(2)

USAISEC

US Army Information Systems Engineering Command
Fort Huachuca, AZ 85613-5300

U.S. ARMY INSTITUTE FOR RESEARCH IN MANAGEMENT INFORMATION, COMMUNICATIONS, AND COMPUTER SCIENCES

ARMY NONPROGRAMMER SYSTEM FOR WORKING ENCYCLOPEDIA REQUESTS TOOLS MANUAL (ASQB-GI-90-006)

December 1989

AIRMICS
115 O'Keefe Building
Georgia Institute of Technology
Atlanta, GA 30332-0800

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704--188
Exp. Date: Jun 30, 1986

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS NONE
2a. SECURITY CLASSIFICATION AUTHORITY N/A	3. DISTRIBUTION / AVAILABILITY OF REPORT N/A
2b. DECLASSIFICATION / DOUWNGRADING SCHEDULE N/A	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) ASQBG-I-90-006	5. MONITORING ORGANIZATION REPORT NUMBER(S) N/A
6a. NAME OF PERFORMING ORGANIZATION AIRMICS 6b. OFFICE SYMBOL (if applicable) ASQBG - I	7a. NAME OF MONITORING ORGANIZATION N/A
6c. ADDRESS (City, State, and ZIP Code) 115 O'Keefe Bldg., Georgia Institute of Technology Atlanta, GA 30332-0800	7b. ADDRESS (City, State, and Zip Code) N/A
8a. NAME OF FUNDING/SPONSORING ORGANIZATION AIRMICS 8b. OFFICE SYMBOL (if applicable) ASQBG - I	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N/A
8c. ADDRESS (City, State, and ZIP Code) 115 O'Keefe Bldg., Georgia Institute of Technology Atlanta, GA 30332-0800	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO.: 62783A PROJECT NO.: DY10 TASK NO.: 04-01 WORK UNIT ACCESSION NO.:

11. TITLE (Include Security Classification)

Army's Nonprogrammer System for Working Encyclopedia Requests (ANSWER) Tools Manual (UNCLASSIFIED)

12. PERSONAL AUTHOR(S)

Dr. Karen R an, Cho-Li Hou, Datta Shetti

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
	FROM 06/16/89 TO 12/15/89	December 15, 1989	33

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Data Encyclopedia, Very Large Database, Distributed Query Processing, Metadata

19. ABSTRACT (Continue on reverse if necessary and identity by block number)

This report describes the current results of research to develop a set of data management tools for use by database planners, developers, administrators, and users within the Army. Specifically, several data management tools that have been prototyped to date are described to include database registration, schema integration, browsing, an AI based standard data element naming tool, and an Information Resource Dictionary System (IRDS) repository. This toolset has been integrated under the X-windows interface management system. Plans for future efforts include additional AI techniques for data management, security, distributed query formulation, and distributed query processing.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION	
☒ UNCLASSIFIED / UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	UNCLASSIFIED	
22a. NAME OF RESPONSIBLE INDIVIDUAL CPT Joseph J. Nealon	22b. TELEPHONE (Include Area Code) (404) 894-3110	22c OFFICE SYMBOL ASQBG - I

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

This research was performed by Honeywell Federal Systems, contract number DAKF11-88-C-0024, for the Army Institute for Research in Management Information, Communications, and Computer Sciences (AIRMICS), the RDTE organization of the U.S. Army Information Systems Engineering Command (USAISEC). The report discusses a set of data management tools developed and/or enhanced during a 6 month effort, the second phase of a four phase effort. Requests to view a demonstration of the prototype may be made by contacting CPT Joe Nealon at 404/894-3110. This research report is not to be construed as an official Army position, unless so designated by other authorized documents. Material included herein is approved for public release, distribution unlimited. Not protected by copyright laws.

THIS REPORT HAS BEEN REVIEWED AND IS APPROVED

s/ Glenn Racine

Glenn Racine, Chief
Computer and Information
Systems Division

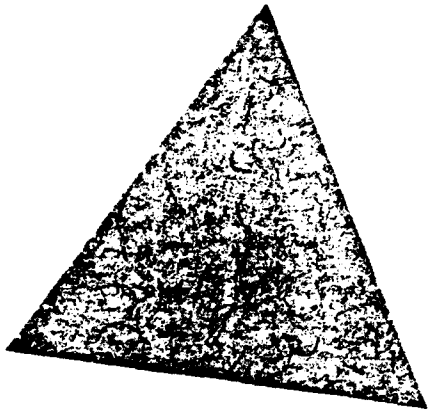
s/ James T. Blake LTC, AV
for

John R. Mitchell
Director
AIRMICS

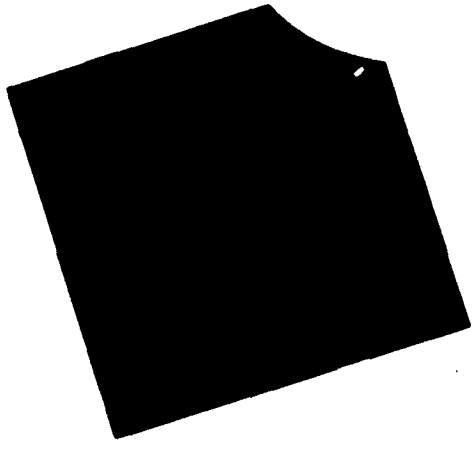
Honeywell

ANSWER Tools Manual

Volume I. Installation and Use



Contract No.
DAKF11-99-C-0024



December 15, 1989

Sensor and System Development Center
1000 Boone Avenue North • Golden Valley • Minnesota • 55427

Honeywell

ANSWER
Tools Manual

Volume I.
Installation and Use

Contract No. DAKF11-99-C-0024

Prepared for:

AIRMICS
Under Subcontract to
Honeywell Federal Systems Inc.

Honeywell Inc.
Sensor and System Development Center
1000 Boone Avenue North
Golden Valley, Minnesota 55427

December 15, 1989

G89386

Table of Contents

List of Figures

Section 1
Introduction

This manual describes the installation and use of the ANSWER tool set developed under Phase II. The tools described in this manual include:

- User interface manager,
- Schema integrator,
- Create schema facility,
- Data element creation tool.

The source code for the user interface manager and the data element creation tool and the scripts for the data element creation tool and the schema integrator are included in Volume II.

Section 2
User Interface Manager

The user interface manager supports the presentation of information and tracks the current status of tool invocations, suspensions and resumptions. When the ANSWER tool set is initially installed, the user interface manager is configured as shown in Figure 2-1.

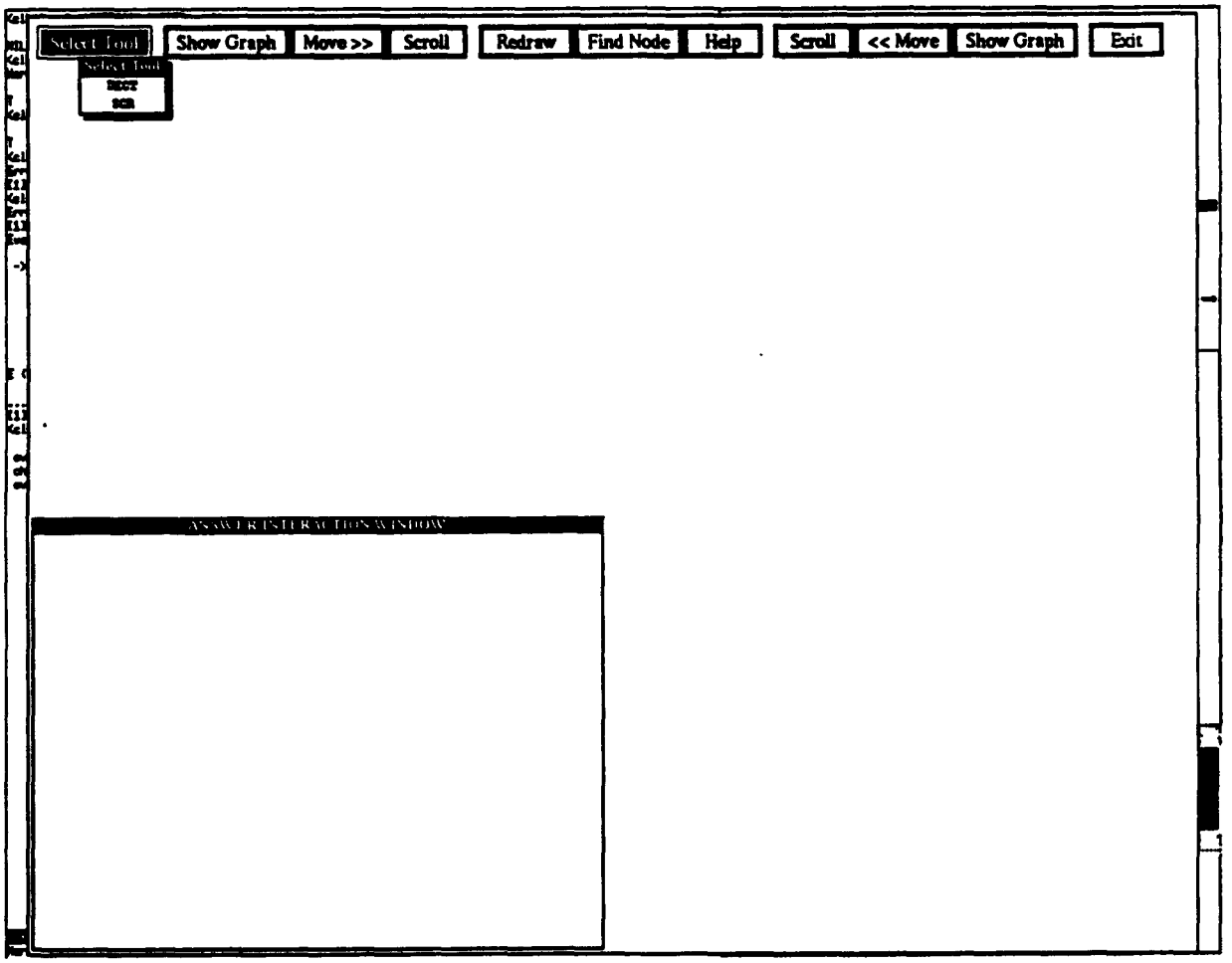


Figure 2-1. User Interface Manager Initial Screen Display

The interface is organized as a pair of fixed-position graphics display windows; a set of global command buttons across the top of the screen; a tool-specific interaction window that changes depending on which tool is invoked. The graphics windows are always available to display schema and Army Data Dictionary information.

The interface is designed to display information about database schemas, Army Data Dictionary information and tool status and local commands. The schemas and ADD information are displayed graphically at user request or as directed by a specific tool. In

certain cases, the user may suspend processing of a tool to inspect schema or ADD information in the graphics display windows.

A number of global commands are available to the user, including:

- Select tool,
- Move left,
- Move right,
- Scroll,
- Find node.

The select tool command provides a list of tools available at any point in time to the user through a pop-up window display. The move left and move right commands will shift the graphics display in the designated window to the graphic display to the left or to the right. The scroll command will provide a topological display of the entire graph currently being displayed in a window. Typically, a graph is too large to completely display in a graphics window at one time. The graphics facilities provide assistance to scroll the graph to expose additional nodes. The find node command will identify the location of any schema that has currently been read into the ANSWER environment.

The graphics display window also supports node-specific commands that are accessible by positioning the mouse on the node and clicking left. The node-specific commands will vary with the type of node being displayed and with the current state of a computation. Typical node-specific commands include:

- List attributes,
- Show definition,
- Create mapping,
- Remove mapping,
- Change level.

The list attribute command is used to expand an attribute node in a schema to display all of the individual attributes represented by that node. Attributes are represented this way to avoid cluttering the graphical display with large numbers of attributes for each individual entity.

Show definition will display definition information of a schema node or an ADD element.

Create mapping will create an association between a schema node and a ADD element. (Note: The committing of the results to IRDS has not yet been implemented.)

Remove mapping will remove an established link between a schema node and a ADD element. (Note: committing of the results to IRDS has not yet been implemented.)

Change level will allow the user to follow the association of ADD elements with schema elements and browse through portions of the ADD.

Once a specific tool has been selected, the interface for that specific tool is presented in the tool-specific interaction window and graphical information relevant to the tool is displayed in the graphics display windows. The global commands remain available throughout any tool interaction. The node-specific commands will change depending on the type of information being displayed.

Section 3
Schema Integrator

The functionality of the schema integrator is essentially unchanged from the Phase I prototype to the Phase II prototype. The interface for the schema integrator is substantially different. The Phase II interface makes use of the interface capabilities provided by the user interface manager to graphically display information and interact with the user through direct manipulation and other interface devices (e.g., pop-up menus).

The schema creation phase, originally part of the Phase I schema integrator prototype is now accessible as a separate tool. It will be discussed separately. The schema integrator itself now consists of the following five major phases:

- Read input schemas from IRDS,
- Equivalence class creation and deletion,
- User assertion collection,
- Display integrated schema,
- Commit integration results to IRDS.

The first phase, reading schemas from IRDS will access IRDS and read schemas into the local representation format required by the schema integrator. The access to IRDS is unchanged from Phase I. The interface in Phase II involves a series of pop-up menus which display the information available in IRDS. The user selects the schemas to be read in from IRDS as shown in Figure 3-1.

Schemas selected for integration are displayed in the two graphics interaction windows.

The second phase invoves collecting information about equivalence class relationships between attributes. The user may scroll through the graphical displays of the two schemas. When the user identifies a pair of attributes, one from an object in schema[i] and one from an object in schema[j], the attributes may be equivalenced by first selecting the attributes with the mouse and then confirming the selection through a pop-up menu. Figure 3-2 shows the schema integrator at a point where two attributes have been selected and the user may confirm the selection.

Once a pair of attributes have been selected, the system provides an explicit prompt to the user to examine the standard data elements associated with the attributes. If the user indicates the standard data elements are not correct, the system will invoke the Data Element Creation tool to provide assistance in creating a new data element. See Figure 3-3.

While the system provides an explicit prompt here to change tools, at other points in the processing, without an explicit prompt, the user may elect to suspend the schema integrator and invoke another tool by selecting a new tool from the set of global commands.

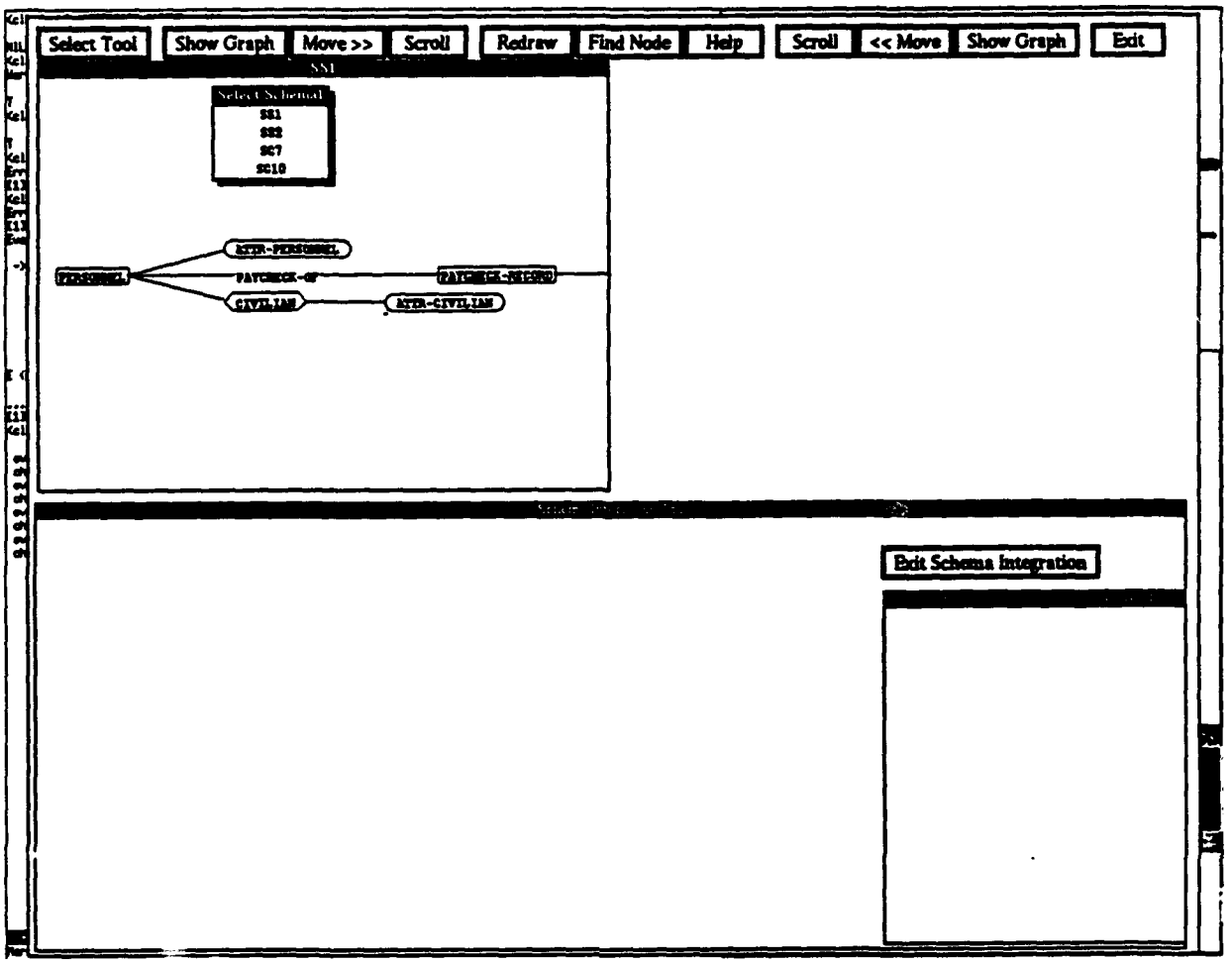


Figure 3-1. Menus for Selection of Input Schemas

At this point, the user has selected a set of attributes that are used to drive the next step of integration, user assertion collection. In this step, the system displays object pairs to the user as driven by the selection of attribute equivalence classes. The user indicates the relationship between the object pairs by selecting from a list of six possible relationships: equal, not equal, subset, superset, disjoint but integratable, disjoint and not integratable. The interaction screen for this step is shown in Figure 3-4.

When the user selects an assertion type for the object pair, there is an explicit prompt to review the prime words associated with the objects. If the prime words are not correct, the system will provide assistance in selecting a new prime word.

The user indicates they are finished with this step in the process by clicking on the DONE button. The list of object pairs is then analyzed by the schema integrator. The schema integrator will identify inconsistent assertion sets and send those back to the interface for redisplay and resolution by the user. For example, if the user asserts that A is contained in B and B is contained in C, and that A and C are disjoint and not integratable, the system will return the assertions to the user for revision.

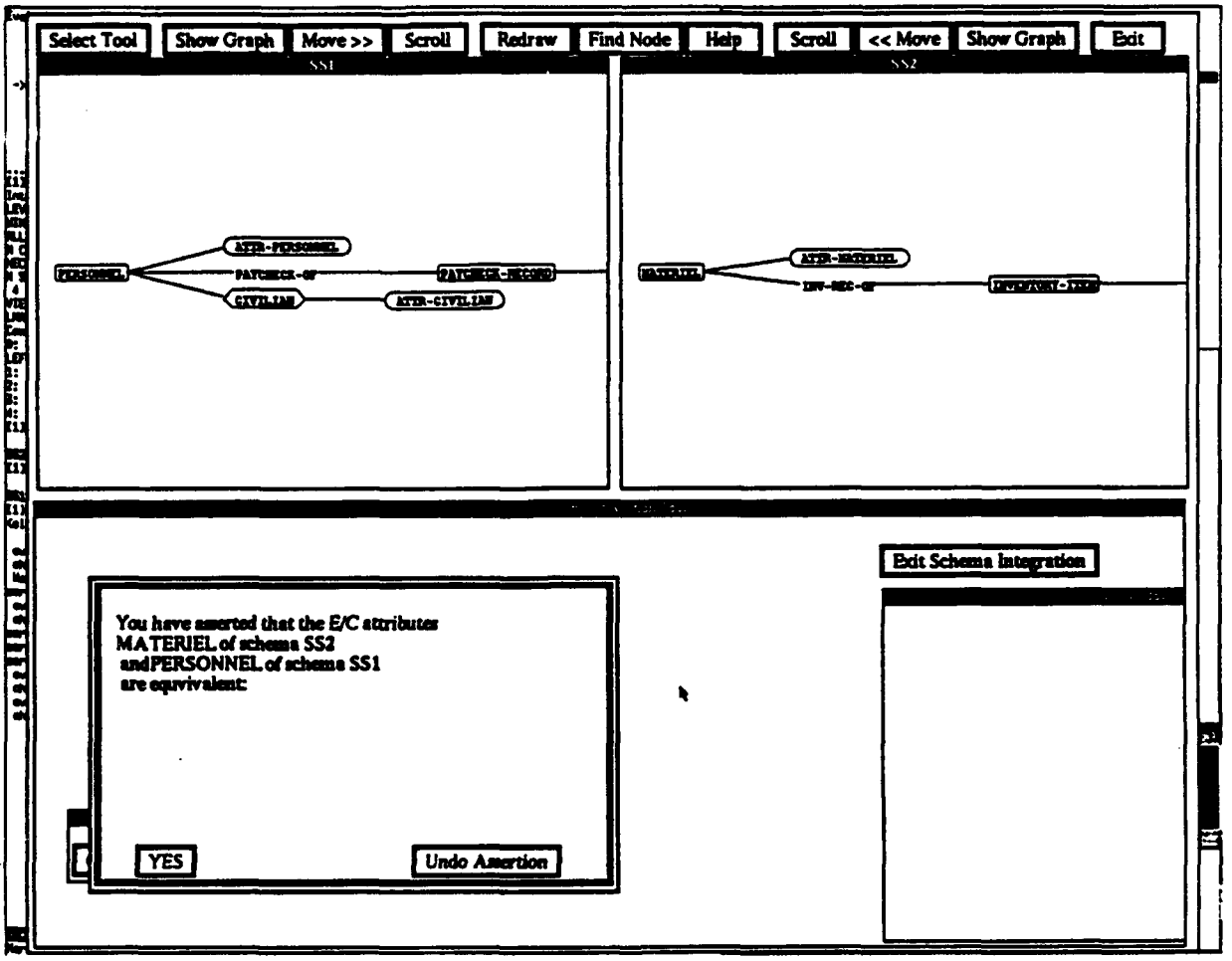


Figure 3-2. Menu to Confirm Attribute Selection

The process of attribute selection and assertion collection may be repeated for attributes of relationships if there are any.

Once the identification of equivalence classes and collection of assertions is completed, the schema integrator will generate an integrated schema. (See the Phase I final report for a discussion of the functional algorithm for schema integration.) The integrated result can be displayed to the user in the graphical windows.

When the user is satisfied with the integrated result, the schema may be committed to IRDS as shown in Figure 3-5.

The current implementation of the user interface manager will not support direct editing of the integrated result until the schema hass been committed to IRDS. Future versions will be able to support this feature.

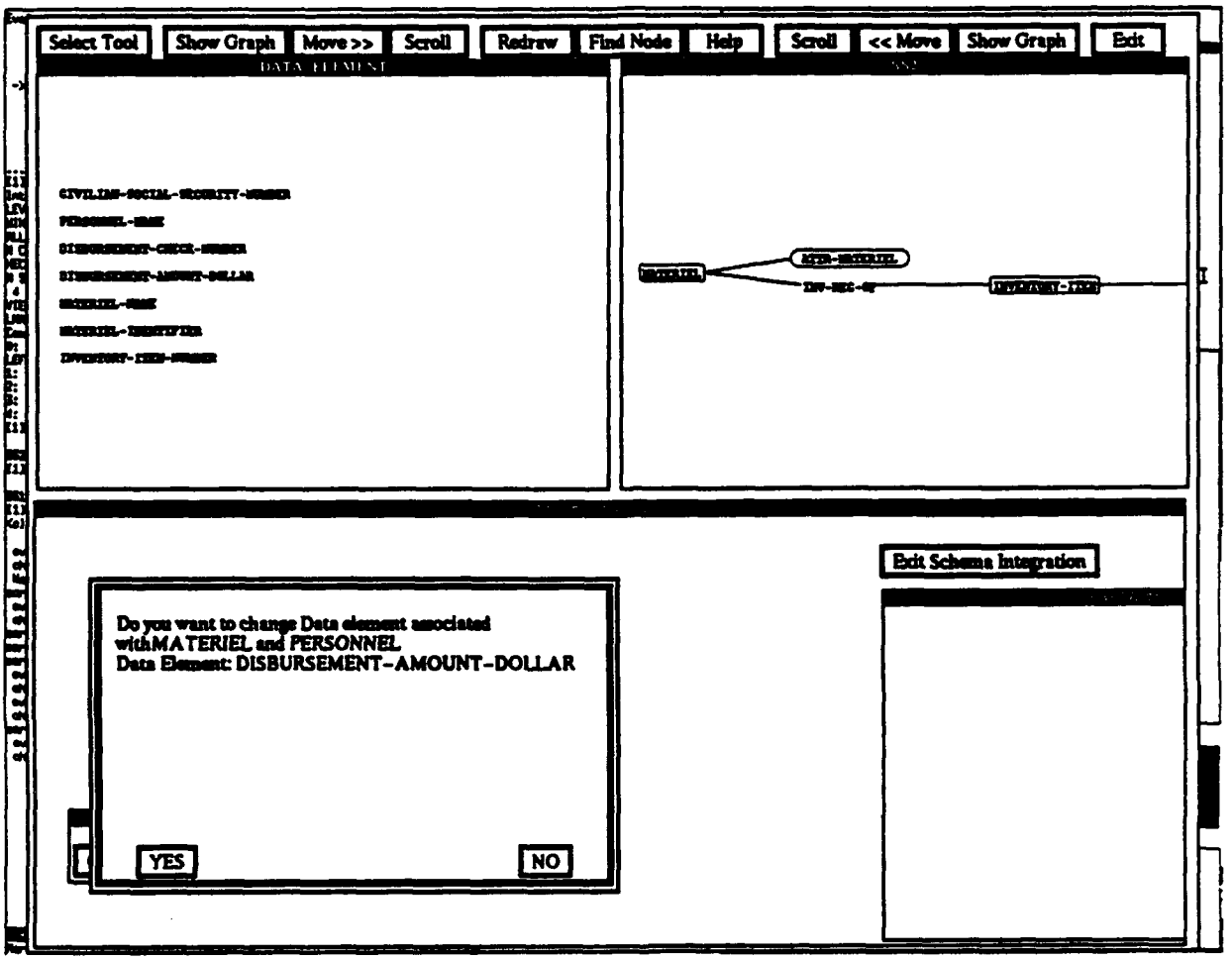


Figure 3-3. Menu to Confirm SDE or Invoke DECT

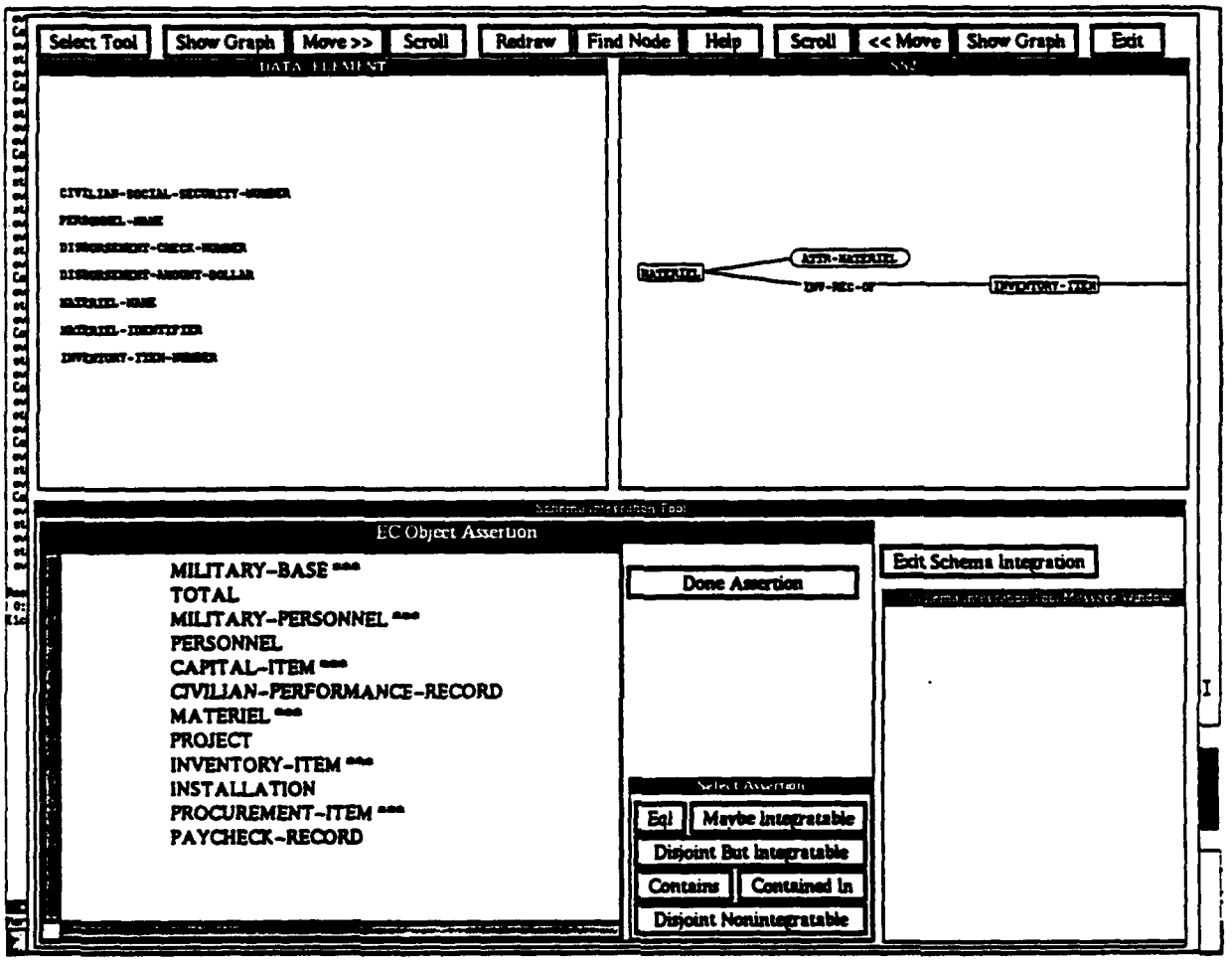


Figure 3-4. Assertion Collection for Objects

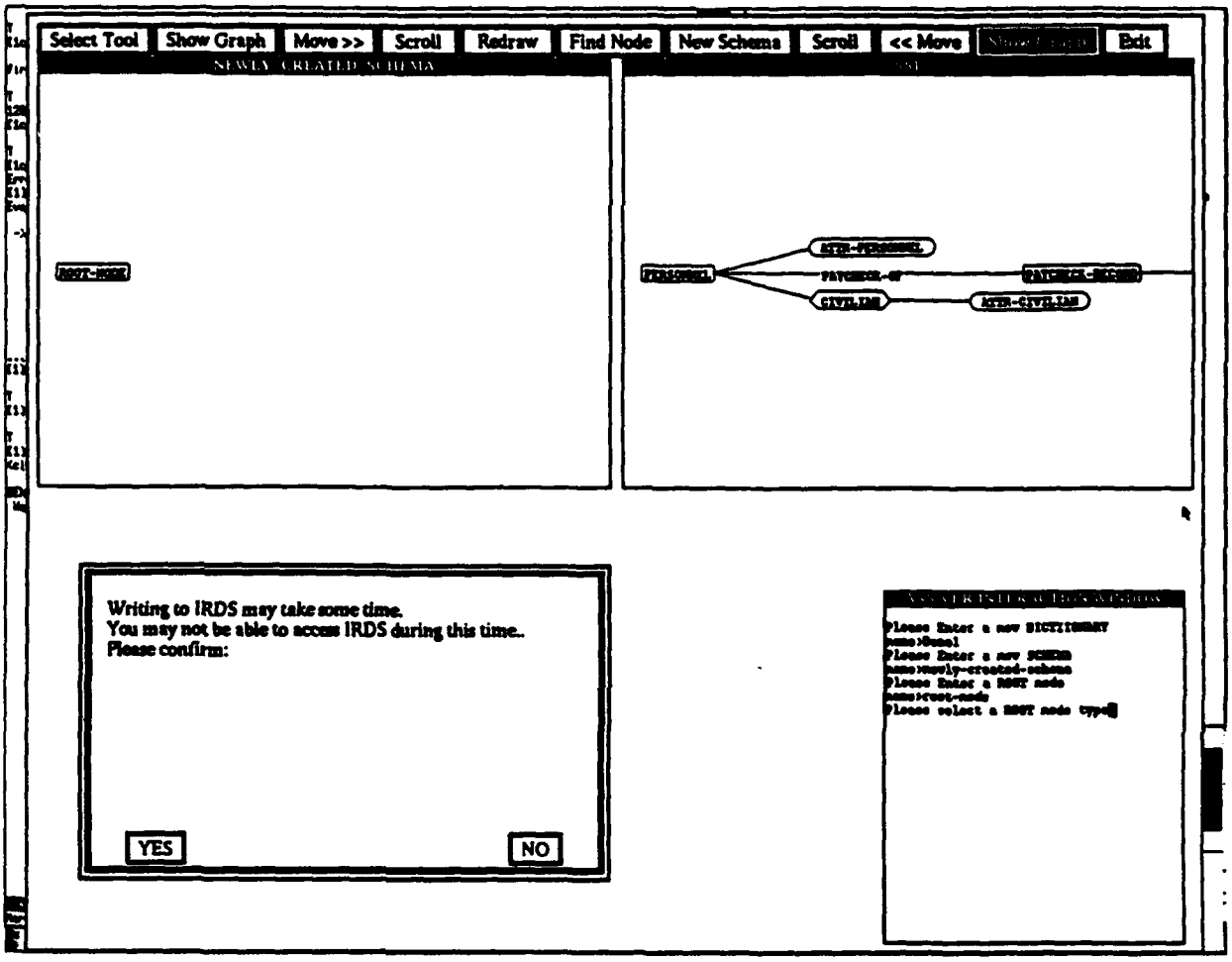


Figure 3-5. Menus for Writing Schema Back to IRDS

Section 4
Create New Schema

The functionality of creating a new schema was provided as part of the Phase I prototype of the schema integrator. In phase II it has been separated out as a distinct tool. The Create Schema tool provides the ability to add and delete nodes from existing schemas or to create an entirely new schema. The initial interface display for the create schema tool is shown in Figure 4-1

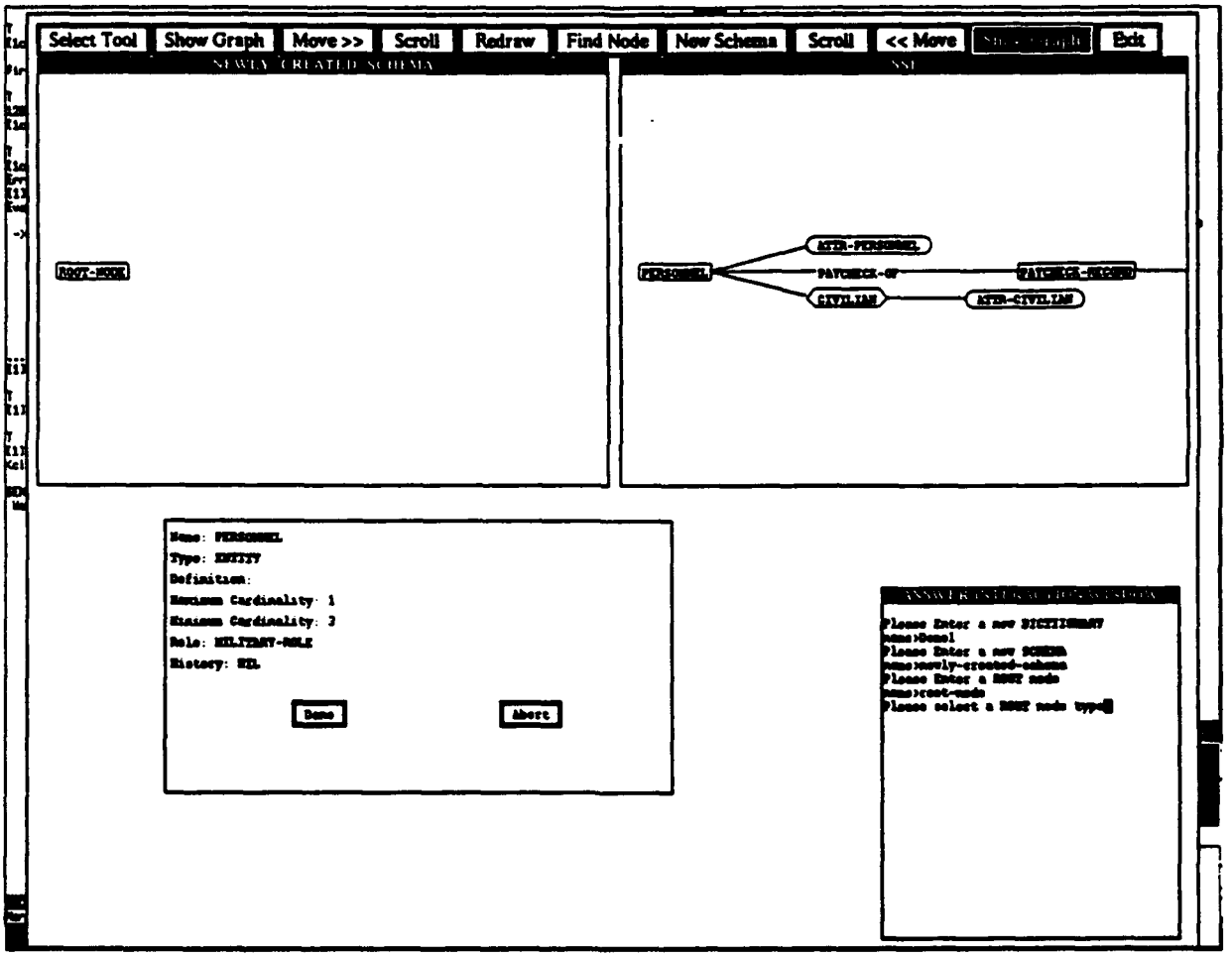


Figure 4-1. Create Schema Tool

The tool is very simple, offering the following commands:

- Add New Node,
- Delete Node,
- Edit Node,
- Commit to IRDS.

A schema may be read in and edited using these commands. Add node creates a new node of a specified type and connects it to the existing schema as indicated by the user.

The delete node command removes the indicated node from the schema.

Edit node will display the information keep about individual nodes and allow the user to modify that information. For example, if the user wished to modify the definition stored for a particular node the interaction would appear as shown in Figure 4-2.

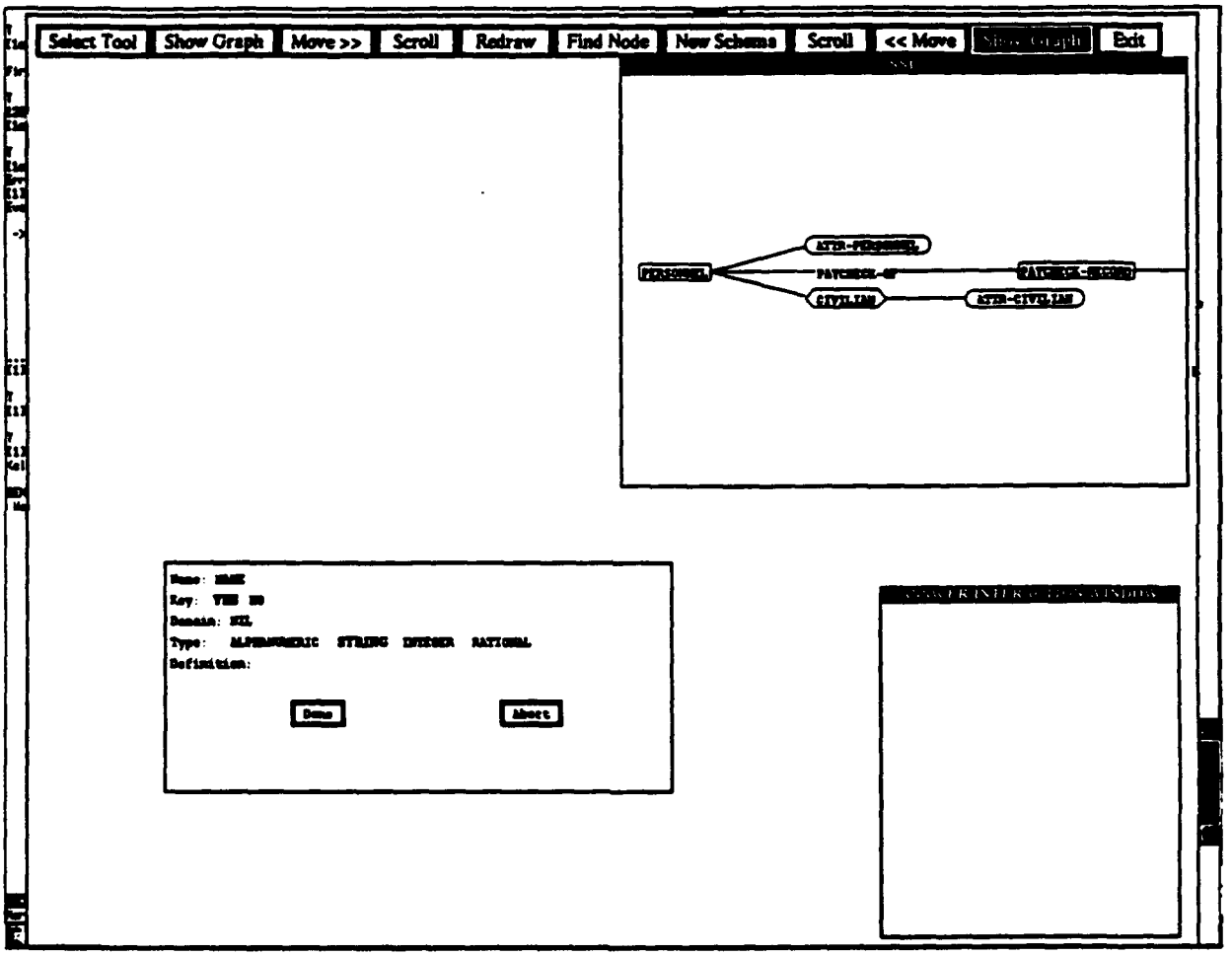


Figure 4-2. Edit Node

Once all editing has been completed, the user may elect to commit the schema to IRDS. The edits are committed as a new version of the schema if it is an existing schema. IRDS manages versions by saving a complete copy of the new version. It is recommended that the user only commit the editing results to IRDS when major modifications have been made to avoid the proliferation of versions with only minor deltas.

Section 5
Data Element Creation Tool

The data element creation tool is an initial prototype tool. It runs under the user interface manager, but it has not yet been integrated with IRDS. All data used by the Data Element Creation tool is currently local to that tool.

The Data Element Creation Tool (DECT) is designed to aid the user in creating new standard data element names. It is based in part on a prototype originally built by MITRE. It has been reimplemented with enhancements to the original MITRE algorithm to improve the definition generation capabilities and to provide some assistance in identifying synonymous data element names.

The DECT leads the user through a session by instructing the user to first select a prime word to be used in the data element name. Definitions of prime words may be displayed on demand as well. Figure 5-1 shows the prime word selection step of DECT.

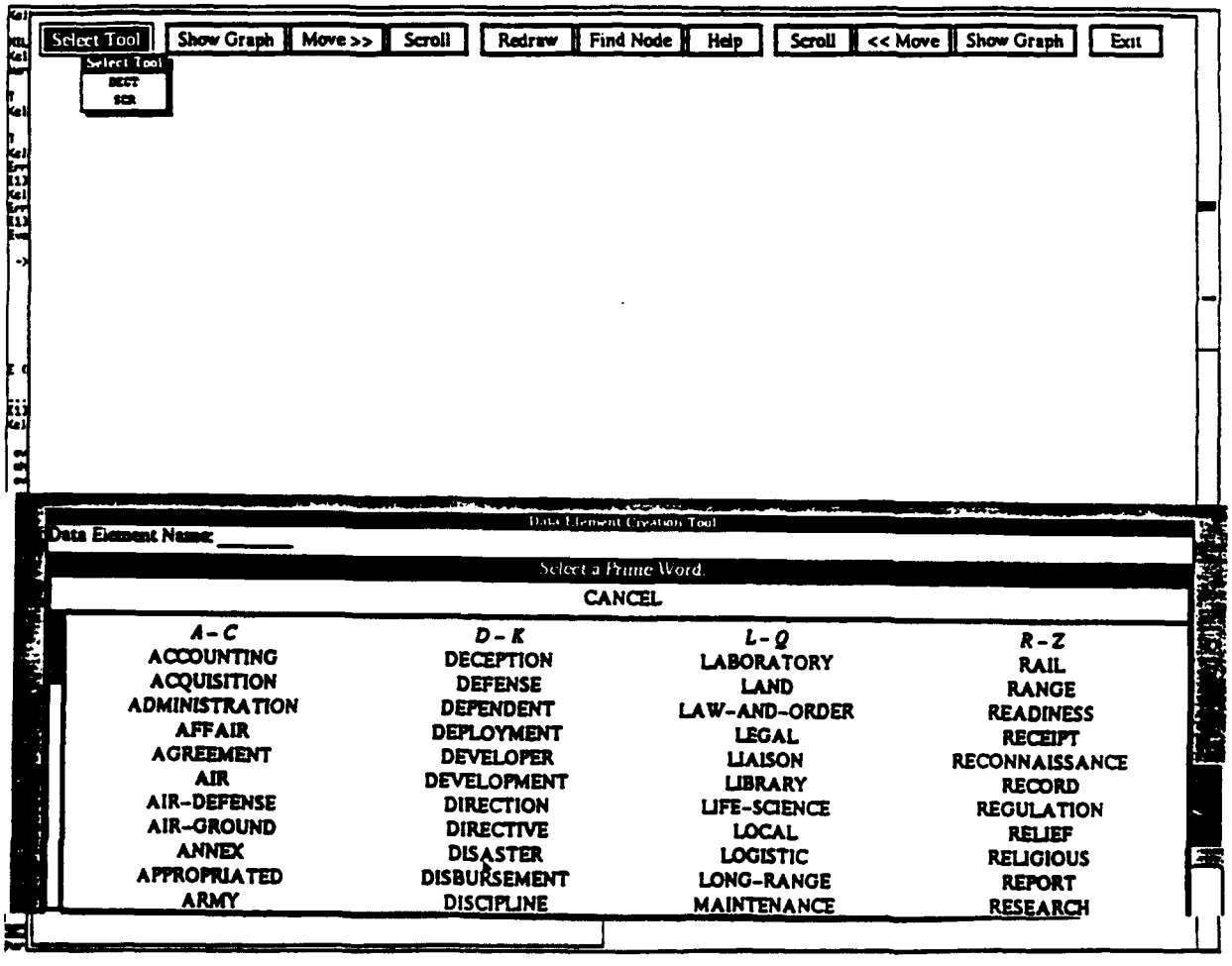


Figure 5-1. Prime Word Selection

Once a prime word has been selected, the user then selects modifiers for the prime word. The modifiers are also selected from a menu of possible modifiers. The modifiers must be predefined to aid in definition generation and data element classification. A data administrator may add new modifiers to the system as necessary. The modifier menu for prime words is shown in Figure 5-2.

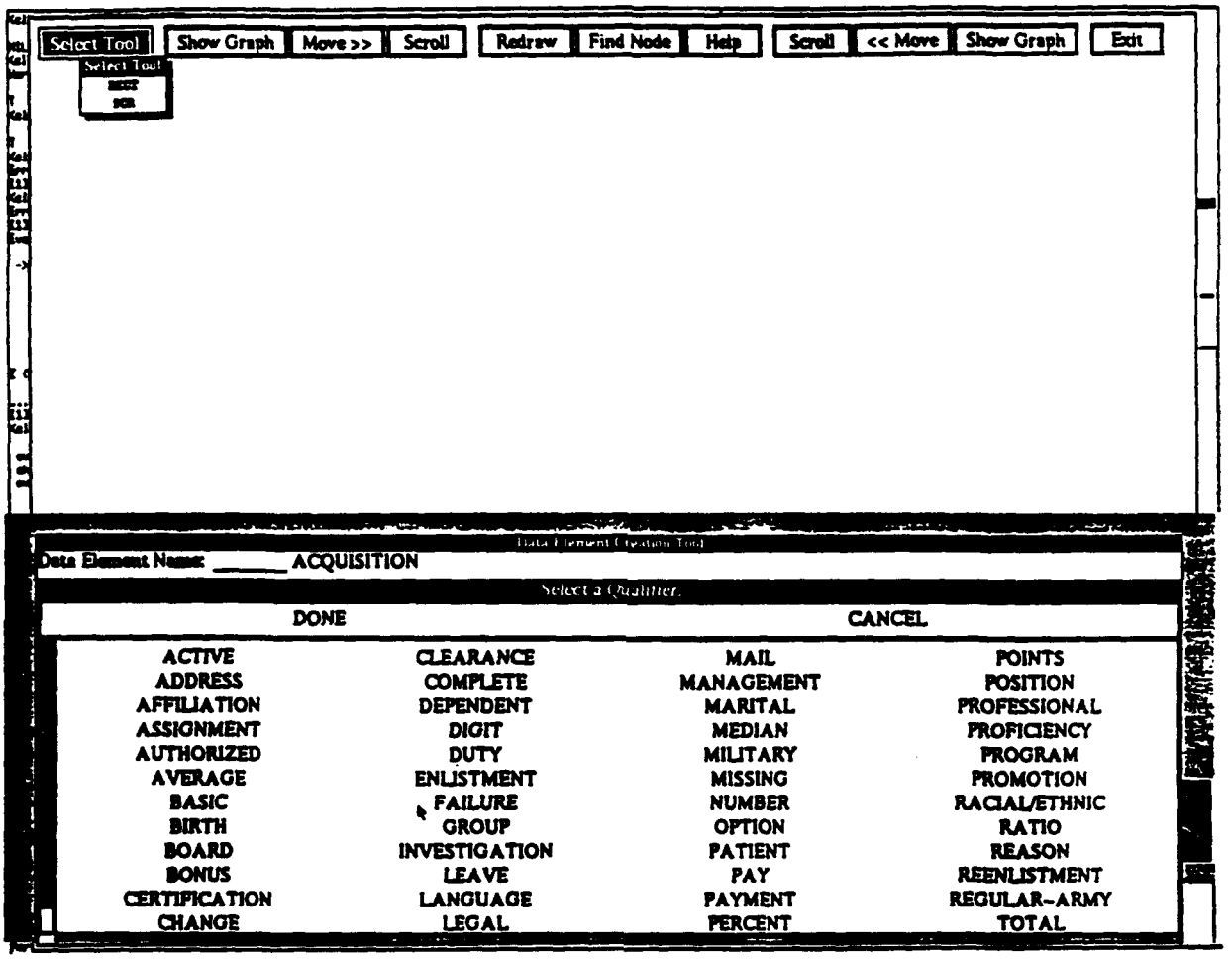


Figure 5-2. Prime Word Modifier Menu

The definitions of qualifiers may also be displayed on request from the user.

Once a prime word and its modifiers have been selected, a class word must be chosen. The system presents the user with a menus of class words to choose from as shown in Figure 5-3.

The user may select a class word directly or may elect to see the definitions of class words before selecting one. It should be noted here that at any point in this process, the user is free to browse through existing schemas and existing prime word and class word hierarchies stored in IRDS. This may aid the user in selecting appropriate prime words and class words by investigating other examples of their use.

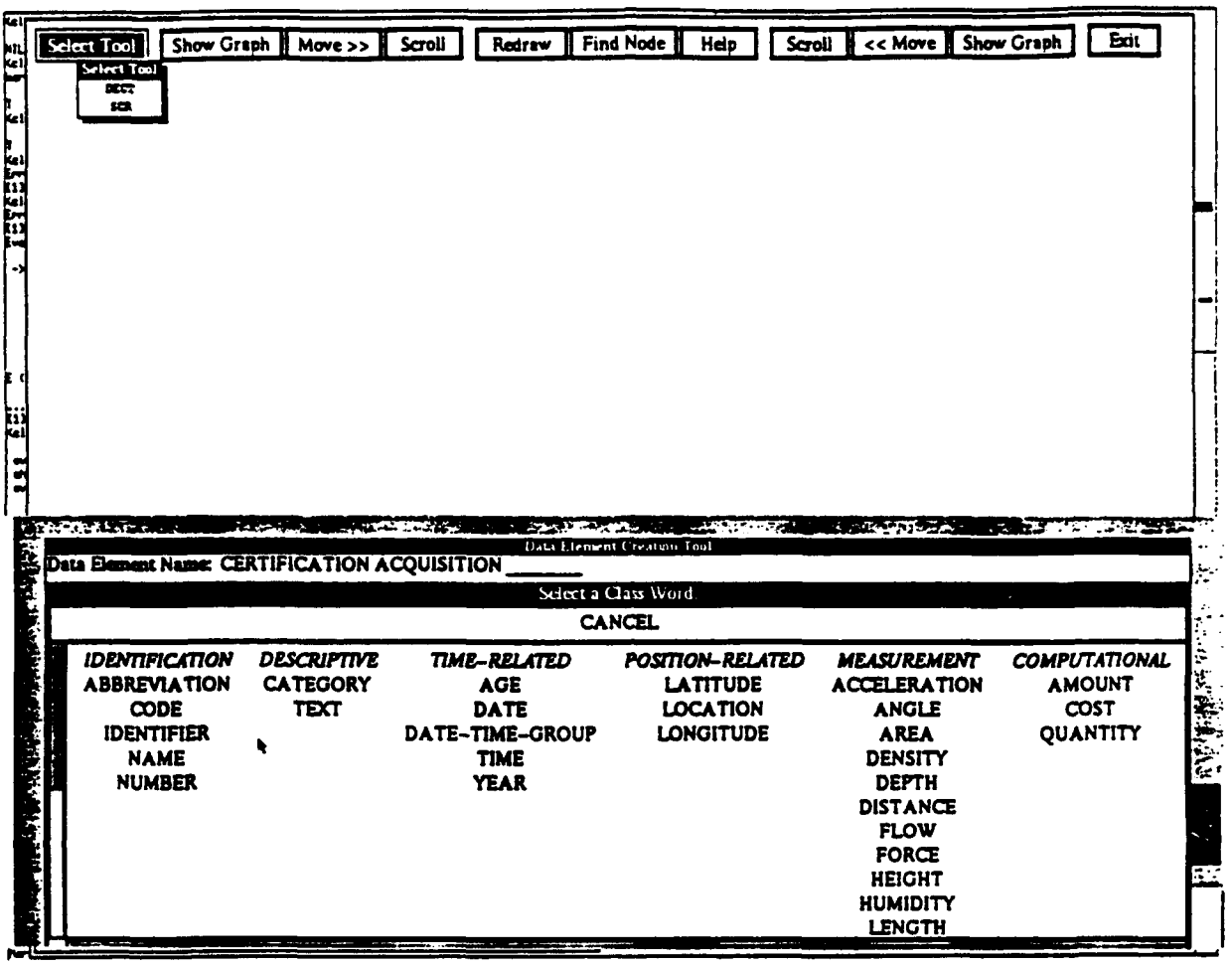


Figure 5-3. Class Word Menu

The user also selects appropriate class modifiers from a fixed list as shown in Figure 5-4.

It should be noted here that the optional qualifiers for a class word have not yet been included in the DECT. They will be added in the next version.

Once the user has assembled a complete data element name, the data element name is classified within the existing structure of data element names available to the DECT. The classification process will aid the user in determining whether or not a similar standard data element already exists the standard data elements known to the DECT. If the DECT finds that the data element may be classified in more than one way it engages the user in a series of questions to determine if modifications are necessary the the standard data element. The content of a sample dialogue is represented below in Figure 5-5.

The dialogue shown in Figure 5-6 represents the series of questions the user is asked in classifying the data element name with respect to the prime word. A different series of questions will be asked to determine if the class word selection is correct. The DECT will offer the user the opportunity to change the standard data element if the classification of the name does not correspond to the intended meaning of the naming being created.

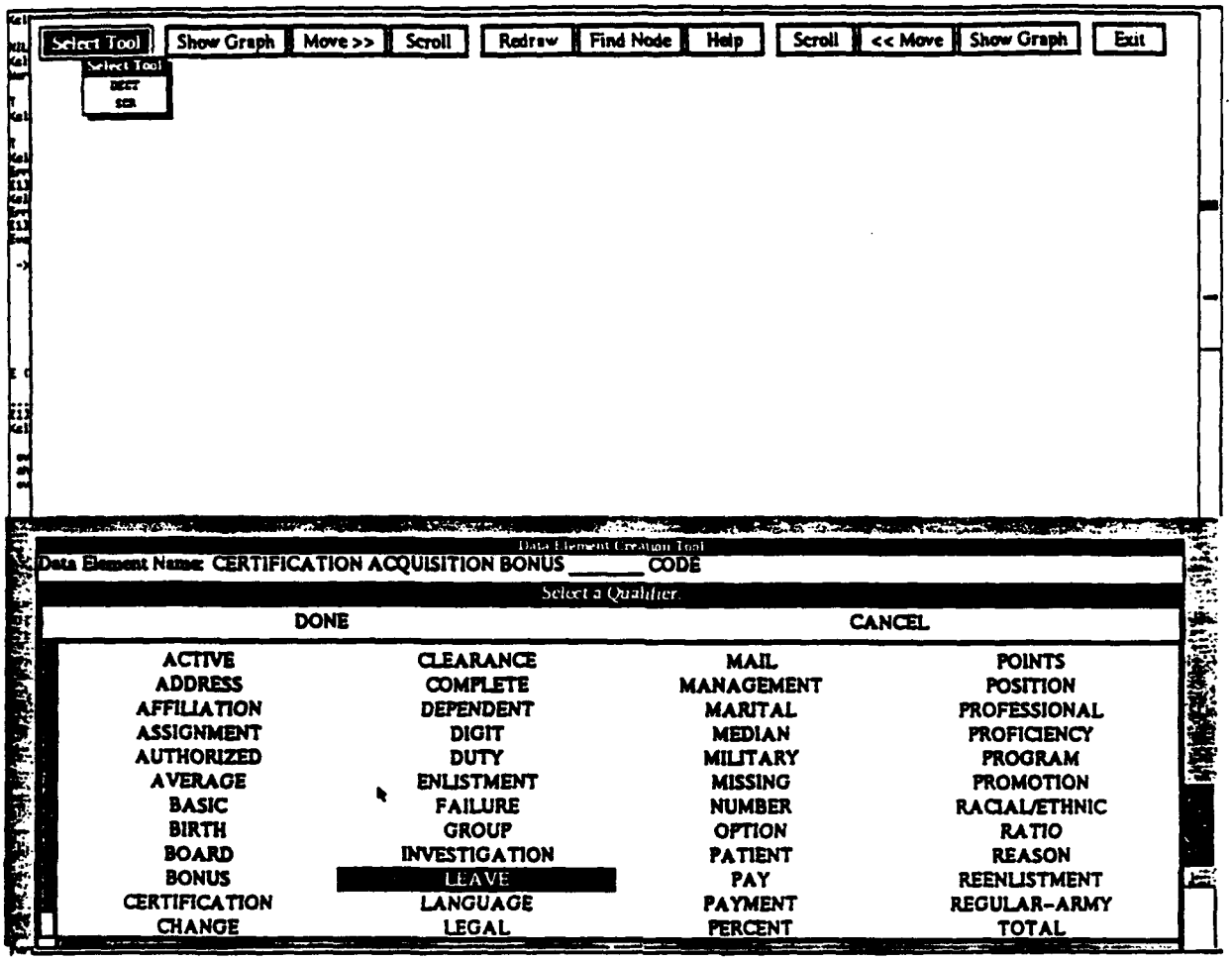


Figure 5-4. Class Word Modifiers

Once the new standard data element name has been created, it will be stored in the local data of the DECT. In the next revision of this tool prototype, the tool will be integrated with IRDS so that the the created data element names will be stored directly in IRDS.

The final step of the DECT is to generate a definition for the new data element name. The definition for the new name may be viewed as shown in Figure 5-6.

The data element definition generation algorithm is a modification of the algorithm originally used in the MITRE prototype. The Phase II version of the algorithm makes use of additional context-sensitive information associated with the modifiers to generate a more accurate defintion. Some sample definitions are given in Figure 5-7.

The DECT is still an initial prototype. Additional testing and modification of user functions can be done to make DECT a useful tool for Army data administration.

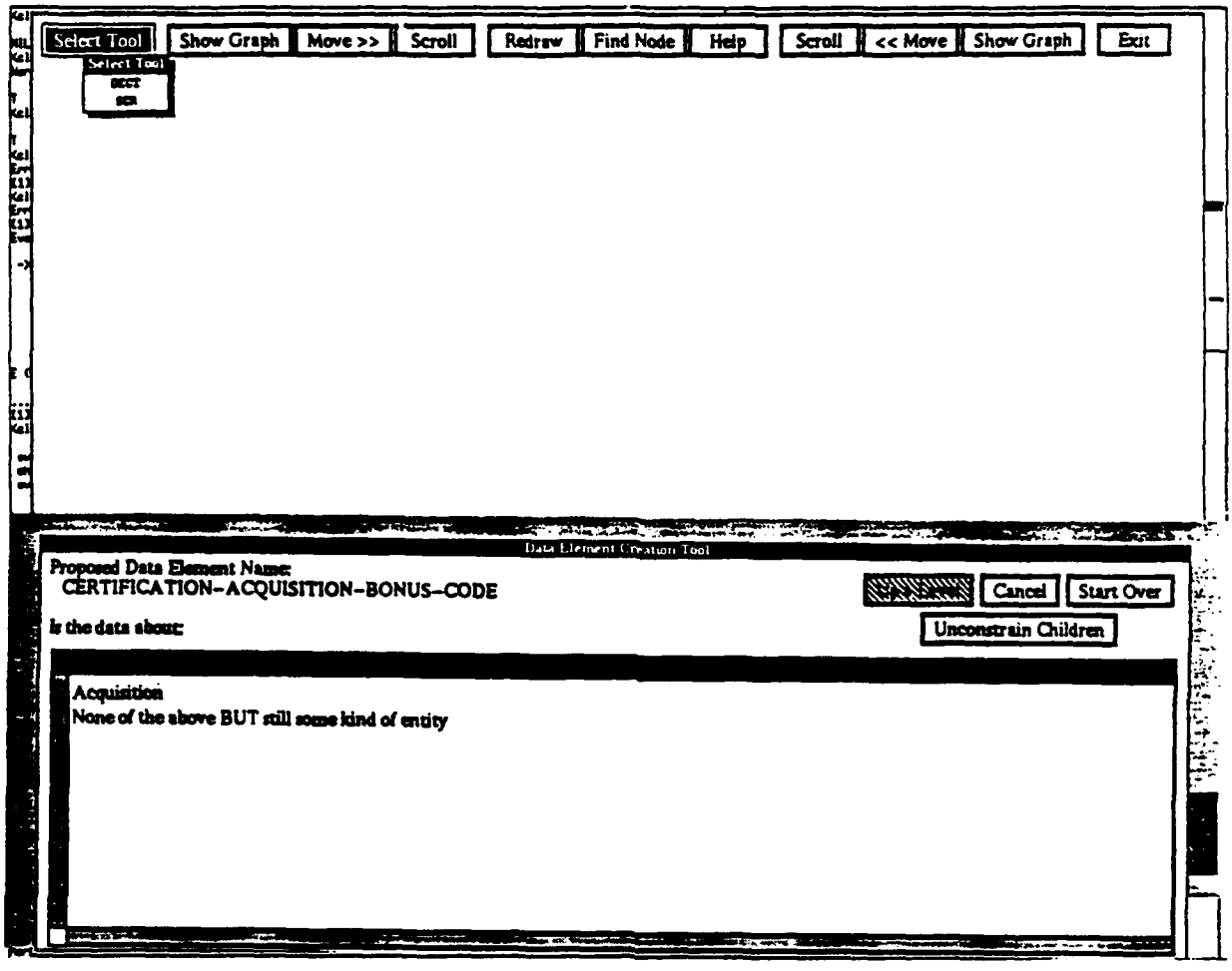


Figure 5-5. Sample Questions on Prime Word Classification

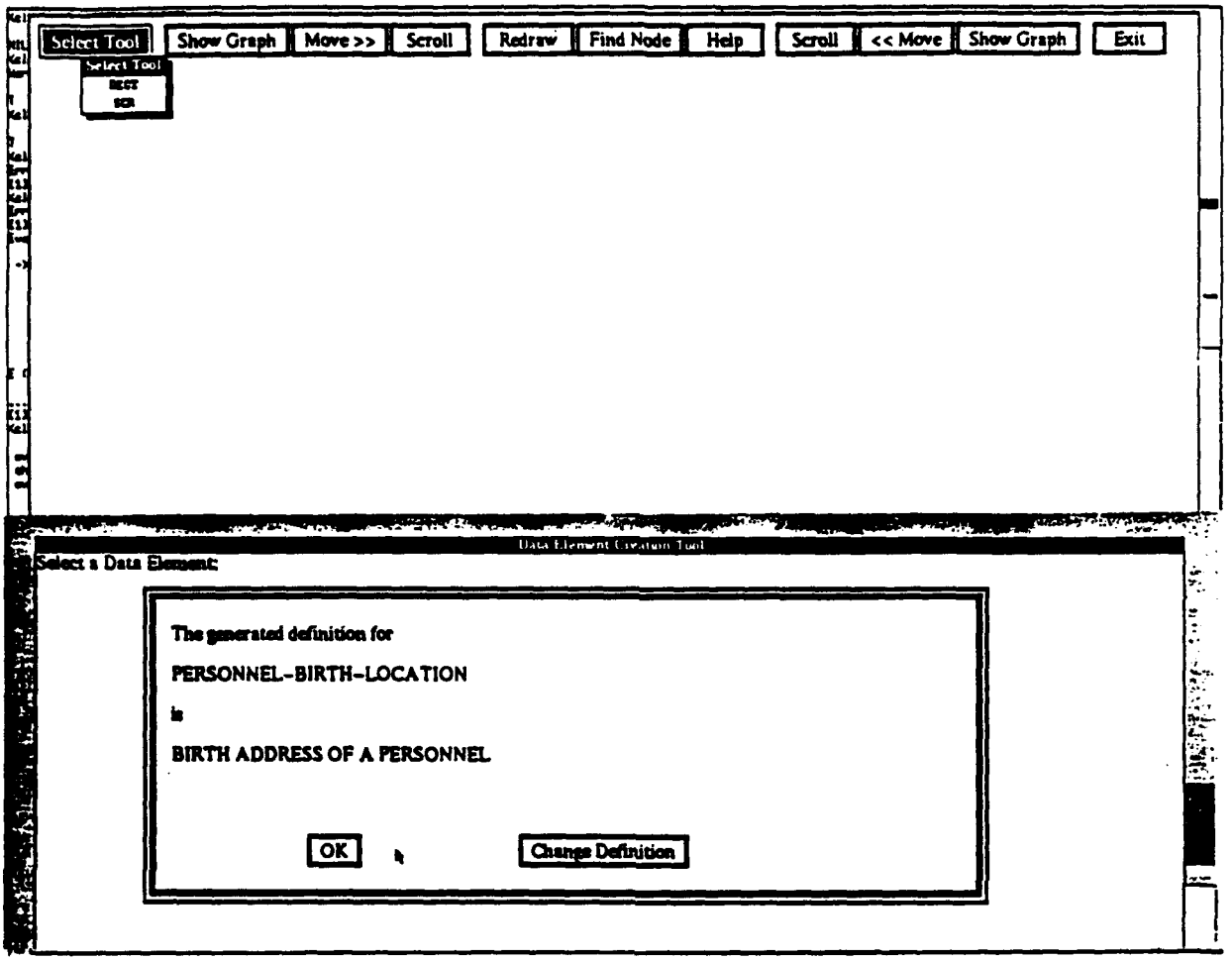


Figure 5-6. New Definition for Data Element

SDE: Personnel birth location
Definition: Birthplace of a person

SDE: Facility mail location
Definition: Mailing address of a facility

SDE: Acquisition divisional reenlistment quantity
Definition: Divisional reenlistment quantity of an acquisition

Figure 5-7. Sample Definitions Generated by DECT

Section 6
Implementation Instructions

6.1 Starting the ANSWER System

The ANSWER system must be invoked in an environment with Common LISP and X-windows running. This document describes only the Common LISP and X-window system features that are related to the ANSWER. Please refer to appropriate manuals for further information regarding the X-window system and the Allegro Common LISP.

The X-windowssystem is started by typing *startx* on the console. The console must be in VT100 mode (i.e., non-SUNTOOLS mode). The X-windows system starts an xterm and an EMACS editor. The EMACS editor is GNU 18.50 with the Allegro Franz Common LISP hooks. The LISP is started by typing:

M-x fi:explicit-common-lisp

At this point the editor asks for an image name. Type *xcw* as the image name and *nil* as the parameter for the image. Once the LISP is started, change the directory to isi by typing:

:cd isi

6.1.1 Starting the ANSWER User Interface Manager

To start the user interface manager and the ANSWER, system execute the following steps (these instructions assume that X-window and LISP are running):

- Change directory by typing:

  cd ~/shetti/answer/scr

- Load the ANSWER files. All the ANSWER files are loaded by loading the master file build.cl. To load this file, type:

  :ld build.cl

  This in turn will load the following files: cw-imp, isi-grapher, paint, irds-parser, control, top-level, gwrite, edit-node, fielded, commands, main, eqv and scr.

- To change the package to answer, type *the* LISP command:

  (in-package 'answer)

- To start the system, type:

  start-browser

6.2 Overview of the Implementation

The code for the answer system is organized in several directories. These are currently organized according to the directories of the developers. The user interface manager is organized under **/usr/what/shetti/answer**. The schema integrator tools are organized under **/usr/what/oracle/Answer**. The Data Element Creation Tools (DECT) is organized under **/usr/olawsky/ans**.

6.2.1 User Interface Manager

The user interface manager is organized under the directory **/usr/what/shetti/answer**. That directory in turn has three main directories: isi, scr and scripts. The code is also organized into several packages. For a complete description of packages, see the Allegro Common LISP manual. The basic idea behind packages is to manage the name space of functions. The isi directory contains code in the isi package. The major portion of the code is in the answer package. The contents of each directory are discussed below:

- **isi:** The isi directory contains the graphics-related code. All the files in this directory are in the isi package. The main code is in the answer package. The answer package uses the isi package. The following files are in the isi directory:
  - **cw-imp:** This file has common windows bindings to the ISI grapher's graphical function calls, such as make window, move, resize windows and draw lines, boxes and texts, etc.
  - **isi-grapher:** This is the heart of the browser. It contains functions for laying out arbitrary graphs and a means of associating commands to the nodes in the graph. The main function of this file is create-graph, which, as its argument takes a set of root nodes and a children function, returns children of a node.
  - **paint:** This file has node paint function customizations, such as boxed node, nodes with semicircular endings, nodes with triangular endings.
- **scr:** The scr directory contains the user interface manager code. All the files are in the answer package.
  - **globals:** As its name indicates, this file contains a set of global variables used by the user interface manager.
  - **irds-parser:** This file contains the definition of the basic data structures used by the answer to store local schemas as well as prime terms and data elements. The schema read from the IRDS are in the form of the communication buffer structures described in Appendix B of the final report. The IRDS parser converts these buffers into LISP data structures.

- **control:** This file has the definition of an event structure, and the definition for the event handler function. The event handler function waits for the certain types of event to occur, and takes appropriate action when the events are signalled.

- **top-level:** This file contains the function that sets up the main window, the top-level mouse handler loop,and the clean-up function that gets rid of the windows when the answer system exits.

- **gwrite:** This file has a function to convert a schema represented in the local LISP data structures to the communication buffer format.

- **fielded:** The fielded file has the underlying code for constructing a node editor.

- **edit-node:** This file has specific instances of the node editor, such as attribute node editor, relation node editor and general node editor for all other types of nodes.

- **commands:** This file has definitions for the numerous commands that are available from the node as well as the global command buttons.

- **build:** This file has load-file functions for all the files in the ANSWER system.

• scripts: The scripts directory contains the script definition files for the scr script. All files are in the answer package.

- **scr:** contains the actual scr script (schema integration script). The scr script is a single function with each state in the script defined as flet in the common LISP, and the main body of the function hosts the script driver. The scripts are given in Appendix A of the final report.

- **main:** This file has scr script supporting functions, such as functions that request from the user retrieve from IRDS the schemas to be integrated and definitions for the dialog boxes used by the scr script.

- **eqv:** also has scr script supporting functions. Some of the functions in the file are as follows: (1) store all the equivalence attribute and convert them to the form required by the SCR; and (2) create stay-up menu and handle events such as button selection, menu item selection, etc.

6.2.2 Data Element Creation Tool

The data element creation tool (DECT) code is in the directory **/usr/olawsky/ans**.

There are three primary components of DECT: data element entry, the classifier and the definition generator. DECT also makes heavy use of several special utilities—stay-up menus, notification boxes and buttons.

The files that contain most of the DECT code are dect.cl, decreate.cl, class-words.cl, prime-words.cl, quals.cl, cw-hier.cl and pw-hier.cl. The DECT stores new data elements only in its local classfication hierarchies.

As the tool is run, newly created elements are stored. If a demonstration of the tool is to be given with an empty set of data element names, the following files must be reloaded: class-words.cl, prime-words.cl, quals.cl, cw-hier.cl and pw-hier.cl. The global variable *data-elements* must also be set to nil.

- **decreate.cl:** This file contains the functions that drive the data element creation tool. The function defined in this file include:
  - **name-DE:** a function that controls the collection of a prime word, its modifiers, a class word, and its modifiers, all in the correct order. The function calls the appropriate interaction objects (menus of prime words, class words and modifiers with mouse-selectable items) and contructs a name.
  - **ask-question:** This function controls the asking of questions about a data element name as it is classified against the prime word and class word heirarchies.
  - **DE-classify-init:** This function controls the overall classification process tor against the prime word and class word hierarchies.
  - **generate-definition:** This function generates a definition by calling other functions to identify synonyms in the appropriate context for prime words, class words and modifiers.
  - **eval-conspec:** This function supports the context-sensitive specification of the synonym definitions.
- **dect.cl:** This file contains the functions that define the script used for the DECT by the user interface manager. It calls many of the other functions defined in this directory.
- **class-words.cl:** This file includes functions that define the list of class words used by the DECT. Synonyms for class words are also defined here.
- **prime-words.cl:** This file includes functions which define the list of prime words used by DECT. Synonyms for prime words are also defined here.
- **quals.cl:** This file contains the functions that define the qualifiers used in data element creation. The same list of qualifiers is used for prime words and class words. Synonyms for modifiers are also defined here.

- **cw-hier.cl and pw-hier.cl:** These files include functions that define the prime word and class word hierarchies against which the newly defined data elements are classified.

Other files contain the functions which define the interaction objects and functions used to build the interaction objects for the DECT. Some functions are also used by the user interface manager:

- **dbuttons.cl, notify.cl, scrollbar.cl, stay-up-menu.cl:** These files contain functions that define the building blocks of interaction objects used by the DECT. Some of these objects are also used by the user interface manager. The objects include a variety of buttons, notification boxes, scroll bars and a stay-up-menu (i.e., a menu that persists on the screen even if the mouse is not positioned on the menu).

- **dglobals.cl:** This file contains functions that define the interaction objects used by the DECT.

6.2.3 Schema Integrator

The schema integrator code is in the **user/what/oracle/Answer** directory. There are six subdirectories under the Answer directory. Each directory contains routines that service user interface requests via messages.

- **Extract_History Directory:** The user interface sends a request to get dictionary information. The extract_history routines will get dictionary names, schema names and schema history information stored in the Information Resource Dictionary System (IRDS). Information such as the names of schemas in the dictionary is returned. In addition, history information for the integrated schema, such as which schemas made up the integrated schema, is returned.

- **Browse_Schema Directory:** The user interface sends a request to get the "existing source schema" stored in IRDS so the user can browse it. The browse_schema routines will get the requested schema information from the specified schema and return the information to the user interface.

  The returned information includes object name, object type, prime word, attribute name, attribute type and standard data element name.

- **Store_New_Schema Directory:** The user interface sends a new schema information message that includes schema name, dictionary_name to store the new schema, object name, object type, prime word, attribute name, attribute type, standard data element name, etc., to the store_new_schema routines. The store_new_schema routines get the new schema information, create AF, OF, RF, and DF four files and IRDS command script, and fork a process, putting the new schema into IRDS.

- **Integrate_Schema Directory:** The user interface sends a request with two source schema names that need to be integrated. There are four subdirectories under this directory: Sub_Pre_Assertions, Sub_Assertions, Sub_Integrate, and Sub_IRDS_Make.

  - **Sub_Pre_Assertions Directory:** The Sub_Pre_Assertions routines extract two user-specified source schemas and create AF, OF, DF and RF files for use by the schema integrator tool.

  - **Sub_Assertions Directory:** The Sub_Assertions routines get E/C or REL object equivalent attributes information from the user interface and send a list of E/C or REL object pairs related to the E/C or REL equivalent attributes to the user interface.

  - **Sub_Integrate Directory:** The Sub_Integrate routines get E/C or REL object pair assertions from the user interface and generate an integrated schema to send to the user inteface.

  - **Sub_IRDS_Make Directory:** The Sub_IRDS Make routines get the new integrated schema name and the decision to commit the new integrated schema into IRDS from the user. The routines will generate an IRDS command script for integrated schema and fork a process to store the integrated schema into IRDS at the user's request.

- **Common_Block Directory:** The Common_Block Directory has the routines shared by the above four directories. There are two subdirectories under the Common_Block directory.

  - **C Source Code Directory:** This subdirectory three subdirectories: Communications, Integration and Utilities. The routines in these directories are all written in C.

    The Communication directory has all UNIX socket communication routines.

    The Integration directory has the routines shared by other schema integator routines. These shared routines include read/write file, IRDS command script, pack schema and other miscellaneous routines.

    The Utilities directory includes routines that perform I/O and structure mapping as well as header files for file formats.

  - **Lisp Directory:** All routines used by the user interface's socket communication portion are in this directory. All routines are written in LISP.

- **IRDS Directory:** There are two files under this subdirectory: IRDS executable code and signal file. The signal file is used to indicate if any process is using IRDS. The signal file exists when no processes are using IRDS.

If the schema integrator tool does not run properly, three things need to be checked before re-running the tool:

- Check the Oracle database:
  - Enter the database using irds/irds user name and password.
  - No view should exist. Drop any existing view.
- Check the Communication Result file. Remove the commresult file under the **user/what/oracle/comm** directory. If all the communications have been established, the commresult file will exist. If this file is left in the directory, you will not be able to re-run the schema integrator tool properly.
- Check the Guard file. Ensure that the signal file exists under the **user/what/oracle/IRDS directory**. Use the command *touch signal* to create the signal file. The signal file exists when no other processes are using IRDS. If it does not exist, ensure that there are no other processes using IRDS. Otherwise, the schema integrator tool will not run properly.

DEPARTMENT OF THE ARMY
AIRMICS
115 O'KEEFE BUILDING
GEORGIA INSTITUTE OF TECHNOLOGY
ATLANTA, GA 30332-0800

OFFICAL BUSINESS
PENALTY FOR PRIVATE USE, $300

THIRD CLASS