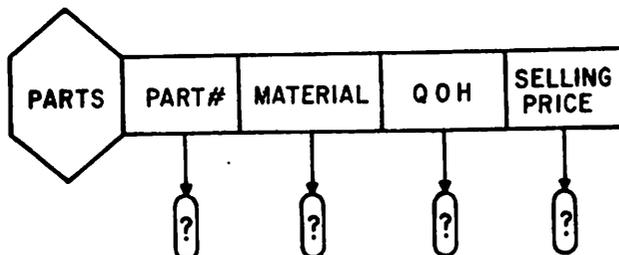
PARTS(PART#, MATERIAL, QUANTITY-ON-HAND, SELLING PRICE)

SUPPLIER(SUPPLIER #, SUPPLIER LOCATION)

ORDER(CUSTOMER #, PART#, CUSTOMER LOCATION, QUANTITY)

PRICE(SUPPLIER #, PART #, BUYING PRICE)

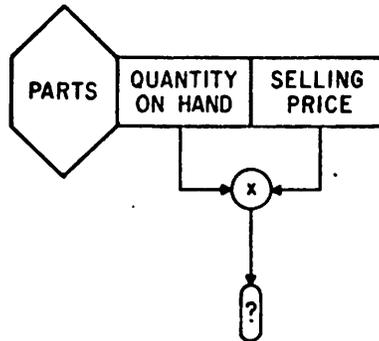
1. List the entire PARTS relation.



Note: 1. Words like "entire" can present problems in natural language processing.

2. The special character "?" indicates what the user wishes to see (i.e. the target list).
3. The ordering of the domains is immaterial.
4. The sequence of steps the user takes in formulating the final query is immaterial.

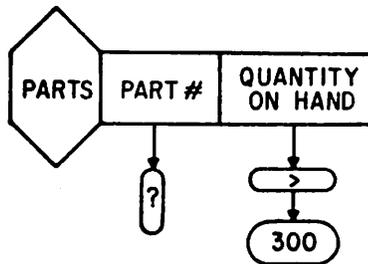
2. Find the total value (SELLING times QUANTITY) of all PART #s.



Note: 1. The English version is ambiguous without the parenthetic remark.

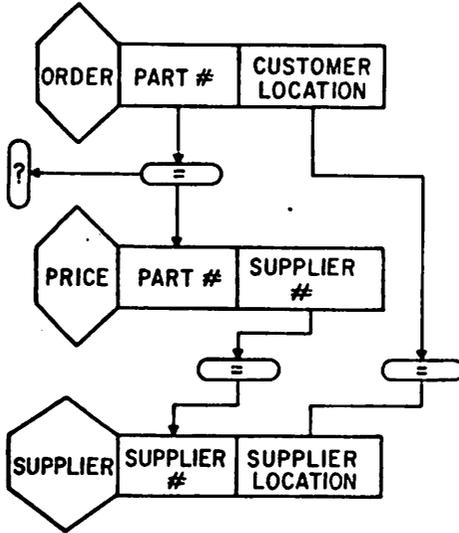
2. Only those domains used need be depicted.

3. Get all PART #s whose QUANTITY-ON-HAND is greater than 300.



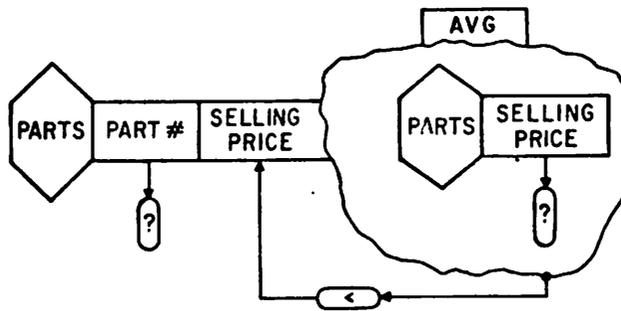
Note: The addition of a qualification to the query allows the only keyboard entry -- the constant.

4. Retrieve the PART #s whose destination and origin are the same city.



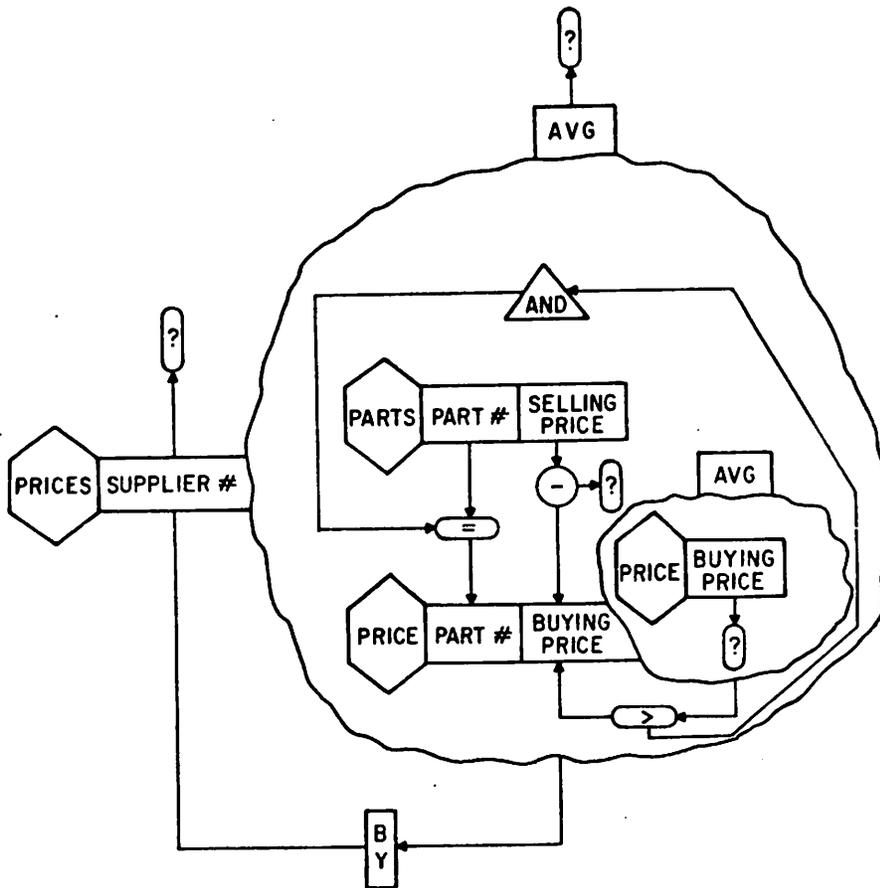
Note: The unmentioned intermediate relation may make processing the English language version difficult. On the other hand, should the CUPID picture be improperly drawn due to such omission, straightforward algorithms (in the next section) can correct the problem.

- List PART #'s whose SELLING PRICE is less than the average SELLING PRICE.



Note: Implied here is the cross-product of the PARTS relation with itself.

- Find the average markup for each SUPPLIER for those PART #'s whose BUYING PRICE exceeds the average BUYING PRICE.



- Note: 1. The last two queries involve aggregation operations.
 2. Query 6 demonstrates the nesting of aggregation.

The advantages of this syntax over a natural language system are now summarized.

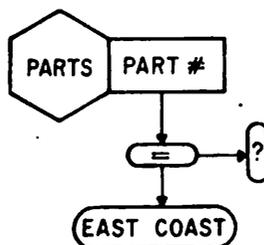
1. No unbounded text strings.
2. Only constants are allowed to be entered from the keyboard thus eliminating most
 - a. typographic errors
 - b. spelling errors
3. Easy parsing.
4. Powerful (in fact, it is QUEL equivalent).
5. User-friendly with its menu-move operations.

We turn now to CUPID's definition facility and its interrelated feature of detection and resolution of improper pictures. This facility offers yet additional assistance to the casual user.

III. DEFINITION CAPABILITY

A definition capability is a mechanism which allows the system to "learn" new concepts, i.e. demonstrate semantic learning. Previous attempts to devise a general semantic learner involved very large programs to recognize a small subset of English within a small universe of discourse [3,7,13,14,15]. In CUPID, the areas of user definition (and, thus the amount of semantic learning necessary) is very restricted. Therefore, our system is not only pragmatically viable, but also moderate in size.

To illustrate this facility we turn to an example. Suppose the user draws the following picture



asking for a listing of "east coast parts". This query presents two problems:

1. Defining the unknown term "east coast".
2. Resolving the misplacement of the constant "east coast" since it is not a known value for PART #.

The following algorithms are applied to all pictures to resolve both problems.

Algorithm 1. For each constant do:

- 1) If constant is numeric, go to 3.
- 2) If constant is non-numeric then check User Definition Table (see Definition Representation section) for previously defined constants and

present user with all established definitions of the term. If one is acceptable to user, use it and exit; otherwise:

3) Ask user to pick from all domain types and compare selected type with type of domain to which new term is connected.

Comment: The set of all domains is categorized at creation time into comparable subsets by the classification referred to here as 'type'. A comparability matrix is generated for easy domain comparisons.

3a) If the types are comparable, perform a QUEL RETRIEVE operation to see if the value exists for the given domain.

1) Value exists, exit.

2) Value is not found -- tell user and offer a list of comparable domains and ask if constant is a value for some domain in this list.

1) If the user selects another domain, redraw the figure with new placement and repeat the algorithm.

2) If user refuses a new domain, utilize a domain type defining procedure to define the new term, then exit

Comment: A defining routine exists for each unique domain type. This routine must be provided whenever a new type is declared. Since "east coast" is of type 'geographic', a map of the U.S. would be displayed and the user asked to circumscribe the area meaning "east coast". Some numeric typed domains will require the user to specify an interval out of the domain range. The last resort is enumeration, or tuple selection.

3b) If types are not comparable, list all comparable domains and ask user to select the correct one. Draw a new picture and reexecute algorithm.

Algorithm 2:

For any two domains connected by a relational or arithmetic operator, check if they are of same type. If so, exit; otherwise:

Beginning at the rear end of the incorrect arrow:

1) List the domain type of the domain and all comparable domains.

Ask user to select the appropriate domain from list.

2a) If the user selects a comparable domain, redraw connection and reexecute algorithm.

2b) If the user insists on a non-comparable domain, process the query allowing the normal software conversion to take place where necessary.

After Algorithms 1 and 2, the query is processed. Expressions with illegal syntax are caught by QUEL parsing and appropriate error messages are returned to the user.

IV. DEFINITION REPRESENTATION

While many definitions are peculiar to their respective definer (user), some global learning can take place in an attempt to provide efficiency for commonly used terms. For this reason, a definition table exists containing the new term, the user's identification, (the domain and relation with which the new term is initially associated,) the type, and a pointer to where the definition is stored.

If enumeration was the method of definition, the definition is stored as a relation with two domains:

1. a duplicate of the associated domain from the original relation.
2. a binary valued domain to indicate which tuples of
 1. fit the definition

For those definitions not accomplished through enumeration, the meaning is stored as one or more program routines - in a manner not unlike Winograd's semantic learner [14].

Certainly, this is a form of learning from the past. When a new user asks for "east coast", the system can flash the picture so defined and ask if that is what is meant. If it is, the established definition is used -- if not, a new definition of "east coast" will be developed and added to the table.

V. IMPLEMENTATION AND SUMMARY

CUPID is implemented on a Digital Equipment Corporation PDP-11/40 computer system with a GT-42 CRT graphical display device. INGRES (including CUPID) runs under the UNIX time sharing system [9]. The picture processing routines are written in Fortran, while the language processing is coded in "C" [8].

Picture processing for CUPID is handled by the graphics modelling system PICASSO. The PICASSO system can be thought of as providing an interactive picture drawing capability and outputting a string corresponding to the picture. This string is then compiled into QUEL using YACC [4].

In summary, we have described CUPID, a graphical human-machine interface for relational data base system. The major innovations as illustrated here include the picture syntax and a definition capability. The advantages include:

- a) it is graphic and interactive
- b) it requires mostly menu-move operations

- c) it has a definition capability
- d) it is relatively easy to implement.

APPENDIX

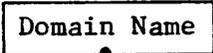
Notational meanings:

Except for aggregates, closed figures contain constants or CUPID provided items. Constants contain keyboard ventries.

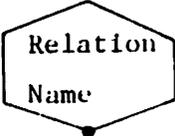
Each terminal symbol has a connecting point, designated by '.'; other terms use the connecting point of their highest precedence binary operator. While only one connection point is shown in the definitions, aesthetic reasons cause us to use more than one point in examples.

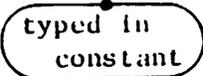
3. The special symbols devised to express picture positioning include:

- a) $a \uparrow b$: connected from a to b by a connecting arrow
- b) $a \Rightarrow b$: a is to the left of b and touches it directly
- c) $a \Downarrow b$: a is above and directly touches b
- d) $a \rightsquigarrow b$: a and b are anywhere on the screen.
- e) $a \approx b$: a is positioned to the left of b and not connected by arrows, but by contiguous picture parts.

Box:  Domain Name

Boxes: Box | Box \Rightarrow Boxes

Hex: 

Const: 

Q box: 

t l 1: Box ↓ Qbox | a-fcn + Qbox

tl: tl1 | tl 1 ~> tl

qual: luop + qual | qual + lbop + clause | clause

clause: a-fcn + rop + a-fcn | a-fcn + rop + a-clause |

set-clause | luop + clause

a-clause: a-fcn + rop + a-fcn

lbop:  | 

luop: 

rop:  |  |  |  |  | 

a-fcn: attrib-fcn | aggr-fcn | ag2-fcn |

a-fcn + bop + a-fcn | Log + const + a-fcn | uop + a-fcn

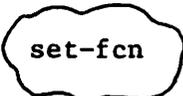
attrib-fcn: const | box

bop:  |  |  |  | 

uop:  | 

Log: 

aggr-fcn 1: aop ↓ ASblob | aop ↓ Sblob

Sblob: 

ASblob: 

aggr-fcn: Hex \Rightarrow Boxes \downarrow By \downarrow aggr-fcn 1 |

aggr-fcn 1

By:

B
Y

ag 2 - fcn 1: aop 2 \Downarrow ASblob

ag 2-fcn: Hex \Rightarrow Boxes \downarrow By \downarrow ag 2-fcn

attrib: box

aop:

SUM

 |

SUMP

 |

AVG

 |

AVGP

 |

MAX

 |

MIN

aop 2:

COUNT

 |

Countp

set-clause: set-fcn \downarrow rop \downarrow set-fcn

set-fcn: set-fcn \downarrow sop \downarrow set-fcn | set-fcn 1 |

SET

 \Downarrow Sblob

a-seq: a-fcn \downarrow a-seq | a-fcn

set-fcn 1:

SET

 \Downarrow ASblob |

Hex \Rightarrow a-seq \downarrow By \downarrow

SET

 \Downarrow ASblob

Sop:

UNION

 |

INTER

 |

DIFF

Que Hex \approx t ℓ | Hex \approx t ℓ \approx qual

Bibliography

1. Austin, D.M. and Holmes, H.H., "PICASSO: A General Interactive Graphics Modelling Program," AEC Contract No. W-7405-eng-48, Lawrence Berkeley Lab, Berkeley, Cal., Jan., 1972.
2. Codd, E.F., "Seven Steps to Rendezvous with the Casual User," IBM Research, San Jose, Calif., RJ 1333, January, 1974.
3. Coles, L.S., "An On-Line Question-Answering System with Natural Language and Pictorial Input," Proc. ACM 23rd Nat. Conf., 1968.
4. Johnson, S.C., "YACC, Yet Another Compiler-Compiler," UNIX Handbook, Bell Telephone Labs, Murray Hill, N.J., July, 1974.
5. Martin, J., Design of Man-Computer Dialogues, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1973.
6. McDonald, N., Stonebraker, M., and Wong, E., "Preliminary Design of INGRES," ERL, Univ. Calif. Berkeley, Mem #ERL-M435, April, 1974.
7. Quillian, M.R., "Semantic Memory," Ph.D. Thesis, Carnegie-Mellon Univ., Pittsburgh, Pa., Feb., 1966.
8. Ritchie, D.M., "C REference Manual," UNIX Handbook, Bell Telephone Labs, Murray Hill; N.J., July, 1974.
9. Ritchie, D.M. and Thompson, K., "The UNIX Time-Sharing System," Proc. 6th Operating Systems Symposium, Yorktown Heights, N.Y., Nov., 1973.
10. Stonebraker, M. and Wong, E., "Access Control in a Relational Data Base Management System by Query Modification," ERL, Univ. Calif. Berkeley, Mem. # ERL-M438, May, 1974.
11. Stonebraker, M. and Wong, E., "INGRES-A Relational Data Base System," ERL, Univ. of Cal., Berkeley, Mem.# ERL-M472, Nov., 1974.

12. Walsh, W.J., "Computer Graphics Applications to Interdisciplinary Problem Solving: CSMP User's Manual, EECS 298-15, Univ. of Cal., Berkeley, Winter/Spring, 1971.
13. Weizenbaum, J., "ELIZA," Comm. ACM, Vol. 9, No. 1, January, 1966.
14. Winograd, T., "Procedures as a Representation for Data in a Computer Program for Understanding Natural Language," Revised Ph.D. Dissertation, M.I.T., January, 1971.
15. Woods, W.A., "Semantic Interpretation of English Questions on a Structured Data Base," Rep. NSF-17, 1967, Computer Lab., Harvard Univ., Cambridge, Mass., August, 1966.
16. Zloof, M.M., "Query by Example," IBM Thomas J. Watson Research Center, Yorktown Hgts., N.Y., RC4917.