A RELATIONAL DATA DICTIONARY
COMPATIBLE WITH THE
NATIONAL BUREAU OF STANDARDS
INFORMATION RESOURCE DICTIONARY SYSTEM

by

Robert A. Kirsch II

December 1985

Thesis Advisor: Daniel R. Dolk

Approved for public release; distribution is unlimited
**ABSTRACT**

Data is a very valuable corporate asset. How it is managed and controlled can often determine the success or failure of a corporate venture. With this fact in mind many organizations are taking a close look at what tools are available to help them in this effort.

This thesis takes a look at two types of data management tools available today, the Relational Data Base Management System (RDBMS) and the Data Dictionary (DD). It discusses desirable DBMS and DD characteristics with particular attention being paid to the shortcomings of DDs. It also describes the effort of the National Bureau of Standards (NBS) to develop a DD standard and examines in detail the NBS Information Resource Dictionary System (IRDS) and how the standard was implemented in a prototype IRDS.
A Relational Data Dictionary
Compatible with the
National Bureau of Standards
Information Resource Dictionary System

by

Robert A. Kirsch II
Captain, United States Army
B.S., University of South Alabama, 1973

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN INFORMATION SYSTEMS

NAVAL POSTGRADUATE SCHOOL
December 1985

Author: Robert A. Kirsch II
Approved by: Daniel R. Dolk, Thesis Advisor
Paul W. Callahan, Second Reader
Will N. Greer, Jr., Chairman
Department of Administrative Sciences

Kneale T. Marshall
Dean of Information and Policy Sciences
ABSTRACT

Data is a very valuable corporate asset. How it is managed and controlled can often determine the success or failure of a corporate venture. With this fact in mind many organizations are taking a close look at what tools are available to help them in this effort.

This thesis takes a look at two types of data management tools available today, the Relational Data Base Management System (DBMS) and the Data Dictionary (DD). It discusses desirable DBMS and DD characteristics with particular attention being paid to the shortcomings of DDs. It also describes the effort of the National Bureau of Standards (NBS) to develop a DD standard and examines in detail the NBS Information Resource Dictionary System (IRDS) and how the standard was implemented in a prototype IRDS.

Page 28 is not missing but is misnumbered.
Per Ms. Elaine Evans, NPS/Code 014
TABLE OF CONTENTS

I. INTRODUCTION ........................................... 9
   A. BACKGROUND ........................................ 9
   B. OBJECTIVES ......................................... 10

II. DATA DICTIONARY FUNCTIONS AND CAPABILITIES ........ 13
   A. GENERAL ............................................. 13
   B. DBMS DATA DICTIONARY CAPABILITIES .............. 13
   C. DATA DICTIONARY CAPABILITIES ..................... 15
      1. Dictionary Schema ............................... 17
      2. User Dialogue .................................. 17
      3. Dictionary Commands ......................... 19
         a. Dictionary Maintenance Command ............ 19
         b. Report and Query Commands ............... 20
         c. Data Structure Interface Commands ...... 20
         d. Extensibility Commands .................... 20
         e. Status-Related Commands .................. 20
         f. Security Commands .......................... 20
         g. Dictionary Processing Control Commands .... 20
         h. Dictionary Administrator Commands ....... 20
      4. Extensibility .................................... 21
      5. Status Facilities ................................ 21
      6. Report Processor ................................ 21
      7. Query Processor ................................ 21
      8. Convert Function ................................. 21
      9. Software Interface ............................... 21
c. Output Syntax ........................................ 47

- Entity-lists ........................................ 47
- Procedures ........................................ 47

3. Schema Maintenance and Output .......................... 47
   - Schema Control .................................... 48
   - Schema Manipulation ............................... 50
   - Schema Output ................................... 51

4. The IRD to IRD Interface ................................. 52

5. IRDS Control Facilities .................................. 53
   - The Versioning Facility ......................... 53
   - The Life-cycle-phase Facility .................. 54
   - Quality-Indicators ............................... 56
   - Views ............................................ 58
   - Core Security ................................... 58

6. User Interfaces ........................................ 60
   - Command Language ................................ 60
   - Panel Interface .................................. 60
   - Operation on the Panel Interface ............... 62

7. IRDS Modules ........................................... 62
   - Entity Level Security ............................. 63
   - Application Program (call) Interface .......... 63
   - Support of Standard Data Models ............... 63

E. CONCLUSION ........................................... 63

IV. NBS IRDS PROTOTYPE ................................ 65
   - PROTOTYPING ..................................... 65

6
1. Advantages of Prototyping .......... 67
2. Disadvantages of Prototyping .......... 68
3. Types of Prototyping .......... 70
4. Reasons for Prototyping .......... 70

B. THE IRDS PROTOTYPE .......... 70
   1. Relational Model of the IRDS .......... 71
   2. Interface .......... 73

C. IRDS START-UP .......... 73

D. SECURITY .......... 76

E. POPULATING AND MAINTAINING THE DICTIONARY .......... 77
   1. Adding Entities .......... 77
   2. Modifying Entities .......... 81
   3. Deleting Entities .......... 84
   4. Adding Relationships .......... 84
   5. Modifying Relationships .......... 86
   6. Deleting Relationships .......... 87

F. THE DICTIONARY OUTPUT FACILITY .......... 87
   1. Entities .......... 87
   2. Relationships .......... 90
   3. Schema .......... 95

G. QUERY .......... 95

H. SCHEMA MAINTENANCE .......... 100
   1. Entity Meta-Data .......... 102
   2. Relationship Meta-Data .......... 103

I. FINAL COMMENTS .......... 104

V. CONCLUSION .......... 107
APPENDIX A: CORE STANDARD SCHEMA .................................. 109
APPENDIX B: COMMAND SPECIFICATIONS .............................. 119
APPENDIX C: BOCKMAN DIAGRAMS ...................................... 127
APPENDIX D: STANDARD DATA MODELS ................................. 127
APPENDIX E: PROTOTYPE SOFTWARE ................................... 133
LIST OF REFERENCES ...................................................... 250
INITIAL DISTRIBUTION LIST ............................................ 254
I. INTRODUCTION

A. BACKGROUND

In the corporate world data is a very valuable resource. Many organizations spend a great deal of time and corporate assets trying to control it. Data is used to facilitate the management decision process by providing the manager with timely, accurate and relevant information. Since the quality of the decisions made by today’s managers is so important, it is very critical that the corporate data resource be easy to access, as accurate as possible, and properly and effectively managed. [Ref. 1]

Concern over corporate information resources has resulted from the explosive growth in the size, complexity and number of data bases available to managers. This data base explosion has also ushered in the need for better tools to manage the corporate data base. A critical software tool that has been developed to control and manage data is the Data Base Management System (DBMS).

E. F. Codd has identified nine functions that the ideal DBMS should have (See Figure 1.1) [Ref. 2]. Kroenke states that DBMS products vary in the degree to which they provide these functions. Currently, no commercial DBMS provides all nine functions entirely satisfactorily. These functions are necessary and important, however, and this situation should change as DBMS products evolve and as new products are developed. [Ref. 3]

Of the nine functions listed in Figure 1.1, the one that is of particular interest to the Data Administrator (the individual who is responsible for the management of the data dictionary and for its effective use in the pursuit of data resource goals) is the function of providing a user-accessible catalog for data descriptions.
B. OBJECTIVES

The changes in today's end-user environment reflects the growth in computer literacy and increased need for data. Users are demanding increasingly better access to data via interactive processing, ad-hoc queries, specialized reports and simpler man-machine communication. At the same time there is growing concern over the timeliness, validity, and relevance, and usability of the data that is available.

As a result, there has been a growing interest in two tools which provide highly visible support for the information processing community-data dictionaries and relational data bases. Most relational data base products provide only rudimentary dictionary capabilities, "the offerings provide little more than a method of defining the schema." [Ref. 6] The relational data dictionary has become the link that connects the user/analyst with the DBMS. [Ref. 7]

The relational data dictionary, that is the data dictionary normally provided with a relational DBMS has additional weaknesses besides the ones mentioned above:

* They do not provide a full range of functions

* Their ability to interface with more than one DBMS is limited or non-existent

* There is a broad divergence concerning the scope of data dictionaries and until recently there has been no universally accepted standard [Ref. 8], [Ref. 9].

It is interesting to note that these problems apply to data dictionaries in general and not just to the relational variety. The purpose of this work is to create a prototype of a relational dictionary based on the
The usefulness of the catalog is greater if it contains not only data descriptions but also data about the relationship between programs and data, e.g., which programs access which data, and what they do with it. [Ref. 4]

1. Store, retrieve, and update data
2. Provide integrity services to enforce data constraints
3. Provide a user-accessible catalog of data descriptions
4. Control concurrent processing
5. Support logical transactions
6. Recover from failure
7. Provide security facilities
8. Interface with communications control programs
9. Provide utility services

Figure 1.1 DBMS Functions

The problem that arises is that some DBMSs have limitations on how well they maintain the meta-data (data that describes other data or data bases). Meta-data include descriptions of the meaning of data items, the ways in which the data are used: the sources of particular data elements: the physical characteristics and rules or restrictions on their forms or uses. When the meta-data deals strictly with where data stored in the DBMS it is referred to as a Data Directory but this capability is not enough. The Data Dictionary (DD) system is an expansion of the DBMS description cataloging capability. The Data Dictionary system is a key tool available to the Data Administrator for the management of meta-data and information resources. The DD provides facilities for recording, storing and processing descriptions of and organization's data and data processing resources. [Ref. 5]
specified standards for dictionaries recently developed by the National Bureau of Standards (NBS). Chapter 2 discusses dictionary concepts in general and reviews functionality of existing dictionary capabilities with special attention on relational systems. Chapter 3 discusses the features and capabilities which form the basis of the NBS draft proposal American National Standards (dp ANSI) Information Resource Dictionary System (IRDS). Chapter 4 outlines and discusses the IRDS features that were selected for inclusion in the relational dictionary prototype and how those features were actually implemented.
II. DATA DICTIONARY FUNCTIONS AND CAPABILITIES

A. GENERAL

The Data Dictionary (DD) is of great importance to the DBMS administrator and user because it allows the administrator to control how data and data bases are described and structured and it provides the link that connects the user to the DBMS. A data dictionary is a repository of data about data and processes associated with a particular system or organization.

B. DBMS DATA DICTIONARY CAPABILITIES

The data stored in a DBMS data base may be organized along hierarchical, network or relational lines. This organizational capability also exists for the data in the Data Dictionary, which in most cases is actually data stored in the DBMS itself. Data dictionaries implemented in this fashion are most often referred to as a DATA DIRECTORY (how the data is stored in the data base). On the other hand the implementation of a data dictionary can be on such a scale that it incorporates all of the data resources available to an organization. An implementation such as this is often referred to as INFORMATION RESOURCE MANAGEMENT. [Ref. 10] This thesis is most concerned with data dictionaries of the information resource management type.

The DBMS acts as a librarian for the data base, storing and retrieving data according to a particular format [Ref. 11]. However, a DBMS does not necessarily provide for the security, integrity, accountability, or maintainability of that data. These objectives are best achieved when a data dictionary is used in conjunction with the DBMS [Ref 12].
A DD is an instrument for describing an organization's meta-data. Meta-data refers to that data which describes other data or data bases and includes descriptions of the meaning of data items, the ways in which the data are used; the sources of particular data elements; the physical characteristics; and rules or restrictions on their forms or uses [Ref. 13].

There are additional capabilities that should be made available to the DBMS user as part of the data dictionary [Ref. 14]:

1. Retrieval and analysis capabilities which assist the user in application development.
2. The ability to generate pre-defined, customized and user defined reports via some type of report writer.
3. The ability to extend the data dictionary as necessary to meet the DBMS user's unique needs.
4. Data management tools that are intended to ensure the security, validity, recoverability and integrity of the data dictionary system and its associated data bases.
5. Software interfaces that allow other software modules to access the data base via the dictionary and the capabilities of translating the meta-data into file definitions usable by the software.

M. T. Vanecek described the capabilities listed above as those most important from a DBMS auditor's standpoint but it is easy to see that they could apply to many types of users. [Ref. 15: pp. 15-16]

P. P. Uhrowczik describes the capabilities listed above as being derived from the "management use mode." He goes on to identify additional DD capabilities that should be available to the DBMS user in what he calls the "computer use mode" [Ref. 16: pp. 332-334]:

1. Data Mapping. Where the user is no longer concerned with what is sometimes called the "physical-equal-logical" environment. This is accomplished by removing the awareness of where data is stored and giving it to the DD.
2. Data Conversion. During the mapping process, data can be converted to a different format. For instance, data physically stored as
character can be retrieved and converted to decimal.

3. Data compaction. Data can be stored in a compacted form (encoded), but presented to the user in a more meaningful format (decoded).

4. Input and output validation. Data entering a program (input) or data entering physical storage (update) can be checked against pre-established editing standards. For example, data can have a specified format, and lie within a specified range of values.

5. Test-data generation. System-generated test data with characteristics as described in the DD can be presented to the user.

6. Logical record and file definitions. A user is generally interested in processing only certain data elements forming a logical record and desires that these logical records be presented to him in a certain sequence. In Figure 2.1 the user defines his logical record as a series of element names and states his desire to process the file sequentially in a DEPT/MANNO sort sequence. The fact that the file comes physically from two different data sets is pre-defined in the DD/D. Thus the system can deliver the logical records properly assembled in the requested sequence. The user and the program do not need to know about the two data sets that are required to produce the view.

7. JCL Generation. Job Control Language (JCL) statements for physical data sets can be automatically generated as required by the particular operating system in use. This not only eliminates the user's pre-occupation with JCL, but also facilitates migration to different operating systems.

8. Access to distributed data bases. Data bases or portions of data bases may be physically stored in different locations on different computers, linked via data communication facilities. The data directory located with each distributed data base would describe the physical data located at that site, as well as, physical data located at other sites. The DBMS can decide based on the information provided by the DD whether to satisfy the request locally or from a remote location.

C. DATA DICTIONARY CAPABILITIES

The capabilities listed above describe the data dictionary capabilities that should be available to any DBMS user. However this view of the desired data dictionary capabilities is limited, since it perceives the data dictionary as an extension of the DBMS itself and not as a true data dictionary.

It is possible, on the other hand, to view the data
USER REQUEST (PROBLEM PROGRAM)

USER ID B7601

RETRIEVE FILE SEQUENTIALLY IN SEQUENCE BY DEPT/MANNO

LOGICAL RECORD DEFINITION

- DEPT
- MANNO
- SALARY
- NAME
- MANAGER
- YRS OF SERVICE
- JOB TITLE
- LAST APPRAISAL

READ ONLY
READ ONLY

AVAILABLE DATA SETS

<table>
<thead>
<tr>
<th>PAYROLL DATA SET</th>
<th>PERSONNEL DATA SET</th>
</tr>
</thead>
<tbody>
<tr>
<td>STORED RECORD DEFINITION</td>
<td>STORED RECORD DEFINITION</td>
</tr>
<tr>
<td>MANNO (KEY)</td>
<td>MANNO (KEY)</td>
</tr>
<tr>
<td>NAME</td>
<td>NAME</td>
</tr>
<tr>
<td>SALARY</td>
<td>DEPT</td>
</tr>
<tr>
<td>YTD GROSS</td>
<td>JOB TITLE</td>
</tr>
<tr>
<td>DEDUCTIONS</td>
<td>HIRE DATE</td>
</tr>
<tr>
<td>BOTH DATA</td>
<td>SETS IS ON</td>
</tr>
<tr>
<td>MANNO</td>
<td>MANAGER</td>
</tr>
</tbody>
</table>

Figure 2.1 Logical Record and File Definitions
dictionary as an entity unto itself whether it is from standing or DBMS dependent. The capabilities and functions shown in Figure 2.2 and described below represent a joining of the capabilities and functions described by Allen et al [Ref. 17: pp. 248-253] and Lefkovits et al [Ref. 18: pp. 2-7 thru 2-29].

1. **Dictionary Schema**

Denotes the structure of the dictionary. Both sources agree that, at a minimum, a DD should allow for the definition of Entities, Relationships and Attributes. Entities are the basic unit of the dictionary and represent real world objects or things about which certain information exists in the dictionary. Relationships provide information about associations between entities whereas attributes provide information about entities and relationships that exist in the dictionary. Figures 2.3 and 2.4 show examples of commonly used entities and relationships.

Data Dictionary Maintenance

<table>
<thead>
<tr>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity-types</td>
</tr>
<tr>
<td>Attribute-types</td>
</tr>
<tr>
<td>Relationship-types</td>
</tr>
</tbody>
</table>

User Dialogue

Dictionary Commands

Extensibility

Status Facilities

Report Processor

Query Processor

Convert Function

Software Interface

Data Management

Security

Integrity

Concurrent Control

Internal access to DD

Figure 2.2 Data Dictionary Capabilities

Entity names should be unique but facilities to track duplicate names in the form of aliases or synonyms should be provided. Additionally,
Figure 2.3 Logical Structure of a Typical DD
(from Allen et al)

Figure 2.4 Hierarchy of Entity-types
(from Lefkovits et al)
the DD should allow a minimum of three groupings of entity-types: Data-Element, Processes, and Usage.

The dictionary system should also provide a means of grouping together dictionary elements that have the same characteristics. This can be accomplished through the establishment of Entity-types, Relationship-types, and Attribute-types. It can also be accomplished through the establishment of a Key-word In-context feature. Neither author provided specific examples of attribute or relationship types. Both did agree that most DD provide enough attribute-types and relationship-types to meet the average user needs. In addition they identified the extensibility feature which would allow a DD user to expand the DD to meet his individual requirements.

2. User Dialogue

The method used by the DD to communicate with the user and vice versa.

a. keyword-driven language
b. position-sensitive transactions
c. interactive, prompted input
d. interactive, performatted screens or menus

3. Dictionary Commands

Provide user with the ability to use the DD system to its fullest extent. Dictionary commands can be divided to the following categories.

a. Dictionary Maintenance Commands

Those commands that allow entities, relationships, and attributes to be created, modified and deleted from the dictionary.
b. Report and Query Commands

Those commands that allow the user to request the system to generate listings of entities, relationships and attributes and generate queries on such things as the usage of dictionary entities, keyword and synonym searches.

c. Data Structure Interface Commands

These commands give the DD system the ability to generate descriptions of data structures in such a way that they can be processed by other language processors, such as language compilers or DBMS schema/subschema utilities.

d. Extensibility Commands

These commands are discussed in 4 below.

e. Status-related Commands

Will be discussed in 5 below.

f. Security Commands

These commands provide the system with the ability to exclude some users from access to the system or restrict his ability to modify and change the system.

h. Dictionary Processing Control Commands

These commands allow the user to perform such functions as log-on, log-off, terminate operation upon error, etc.

h. Dictionary Administrator Commands

These commands will allow the dictionary administrator to:

* initially create the dictionary system
* recover the dictionary after a failure
* set default values
create back-up copies of the dictionary

4. **Extensibility**

A feature that allows the DD structure to be extended by definition of additional entities, relationships, and attributes.

5. **Status Facilities**

Allows the dictionary system to be used in a System Life Cycle environment, that is the system would allow for the designation of an entity as being "Under Development," "Production" or "Archive" for example.

6. **Report Processor**

This capability allows the user of the DD system to produce predefined reports, the ability to customize reports and produce user-defined reports.

7. **Query Processor**

This capability would give the DD user the ability to generate English-like queries of the system. This query capability is analogous to the corresponding function in DBMSs for access to data bases.

8. **Convert Function**

This function allows the DD system to read application programs, libraries and schemata and generate DD maintenance input transactions to automatically create a DD schema.

9. **Software Interface**

This capability provides a formatted pathway, enabling the DD system to provide meta-data to other software systems such as compilers.

10. **Data Management**

This function would provide for the data base management tasks such as:
* Security
* Integrity
* Concurrent control
* Internal access of the DD

Not all data dictionary systems possess the capabilities listed above in fact, early data dictionary systems were little more than document generators, taking the meta-data that had been stored in them and printing out reports describing file and record structures. Other DD which are DBMS-dependent obtain the capabilities listed above from the DBMS they are associated with. Unfortunately even DBMS products that are currently being marketed are limited in the data dictionary capabilities they offer and very few if any offer what could be classified as information resource dictionary systems. In addition to the limited DD capabilities associated with DBMS, the additional problem of lack of standardization exists.

D. ADVANTAGES OF DATA DICTIONARIES

The main advantage of a dictionary lies not in its ability to store and catalogue information about data, but in its ability to assist in the discipline of data design [Ref. 19].

This advantage can be expanded into a number of beneficial areas:

1. Information about data/corporate asset. Accurate information about how a company functions, about its employees and clients can be stored in a DBMS and defined in a data dictionary. By storing this knowledge on a magnetic media and providing for adequate backup and recovery to the data dictionary, the corporate asset is being saved from catastrophe.

2. Public vs. Private Information. The situation where only a programmer knows all of the information (institutional knowledge) about a particular application, can cause many problems not only for those who must pick up a project in mid-stream, but even for the programmer himself if it has been several months since he last worked
on the application. By incorporating his institutional knowledge about each application into a data dictionary as each new application is developed, the information becomes public knowledge for the application developer and anyone who follows him. This will substantially reduce the effort required to modify and enhance existing applications.

3. Communication tool. The data dictionary can become a repository of corporate information, i.e., minutes of meetings, memos, notes, manuals and reference texts, which can be accessed by all areas of a company. The central area of Figure 2.5 represents the communications value of a data dictionary.

Figure 2.5 Communication Value of a Data Dictionary

4. Safeguard against Data Redundancy. Old systems are difficult to maintain because of lack of information, process redundancy and data redundancy. Information availability has been discussed above. Process redundancy can be reduced through structured programming techniques. Data redundancy however requires a different approach. Data redundancy is a situation where the same data element proliferates throughout the system.

It is not uncommon in an older system to find the same data element stored in ten different locations and requiring ten different update transactions to maintain it. This same data element may be referenced by 50 different names through the system. Is it any wonder that such systems are difficult to maintain. [Ref. 20]

Listed below are various types of data redundancy:
a. Reference Redundancy - when the same data element is referenced by different names.

b. Format Redundancy - when the same data element appears in the system in different formats.

c. Group Redundancy - when data elements are grouped under a group name when no requirement exists from them in the first place.

d. Occurrence Redundancy - when repetitious data names are used to identify multiple generations of the same data element.

e. Definition Redundancy - when a data element is used for more than one purpose thus the element has more than one definition.

f. Storage Redundancy - when the same data element is stored in more than one location (redundancy of this type, sometimes serves a purpose, in distributed systems for example).

These as well as other types of redundancy not mentioned can be controlled through the use of a data dictionary.

5. Glossary of Terms. Another benefit of implementing a data dictionary is to use it as a glossary of terms. Which could be used in the development of software and as a training tool.

The data dictionary can be very effective when used as a tool to support structured analysis and design. It can be used to document data store, data flow, and process entity types. The data dictionary can also be used to generate, file segment, and record definitions for a variety of programming languages. By doing so, we can centralize the control of program data definitions. [Ref. 21]

6. Documentation. The data dictionary can serve as an effective medium for the presentation of documentation. The nature of a data dictionary makes maintenance of documentation easier and anyone who has access to a computer terminal can subsequently access the documentation.

7. System development. The "data dictionary is one more tool to increase user effectiveness in system development." [Ref. 22] the traditional approach to systems development (see Figure 2.6) can be enhanced to allow all involved in the development process, access to the necessary information as it is generated. (see Figure 2.7) This is accomplished by incorporating the DD into the traditional development network.

All of the capabilities and benefits listed above are important, but very few if any data dictionary systems available today can provide them all. In other words there is no current standard from which all data
Figure 2.6 Systems Development Traditional Approach

Figure 2.7 Expanded System Development Approach

25
dictionary products are developed. This situation is in the process of being eliminated now that the National Bureau of Standards (NBS) has formalized and published a standard for data dictionaries in the form of the Information Resource Dictionary System (IRDS) standard. The feature and functions found in that standard are discussed in the next chapter.

E. EXISTING DBMS DATA DICTIONARY CAPABILITIES

As stated earlier, very few DBMSs contain DDs that exhibit all the capabilities discussed above, and even fewer Relational DBMSs offer the previously identified minimum DBMS dictionaries capabilities. Tables 2.1 thru 2.3 list the DD capabilities provided by the INGRES and ORACLE DBMS products. It is easy to see from the list above that the dictionary capabilities provided by ORACLE and INGRES are very limited from the standpoint of offering full data dictionary capabilities.

But what alternatives exist to improve this situation? The NBS IRDS standards offers a convenient vehicle to improve this situation. By adopting this standard as an industry-wide starting point, all products that use data dictionaries and the data dictionary itself will improve. The next chapter discusses the NBS IRDS standard in detail.
TABLE 2.1
DBMS DICTIONARY CAPABILITIES

<table>
<thead>
<tr>
<th>CAPABILITY</th>
<th>INGRES</th>
<th>ORACLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Dictionary Maintenance</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>Schema</td>
<td>****</td>
<td>****</td>
</tr>
<tr>
<td>Entity-types</td>
<td>L</td>
<td>A</td>
</tr>
<tr>
<td>Attribute-types</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>Relationship-types</td>
<td>L</td>
<td>A</td>
</tr>
<tr>
<td>User Dialogue</td>
<td>****</td>
<td>****</td>
</tr>
<tr>
<td>Keyword-drive</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>Position-sensitive Trans</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Interactive</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>Prompted input</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>Preformatted screen</td>
<td>A</td>
<td>L</td>
</tr>
<tr>
<td>Menus</td>
<td>L</td>
<td>L</td>
</tr>
</tbody>
</table>

Facility/capability availability = A
Facility/capability available but limited = L
Facility/capability not available = N
Facility/capability as part of DBMS only = F
## TABLE 2.2

**DBMS DICTIONARY CAPABILITIES**

<table>
<thead>
<tr>
<th>CAPABILITY</th>
<th>INCRES</th>
<th>ORACLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dictionary Commands</td>
<td>****</td>
<td>****</td>
</tr>
<tr>
<td>Maintenance</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>Add</td>
<td>N</td>
<td>L</td>
</tr>
<tr>
<td>Modify</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>delete</td>
<td>N</td>
<td>L</td>
</tr>
<tr>
<td>Report</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>Query</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>Data Structure interface</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Extensibility</td>
<td>N</td>
<td>P</td>
</tr>
<tr>
<td>Status-related</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Security</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>Processing control</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>

Facility/capability availability = A  
Facility/capability available but limited = L  
Facility/capability not available = N  
Facility/capability as part of DBMS only = P
<table>
<thead>
<tr>
<th>CAPABILITY</th>
<th>INGRES</th>
<th>ORACLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administration</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>Extensibility</td>
<td>N</td>
<td>P/L</td>
</tr>
<tr>
<td>Status Facilities</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Report Processor</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>Query Processor</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>Convert Function</td>
<td>A</td>
<td>N</td>
</tr>
<tr>
<td>Software Interface</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>Data Management</td>
<td>L</td>
<td>P/L</td>
</tr>
<tr>
<td>Security</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>Integrity</td>
<td>N</td>
<td>P</td>
</tr>
<tr>
<td>Concurrent Control</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Internal access to DD</td>
<td>N</td>
<td>L</td>
</tr>
</tbody>
</table>

Facility/capability availability = A
Facility/capability available but limited = L
Facility/capability not available = N
Facility/capability as part of DBMS only = P
III. INFORMATION RESOURCE DICTIONARY SYSTEM

This chapter discusses the features and characteristics which form the basis of the draft proposal American National Standards (dp ANS) Information Resource Dictionary System (IRDS). The chapter that follows will outline which of these features were chosen for incorporation into the Prototype IRDS.

A. BACKGROUND

As the world's largest user of information processing technology, the U. S. Government depends on this technology to carry out Government-wide programs and deliver essential public services. As with most new technologies Data Dictionary/Directory Systems (DD/DS) were being developed by numerous software suppliers each from a different set of standards. Since it is estimated that the federal government could save "$120 million in benefits by the early 1990s from use of a standard (IRDS)" [Ref. 23], the American National Standards Institute (ANSI) and the National Bureau of Standards (NBS) of the United States Department of Commerce were prompted to initiate efforts to develop standards for dictionary systems. To this end the ANSI committee for Information Systems (X3) convened a Technical Committee X3H4 to develop the standard for an IRDS in 1980. NBS at the same time established a similar committee to develop the "Federal Information Processing Standards for Data Dictionary Systems" (FIPS DDS).

Although the ANSI X3H4 and the NBS committees used different titles for standards they were developing, the two groups had identical goals and similar development approaches. The two efforts came together with the adoption of Proposal A83-020 in August 1983. The proposal called for the acceptance of the draft FIPS DDS as the Base Document for any further
development of IRDS standards and has since been developed into the dp ANS IRDSs [Ref. 24], [Ref. 25], [Ref. 26], [Ref. 27].

B. IRDS DESIGN OBJECTIVES

When specifications for the standard IRDS were being developed three key objectives were always in the forefront of consideration. They were:

* The IRDS should contain the major features and capabilities found in existing Data Dictionary Systems.
* The IRDS should be modularized to promote ease of implementation and cost efficient development.
* The IRDS should support portability of skills and a wide range of user environments.

In pursuit of this goal the Institute for Computer Science and Technology of the National Bureau of Standards took the following steps:

* Preparing and disseminating the Prospectus for Data Dictionary System Standard [Ref. 28] in 1980. This document discussed the use of Data Dictionaries and plans to develop a FIPS standard.
* Conducted a Data Base Directions workshop in October, 1980 that investigated how managers can evaluate, select, and effectively use information resource management tools, in particular data dictionary systems.
* Conducted interviews with government employees that were knowledgeable in the area of data dictionaries to determine current and future requirements for data dictionary systems. The Federal Requirements for a Federal Information Processing Standard Data Dictionary System [Ref. 29] was published as a result of those interviews.
* Conducted numerous workshops for users and vendors between 1982-84 to obtain feedback on previously published documents.
* Developed a functional specification for the development of a data dictionary standard [Ref. 30].
* Prepared and disseminated in August 1983 the draft specifications for the planned Federal Information Processing Standard for Data Dictionary Systems, the document that later became the baseline standard.

1. Outgrowth of Existing Systems

All vendors who were marketing developed IRDSs or were developing IRDS were asked to review the proposed IRDS specification and make
recommendations and suggestions on what should be included in or excluded from the draft standards. Many of their recommendations were subsequently included in the draft specifications.

2. **Flexibility**

The proposed IRDS includes a "CORE" dictionary system (which is the basis for the prototype to be discussed in Chapter 5) plus three modules. The modules are designed to interface with the core system but be independent of each other so that any or all of the modules can be implemented with the core system when desired. To provide additional flexibility, capabilities are specified in the core IRDS that allow organizations to customize or extend the IRDS as required.

3. **Portability of Skill**

The core IRDS contains two user interfaces: a menu driven "Panel" interface and a command language interface. The panel interface allows the system to be used by the inexperienced user. It incorporates a series of interrelated screens that guide the user through the system. The command language interface on the other hand is designed to allow the more experienced user to access the system without viewing the panels. The command language interface may be used in a batch or interface mode.

An implementation of the IRDS standard is considered complete if either of the interfaces are implemented.

C. **IRDS DATA ARCHITECTURE**

This section presents an overview of the framework in which IRDS data is organized and presented to the user.
1. Framework

The IRDS standard is specified in terms of entities, relationships, and attributes (see Figure 3.1).

An IRDS entity represents or describes a real world concept, person, event, or quantity, but is not the actual data that exists in an application file or data base. [Ref. 31]

A relationship is an association between two entities. An attribute represents a property about an IRDS entity or relationship as the IRDS also allows relationships to have attributes. Relationships in the Core IRDS are binary, denoting that an association exists between two entities in the IRDS.

The Core system was restricted to binary relationships because (1) the vast majority of current implementations use binary relationships and (2) it was desired that the Core system be simple enough to implement on microcomputers.

ENTITY u8-20 ENTITY-TYPE = SYSTEM
ASCAD Database_Update
WITH ATTRIBUTES
DESCRIPTION (START = 100 INCREMENT = 10) =
"This subsystem provides the capability for the staff to update the contents of the ASCAD Database."
SYSTEM-CATEGORY = "subsystem"
SECURITY = "datamgr";

Figure 3.1 Sample Entity Representation

An important aspect of the IRDS standard is the concept of TYPE which is used as a way of classifying entities, relationships and attributes. Different attributes have different meanings, for example the
length of Payroll-Number or number of fields in a Payroll-Record are different. But these attributes may appear many times in relationship to other entities, length of name, length of address or number-of-fields in an Accounts-Payable record. The IRDS standard handles this situation by declaring that each attribute is a specific type called an "attribute-type." Thus there are attribute-types called length and number-of-fields.

The concept of types is extended to the IRDS relationship and entity in the form of "relationship-types" and "entity-types" see Appendix A.

Relationships within the IRDS can also have attributes, for example the relationship in Figure 3.1 between Payroll-Record and Payroll-Number could have position attribute-type with a value of 3 indicating that the Payroll-Number appears as the third element in the Payroll-Record.

The IRDS standard also allows for ordered sets of attributes called attribute-groups. This capability was incorporated into the standard because individual attribute-types don't always convey the complete message about an entity. An example of this might be the allowable-range of an entity. The allowable range has a high value and a low value which a singular attribute would not be able to convey. An attribute-group on the other hand would be able to convey this information quite easily.

2. **IRDS Schema**

The IRD schema describes the structure of the IRD. Thus for every entity, relationship, attribute and attribute-group that can exist in the IRD, a corresponding description of the entity-type, relationship-type,
attribute-type and attribute-group-type must exist in the IRD schema. The proposed IRDS standard specifies a set of specifically allowable entries of the types listed above which are collectively called the "Core-System-Standard Schema" which will be discussed in 3. below.

The IRD schema is important for two reasons. First, the IRDS specifications allow for facilities to modify and expand the core-system-standard schema to meet the unique needs of individual users. Second, the IRD schema supports the core system plus modules approach as discussed in Section 1 of this chapter and the IRD schema allows not only extension of the schema data but also definition of additional IRDS functions.

3. The System-Standard Schema

The system-standard schema defines the allowable contents of the IRDS and is expected to be part of every IRDS implementation (the prototype IRD in Appendix E only implements a subset of the system-standard schema, this will be explained in Chapter 4). The core-system-standard schema does not contain all possible entity, relationship and attribute-types that an organization might desire. It does however represent the consensus of the organizations which participated in the original IRDS workshops and reviews. An overview of the core-system-standard schema is provided below and a complete core-system-standard schema is provided in Appendix A.

a. Entity Types

The core system-standard schema contains twelve entity-types that conceptually can be grouped into three categories, Data, Process, and External. [Ref. 32]
**Data Entity-Types**

* DOCUMENT, describes instances of human readable data, such as tax forms and annual reports.

* FILE, describes collections of records which represent an organization's data, such as inventory and accounts receivable files.

* RECORD, describes instances of logically associated data, such as a payroll record.

* ELEMENT, describes an instance of data, such as a social-security-number.

* BIT-STRING, describes a string of binary digits, such as 01000101.

* CHARACTER-STRING, describes a string of characters, such as "house."

* FIXED-POINT, describes exact representations of numeric values.

* FLOAT, describes exact representations of approximate numeric values.

The last four are not used to represent application entities, but are instead used by the "REPRESENTED-AS" relationship to describe the characteristics of elements:

**PROCESS Entity-types**

* SYSTEM, describes a collection of processes and data, such as a payroll-system or accounts-payable-system.

* PROGRAM, describes a particular process, such as print accounts-payable check

* MODULE, describes a group of programs that are logically associated, such as a sort module.

**EXTERNAL Entity-types**

* USER, describes an individual or organization that is using the IRDS, such as the accounting department.

**b. Relational-types**

The relationship-types provided for in the IRDS core system-standard schema represent virtually all connections that might be useful to users. These relationship-types are grouped into eight classes [Ref. 33]:
* CONTAINS, describes a situation where an entity-type contains other entity-types, such as Accounts payable-file CONTAINS Accounts Payable-record.

* PROCESSES, describes a situation where an entity-type acts upon another entity-type, such as Payroll-program PROCESSES Payroll-records.

* RESPONSIBLE-FOR, describes an association between entities representing organizational components and other entities, to indicate organizational responsibility. An example of such a relationship is Accounting-department RESPONSIBLE-FOR General-ledger-file.

* RUNS, describes an association between user and process entities, such as user RUNS program.

* GOES-TO, describes a situation where one process transfers control to another process. An example of this relationship is Accounts-payable-aging-program GOES-TO Aging-report-program.

* DERIVED FROM, describes a situation where an entity is derived from another entity such as Annual-report DERIVED-FROM program-file.

* CALLS, describes a situation where one entity calls another entity such as Data-entry-program CALLS Aging-program.

* REPRESENTED-AS, describes associations between ELEMENTs and certain other entities that document the ELEMENTs format. An example of such a relationship-type is Employee-Name REPRESENTED-AS Ascii-char-string.

c. Attribute Types

The attribute-types available as part of the core-system-standard schema are the ones selected by conscientious of participating DD users and DD software developers during the development of the IRDS standard. They represent most of the attributes that an organization would need to describe the core-system-standard entity and relationship-types. The attribute-types provide [Ref. 34]:

* Audit trail information, a typical audit attribute-type is DATE-CREATED.

* General documentation for entities, for example, DESCRIPTION and COMMENTS.

See Appendix C for a complete list of the attribute-types.
4. Entity Names

The core IRDS allows flexibility in the assigning of entity names. The system also allows for several distinct names to be associated with an entity and for each name to serve a specific purpose. The core system allows for ACCESS NAME, DESCRIPTIVE NAME and ALTERNATE NAME.

The access name is the entity's primary identifier and it is the basis for the structure of most commands and panels. The access name is designed to be short, for ease of use by the system and user. Normally a user will provide the access name of an entity. However an option exists for the IRDS to generate the access names for all entities of a given type. The names that are generated by the system may be modified at a later date.

The descriptive name provides detailed information about the object represented by the entity. So the brevity of the access name poses no disadvantage to the system or user.

The IRDS does place a requirement on the user that all access and descriptive names be unique throughout the system. This requirement was generated by the ANSI X3H4 and workshop participants to insure simplicity in the command language and panel interfaces.

The core IRDS also allows for user assignment of ALTERNATE NAMES for an entity. The term alternate name is used here in the same sense as the terms "synonym" and "alias." The alternate name documents different names used to represent the same real world things. For example, the element whose access name is Social-Security-Number might have alternate names, SSN, Soc-Sec, No, and Social-Security-Number.

D. FUNCTIONS AND PROCESSES

This section describes the functions and processes provided as part of the core IRDS.
1. Populating and Maintaining the IRD

The core IRDS provides functions to add, modify, and delete entities and relationships.

a. Entities

(1) Adding Entities. This function allows the user to add/create entities to the IRD. Some important aspects of adding a new entity are:

* Declaring the type of the entity.
* Designating the assigned access name.
* Assigning a descriptive name to the entity.
* Declaring attributes and attribute-groups for the new entity.

The designated entity-type must be one that exists in the IRD schema.

In order for the access name to be valid it must conform to the following rules:

* The access name must conform to the length and picture requirements of IRD schema.
* The access name used must not previously exist in the dictionary.
* If the system is to generate the access name the user must supply the entity type and starting value see Figure 3.2 for examples.

(2) Modifying Entities. This function is used to change the attributes of existing entities. When using the modify function the user may accomplish the following:

* Creation of new attributes.
* Modification of existing attributes.
* Deletion of existing attributes.

The core IRDS also offers a modification option that allows the user
ADD ENTITY  u8-20  ENTITY-TYPE = SYSTEM
  DESCRIPTIVE-NAME = ASCAD_Database_Update
  WITH ATTRIBUTES
    DESCRIPTION (START = 100  INCREMENT = 10)
      "This subsystem provides the capability for
      the staff to update the contents of the
      ASCAD Database."
    SYSTEM-CATEGORY = "subsystem",
    SECURITY = "datamgr";

Figure 3.2 Sample Command for Adding Entity
to create a new entity which has all the values of the old entity but
with some desired modification. This option allows for the easy genera-
tion of a new version of an existing entity which would be identified as
a different form the original entity by a version number (Figure 3.3).

```
MODIFY ENTITY  dd_01093
WITH ATTRIBUTES
  DESCRIPTION = "A shared data field occupied by
               either cntry_code or state_code",
  SECURITY = "datamgr"
  DATA-CLASS = "alphanumeric",
  IDENTIFICATION-NAMES =
    (ALTERNATE-NAME = "cntry_st_code",
     ALTERNATE-NAME-CONTEXT = "pl1")
```

Figure 3.3 Sample Command for Modifying Entity

(3) **Deletion of Entities.** The core IRDS allows entities to
be deleted by specifying any of the following:

* The access name.
* Entity selection criteria (access names) which will result in the
  creation of a new entity-list.
* The name of an existing entity-list created earlier in the session
  or saved from a previous session.

b. Relationships

(1) **Adding Relationships.** The core IRDS allows for the
creation of new relationships other than those provided as part of the
core. The important considerations in creating a new relationship in-
clude designating:

* The entities that are to be members of the relationship.
* The relationship type.
* Optionally, attributes and attribute groups for the new relationship.
* The entity sequence for ordered relationships.
In creating a new relationship the user need only identify the access-names of those entities associated with the relationship.

(2) Modifying Relationships. The core IRDS allows the user to modify any existing relationship by identify the relationship by type and the access associated with it. Using this function, allows the user to:
* Change a relationship's attributes.
* Create new attributes.
* Delete existing attributes.
* Change the sequence of entities associated with the relationship.

(3) Deleting Relationships. The function is provided by the core system to allow for deletion of relationships.

c. Copying Entities and Relationships.

The core IRDS allows for the creation of new entities with the same attributes, attribute groups and relationships as an existing entity. In order for the new entity to be created the user must activate the copy function and specify a new access name which is not duplicated in the system. Optionally the user may designate a new full descriptive name for the entity to be copied.

2. IRDS Output Facility

The core IRDS provides a GENERAL OUTPUT function for producing output of IRD entities, relationships, and attributes. The general output capabilities are discussed in a. below. The core IRDS also provides two additional output facilities the IMPACT-OF-CHANGE function, which provides a report of all entities that might be affected by a change to a specific entity, and the SYNTAX-OUTPUT function which generates output in the same
format as data was entered to create the entity in the first place.

These two functions are discussed in detail below.

a. General Output

The core IRDS requires that seven steps be completed before any output can be generated. Some of the steps are optional and therefore default values are available. The seven steps required for output generation are:

(1) Specify the views to which retrieval applies. The view is associated with the life cycle phase that the particular entity belongs to (See Figure 3.4 for an example).

Select ENTITIES "Program-2 (*.*)"

Where *:* means all revision-numbers and all variation-numbers

Figure 3.4 Sample Command Line

(2) Selection of the entities to be output. This selection is performed via the entering of selection criteria. Criteria is generally entered at the initiation of the output process. Selection criteria includes (See Figure 3.4

* The type(s) of entities to be retrieved.
* Characteristics of the assigned access or descriptive name.
* Characteristics of the associated version identifier.
* Designated attributes or attribute groups.
* Life-cycle-phases.
* Relationships

(3) Sorting the entities on a series of sort parameters. The available parameters are the same as those listed in D.2 above. Suppose a user wishes to sort the selected entities based on entity-type, variation name, assigned-access-name, and revision-number. Figure 3.5 shows how the command might look.

44
(4) Designating what information is to be displayed include:

* The kind of entity name (access, descriptive or alternate)
* The life-cycle-phase of the entity.
* One or more of the entity’s attributes or attribute groups.
* One or more relationships in which the entity participates.

See Figure 3.6 for an example.

```
entity-type (ascending), variation
(ascending), assigned-access-name
(ascending), revision (descending)
```

Figure 3.5 Sample Parameters

(5) Routing information which sends the output to a particular destination.

(6) Assigning a title to the output.

(7) Providing a name for the output procedure to allow it to be recalled at a later time, when the same output is required.

```
SHOW ASSIGNED-ACCESS NAME
ASSIGNED-DESCRIPTIVE-NAME
REVISION-NUMBER, VARIATION NAME
```

Figure 3.6 Sample Output Format Command Line

b. Output IMPACT-OF-CHANGE

As previously stated, the IRDS allows for the printing or displaying of an Impact of Change report. This report is generated by a function that has two options. First, there is a cumulative impact-of-change option that lists all entities that will be impacted by a proposed change(s). Second, the Individual-Impact-Of-Change option produces a separate list of entities for each of the originally specified entity changes.
ENTITY-1
[All ENTITY-1 information in the order in which it was originally entered].

RELATIONSHIP-1 The first relationship that the entity participates in and all the information associated with the relationship.

RELATIONSHIP-j The jth relationship that the entity participates in and all the information associated with the relationship.

ENTITY-n
[All ENTITY-n information in the order in which it was originally entered].

RELATIONSHIP-1 The first relationship that the entity participates in and all the information associated with the relationship.

RELATIONSHIP-k The kth relationship that the entity participates in and all the information associated with the relationship.

Figure 3.7 Sample Output Syntax Report Format

ENTITY-1
[All ENTITY-1 information in the order in which it was originally entered].

ENTITY-n
[All ENTITY-n information in the order in which it was originally entered].

RELATIONSHIP-1 The first relationship that the entities participated in and all the information associated with the relationship.

RELATIONSHIP-k The kth relationship that the entities participated in and all the information associated with the relationship.

Figure 3.8 Sample Output Syntax Report Format
c. Output Syntax

The output syntax function produces output that includes all information about the entity that was entered during the add-entity or add-relationship process. The output for this function has two formats. The first, involves the listing of each entity and all relationships associated with the entity (See Figure 3.7). The second, lists all for the entities first and then lists all the relationships associated with those entities (See Figure 3.8).

d. Entity-lists

The IRDS allows a user to create and manipulate lists of access names which may then be used as input to other IRDS output functions. The IRDS has functions that allow for the creation of entity lists, maintenance of entity lists, assigning of names to entity lists, output of entity lists, output of entity list names and the performance of set operations on entity lists which include union, intersection and symmetric difference.

e. Procedures

Finally the IRDS provides a PROCEDURE FACILITY that allows the user to save a sequence of operations, used to produce an output. This facility also allows for the saving of previously defined procedures under unique names, execution of previously saved procedures by specifying its name and outputting the names and structures of existing procedures.

3. Schema Maintenance and Output

This section expands the discussion of the IRD Schema which was introduced in Section C.2 and also discusses schema maintenance and output. In the previous sections the schema was shown to include:

ENTITY-TYPES, RELATIONSHIP-TYPES, RELATIONSHIP-CLASS-TYPES, ATTRIBUTES-TYPES,
and ATTRIBUTE-GROUP-TYPES all of which are described in the schema as meta-entities. Meta-entities represent real world entities in the IRD schema. Real world entities are objects of concepts such as sales manager, account, balance sheet and others. The entities that represent these objects, such as user, record or report are in turn linked by meta-relationships and both can have meta-attributes associated with them.

a. Schema Control

As stated in D.3 above the IRD schema contains meta-entities which are linked by meta-relationships with both the entities and relationships being described via meta-attributes.

(1) Meta-entity. The IRD schema allows for the following meta-entities:

* Entity-type
* Relationship-type
* Attribute-type
* Relationship-class-type
* Attribute-class-type
* Attribute-group-type
* Attribute-type-validation-procedure
* Attribute-type-validation-data
* Variation-names-data
* Life-cycle-phase
* Quality-indicator
* Schema-defaults

See Figure 3.9 for an example of an instance of each.

(2) Meta-relationships. Meta-relationships represent relationships between two meta-entities. The core IRDS only allows one
occurrence of a relationship between any two meta-entities. Also
meta-relationships are not given individual names in the core IRDS.

Entity-type
Relationship-type
Attribute-type
Relationship-class-type
Attribute-group-type
Attribute-type-validation-procedure
Attribute-type-validation-data
Variation-names-data
Life-cycle-phase
Quality-indicator
Schema-defaults

Figure 3.9 Instances of Meta-Entities

The general form for a meta-relationship is meta-entity,
meta-relationship, meta-entity. See Figure 3.10 for an example of the
general form of a meta-relationship.

(3) Meta-attributes. Meta-attributes perform a descriptive
role with respect to meta-entities and meta-relationships. The core
IRDS allows for four general types:

* ADDED BY
* ALLOWABLE-VALUE
* DESCRIPTION
* LAST-MODIFIED-BY
* NUMBER-OF-LINES-OF CODE

Figure 3.10 Sample Meta-Attributes

* Documentation meta-attributes are used to document the purpose of
the meta-entity, See Figure 3.10.
* Audit meta-attributes serve the same general purpose as the audit attribute in the IRD, that being to provide an audit trail of what has happened in the schema, see Figure 3.10.

* Schema control meta-attributes provide certain controls over what can and cannot be done to the schema.

* Dictionary control meta-attributes which provide control over the dictionary itself.

(4) A Sample Schema Structure. Figure 3.2 shows a sample schema structure involving files. It demonstrates the use of meta-entities, meta-relationships and meta-attributes in the formation of schema.

b. Schema Manipulation

The core IRDS allows for the modification of the schema via adding, modifying and deleting of meta-entities and relationships. These functions are designed to be performed by only those individuals with the proper access authorization.

(1) Adding meta-entities. The core IRDS will allow those users with the proper authorization to add new meta-entities. The kinds of meta-entities that can be added are listed in D.3.a.(1) above. New meta-entities may not be assigned the name of a meta-entity that already exists.

(2) Modifying meta-entities. The core IRDS allows the user to modify meta-entities by associating a new meta-attribute with the meta-entity, by changing an existing meta-attribute or by deleting a meta-attribute that already is associated with the meta-entity. In the case of a changed or deleted meta-attribute the IRDS will insure that the change did not adversely effect the dictionary.

(3) Deleting meta-entities. The core IRDS provides the user with the ability to delete an existing meta-entity from the schema.
However the IRDS will insure that the integrity of the dictionary is not violated.

(4) Adding Meta-relationships. The core IRDS gives the IRDS user the ability to add new meta-relationships as he sees necessary. As stated earlier meta-relationships are associations between meta-entities. The process of adding a meta-relationship requires that the user specify the meta-entities that are to be members of the relationship and any meta-attributes that will be associated with the meta-relationship.

(5) Modifying, Deleting and Replacing Meta-relationships. The core IRDS provides the user with the ability to modify, delete and replace meta-relationships. The modifying and deleting of meta-relationships is performed in the same manner as the modification and deletion of entities as explained in D.3.b.(2) and D.3.b.(3) above. The replacement of meta-relationships actually involves the combination of the delete meta-relationship and add meta-relationship functions. The replacement function is organized in this manner to insure the integrity of the IRD.

(6) Modification of Meta-entity Names. The core IRDS allows the user to modify the meta-entity name. This process however falls along the same lines as the meta-relationship replacement function. It is the forced combination of the meta-entity deletion and add functions. This process is again used to insure integrity of the IRD. One additional requirement exists and that is that the meta-entity name not be duplicated anywhere in the IRD.

c. Schema Output

The core IRDS allows those authorized to work with the schema the ability to output information about it. In order to produce the
output the user must select the meta-entities to be displayed. This selection is accomplished by choosing one of the following:

* That all meta-entities be displayed.
* That all meta-entities of a specific type(s) be displayed.
* The name of a specific meta-entity.

The resulting set of meta-entities may then be sorted on one of the following parameters:

* meta-entity-type
* meta-entity-name
* Non repeating meta-attribute-types

Before the sorted list is displayed the user must specify the information about each meta-entity he wishes to see. The display options available to him are one of the following:

* meta-name
* meta-type
* One or more of the associated meta-attributes
* All or none of the associated meta-relationships in which the meta-entity participates

4. **The IRD to IRD Interface**

The IRD to IRD interface is an important feature of the core standard IRDS because it is the only controlled means for moving data between two IRDS. This facility allows an organization with more than one IRD to transfer information between them. The facility is also designed to allow IRDSs developed by different vendors to interface and exchange information, provided a communication link exists and they have
followed IRDS standards. The core standard IRDS only allows for the transfer and does not have any means of providing the physical connection between the IRDS. In allowing for the interface the only important issue stressed is that the exporting and importing dictionaries and the exporting and importing schema's must be compatible.

5. IRDS Control Facilities

The core IRDS contains five control facilities that are important in populating and maintaining the IRD. These are:

* The Versioning Facility
* The Life-Cycle-Phase Facility
* Quality-Indicators
* Views
* Security

An overview of these was provided in Section D. This section presents additional detail on the structure and use of these facilities.

a. The Versioning Facility

The versioning facility provides the user with the ability to distinguish between entities that would otherwise be considered the same. The distinction is generated via the version-identifier which is composed of two parts: (1) a required revision-number and (2) an optional variation-name.

In the command language syntax the user encloses the version-identifier in parentheses and appends it to the access or descriptive entity name. Within the parentheses the variation-name (if used) is followed by the revision-number, separated by a colon. If the user does not specify a revision-number the system will default with a value of
1 to indicate that no revision exists and a value of 1 greater than
the current value for any subsequent revisions.

For example, suppose a certain payroll module exists that
calculates state taxes for Alabama, Georgia, and Florida and another pay-
roll module of the same functionality calculates state taxes for California
and Texas. We can describe both with the same access name PAYROLL-MODULE,
and differentiate between the two with different variation-names. Thus
we could have PAYROLL-MODULE (AL-GA-FL:1) which would represent the Ala-
bama, Georgia, and Florida capable payroll module with no revision. The
California and Texas module which has had three revisions would be re-
presented as PAYROLL-MODULE (CA-TX:4).

b. The Life-Cycle-Phase Facility

The life-cycle-phase facility of the core IRDS: (1) allows
the user to define the life cycle phase to meet the methodology currently
being used; (2) Provides facilities to assign each entity to a particu-
lar phase; (3) Provides integrity rules concerning the passing of an
entity from one phase to another. Each phase is represented in the
schema as a meta-entity.

Every life-cycle-phase belongs to a "phase class" and the
core IRDS recognizes three such classes:

* UNCONTROLLED -- Uncontrolled phases are "specification," "design"
or "non-operational." There are no integrity rules for this class
and a user may identify as many phases with this class as desired.

* CONTROLLED -- Controlled phases are those that are considered to be
"operational." The core IRDS allows only one such phase the
"CONTROLLED-PHASE" with its associated integrity rules. The
integrity rules will be covered in the next section.

* ARCHIVED -- The core IRDS can only have one ARCHIVED life-cycle-
phase, called the "ARCHIVED-PHASE" and it is used to document and
classify entities no longer in use. This class also has special integrity rules associated with it, those will also be discussed in the next section.

(1) Integrity Rules. As mentioned previously, integrity rules for the CONTROLLED and ARCHIVED life-cycle-phases are enforced by the core IRDS. These rules are based on a hierarchy of system-standard entity-types as defined by the following list. The highest in the hierarchy is the first and the lowest is the last:

* SYSTEM
* PROGRAM
* MODULE
* FILE
* DOCUMENT
* RECORD
* ELEMENT

This means that the entities are "Phase-related." The hierarchy only applies to the core standard IRDS entity types and not to any entities added by the user via the extendability facility.

These are integrity rules in the sense of controlled and archived but not in the sense of allowable ranges of attribute data values, e.g.: "sex must be 'M' or 'F'." This type of integrity is handled through the ATTRIBUTE-TYPE-VALIDATION-PROCEDURE-META-ENTITIES:

* RANGE-VALIDATION, which is used to restrict an attribute-type to an allowable set of ranges.
* VALUE-VALIDATION, which is used to restrict an attribute-type to an allowable set of values.

There are two relationship-class-types that are designated as phase-related, they are CONTAINS and PROCESSES they are combined...
with the entity-type to form phase-related relationship-types. Listed below in Table 3.1 are the relationship-types generated by this combination:

The general integrity rule for entities in the controlled life-cycle-phase is:

An entity can be in the CONTROLLED life-cycle-phase only if all entities whose types are below its type on the above hierarchy and that are connected to it with phase-related relationships are also in the CONTROLLED life-cycle-phase.

The ARCHIVED life-cycle-phase has an integrity rule similar to that above:

An entity can be in the ARCHIVED life-cycle-phase only if all entities whose types are below its type in the above hierarchy and that are connected to it with phase-related relationships are in either the CONTROLLED or ARCHIVED life-cycle-phase.

The integrity rules are designed to insure that when an entity, for example "PAYROLL-SYSTEM" is moved to a new phase, for example "OPERATIONAL-PHASE" that all of the programs and modules associated with the system are either already in the operational-phase or ready to be moved to it, thus insuring the integrity of the system.

c. Quality-Indicators

The core IRDS allows the user to define quality-indicators and assign them to entities. These quality-indicators denote such things as:

* The level of standardization of an entity (e.g., program standards, organization standards, company standards, and international standards).

* The degree to which an entity meets the user quality assurance standards, etc.

All quality-indicators must be added to the IRD schema as a meta-entity. Also the core system-standard schema does not include any indicators, so all indicators must be user defined.
<table>
<thead>
<tr>
<th>Table 3.1</th>
<th>Phrase-Related Relationships - Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>System-contains-system</td>
<td>System-contains-problem</td>
</tr>
<tr>
<td>System-contains-module</td>
<td>Program-contains-program</td>
</tr>
<tr>
<td>Program-contains-module</td>
<td>Module-contains-module</td>
</tr>
<tr>
<td>File-contains-document</td>
<td>File-contains-record</td>
</tr>
<tr>
<td>File-contains-element</td>
<td>Document-contains-document</td>
</tr>
<tr>
<td>Document-contains-record</td>
<td>Record-contains-record</td>
</tr>
<tr>
<td>Record-contains-element</td>
<td>Element-contains-element</td>
</tr>
<tr>
<td>System-processes-file</td>
<td>System-processes-document</td>
</tr>
<tr>
<td>System-processes-record</td>
<td>System-processes-element</td>
</tr>
<tr>
<td>Program-processes-file</td>
<td>Program-processes-document</td>
</tr>
<tr>
<td>Program-processes-record</td>
<td>Program-processes-element</td>
</tr>
<tr>
<td>Module-processes-file</td>
<td>Module-processes-document</td>
</tr>
<tr>
<td>Module-processes-record</td>
<td>Module-processes-element</td>
</tr>
</tbody>
</table>
d. Views

Views are how the user logically perceives the dictionary and as such it is generally a subset of the complete dictionary. A view may be: (1) a set of entities with associate entities, attributes, and attribute-groups; (2) a set of relationships with its associated entities, attributes, and attribute-groups or (3) a set of specifications of operations that may be performed by the user.

Structurally, VIEW is an entity-type in the core IRDS system-standard schema and each view in the IRD is an instance of that entity-type. For example, if a particular programmer is working on the Payroll-system of an organization. His view of the IRD would be all the programs, modules, files, records and elements contained in or processed by the Payroll system.

The core IRDS allows an organization to define what views are available to a user thus limiting his access to the dictionary. If more than one view is available to a user, one will be designated as the default-view and will be presented to the user each time he uses the system unless he specifically specifies otherwise. Views associated with each user are stored in the IRD as attributes of the DICTIONARY-USER entity.

e. Core Security

The general mechanism that implements core IRDS security consists of the following:

* For each authorized user of the IRDS, one DICTIONARY-USER entity exists. Associated with this entity are attributes that define the user's level of access.

* Associated with each VIEW entity are attributes that define the permissions and restrictions that apply to all IRDS users allowed
to use the view. These include the abilities (independently specified for each entity-type), to read, add to, modify, and delete the entities that comprise the view.

* Finally, each DICTIONARY-USER entity is linked to those views that the user can access.

(1) **Access Permission.** Most IRD ACCESS PERMISSION is associated with view entities, and, for each view, the permission applies to all entities in that view. Each permission consists of several parts:

* The name of the entity-type for which the permissions are specified.
* An indicator showing if permission exists to read entities of the specified type.
* An indicator showing if permission exists to add entities of the specified type.
* An indicator showing if permission exists to modify entities of the specified type.
* An indicator showing if permission exists to delete entities of the specified type.
* An indicator showing which relationships are explicitly excluded from that view.
* An indicator showing if permission exists to modify the life-cycle-phase of entities of the specified type.

These permissions are stored in the IRD as a DICTIONARY-PERMISSION attribute-group. Each view may have multiple permissions associated with it.

The core IRDS specified five categories of permission:

* **GLOBAL PERMISSION:** All schema functions are allowed.

* **GLOBAL PERMISSION FOR UNLOCKED META-ENTITIES:** Permission to perform all schema functions except those that modify or delete meta-entities that have installation-lock set on.

* **ATTRIBUTE-TYPE-VALIDATION-DATA WRITE PERMISSION:** Read attribute type
validation data and modify their meta-attribute.

* ATTRIBUTE-TYPE-VALIDATION-DATA READ PERMISSION: permission to read attribute-type-validation-data and their meta-attribute.

* REPORTING PERMISSION: permission to read the complete schema.

This facility is implemented through attributes of the DICTIONARY-USER entity.

6. User Interfaces

This section discusses the command language and panel interfaces. An implementation of the IRDS may contain either or both of the interfaces but each interface will support the full capabilities of the IRDS.

As stated earlier the IRDS interfaces are designed to allow the system to communicate with the user and vice versa. The panel interface is designed to prompt the novice through the system while the command language interface is designed for the more experienced user and thus skips most of the panels used in the panel interface.

a. Command Language

The COMMAND LANGUAGE interface supports both batch and interactive modes. The commands used by the command language interface correspond closely with the functions discussed throughout this chapter. The syntax of each of the command language commands is presented in the Bacus-Naur form. Since the command language closely parallels the discussion presented in the previous sections a detailed discussion of each command will not be attempted. A summary listing of the commands and their associated functions is provided in Appendix E. However the command language is discussed and illustrated in depth in [Ref. 35].
b. Panel Interface

The core IRDS provides the user of the system a structured set of logical screens (or panels) which, when used in the proper sequence perform the functions of the system. The panels can be considered to be user friendly in that they guide the user through the procedures for a function.

The core IRDS does not specifically identify a panel structure of physical implementation of the panel interface. It is therefore up to the user to define his own panel structure and panel map (which panel follow which) for each function.

The core IRDS does provide rules for the structure of the panels used by the IRD. They are:

* Each panel shall have a unique name.

* The panel interface is to have an inter-panel structure that defines a default progression of panels.

* The first panel encountered is the HOME panel.

* The user may return to the HOME panel at anytime.

The structure of the panel interface is defined in terms of panel trees and panel areas. A panel tree is the collection of one or more panels used to perform a single function. A panel area is a portion of a panel that is associated with a particular category of information, and deals with the user interaction with the IRDS. The core IRDS identifies six different areas associated with the panel, not all of which are shown to the user at one time:

* STATE AREA -- This area will always be displayed to the user. It informs the user of the name of the dictionary being accessed, and what is being done with the current panel, for example, adding a record.
* DATA AREA -- The data area supports the user in one of two ways: It displays labels that guide the user while he/she performs data entry; and, if the user is retrieving information, it displays the results.

* SCHEMA AREA -- The schema area is primarily used during dictionary update operations. Examples of the use area include:
  - The listing of all valid entity-types, when adding an entity.
  - Displaying names of attribute-types that may be associated with an entity-type being entered.

* ACTION AREA -- The action area displays the options that a user has when proceeding from the current panel to another.

* MESSAGE AREA -- This panel area displays any errors and warning messages.

* HELP AREA -- The help area displays information that the system can provide in response to a request for help.

   c. Operation on the Panel Interface

   The panel interface will generally be available to all IRDS users. The core IRDS does not however, require that the panel made available to a user be tailored to meet his view of the system. The panel interface will still only allow the user to perform those functions and operations allowed according to his view and current security.

7. IRDS Modules

   The draft proposed IRDS standard contains specifications for three modules which may be implemented along with the core IRDS. They are:

   * ENTITY LEVEL SECURITY. [Ref. 36]
   * APPLICATION PROGRAM (CALL) INTERFACE. [Ref. 37]
   * SUPPORT OF STANDARD DATA MODULES. [Ref. 38]

   Since the scope of this thesis deals primarily with the capabilities of the core IRDS, the references listed above should be consulted if any additional information beyond that provided is required.
a. Entity Level Security

This module allows the user the ability to assign read and write limitations to individual entities. This facility operates in addition to the security function provided in the core IRDS.

To accomplish entity level security, the module introduces the entity-type ACCESS-CONTROLLER, and a set of SECURED-BY relationship-types that allow an ACCESS-CONTROLLER entity to be connected with entities of all other types.

b. Application Program (call) Interface

This module provides an interface from a standard programming language to the IRDS. This is accomplished by using the call feature of the programming language. In this way the IRDS is treated as an application program subroutine.

c. Support of Standard Data Models

An implementation of the specifications of this module would assist an organization in describing network and relational databases, particularly those supported by NDL and SQL command languages. The describing of network and relational databases is accomplished through the addition of three new entity-types, twelve new relationship-types, and fourteen new attribute-types to the Core System-Standard Schema, See Appendix D.

E. CONCLUSION

The NBS IRDS standards provide the Information Resource Management arena a valuable tool. An implementation of an IRDS using the core standards as discussed above would deliver to the DBMS user tremendous capability, flexibility and uniformity in describing and controlling an
organizations data. Finally the capabilities described above far exceed that which is currently available with most of the dictionaries provided with DBMS products.

But is an IRDS implementation possible. The next chapter discusses just such an implementation.
IV. NBS IRDS PROTOTYPE

This chapter discusses the implementation of selected portions of the NBS IRDS standards in the form of a relational prototype IRDS provided as Appendix E. Before discussing the NBS IRDS capabilities included in the prototype. It is necessary to discuss prototyping, its advantage/disadvantages and why prototyping was chosen as the method for implementing an IRDS.

A. PROTOTYPING [Ref. 39]

Webster’s dictionary defines a prototype as one of three possible things:
* An original or model after which anything is formed
* The first thing of its kind
* A pattern, an exemplar, an archetype

The second definition is probably the most relevant to this discussion because prototypes are being used in data processing as a first attempt at design which is then extended or enhanced. In general systems development, a prototype is known as

... a partially complete functional model of a target system whose purpose is to provide a better understanding of the target system’s requirements [Ref. 40].

A software prototype is characterized by the following feature. It is a working system, although of limited capability, rather than just an idea on paper. A prototype may become, after iterative enhancement, a production system. Its original purpose is to test assumptions about requirements and/or system design architecture. A prototype is created quickly. This has become possible only in recent years with more powerful languages such as dBase II and III which are less procedurally
oriented. Some would argue however, that prototyping was the way software was developed before the advent of functional decomposition and the system development life cycle which is generally accepted and used today.

In the early days of software development writing programs was the thing to do. After an explanation of the problem, a period of questions and answers, and research into the nuts and bolts of a method, the programmer began his or her work. Starting with that portion of the problem that was well understood, lines of FORTRAN, COBOL or ALGOL would begin to appear. As time passed additional portions were coded until the entire program was complete. Design was conducted implicitly, if at all! [Ref. 41]

A prototype should be inexpensive to build, at least less than it would cost if a conventional high level language were used. Indeed, prototyping in data processing originated only recently because until recently, programming a prototype was just as costly as programming the working system [Ref. 42]. The important point is to get something running soon to establish effective communications with the user without the use of extravagant resources. The follow-on development of a prototype is an iterative process in which improvements are made in small increments as the user developer work together and discover new requirements. [Ref. 43]

Mitchell Spiegel, formerly of Wang Laboratories, explains the prototyping approach as:

... a process of modeling user requirements in one or more levels of detail, including working models. Project resources are allocated to produce scaled down versions of the software described by requirements. The prototype version makes the software visible for review by users, designers and management. This process continues as desired, with running versions ready for release after several iterations. [Ref. 44]

Traditional management information system development follows a series of steps (see Figure 4.1). Prototyping is considered as an
adjunct activity to the specification of requirements (See Figure 4.2).

The results of prototyping are input to the steps following requirements:

- Feasibility Study
- Requirements
- Product/Preliminary Design
- Detailed Design
- Coding
- Integration
- Implementation
- Operations and Maintenance

Figure 4.1 Steps in Traditional System Development

analysis, but may or may not be used actively in those steps.

1. **Advantages of Prototyping**

   There are several advantages associated with the use of prototyping. First, a prototype usually gets the product into use as early as possible. Early use can provide assistance to the decision makers and feedback to the builders. Second, prototyping is considerably cheaper than a "full-build" approach, which delays installation until the product is complete. Third, prototyping is a convenient way of keeping the product simple, which is valuable to both builders and users. Fourth, prototyping lowers risk and expectations. [Ref. 45] Fifth, it is easy to write statements in a requirements document which say "the system shall do x" and "the system shall be capable of y." However, both the developer and the user get a more realistic feeling for the effort and cost of a feature when they must actually add it to a working model. Thus, the eventual model better represents what is feasible than a document with a series of "shall statements." Even though the functionality of a prototype product is minimal, the user is forced to think
more carefully about the task being automated. This should produce a more accurate understanding of the problem [Ref. 29]. Finally, prototyping unlike traditional methods builds an effective brigade across the communication gap between the user and the developer.

<table>
<thead>
<tr>
<th>Feasibility Study</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements------</td>
</tr>
<tr>
<td>Product/ Preliminary Design</td>
</tr>
<tr>
<td>Detailed Design</td>
</tr>
<tr>
<td>Coding</td>
</tr>
<tr>
<td>Integration</td>
</tr>
<tr>
<td>Implementation</td>
</tr>
</tbody>
</table>

1. Identify basic needs
2. Develop working model
3. Demo in context & Solicit refinements
4. Implement revisions
5. Is prototype done?
   If YES, go on to detailed design
   If NO, go back to step 3

Figure 4.2 System Development Using Prototyping

2. Disadvantages of Prototyping

Prototyping has some decided disadvantages as well. Prototyping makes it difficult to plan resource use because a clear picture of what the finished product will look like is not provided. It also makes it difficult to decide whether to enhance an old version or build a new one. Analysts and user can become bored after the nth iteration of the prototype.
In using the traditional development process there are specific requirements which, when met by proof of validation, clearly mark the job as complete. Because the prototype is changing continually, it creates a problem keeping users abreast of the current version and what has been validated and what has not. Prototyping can cause a reduction in discipline for proper documentation and testing (although this has nothing to do with the prototype itself). Because there is less emphasis on hard thinking and "desk checking" there is a greater chance of missing a basic problem which could negate assumptions essential to the product being developed. Also there is the chance users may become so happy with the prototype that they consider it a functional product and want the data processing people to start work on something else. A study using the ACT/I software package for prototyping showed increased needs for computing resources. If the productivity gained from using prototyping doesn't offset the cost of the increased computing power, then the prototyping approach is serving at a disadvantage.

3. Types of Prototyping

There are two approaches to prototyping: the throwaway prototype. The throwaway prototype development process has the advantage that when the developer can show the user an immediate capability when he is through, he can just discard the product. This lowers the developer's risk and the user's expectations. The evolving prototype process on the other hand is better suited for the development of an initial capability that will evolve into a finished product. The evolving prototype has the disadvantage that the user may accept the first version and thus short circuit full development.
4. Reasons for Prototyping

Prototyping was chosen as opposed to full life cycle development, because time constraints prevented full development of a DD system whereas prototyping allowed a viewable product to be produced in the given timeframe. Additionally the evolving prototype process was used to develop the IRDS with the anticipation that additional capabilities as specified in the NBS IRDS standards would be added according to user needs as additional versions were implemented.

B. THE IRDS PROTOTYPE

dBASE III a Data Base Management System (DBMS) was selected as the development tool for the IRDS prototype, because data dictionary systems are essentially a specialized kind of database system. The prototype could have been written in Pascal or COBOL but the time required to produce a usable product would have been prohibitive. Additionally since the prototype was developed using a DBMS system certain capabilities were already available, i.e., a query processor, file maintenance routines, and high level language. The intention was not to develop a marketable product but to demonstrate and evaluate the capabilities described in the NBS IRDS standard.

The IRDS prototype is based on a reasonable subset of the core features presented in Chapter 3. The features listed below constitute IRDS Prototype Version 1.0 (See Appendix C):

* Panel Interface
* Security
* Add Entity
* Modify Entity
The remainder of the features listed in Chapter 4, though desirable, will be left for implementation in later versions.

1. A Relational Model of the IRDS

The IRDS prototype accounts for several different relations including users, systems, programs, modules, document, files, records and elements. The generalized format of these relations is as follows:

* USER (access-name, id-name, duration-type, description, date-added, added-by, comments, last-modification-date, last-modified-by, number-of-modifications)

* SYSTEM (access-name, id-name, duration-type, description, date-added, added-by, system-category, comments, last-modification-date, last-modified-by, number-of-modifications)

* PROGRAM (access-name, id-name, duration-type, description, date-added, added-by, lines-of-code, comments, last-modification-date, last-modified-by, number-of-modifications)

* MODULE (access-name, id-name, duration-type, description, date-added, added-by, lines-of-code, comments, last-modification-date, last-modified-by, number-of-modifications)

* DOCUMENT (access-name, id-name, duration-type, description, date-added, added-by, comments, last-modification-date, last-modified-by, number-of-modifications)
* FILE \((access-name, id-name, duration-type, description, date added, added-by, number-of-records, comments, last-modification-date, last-modified-by, number-of-modifications)\)

* RECORD \((access-name, id-name, duration-type, description, date-added, added-by, number-of-elements, size, comments, last-modification-date, last-modified-by, number-of-modifications)\)

* ELEMENT \((access-name, id-name, duration-type, description, date-added, added-by, element-type, element-length, low-of-range, high-of-range, allowable-value, comments, last-modification-date, last-modified-by, number-of-modifications)\)

for a detailed explanation of the attributes for these relations see [Ref. 47].

Relationships among the various relations are tracked by having relations with a verb name reflecting how one entity relates to another. For example, since a program can contain several modules, a program-contains-module relations is included in the dictionary. Its format is as follows:

* PROGRAM-CONTAINS-MODULE \((program-name, module-name)\). An example of this relation would be:

PROGRAM-CONTAINS-MODULE \((u-8, u-8-10)\)

\((u-8, u-8-20)\)

\((u-8, u-8-30)\)

The prototype implements twelve of the sixty-four relationships specified in the NBS IRDS standard. See Appendix A for a complete listing of the allowable relationships. Listed below are the twelve relationships included in the prototype:

* PROGRAM-PROCESSES-RECORD \((program-name, record-name)\)

* PROGRAM-PROCESS-FILE \((program-name, file-name)\)

* SYSTEM-CONTAINS-FILE \((system-name, file-name)\)

72
2. Interface

The NBS IRDS standard provides for two user interface capabilities: the Command Language Interface and the Panel Interface. The Panel Interface method was chosen because it provides a "user friendly" communication link between the IRDS and the user. Figures 4.3 thru 4.6 provide a series of panel trees that diagrammatically represent panel interface system used.

The panel structure itself followed the guidelines provided in the IRDS standard (See Figure 4.7). The IRDS standard allows for six possible areas to be defined in the panel state area, data area, schema area, action area, message area and help area. All areas except the help area are included in this prototype. Figure 4.7 shows what portions of the screen are used for each of the areas.

C. IRDS START-UP

This IRDS prototype was written in dBASE III and uses panel interfacing as the means of communication with the user. The first panel that
Figure 4.3 The Panel Interface -- Overall Structure

Figure 4.4 Dictionary Maintenance Panel Tree
Figure 4.7 Panel Structure
that a user sees when signing on the system, is one that requires the individual to insure that he/she has the computer in the proper mode (Figure 4.8)

D. SECURITY

Security is provided in two ways. First, the system requires the user to enter a user ID and password which are stored as attributes of the SECURITY-ACCESS ENTITY. Second, the SECURITY-ACCESS ENTITY contains additional attributes that pertain to which entities the user can view, display and/or modify (See Appendix F for a detailed description the entity structure). Figure 4.9 depicts the panel that requires the user

Figure 4.8 Initial Panel
to log into the system using his user ID and password. Once a user has entered his ID and password the system will grant or deny access to the system. The system will allow the user three chances to enter his ID and password correctly, if a proper logon has not been accomplished at that time the system will terminate. If access is granted additional variables will be loaded to the system that will restrict the user ability to add, modify and change relations and relationships during the current session. The data administrator is the only user capable of modifying the attributes associated with a user's security-access entity. Once the user has successfully logged in, the system will display the main menu (Figure 4.10). From this point the user can proceed to any other panel. This panel must always be returned before any other function can be used.

E. POPULATING AND MAINTAINING THE DICTIONARY

The routines to add, modify and delete entities and relationships are executed from the maintenance menu (See Figure 4.11). The user decides which maintenance activity he want to do and makes the appropriate menu selection. The system will then activate the appropriate maintenance module and present a panel to the user showing him what his options are or what input is required.

The following sections describe each of the five dictionary maintenance functions available to the user as part of the prototype.

1. Adding Entities

If the prototype user selects the add entity option from the maintenance menu, the system will prompt him as to which type of entity he would like to add (See Figure 4.12). Once the user has indicated his
Figure 4.9 Security Panel
Figure 4.10 Main IRDS Panel
1. INFORMATION RESOURCE DICTIONARY SYSTEM
   MAINTENANCE MENU
   1) ADD ENTITY
   2) MODIFY ENTITY
   3) DELETE ENTITY
   4) ADD RELATIONSHIP
   5) DELETE RELATIONSHIP
   6) RETURN TO MAIN MENU
   ENTER YOUR CHOICE (1-6) FROM ABOVE:

Figure 4.11 Maintenance Panel
choice, the system will present a panel prompting the user to enter the appropriate attributes about the entity (See Figure 4.13). For a

![Figure 4.12 Add Entity Panel](image)

complete list of all allowable entity attributes see Appendix A.

2. **Modifying Entities**

If the prototype user elects to modify an existing entity, the system displays a panel asking which entity he desires to modify (Figure 4.14 and 4.15). Once the user makes his selection as to which entity to modify, the system retrieves the desired tuple and presents a panel displaying it's current contents. The user can then modify the tuple as desired (Figure 4.16)
Figure 4.13 Add Entity Data Input Panel

Figure 4.14 Modify Entity Panel
Figure 4.15 Modify Entity Select Panel

Figure 4.16 Modify Entity Input Panel
3. Deleting Entities

If the prototype user selects the delete entity option, the system presents a panel requesting that the user select an entity type to delete.

Figure 4.17 Delete Entity Panel

The system then requests the user to identify the particular entity tuple to be deleted and provides instructions on how to complete or short the deletion. Once the user indicates which tuple he wishes to delete, the system displays the tuple and waits for the user to complete the transaction (Figure 4.16 thru 4.18)

4. Adding Relationships

If the prototype user elects to add a relationship the system present a panel asking him to select which type of relationship he wishes
1.1.2.1.0.0 INFORMATION RESOURCE DICTIONARY SYSTEM

MODIFY ENTITY

ENTER TUPLE NUMBER OF THE ELEMENT
YOU WISH TO DELETE. THE RECORD
WILL BE DISPLAYED FOR YOU TO
EXAMINE. IF YOU ARE SURE THAT
YOU ARE DELETING THE RIGHT
RECORD DEPRESS 'U'.

IF YOU DO NOT WANT IT DELETED
DEPRESS 'O' TO RETURN TO THE
MAINTENANCE MENU.

ENTER THE TUPLE NUMBER NOW:

Figure 4.18 Delete Entity Selection Panel

Figure 4.19 Delete Entity Confirmation Panel
to add (Figure 4.20). When the user makes his choice the system executes the relationship add module and prompts the user for the required input (Figure 4.21) This prototype version allows 12 relationships. See Appendix A for a complete list of all relationships allowed in the IRDS standard.

1.1.4.0.8 INFORMATION RESOURCE DICTIONARY SYSTEM

ADD TO RELATIONSHIP

1) USER CONTAINS SYSTEM  8) FILE CONTAINS RECORDS
2) SYSTEM CONTAINS PROGRAM  9) RECORD CONTAINS ELEMENT
3) PROGRAM PROCESSES FILE  10) USER RESPONSIBLE FOR SYSTEM
4) PROGRAM PROCESSES RECORD 11) USER RESPONSIBLE FOR FILE
5) PROGRAM PROCESSES ELEMENT 12) PROGRAM PRODUCES DOCUMENT
6) SYSTEM CONTAINS PROGRAM  13) RETURN TO PREVIOUS MENU
7) PROGRAM CONTAINS MODULE  14) RETURN TO MAIN MENU

ENTER YOUR CHOICE (1-14) FROM ABOVE:

Figure 4.20 Add Relationship Selection Panel

5. Modifying Relationships

This version of the IRDS prototype does not contain a modify relationship capability as the add relationship module serves the same purpose.
6. **Deleting Relationships**

This module of the IRDS prototype allows the user to select a tuple of a particular relationship and mark it for deletion. The user must identify which type of relationship he wants to modify (Figure 4.24). After the user makes a selection, the delete module is loaded which prompts the user to identify which tuple to delete and provides him with instructions on how to complete the transaction. The system then retrieves the tuple and displays it for verification and transaction completion (See Figures 4.25 and 4.26).

**F. THE DICTIONARY OUTPUT FACILITY**

The IRDS prototype allows the user to generate dictionary output in two forms, screen and printer. When the user selects the dictionary output
**1.1.5.0.0**

**INFORMATION RESOURCE DICTIONARY SYSTEM**

**DELETE FROM RELATIONSHIP**

1) **USER CONTAINS SYSTEM**
2) **SYSTEM CONTAINS PROGRAM**
3) **PROGRAM PROCESSES FILE**
4) **PROGRAM PROCESSES RECORD**
5) **PROGRAM PROCESSES ELEMENT**
6) **SYSTEM CONTAINS PROGRAM**
7) **PROGRAM CONTAINS MODULE**
8) **FILE CONTAINS RECORDS**
9) **RECORD CONTAINS ELEMENT**
10) **USER RESPONSIBLE FOR SYSTEM**
11) **USER RESPONSIBLE FOR FILE**
12) **PROGRAM PRODUCES DOCUMENT**
13) **RETURN TO PREVIOUS MENU**
14) **RETURN TO MAIN MENU**

ENTER YOUR CHOICE (1-14) FROM ABOVE:

---

**Figure 4.24 Delete Relationship Selection Panel**

---

**1.1.5.1.0.0**

**INFORMATION RESOURCE DICTIONARY SYSTEM**

**DELETE FROM RELATIONSHIP**

ENTER TUPLE NUMBER OF THE

USER-PROCESSES-SYSTEM

TUPLE THAT YOU WISH TO HAVE DELETED
THE TUPLE WILL BE DISPLAYED FOR
YOU TO EXAMINE. IF YOU ARE SURE
THAT YOU ARE DELETING THE RIGHT
TUPLE DEPRESS "U". IF YOU DO NOT
WANT IT DELETED, TYPE "END", IF
YOU WANT TO EXIT THE MODULE WITHOUT
IDENTIFYING A TUPLE DEPRESS "7" TO
RETURN TO THE PREVIOUS MENU.

ENTER THE TUPLE NUMBER NOW _

---

**Figure 4.25 Delete Relationship Panel**
option from the main menu, the system executes the dictionary output module and presents to the user a panel (Figure 4.27) requesting that he choose entities or relationships as output.

Figure 4.26 Delete Relationship Tuple Verification Panel

1. **Entities**

If the user chooses the entity output option, the system presents a panel requesting the type of entity to be output (Figure 4.28). The system then prompts for whether output is to be generated and displayed on the screen or sent to the printer (Figure 4.29). The system then displays all tuples of the entity-type selected, one at a time for screen output and all at once for printer output (Figure 4.30). The current version of the IRDS prototype does not allow the user to select which attributes will be displayed or limit the number of entities displayed. However the query function does give the user the ability
to display selected entity types. This capability will be discussed in Section G.

Figure 4.27 Dictionary Output Selection Panel

2. Relationships

If the prototype user decides to output the tuples associated with a particular relationship, he makes the appropriate choice on the dictionary output panel (Figure 4.27). The system activates the appropriate module and then requests that the user identify the relationship to be output (See Figure 4.31. After the user selects the relationship, the system prompts for whether output is to be generated to the screen or printer (Figure 4.32). The system then displays all tuples of the entity-type selected (Figure 4.33). This version of the IRDS prototype does not allow the user to select which entities associated with the relationship are to be displayed. However the query function does give the user the ability to display selected entities with a relationship.
1.2.1.0.0

INFORMATION RESOURCE DICTIONARY SYSTEM

ENTITY OUTPUT

1) USER  6) FILE
2) SYSTEM  7) RECORDS
3) PROGRAM  8) ELEMENT
4) MODULE  9) RETURN TO PREVIOUS MENU
5) DOCUMENT  10) RETURN TO MAIN MENU

ENTER YOUR CHOICE (1-10) FROM ABOVE: 1

Figure 4.28 Entity Output Panel

1.2.1.1.0

INFORMATION RESOURCE DICTIONARY SYSTEM

ENTITY OUTPUT

LISTED BELOW ARE THE CHOICES FOR HOW YOU CAN HAVE THE RELATION USER DISPLAYED.

1) SCREEN OUTPUT
2) PRINTER OUTPUT
3) RETURN TO PREVIOUS MENU

ENTER YOUR CHOICE (1-3) FROM ABOVE: ___

Figure 4.29 Output Selection Panel

91
<table>
<thead>
<tr>
<th>Tuple No.</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>USER</td>
<td>PAY-DEPT</td>
</tr>
<tr>
<td>ID NAME</td>
<td>PAYROLL DEPARTMENT</td>
</tr>
<tr>
<td>DESCRIPT</td>
<td>The department within the organization that produces the company's weekly and monthly payroll.</td>
</tr>
<tr>
<td>DATE ADDED</td>
<td>06/01/85</td>
</tr>
<tr>
<td>ADDED BY</td>
<td>Robert A. Kirsch II</td>
</tr>
<tr>
<td>COMMENTS</td>
<td>This is a standard attribute of the IRDS.</td>
</tr>
<tr>
<td>LST MOD DT</td>
<td>06/01/85</td>
</tr>
<tr>
<td>LST MOD BY</td>
<td>Kirsch</td>
</tr>
<tr>
<td>NUM OF MOD</td>
<td>881</td>
</tr>
<tr>
<td>LOCATION</td>
<td>Schema</td>
</tr>
<tr>
<td>SECURITY</td>
<td>none</td>
</tr>
</tbody>
</table>

PRESS RETURN TO SEE THE NEXT TUPLE.

Figure 4.30 Entity Output
Figure 4.31 Relationship Output Selection Panel
1.2.1.1.0

INFORMATION RESOURCE DICTIONARY SYSTEM
RELATIONSHIP OUTPUT

LISTED BELOW ARE THE CHOICES FOR HOW YOU CAN HAVE THE RELATIONSHIP.

USER PROCESSES SYSTEM DISPLAYED.
1) SCREEN OUTPUT
2) PRINTER OUTPUT
3) RETURN TO PREVIOUS MENU

ENTER YOUR CHOICE (1-3) FROM ABOVE: _

Figure 4.32 Output Selection Panel

USER PROCESSES SYSTEM

RECORD 1

USER ACCESS NAME: PAY-DEPT
SYSTEM ACCESS NAME: SAL-PAY

PRESS RETURN TO SEE NEXT TUPLE

Figure 4.33 Relationship Output
3. Schema

The IRDS prototype also allows the user to display the schema for all entity-types and relationship-types. In order to execute this portion of the prototype the user selects SCHEMA OUTPUT from the main menu (Figure 4.10). Once the selection has been made, the system executes the schema output module and requests that the user choose which type of schema to output and it's ACCESS-NAME (Figure 4.34 and 4.35). The user is then prompted by means of a panel to select the output medium. The system will then display the requested schema structure (Figure 4.36 and 4.37). Figures 4.36 and 4.37 provide below depict the output of an entity-type schema. The process for displaying a relationship schema is identical and will not be explained further here.

G. QUERY

The IRDS prototype query function give the prototype user the ability to generate ad hoc queries about any relationships that the system
### Entity Select Panel

**1.1.3.0.0**

**INFORMATION RESOURCE DICTIONARY SYSTEM**

**ENTITY SCHEMA OUTPUT**

1) USER  
2) SYSTEM  
3) PROGRAM  
4) MODULE  
5) DOCUMENT  
6) FILE  
7) RECORDS  
8) ELEMENT  
9) RETURN TO PREVIOUS MENU  
10) RETURN TO MAIN MENU

ENTER YOUR CHOICE (1-10) FROM ABOVE: 1

---

### Output Selection Panel

**1.5.1.1.0.0**

**INFORMATION RESOURCE DICTIONARY SYSTEM**

**ENTITY SCHEMA OUTPUT**

LISTED BELOW ARE THE CHOICES FOR HOW YOU CAN HAVE THE SCHEMA FOR RELATION USER DISPLAYED.

1) SCREEN OUTPUT  
2) PRINTER OUTPUT  
3) RETURN TO PREVIOUS MENU

ENTER YOUR CHOICE (1-3) FROM ABOVE: 1
A RELATIONAL DATA DICTIONARY COMPATIBLE WITH THE NATIONAL BUREAU OF STAND. (U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA R A KIRSCH DEC 85
maintains. Not all of the IRDS standard relationships are im-
plemented in this version of the IRDS prototype (See Appendix A for a

<table>
<thead>
<tr>
<th>Field</th>
<th>Field name</th>
<th>Type</th>
<th>Width</th>
<th>Dec</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>USER</td>
<td>Logical</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>ACC_NAME</td>
<td>Character</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>ID_NAME</td>
<td>Character</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>DESCRIPT</td>
<td>Character</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>DATE_ADDED</td>
<td>Date</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>ADDED_BY</td>
<td>Character</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>COMMENTS</td>
<td>Character</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>LST_MOD_DT</td>
<td>Date</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>LST_MOD_BY</td>
<td>Character</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>NUM_OF_MOD</td>
<td>Numeric</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

Figure 4.37 Sample Schema Output

list of the allowable relationships). The remainder of the relationships
will be reserved for implementation in subsequent versions of the proto-
type. The prototype uses a keyword selection process to generate a query
of the form SUBJECT-VERB-OBJECT and a query processor to process the
query and generate the resulting output. When the user selects the
query option from the main menu (Figure 4.10), the system executes the
query module and present a panel (See Figure 4.38) requesting that the
user choice which entity-type is to be the subject of the query. The
system then requires the user to enter the ACCESS-NAME of the entity
to be queried and select whether entries are to be verified before
Figure 4.38 Query Entity-Type Selection Menu
Figure 4.39 Entity-1 Selection Menu

Figure 4.40 Relationship Selection Menu
being accepted by the system (Figure 4.39). The system next prompts the user for the relationship-type that is the verb of the query. Finally the system request the entity-type which acts as the object to form the query (Figures 4.40 and 4.41). When the final form of the

---

**Figure 4.41 Entity-2 Selection Menu**

query has been specified the system process the query, requests the selection of an output medium for the query results, and then generates the output (See Figure 4.42 thru 4.43).

H. SCHEMA MAINTENANCE

Even though the Core IRDS Standard Schema limits entity and relationship meta-data (See Appendix A), it allows for extensibility in that additional attributes may be added by the user. The schema maintenance facility of the IRDS prototype allows an authorized (authorization is determined through the security function) user to add new attributes and modify or delete existing ones. Note: That although the prototype
I.2.1.0.0

INFORMATION RESOURCE DICTIONARY SYSTEM

ENTITY OUTPUT

LISTED BELOW ARE THE CHOICES FOR HOW
YOU CAN HAVE THE QUERY

PAY-DEPT RESPONSIBLE FOR SYSTEM
1) SCREEN OUTPUT
2) PRINTER OUTPUT
3) RETURN TO PREVIOUS MENU

ENTER YOUR CHOICE (1-3) FROM ABOVE: __

Figure 4.42 Output Selection Panel

I.3.9.1.0.0

INFORMATION RESOURCE DICTIONARY SYSTEM

QUERY RESULTS FOR
PAY-DEPT RESPONSIBLE FOR SYSTEM

IDENTIFICATION NAME: SALARY PAYROLL
DESCRIPTION: This system is used to produce the monthly sal-
ary payroll for the company.

IDENTIFICATION NAME: WEEKLY PAYROLL
DESCRIPTION: This system is used to produce the weekly pay-
roll for the company.

Figure 4.43 Query Result Panel
allows for the addition of entity and relationship relations
the panel structure would require modification to make full use of any
relations that were added. When the user selects the schema maintenance
option from the main menu (See Figure 4.10), the system activates the
schema maintenance module and displays a panel requesting that the user
choose which type of schema is to be modified (Figure 4.44).

```
1.4.0.0.0
INFORMATION RESOURCE DICTIONARY SYSTEM
SCHEMA MAINTENANCE MENU
1) ADD, MODIFY OR DELETE ENTITY SCHEMA
2) ADD, MODIFY OR DELETE RELATIONSHIP SCHEMA
ENTER YOUR CHOICE (1-3) FROM ABOVE: 1
```

Figure 4.44 Schema Maintenance Selection Panel

The user will then be allowed to identify particular entity or relation-
ship type and perform maintenance. The following sections describe how
the IRDS prototype performs the schema maintenance functions of the IRDS
prototype.

1. **Entity Meta-data**

   When the authorized user indicates that he desires to add, modify
or delete meta-data associated with the entity-type schema, the system
presents a panel requesting that the user choose which entity he desires to maintain (Figure 4.45).

Figure 4.45 Delete Entity Schema Maintenance Panel

a. Adding, Modifying or Deleting Entity Meta-data

After the user identifies which entity-type is to be maintained, the system retrieves the schema structure, displays it, and allows the authorized user to perform the desired maintenance (See Figure 4.46).

2. Relationship Meta-data

When the authorized user indicates that he desires to add, modify, or delete meta-data associated with the entity-type schema, the system
presents a panel requesting that the user choose which entity he desires to maintain (Figure 4.47).

a. Adding, Modifying or Deleting Relationship

After the user has identified which relationship-type is to be maintained, the system retrieves the schema structure, displays it, and allows the authorized user to perform the desired maintenance (Figure 4.48).

I. FINAL COMMENTS

Although this prototype IRDS does not possess all of the features that were described in Chapter 3, it does demonstrate that a relational DBMS-dependent implementation of the NBS IRDS is feasible as demonstrated by the prototype. The extensibility feature described in the standard
1.4.2.0.0  INFORMATION RESOURCE DICTIONARY SYSTEM

ADD, MODIFY OR DELETE RELATIONSHIP SCHEMA

1) USER CONTAINS SYSTEM  8) FILE CONTAINS RECORDS
2) SYSTEM CONTAINS PROGRAM  9) RECORD CONTAINS ELEMENT
3) PROGRAM PROCESSES FILE 10) USER RESPONSIBLE FOR SYSTEM
4) PROGRAM PROCESSES RECORD 11) USER RESPONSIBLE FOR FILE
5) PROGRAM PROCESSES ELEMENT 12) PROGRAM PRODUCES DOCUMENT
6) SYSTEM CONTAINS PROGRAM 13) RETURN TO PREVIOUS MENU
7) PROGRAM CONTAINS MODULE 14) RETURN TO MAIN MENU

ENTER YOUR CHOICE (1-14) FROM ABOVE:

Figure 4.47 Relationship Selection Panel

C:USER.dbf

Bytes remaining  3768
Fields defined  10

Field name Type Width Dec

1  U_NAME Character  10
2  S_NAME Character  10

Names start with letter; the remainder may be letters, digits or underscore
DEPRESS FI FOR INSTRUCTIONS

Figure 4.48 Relationship Schema Maintenance Panel
is enhanced because of the inherent flexibility of the relational environment. Finally the NBS standards provide a firm foundation from which to consider dictionary system implementation.
V. CONCLUSION

This thesis has discussed and evaluated the value of data as a corporate asset and how Data Base Management Systems (DBMS) can be used to manipulate this corporate asset. It has described how the concern over corporate data has led to the development and increased use of Relational Data Bases and in particular Data Dictionaries (DD). Desirable DBMS and DD characteristics, capabilities and features were identified and discussed. Two existing relational DBMS were evaluated concerning the data dictionary features they provided. The result of that evaluation was that relational systems lack a majority of those dictionary features deemed necessary and desirable. Further, all existing DD products were developed.

This thesis then presented, described and discussed the National Bureau of Standards (NBS) Information Resource Dictionary System (IRDS) standard. The standard provides a synthesis of baseline features, capabilities and functions found in existing DD systems plus the additional capabilities of being able to handle all three major types of data base organization: hierarchical, network and relational. Of equal significance, it offers the flexibility for user to expand the dictionary schema to accommodate unique requirements.

This thesis developed a relational model of the NBS IRDS which was implemented as a prototype using a personal computer and dBase III. The prototype demonstrates that the features presented as part of the NBS IRDS are implementable and usable in a relational environment.
It is recommended that the IRDS prototype undergo additional development with the goal of implementing an operational production version of the NBS IRDS standard.
APPENDIX A
CORE STANDARD SCHEMA

This appendix describes the Core System-Standard Schema and its structural characteristics. The Core System-Standard Schema is defined as that specific set of entity-types, relations-types, attribute-types, and other schema descriptors supported by the Core Standard IRDS. While this Core System-Standard Schema satisfies the requirements of many IRDS environments, an organization can customize its IRDS Schema using the Schema Extensibility Facility discussed in previous chapters.

A.1 ATTRIBUTE-TYPES AND ENTITY-TYPES

In this section, the attribute-types and attribute-group-types associated with each entity-type are given. The following are the entity-types in the Core System-Standard Schema:

* USER
* SYSTEM
* PROGRAM
* MODULE
* FILE
* DOCUMENT
* RECORD
* ELEMENT
* BIT-STRING
* CHARACTER-STRING
* FIXED-POINT
* FLOAT

The other entity-types found in the Core System-Standard Schema are:

* DICTIONARY-USER, in support of the Security Facility.
* VIEW which supports the Security and View Facilities.

The following two tables present the attribute-types and attribute-group-types associated with the non-security related entity-types listed above. Attribute-group-types can be identified by the existence of their component attribute-types, which are indented and immediately follow the attribute-group-type name. At the intersection of a row and column, the following denote that an entity of the given type:

S Can have no more than a single attribute of the given type.
Can have multiple attributes of the given type.

The first table shows the attribute-types associated with the following entity-types:

<table>
<thead>
<tr>
<th>ATTRIBUTE-GROUP-TYPE</th>
<th>ENTITY-TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD-EDITED-By</td>
<td>USR SYS PGM MDL FIL DOC REC ELE</td>
</tr>
<tr>
<td>ALLOWABLE-RANGE</td>
<td>P</td>
</tr>
<tr>
<td>LOW-OF-RANGE</td>
<td></td>
</tr>
<tr>
<td>HIGH-OF-RANGE</td>
<td></td>
</tr>
<tr>
<td>ALLOWABLE-VALUE</td>
<td>P</td>
</tr>
<tr>
<td>CLASSIFICATION</td>
<td>P P P P P P P P</td>
</tr>
<tr>
<td>CODE-LIST-LOCATION</td>
<td>P</td>
</tr>
<tr>
<td>COMMENTS</td>
<td>S S S S S S S S</td>
</tr>
<tr>
<td>DATA-CLASS</td>
<td>S</td>
</tr>
<tr>
<td>DATE-ADDED</td>
<td>S S S S S S S S</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>S S S S S S S S</td>
</tr>
<tr>
<td>DOCUMENT-CATEGORY</td>
<td>S</td>
</tr>
<tr>
<td>DURATION</td>
<td>S S S</td>
</tr>
<tr>
<td>DURATION-VALUE</td>
<td></td>
</tr>
<tr>
<td>DURATION-TYPE</td>
<td></td>
</tr>
<tr>
<td>IDENTIFICATION-NAME</td>
<td>P P P P P P P P</td>
</tr>
<tr>
<td>ALTERNATE-NAME</td>
<td></td>
</tr>
<tr>
<td>ALTERNATE-NAME-CONTEXT</td>
<td></td>
</tr>
<tr>
<td>LAST-MODIFICATION-DATE</td>
<td>S S S S S S S S</td>
</tr>
<tr>
<td>LAST-MODIFIED-By</td>
<td>S S S S S S S S</td>
</tr>
</tbody>
</table>
A.2 RELATIONSHIP-CLASS-TYPES AND RELATIONSHIP-TYPES

This section presents the relationship-class-types and relationship-types in the Core System-Standard Schema. The relationship-class-types, where they exist, are provided in bold print as headers to the relationship-types to which they apply. The inverse-name (which allows the specification of the member entity-types in reverse order) and abbreviated inverse-name are given for each relationship-class-type, so the inverse-name and abbreviated inverse-name for each relationship-type may be inferred. Where no relationship-class-type applies to a particular relationship-type, its inverse-name and abbreviated inverse-name are given directly.

<table>
<thead>
<tr>
<th>(ATTRIBUTE-GROUP-TYPE)</th>
<th>ABBREVIATION</th>
<th>INVERSE-NAME</th>
<th>ABBREVIATED INVERSE-NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONTAINS</td>
<td>CON</td>
<td>CON-CONTAINED-IN</td>
<td>CON-IN</td>
</tr>
<tr>
<td>SYSTEM-CONTAINS-SYSTEM</td>
<td>SYS-CON-SYS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYSTEM-CONTAINS-PROGRAM</td>
<td>SYS-CON-PSM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYSTEM-CONTAINS-MODULE</td>
<td>SYS-CON-MDL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROGRAM-CONTAINS-PROGRAM</td>
<td>PGM-CON-PSM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROGRAM-CONTAINS-MODULE</td>
<td>PGM-CON-MDL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MODULE-CONTAINS-MODULE</td>
<td>MDL-CON-MDL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FILE-CONTAINS-FILE</td>
<td>FIL-CON-FIL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FILE-CONTAINS-DOCUMENT</td>
<td>FIL-CON-DOC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FILE-CONTAINS-RECORD</td>
<td>FIL-CON-REC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FILE-CONTAINS-ELEMENT</td>
<td>FIL-CON-ELE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOCUMENT-CONTAINS-DOCUMENT</td>
<td>DOC-CON-DOC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOCUMENT-CONTAINS-RECORD</td>
<td>DOC-CON-REC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOCUMENT-CONTAINS-ELEMENT</td>
<td>DOC-CON-ELE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RECORD-CONTAINS-RECORD</td>
<td>REC-CON-REC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RECORD-CONTAINS-ELEMENT</td>
<td>REC-CON-ELE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ELEMENT-CONTAINS-ELEMENT</td>
<td>ELE-CONT-ELE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------------------</td>
<td>--------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROCESSES</td>
<td>PR PROCESSED-BY PR-BY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>USER-PROCESSES-FILE</td>
<td>USREC-PR-FIL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>USER-PROCESSES-DOCUMENT</td>
<td>USR-PR-DOC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>USER-PROCESSES-RECORD</td>
<td>USR-PR-REC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>USER-PROCESSES-ELEMENT</td>
<td>USR-PR-ELE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYSTEM-PROCESSES-FILE</td>
<td>SYS-PR-FIL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYSTEM-PROCESSES-DOCUMENT</td>
<td>SYS-PR-DOC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYSTEM-PROCESSES-RECORD</td>
<td>SYS-PR-REC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYSTEM-PROCESSES-ELEMENT</td>
<td>SYS-PR-ELE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROGRAM-PROCESSES-FILE</td>
<td>PGM-PR-FIL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROGRAM-PROCESSES-DOCUMENT</td>
<td>PGM-PR-DOC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROGRAM-PROCESSES-RECORD</td>
<td>PGM-PR-REC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROGRAM-PROCESSES-ELEMENT</td>
<td>PGM-PR-ELE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MODULE-PROCESSES-FILE</td>
<td>MDL-PR-FIL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MODULE-PROCESSES-DOCUMENT</td>
<td>MDL-PR-DOC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MODULE-PROCESSES-RECORD</td>
<td>MDL-PR-REC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MODULE-PROCESSES-ELEMENT</td>
<td>MDL-PR-ELE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RESPONSIBLE-FOR</td>
<td>R-FOR RESPONSIBILITY-OF R-OF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>USER-RESPONSIBLE-FOR-SYSTEM</td>
<td>USR-R-FOR-SYS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>USER-RESPONSIBLE-FOR-PROGRAM</td>
<td>USR-R-FOR-PGM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>USER-RESPONSIBLE-FOR-MODULE</td>
<td>USR-R-FOR-MDL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>USER-RESPONSIBLE-FOR-RECORD</td>
<td>USR-R-FOR-REC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>USER-RESPONSIBLE-FOR-DOCUMENT</td>
<td>USR-R-FOR-DOC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>USER-RESPONSIBLE-FOR-RECORD</td>
<td>USR-R-FOR-REC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>USER-RESPONSIBLE-FOR-ELEMENT</td>
<td>USR-R-FOR-ELE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RUNS</td>
<td>RUN-BY RUN-BY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>USER-RUNS-SYSTEM</td>
<td>USR-RUN-SYS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>USER-RUNS-PROGRAM</td>
<td>USR-RUN-PGM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>USER-RUNS-MODULE</td>
<td>USR-RUN-MDL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GOES-TO</td>
<td>TO COMES-FROM FR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYSTEM-GOES-TO-SYSTEM</td>
<td>SYS-TO-SYS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROGRAM-GOES-TO-PROGRAM</td>
<td>PGM-TO-PGM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MODULE-GOES-TO-MODULE</td>
<td>MDL-TO-MDL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DERIVED-FROM</td>
<td>D-FR PRODUCES PRD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOCUMENT-DERIVED-FROM-FILE</td>
<td>DOC-D-FR-FIL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOCUMENT-DERIVED-FROM-DOCUMENT</td>
<td>DOC-D-FR-DOC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOCUMENT-DERIVED-FROM-RECORD</td>
<td>DOC-D-FR-REC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ELEMENT-DERIVED-FROM-FILE</td>
<td>ELE-D-FR-FIL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ELEMENT-DERIVED-FROM-DOCUMENT</td>
<td>ELE-D-FR-DOC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ELEMENT-DERIVED-FROM-RECORD</td>
<td>ELE-D-FR-REC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ELEMENT-DERIVED-FROM-ELEMENT</td>
<td>ELE-D-FR-ELE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FILE-DERIVED-FROM-DOCUMENT</td>
<td>FIL-D-FR-DOC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FILE-DERIVED-FROM-FILE</td>
<td>FIL-D-FR-FIL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RECORD-DERIVED-FROM-DOCUMENT</td>
<td>REC-D-FR-DOC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RECORD-DERIVED-FROM-FILE</td>
<td>REC-D-FR-FIL</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

112
A.3 ENTITY-TYPES AND RELATIONSHIP-TYPES

The following two tables depict the entity-types participating as members of the non-security related relationship-types in the Core System-Standard Schema. The following notation is to denote that the entity-type is:

1 The first member of the relationship-type.
2 The second member of the relationship-type.
R Both the first and second member of the relationship-type

The first table shows the relationship-types associated with the following entity-types:

* USER
* SYSTEM
* PROGRAM
* MODULE
* FILE
* DOCUMENT
* RECORD
* ELEMENT
<table>
<thead>
<tr>
<th>RELATIONSHIP-CLASS-TYPE AND RELATIONSHIP-TYPE</th>
<th>USR</th>
<th>SYS</th>
<th>PGM</th>
<th>MDL</th>
<th>FIL</th>
<th>DOC</th>
<th>REC</th>
<th>ELE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONTAINS SYSTEM CONTAINS SYSTEM</td>
<td>.</td>
<td>R</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>CONTAINS SYSTEM CONTAINS PROGRAM</td>
<td>.</td>
<td>1</td>
<td>2</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>CONTAINS SYSTEM CONTAINS MODULE</td>
<td>.</td>
<td>1</td>
<td>2</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>CONTAINS PROGRAM CONTAINS PROGRAM</td>
<td>.</td>
<td>.</td>
<td>R</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>CONTAINS PROGRAM CONTAINS MODULE</td>
<td>.</td>
<td>1</td>
<td>2</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>CONTAINS MODULE CONTAINS MODULE</td>
<td>.</td>
<td>.</td>
<td>R</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>CONTAINS FILE</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>R</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>CONTAINS DOCUMENT</td>
<td>.</td>
<td>.</td>
<td>1</td>
<td>2</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>CONTAINS RECORD</td>
<td>.</td>
<td>.</td>
<td>1</td>
<td>2</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>CONTAINS ELEMENT</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>1</td>
<td>2</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>CONTAINS DOCUMENT</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>1</td>
<td>2</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>CONTAINS RECORD</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>CONTAINS ELEMENT</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>R</td>
</tr>
<tr>
<td>PROCESSES USER PROCESSES-FILE</td>
<td>1</td>
<td>.</td>
<td>.</td>
<td>2</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>PROCESSES USER PROCESSES-DOCUMENT</td>
<td>1</td>
<td>.</td>
<td>.</td>
<td>2</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>PROCESSES USER PROCESSES-RECORD</td>
<td>1</td>
<td>.</td>
<td>.</td>
<td>2</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>PROCESSES USER PROCESSES-ELEMENT</td>
<td>1</td>
<td>.</td>
<td>.</td>
<td>2</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>PROCESSES SYSTEM PROCESSES-FILE</td>
<td>1</td>
<td>.</td>
<td>.</td>
<td>2</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>PROCESSES SYSTEM PROCESSES-DOCUMENT</td>
<td>1</td>
<td>.</td>
<td>.</td>
<td>2</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>PROCESSES SYSTEM PROCESSES-RECORD</td>
<td>1</td>
<td>.</td>
<td>.</td>
<td>2</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>PROCESSES SYSTEM PROCESSES-ELEMENT</td>
<td>1</td>
<td>.</td>
<td>.</td>
<td>2</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>PROCESSES PROGRAM PROCESSES-FILE</td>
<td>1</td>
<td>.</td>
<td>.</td>
<td>2</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>PROCESSES PROGRAM PROCESSES-DOCUMENT</td>
<td>1</td>
<td>.</td>
<td>.</td>
<td>2</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>PROCESSES PROGRAM PROCESSES-RECORD</td>
<td>1</td>
<td>.</td>
<td>.</td>
<td>2</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>PROCESSES PROGRAM PROCESSES-ELEMENT</td>
<td>1</td>
<td>.</td>
<td>.</td>
<td>2</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>PROCESSES MODULE PROCESSES-FILE</td>
<td>.</td>
<td>.</td>
<td>1</td>
<td>2</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>PROCESSES MODULE PROCESSES-DOCUMENT</td>
<td>.</td>
<td>.</td>
<td>1</td>
<td>2</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>PROCESSES MODULE PROCESSES-RECORD</td>
<td>.</td>
<td>.</td>
<td>1</td>
<td>2</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>PROCESSES MODULE PROCESSES-ELEMENT</td>
<td>.</td>
<td>.</td>
<td>1</td>
<td>2</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>RESPONSIBLE-FOR USER RESPONSIBLE-FOR-SYSTEM</td>
<td>1</td>
<td>2</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>RESPONSIBLE-FOR USER RESPONSIBLE-FOR-PROGRAM</td>
<td>1</td>
<td>2</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>RESPONSIBLE-FOR USER RESPONSIBLE-FOR-MODULE</td>
<td>1</td>
<td>2</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>RESPONSIBLE-FOR USER RESPONSIBLE-FOR-RECORD</td>
<td>1</td>
<td>2</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>RESPONSIBLE-FOR USER RESPONSIBLE-FOR-DOCUMENT</td>
<td>1</td>
<td>2</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>RESPONSIBLE-FOR USER RESPONSIBLE-FOR-RECORD</td>
<td>1</td>
<td>2</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>RESPONSIBLE-FOR USER RESPONSIBLE-FOR-ELEMENT</td>
<td>1</td>
<td>2</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>RUNS</td>
<td>USER-RUN-SYSTEM</td>
<td>USER-RUN-PROGRAM</td>
<td>USER-RUN-MODULE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------</td>
<td>------------------</td>
<td>-----------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GOES-TO</th>
<th>SYSTEM-GOES-TO-SYSTEM</th>
<th>PROGRAM-GOES-TO-PROGRAM</th>
<th>MODULE-GOES-TO-MODULE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DERIVED-FROM</th>
<th>DOCUMENT-DERIVED-FROM-FILE</th>
<th>DOCUMENT-DERIVED-FROM-DOCUMENT</th>
<th>DOCUMENT-DERIVED-FROM-RECORD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CALLS</th>
<th>PROGRAM-CALLS-PROGRAM</th>
<th>PROGRAM-CALLS-MODULE</th>
<th>MODULE-CALLS-MODULE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>ELEMENT-STANDARD-FOR-ELEMENT</th>
<th>FILE-HAS-SORT-KEY-ELEMENT</th>
<th>FILE-HAS-ACCESS-KEY-ELEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

The last three relationship-types are not members of a relationship-class, and so are listed separately.

The second table shows the relationship-types associated with the following entity-types:

* ELEMENT
* BIT-STRING
* CHARACTER-STRING
* FIXED-POINT
* FLOAT

<table>
<thead>
<tr>
<th>RELATIONSHIP-CLASS-TYPE</th>
<th>ELE</th>
<th>BIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>RELATIONSHIP-TYPE</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

115
A.4 ATTRIBUTE-TYPES AND RELATIONSHIP-TYPES

The following are the attribute-types associated with the relationship-class-types and relationship-types in the Core System-Standard Schema:

* The relationship-types
  - SYSTEM-PROCESSES-FILE
  - PROGRAM-PROCESSES-FILE
  - MODULE-PROCESSES-FILE
  have the single-valued attribute-type ACCESS-METHOD associated with them.

* All PROCESSES and RUNS relationship-types have the single-valued attribute-type FREQUENCY associated with them.

* The relationship-type RECORDS-CONTAINS-ELEMENT has the single-valued attribute-type RELATIVE-POSITION associated with it.

* The relationship-type ELEMENT-REPRESENTED-AS-BIT-STRING has the single-valued attribute-type LENGTH and the multiple-valued attribute-type USAGE associated with it.

* The relationship-type ELEMENT-REPRESENTED-AS-CHARACTER-STRING has the single-valued attribute-types LENGTH and JUSTIFICATION and the multiple-valued attribute-type USAGE associated with it.

* The relationship-types
  - ELEMENT-REPRESENTED-AS-FIXED-POINT
  - ELEMENT-REPRESENTED-AS-FLOAT
  have the single-valued attribute-types LENGTH, PRECISION, and SCALE, and the multiple-valued attribute-type USAGE associated with them.

A.5 SUPPORT FOR THE CORE SECURITY FACILITY

In addition to the entity-types DICTIONARY-USER and VIEW the Core System-Standard Schema also contains the relationship-type DICTIONARY-USER-HAS-VIEW, which associates a IRDS user with the views he/she may use. A number of
attributes-types and attribute-group-types in the Core System-Standard schema are used to specify the categories of permissions that can be assigned to a IRDS user with a particular view.

A.6 THE ATTRIBUTE-TYPE-VALIDATION-PROCEDURE META ENTITIES

The Core System-Standard Schema contains the following two attribute-type-validation-procedure meta-entities:

* RANGE-VALIDATION, used to restrict the attributes of a given attribute-type to a predefined set of ranges.
* VALUE-VALIDATION, used to restrict the attributes of a given attribute-type to a predefined set of values.

A.7 THE ATTRIBUTE-TYPE-VALIDATION-DATA META-ENTITIES

There are no attribute-type-validation-data meta-entities specified in the Core System-Standard Schema. To use this feature, an organization must define and add these meta-entities to the schema.

A.8 THE LIFE-CYCLE-PHASE META-ENTITIES

The Core System-Standard Schema contains four Life-Cycle-Phase meta-entities. These are:

* UNCONTROLLED-PHASE - Entities are in this life-cycle-phase when they are added to the IRD.
* CONTROLLED-PHASE - Entities used in an operational environment, for which structural integrity controls are provided by the IRDS, are in this life-cycle-phase.
* ARCHIVED-PHASE - This life-cycle-phase is used to document those entities no longer in use.
* SECURITY-PHASE - This life-cycle-phase, of phase class UNCONTROLLED is used for DICTIONARY-USER entities associated with the Security Facility of the Core Standard IRDS.

A.9 THE QUALITY-INDICATOR META-ENTITIES

The Core System-Standard Schema does not contain any pre-defined QUALITY-INDICATOR meta-entities. These meta-entities may be defined by an organization.

A.10 THE VARIATION-NAME META-ENTITIES

There are also no pre-defined VARIATION-NAME meta-entities in the Core System-Standard Schema. These meta-entities may be defined by an organization.
A.11 THE SCHEMA-DEFAULTS META-ENTITIES

There is one SCHEMA-DEFAULTS meta-entity in the Core System-Standard Schema. This meta-entity, called EXISTING-SCHEMA-DEFAULTS, is used to establish minimum and maximum name lengths and minimum and maximum attribute lengths in IRD.
APPENDIX B
COMMAND SPECIFICATIONS

SYNTAX:
All words shown in captials are required.
[] = Optional
<> = user supplied
() = May be repeated as required

1. Schema Commands

1.1 Schema Maintenance

* Add Meta-Entity Command

ADD META-ENTITY <Meta-entity-name>
META-ENTITY-TYPE = <Meta-entity-type>
WITH META-ATTRIBUTES
[<Meta-attribute-name> = <Initial value>];

* Modify Meta-Entity Command

MODIFY META-ENTITY <Meta-entity-name>
WITH META-ATTRIBUTES
(<Meta-attribute-name> = <New value>);

* Delete Meta-Entity Command

DELETE META-ENTITY <Meta-entity-name>;

* Add Meta-Relationship Command

ADD META-RELATIONSHIP
FROM <Meta-entity-name-1> TO <Meta-entity-name-2>
WITH META-ATTRIBUTES
[<Meta-attribute-name> = <value>];

* Modify Meta-Relationship Command

MODIFY META-RELATIONSHIP
FROM <Meta-entity-name-1> TO <Meta-entity-name-2>
WITH META-ATTRIBUTES
(<Meta-attribute-name> = <value>)
[<Meta-entity-name-1> = <Meta-entity-name>]
[<Meta-entity-name-2> = <Meta-entity-name>]
[<Meta-attribute-name> = <New-value>];

* Delete Meta-Relationship Command

DELETE META-RELATIONSHIP
FROM <Meta-entity-name-1> TO <Meta-entity-name-2>
WITH META-ATTRIBUTES
[<Meta-attribute-name> = <value>];
* Replace Meta-Relationship Command

REPLACE META-RELATIONSHIP FROM <Meta-entity-name-1> TO <Meta-entity-name-2> WITH META-ATTRIBUTES [[<Meta-attribute-name-1> = <value>]] BY FROM <Meta-entity-name-1> TO <Meta-entity-name-3> WITH META-ATTRIBUTES [[<Meta-attribute-name-2> = <value>]];

* Modify Meta-Entity Name Command

MODIFY META-ENTITY-NAME FROM <Meta-entity-name-1> TO <Meta-entity-name-2>;

* Install Meta-Entity Command

INSTALL <Meta-entity-name>;

1.2 Schema Output Command

* OUTPUT SCHEMA

SELECT [ALL] or [<meta-entity-name-list>] [WHERE <restriction-expression> boolean operator <restriction-expression>] [<Title>] [SHOW ALL] or [SHOW ALL META-ATTRIBUTES or <Meta-attribute-list>] and/or [SHOW ALL META-RELATIONSHIPS or <Meta-relationships-list>] and/or [ROUTE TO <Destination-list>];

2. Dictionary Commands

2.1 Dictionary Maintenance Commands

* Add Entity Command

ADD ENTITY <entity-name>
ENTITY-TYPE = <entity-type>
WITH ATTRIBUTES [[<attribute-name> = <Initial value>]];

* Modify Entity Command

MODIFY ENTITY <entity-name> [[<attribute-name> = <New value>]];
* Delete Entity Command

DELETE ENTITY
   [Entity-name] or
   [USING = Entity-list-name] or
   [USING PROCEDURE = Procedure-name] or
   [SELECT WHERE (restriction-expression) boolean
    operator (restriction-expression)];

* Add Relationship Command

ADD RELATIONSHIP
   Entity-name-1 (Relationship-type) Entity-name-2
   WITH ATTRIBUTES
      [(<attribute-name> = <Initial value>)];

* Modify Relationship Command

MODIFY RELATIONSHIP
   Entity-name-1 (Relationship-type) Entity-name-2
   [(<attribute-name> = <New value>)];

* Delete Relationship Command

MODIFY RELATIONSHIP
   Entity-name-1 (Relationship-type) Entity-name-2
   (Relationship-list-name);

* Modify Access-Name Command

MODIFY ACCESS-NAME
   Current access-name TO New access-name;

* Modify Descriptive-Name Command

MODIFY DESCRIPTIVE-NAME
   Current descriptive-name TO New descriptive-name;

* Modify Entity Life-Cycle-Phase Command

MODIFY ENTITY LIFE-CYCLE-PHASE
   FOR Entity-name or Entity-list-name
   FROM Current life-cycle-phase TO New life-cycle-phase;

* Copy Entity Command

COPY ENTITY Entity-name
   [WITH RELATIONSHIPS]
   TO New entity-name
   [DESCRIPTION-NAME = (Descriptive-name)]
2.2 Dictionary Output Commands

* General Output Command

OUTPUT DICTIONARY
[USING VIEW = ALL]
[USING VIEW = (view-name) or (view-name-list)]
SELECT [ALL] or
[ENTITIES]
<restriction-expression>
<boolean operator>
<restriction-expression>
[SORT SEQUENCE = (sort-parm-list)]
SHOW <show-options>
[SHOW <Title>]
[ROUTE TO (destination-list)]
[PROCEDURE-NAME = (procedure-name)];

* Output Impact-of-Change Command

OUTPUT IMPACT
[USING VIEW = ALL]
[USING VIEW = (view-name) or (view-name-list)]
SELECT [ALL] or
[ENTITIES]
<restriction-expression>
<boolean operator>
<restriction-expression>
[SORT SEQUENCE = (sort-parm-list)]
SHOW <show-options>
[SHOW <Title>]
[SHOW LIFE-CYCLE-PHASE]
[SHOW QUALITY-INDICATOR]
[SHOW ATTRIBUTES [ALL] or [NO] or
[[<attribute-name>]]
[SHOW DESCRIPTIVE-NAME]
[ROUTE TO (destination-list)]
[PROCEDURE-NAME = (procedure-name)];

* Output Syntax Commands

OUTPUT SYNTAX
[USING VIEW = ALL]
[USING VIEW = (view-name) or (view-name-list)]
SELECT [ALL] or
[ENTITIES]
<restriction-expression>
<boolean operator>
<restriction-expression>
[SORT SEQUENCE = (sort-parm-list)]]
SHOW (show-options)
  [SHOW <Title>]
  [SHOW LIFE-CYCLE-PHASE]
  [SHOW QUALITY-INDICATOR]
  [SHOW RELATIONSHIP <relationship-display-options>]
  [SHOW RELATIONSHIP SYNTAX FOR EACH <entity-name>]
  [ROUTE TO <destination-list>]
  [PROCEDURE-NAME = <procedure-name>];

2.3 Dictionary Entity-List Commands

* Build Entity-List Command

BUILD ENTITY-LIST
SELECT [ALL] or
  [ENTITIES]
  <restriction-expression>
  <boolean operator>
  <restriction-expression>
  [LIST-NAME = <entity-list-name>]
  [USING VIEW = ALL]
  [USING VIEW = <view-name> or <view-name-list>]
  [PROCEDURE-NAME = <procedure-name>]
  [PROCEDURE-DESCRIPTION = <short-string-literal>];

* Entity-List Union Command

UNION
  <existing entity-list-name>,
  ({<existing entity-list-name>}
  = <new entity-list-name>);

* Entity-List Intersection Command

INTERSECTION
  <existing entity-list-name>,
  ({<existing entity-list-name>}
  = <new entity-list-name>);

* Entity-List Difference Command

DIFFERENCE
  <entity-list-1-name>,<entity-list-2-name>
  = <new entity-list-name>;

* Entity-List Subtraction Command

SUBTRACTION
  <entity-list-1-name>,<entity-list-2-name>
  = <new entity-list-name>;

123
* Name Current Entity-List Command

```
NAME CURRENT ENTITY-LIST <entity-list-name>;
```

* Output Entity-List Command

```
OUTPUT ENTITY-LIST
  [LIST-NAME = <entity-list-name>]
  [SHOW <Title>]
  [ROUTE TO <destination-list>];
```

* Output Entity-List Names Command

```
OUTPUT ENTITY-LIST NAME
  [SHOW <Title>]
  [ROUTE TO <destination-list>];
```

2.4 Dictionary Procedure Commands

* Output Procedure Syntax command

```
OUTPUT PROCEDURE SYNTAX
  ALL or <procedure-name>
  [SHOW <Title>]
  [ROUTE TO <destination-list>];
```

* Output Procedure Names Command

```
OUTPUT PROCEDURE-NAME
  [SHOW PROCEDURE-DESCRIPTION]
  [ROUTE TO <destination-list>];
```

* Run Output Procedure Command

```
RUN OUTPUT PROCEDURE <procedure-name>
  [USING VIEW = ALL]
  [USING VIEW = <view-name> or <view-name-list>]
  [ROUTE TO <destination-list>];
```

* Run Entity-List Procedure Command

```
RUN ENTITY-LIST PROCEDURE <procedure-name>
  [LIST-NAME = <entity-list-name>]
  [USING VIEW = ALL]
  [USING VIEW = <view-name> or <view-name-list>];
```

* Save Output Procedure Command

```
SAVE OUTPUT PROCEDURE
  PROCEDURE-NAME = <procedure-name>
  [PROCEDURE-DESCRIPTION = <short-string-literal>];
```
* Save Entity-List Procedure Command

```
SAVE ENTITY-LIST PROCEDURE
    PROCEDURE-NAME = <procedure-name>
    [PROCEDURE-DESCRIPTION = <short-string-literal>];
```

* Delete Procedure Command

```
DELETE <procedure-type> PROCEDURE <procedure-name>;
```

3. General Commands

3.1 IRD-IRD Interface Commands

* Create Dictionary Command

```
CREATE DICTIONARY <new-dictionary-name>
    [LOCATION CLAUSE <implementor-defined>]
    SCHEMA IS
        [IN DICTIONARY <dictionary-name>]
        [IN FILE <file-name>]
        [STANDARD]
    [LOAD DICTIONARY FROM <file-name>];
```

* Export Dictionary Command

```
EXPORT DICTIONARY
    [USING VIEW = ALL]
    [USING VIEW = <view-name> or <view-name-list>]
    [USING ENTITY-LIST = <Entity-list-name>]
    [EXCLUDE RELATIONSHIP OF [<relationship-type> or
        [<relationship-list-name>]]]
    [SCHEMA EXPORT FILE = <export-file-name>]
    [SYNTAX = <short-string-literal>];
```

* Check Schema Compatibility Command

```
CHECK SCHEMA
    [SOURCE] or [TARGET] SCHEMA IS
        [IN DICTIONARY <dictionary-name>]
        [IN FILE <file-name>]
        [STANDARD];
```

* Import Dictionary Command

```
IMPORT DICTIONARY
    SCHEMA EXPORT FILE = <export-file-name>
    DICTIONARY EXPORT FILE =
        <dictionary-export-file-name>
    [IN DICTIONARY <dictionary-name>]
    [IN FILE <file-name>]
```
3.2 Utility Commands

* Set Session Default Command

SET
[VIEW = <view-name>]
[MODE = <mode-type>]
[SHOW ATTRIBUTES
[ENCODED or DECODED]
[<implementor-defined-options>]]
[SAVE];

* Session Status Command

STATUS
[ALL]
[DICTIONARY]
[ENTITY-LIST]
[MODE]
[VIEWS]
[PROFILES]
[DEFAULTS]
[<implementor-defined-options>];

* Help Command

HELP
[ALL]
[<command-imperative-substring>];

* Exit Dictionary System Command

EXIT;

* Enter Panel Dialogue Command

PANEL NAME = <panel-name>;}
This appendix describes the new entity-types, relationship-types and attribute-types which can be added to the IRD to allow the system to map into NDL and SQL data structures. [Ref 9, pages 16-20]

D.1 NETWORK MODEL MAPPINGS

The following tables describe the mappings between the generic entity-types of the Core IRD and the Network Model entities and relationships.

<table>
<thead>
<tr>
<th>Network Data Model Mapping - Entity types</th>
</tr>
</thead>
<tbody>
<tr>
<td>NDL</td>
</tr>
<tr>
<td>Schema</td>
</tr>
<tr>
<td>Subschema</td>
</tr>
<tr>
<td>Database</td>
</tr>
<tr>
<td>Record</td>
</tr>
<tr>
<td>Set</td>
</tr>
<tr>
<td>Component</td>
</tr>
<tr>
<td>Module</td>
</tr>
<tr>
<td>Database Procedure</td>
</tr>
<tr>
<td>Data Type</td>
</tr>
<tr>
<td>-------------------------------------------</td>
</tr>
<tr>
<td>IRD Generic Model</td>
</tr>
<tr>
<td>Schema</td>
</tr>
<tr>
<td>Subschema</td>
</tr>
<tr>
<td>Database</td>
</tr>
<tr>
<td>Record *</td>
</tr>
<tr>
<td>Set</td>
</tr>
<tr>
<td>Element *</td>
</tr>
<tr>
<td>Module *</td>
</tr>
<tr>
<td>Program *</td>
</tr>
<tr>
<td>Element *</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Network Data Model Mappings - relationship-types</th>
</tr>
</thead>
<tbody>
<tr>
<td>NDL</td>
</tr>
<tr>
<td>Subschema in</td>
</tr>
<tr>
<td>Schema is</td>
</tr>
<tr>
<td>Owner is</td>
</tr>
<tr>
<td>Members are</td>
</tr>
<tr>
<td>Contains</td>
</tr>
<tr>
<td>Identifier</td>
</tr>
<tr>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>IRD Generic Model</td>
</tr>
<tr>
<td>Schema-Contains-Schema</td>
</tr>
<tr>
<td>Schema-Contains-Set</td>
</tr>
<tr>
<td>Schema-Contains-Record</td>
</tr>
<tr>
<td>Set-Owner-Contains-Record</td>
</tr>
<tr>
<td>Record-Contains-Element</td>
</tr>
<tr>
<td>Element-Contains-Element</td>
</tr>
<tr>
<td>Set-Has-Set-Key-Element</td>
</tr>
<tr>
<td>Record-Redefines-Record</td>
</tr>
<tr>
<td>Schema-Defines-Database</td>
</tr>
<tr>
<td>Element-Associated-With-Element</td>
</tr>
</tbody>
</table>

* Note - these are already defined in the Core IRDS Standard.

++ Note - these NDL relationships are in addition to those appearing in the IRDS Specifications in the Core IRDS. A relationship that is in the Core Standard does not appear here unless a different NDL relationship maps into it.
D.2 RELATIONAL MODEL MAPPINGS

The following tables show the mappings between the generic and Relational Model entities and relationships:

### Network Data Model Mapping - Entity types

<table>
<thead>
<tr>
<th>SQL</th>
<th>IRD Generic Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schema</td>
<td>Schema</td>
</tr>
<tr>
<td>Table</td>
<td>Record *</td>
</tr>
<tr>
<td>Column</td>
<td>Element *</td>
</tr>
<tr>
<td>Data Type</td>
<td>Element *</td>
</tr>
<tr>
<td>Query &amp; Operations</td>
<td>Set</td>
</tr>
<tr>
<td>(Join, Projection, etc.)</td>
<td></td>
</tr>
</tbody>
</table>

### Relational Data Model Mappings - Relationship-types

<table>
<thead>
<tr>
<th>SQL</th>
<th>IRD Generic Model ++</th>
</tr>
</thead>
<tbody>
<tr>
<td>Submodel id tables</td>
<td>Schema-Contains-Schema</td>
</tr>
<tr>
<td>is made up of Table</td>
<td>Schema-Contains-Set</td>
</tr>
<tr>
<td>identified by</td>
<td>Schema-Contains-Record</td>
</tr>
<tr>
<td>is made up of Columns</td>
<td>Record-Has-Access-Key-Element</td>
</tr>
<tr>
<td></td>
<td>Element-Identifies-Element</td>
</tr>
<tr>
<td></td>
<td>Element-Identifies-Record</td>
</tr>
<tr>
<td></td>
<td>Record-Contains-Element</td>
</tr>
<tr>
<td></td>
<td>Element-Associated-With-Element</td>
</tr>
</tbody>
</table>

* Note - these are already defined in the Core IRDS Standard.

++ Note - these SQL relationships are in addition to those appearing in the IRDS Specifications in the Core IRDS. A relationship that is in the Core Standard does not appear here unless a different NDL relationship maps into it.
D.3 ENTITY-TYPES AND RELATIONSHIP-TYPES

The following table identifies new and existing entity-types and gives their applicability to the SQL and NDL database models:

<table>
<thead>
<tr>
<th>Applicability Matrix of Entity-Types</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>New Entity type</td>
</tr>
<tr>
<td>SQL Model</td>
</tr>
<tr>
<td>NDL Model</td>
</tr>
</tbody>
</table>

The following table identifies new and existing relationship-types and gives their applicability to the SQL and NDL database models:

<table>
<thead>
<tr>
<th>Applicability Matrix of Relationship-Types</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>New Relationship-type</td>
</tr>
<tr>
<td>NDL Model</td>
</tr>
<tr>
<td>SQL Model</td>
</tr>
</tbody>
</table>

DESCRIPTION LEGEND:

- **ECE** = Relationship-Type "Element-Contains-Element"
- **RCE** = Relationship-Type "Record-Contains-Element"
- **RAE** = Relationship-Type "Record-Has-Access-Key-Element"
- **RRR** = Relationship-Type "Record-Redefines-Record"
- **SCS** = Relationship-Type "Schema-Contains-Schema"
- **SCT** = Relationship-Type "Schema-Contains-Set"
- **SOE** = Relationship-Type "Set-Has-Sort-Key-Element"
- **SMR** = Relationship-Type "Set-Member-Is-Record"
- **SOR** = Relationship-Type "Set-Owner-Is-Record"
- **EIR** = Relationship-Type "Element-Identifies-Record"
- **SDD** = Relationship-Type "Schema-Defines-Database"
- **EIE** = Relationship-Type "Element-Identifies-Element"
EAE = Relationship-Type "Element-Associated-With Element"
SCR = Relationship-Type "Schema-Contains-Record"

### D.4 ATTRIBUTE-TYPE ASSOCIATIONS

The following table depicts the association between attribute-types and the entity-types to which they apply. The "common" attribute-types defined as part of the Core Standard IRD apply as well.

<table>
<thead>
<tr>
<th>Attribute-Type</th>
<th>SCH</th>
<th>DBA</th>
<th>RCD</th>
<th>SET</th>
<th>ELM</th>
<th>MDL</th>
<th>PGM</th>
</tr>
</thead>
<tbody>
<tr>
<td>LANGUAGE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INITIAL-POPULATION</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RATE-OF-ARRIVALS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RATE-OF-DEPARTURES</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RATE-OF-ACCESS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RATE-OF-UPDATE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEFAULT-CLAUSE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>USAGE</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**DESCRIPTION LEGEND:**

SCH = Entity-type "SCHEMA"
DBS = Entity-type "DATABASE"
RCD = Entity-type "RECORD"
SET = Entity-type "SET"
ELM = Entity-type "ELEMENT"
MDL = Entity-type "MODULE"
PGM = Entity-type "PROGRAM"
The following table shows the attribute-types associated with relationship-types:

<table>
<thead>
<tr>
<th>Attribute-Type</th>
<th>SMR</th>
<th>RAEE</th>
<th>AEE</th>
<th>SOE</th>
<th>SCS</th>
<th>SCS</th>
<th>SOR</th>
<th>RRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCESS-METHOD</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KEY-SELECT</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ORDER-CLAUSE</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>INSERTION-MODE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RETENTION-MODE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>ORDER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DUPLICATES</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OCCURS-CLAUSE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LANGUAGE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>USAGE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

DESCRIPTION LEGEND:

- **SMR** = Relationship-type "SET-MEMBER-IS-RECORD"
- **RAE** = Relationship-type "RECORD-SA-ACCESS-KEY-ELEMENT"
- **EAE** = Relationship-type "ELEMENT-ASSOCIATION-WITH-ELEMENT"
- **SOE** = Relationship-type "SET-HAS-SORT-KEY-ELEMENT"
- **SCS** = Relationship-type "SCHEMA-CONTAINS-SCHEMA"
- **SCT** = Relationship-type "SCHEMA-CONTAINS-SET"
- **SOR** = Relationship-type "SET-OWNER-IS-RECORD"
- **RRR** = Relationship-type "RECORD-REDEFINES-RECORD"
APPENDIX E
IRDS PROTOTYPE SOFTWARE

MAIN.PRG

* MODULE NAME: MAIN
* INPUT FILES: NONE
* OUTPUT FILES: NONE
* ROUTINES THAT THE MODULE CALLS: 1.1.0.0.0.0, 1.2.0.0.0.0, 1.3.0.0.0.0,
  1.4.0.0.0.0, 1.5.0.0.0.0, 1.6.0.0.0.0
* LOCAL VARIABLES USED:
  * choice : CONTAINS THE NUMBER OF ACTION SELECTED.
  * one-time: USED TO INSURE THAT THE ASSOCIATED ROUTINE IS RUN ONLY ONCE.
  * t : REPRESENTS THE BOOLEAN TRUE IS USED TO CREATE A CONTINUES
    LOOP.
  * test : USED TO ALLOW THE USER TO TEST FOR CAPS LOCK DOWN.
* INPUT FILES: NONE
* OUTPUT FILES: NONE
* DESIGNED BY: ROBERT A. KIRSCH II
* WRITTEN BY: ROBERT A. KIRSCH II
* BASIC FUNCTION OF MODULE:
* THIS PROGRAM STARTS THE INFORMATION RESOURCE DICTIONARY SYSTEM
* IT ALLOW THE USER TO CHOOSE WHICH FUNCTION WITHIN THE SYSTEM
* HE WOULD LIKE TO DO.
* SET SAFETY OFF
STORE .t. TO one_time
DO WHILE one_time
CLEAR
STORE ' ' TO test
@ 1.1 SAY "MAIN"
@ 2.22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 11.31 SAY "PLEASE INSURE THAT YOU"
@ 12.31 SAY "HAVE THE 'CAPS LOCK '"
@ 13.31 SAY "ON AS ALL ANSWERS TO"
@ 14.31 SAY "QUESTIONS NEED TO BE"
@ 15.31 SAY "IN UPPER CASE"
@ 17.31 SAY "TEST HERE"
@ 17.42 GET test
@ 18.31 SAY "PRESS RETURN TO CONTINUE"
READ
STORE .f. TO one_time
SAVE TO mem_var
do while .t.
clear
@ 0.1 SAY "MAIN"
@ 1.22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3.36 SAY "MAIN MENU"
@ 6.22 SAY "1) DICTIONARY MAINTENANCE"
@ 8.22 SAY "2) DICTIONARY OUTPUT"
@ 10.22 SAY "3) DICTIONARY QUERY"
@ 12.22 SAY "4) SCHEMA MAINTENANCE"
@ 14.22 SAY "5) SCHEMA OUTPUT"
@ 16.22 SAY "6) EXIT DICTIONARY SYSTEM"
@ 17.22 SAY "" ACCEPT ' ENTER YOUR CHOICE (1-6) FROM ABOVE: ' TO choice
DO CASE
CASE choice = "1"
do 110000
CASE choice = "2"
do 120000
CASE choice = "3"
do 130000
CASE choice = "4"
do 140000
CASE choice = "5"
do 150000
CASE choice = "6"
CLEAR
RELEASE ALL

133
RETURN
OTHERWISE
CLEAR
@ 2,4 SAY "PLEASE ENTER VALUES BETWEEN 1 AND 6 ONLY"
@ 3,4 SAY "PRESS RETURN TO TRY AGAIN!"
ACCEPT TO hold
ENDCASE
ENDDO
RETURN
* 110000.PRG
* MODULE NAME: 1.1.0.0.0.0
* INPUT FILES: NONE
* OUTPUT FILES: NONE
* ROUTINES THAT CALL THE MODLUE: MAIN, 1.1.1.0.0.0, 1.1.2.0.0.0, 1.1.3.0.0.0
* 1.1.4.0.0.0.0, 1.1.5.0.0.0.
* ROUTINES THAT THE MODULE CALLS: 1.1.1.0.0.0, 1.1.2.0.0.0, 1.1.3.0.0.0,
* 1.1.4.0.0.0.0, 1.1.5.0.0.0, MAIN.
* LOCAL VARIABLES USED: choice: CONTAINS THE NUMBER OF ACTION SELECTED.
* t: REPRESENTS NO VALUE AT ALL.
* hold: USED TO STOP ACTION FOR USER DECISION.
* DESIGNED BY: ROBERT A. KIRSCH II
* WRITTEN BY: ROBERT A. KIRSCH II
* BASIC FUNCTION OF MODULE:
* THIS PROGRAM ALLOWS FOR THE MAINTENANCE OF ENTITY RELATIONS,
* AND RELATIONSHIP RELATIONS.
* do while .t.
CLEAR
@ 0.1 SAY "1.1.0.0.0.0"
@ 1.22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3.31 SAY "MAINTENANCE MENU"
@ 6.22 SAY "1) ADD ENTITY"
@ 8.22 SAY "2) MODIFY ENTITY"
@ 10.22 SAY "3) DELETE ENTITY"
@ 12.22 SAY "4) ADD RELATIONSHIP"
@ 14.22 SAY "5) DELETE RELATIONSHIP"
@ 16.22 SAY "6) RETURN TO MAIN MENU"
@ 17.22 SAY "" ACCEPT ' ENTER YOUR CHOICE (1-6) FROM ABOVE: ' TO choice
DO CASE
CASE choice = "1"
do 111000
CASE choice = "2"
do 112000
CASE choice = "3"
do 113000
CASE choice = "4"
do 114000
CASE choice = "5"
do 115000
CASE choice = "6"
RETURN TO MASTER
OTHERWISE
CLEAR
@ 2.18 SAY choice
@ 2.21 SAY "IS NOT A VALID CHOICE"
@ 3.18 SAY "PLEASE ENTER VALUES BETWEEN 1 AND 6 ONLY"
@ 4.18 SAY "PRESS RETURN TO TRY AGAIN!"
WAIT TO hold
ENDCASE
ENDDO
RETURN
* 111000.PRG
* MODULE NAME: 1.1.1.0.0.0
* INPUT FILES: NONE
* OUTPUT FILES: NONE
* ROUTINES THAT CALL THE MODULE: 1.1.0.0.0.0
* ROUTINES THAT THE MODULE CALLS: 1.1.1.0.0.0, 1.1.1.2.0.0, 1.1.1.3.0.0,
  1.1.1.4.0.0, 1.1.1.5.0.0, 1.1.1.6.0.0, 1.1.1.7.0.0, 1.1.1.8.0.0, 1.1.0.0.0.0
* MAIN
* LOCAL VARIABLES USED: choice: CONTAINS THE NUMBER OF ACTION SELECTED.
  t: REPRESENTS NO VALUE AT ALL.
  hold: USED TO STOP ACTION FOR USER DECISION.
* DESIGNED BY: ROBERT A. KIRSCH II
* WRITTEN BY: ROBERT A. KIRSCH II
* BASIC FUNCTION OF MODULE:
* THIS MODULE ALLOW THE USER TO CHOOSE WHICH TYPE OF ENTITY RELATION
  TO ADD TUPLES TO.
* set color to 0/3,3
set talk off
CLEAR
do while .t.
CLEAR
@ 0.1 SAY "1.1.1.0.0.0"
@ 1.22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3.35 SAY "ADD ENTITY"
@ 5.15 SAY "1) USER 6) FILE"
@ 5.15 SAY "2) SYSTEM 7) RECORD"
@ 10.15 SAY "3) PROGRAM 8) ELEMENT"
@ 12.15 SAY "4) MODULE 9) RETURN TO PREVIOUS MENU"
@ 14.15 SAY "5) DOCUMENT 10) RETURN TO MAIN MENU"
@ 15.22 SAY ""
ACCEPT ' ENTER YOUR CHOICE (1-10) FROM ABOVE: ' TO choice
SET EXACT ON
DO CASE
CASE choice = "1"
do 111100
CASE choice = "2"
do 111200
CASE choice = "3"
do 111300
CASE choice = "4"
do 111400
CASE choice = "5"
do 111500
CASE choice = "6"
do 111600
CASE choice = "7"
do 111700
CASE choice = "8"
do 111800
CASE choice = "9"
RETURN
CASE choice = "10"
RETURN TO MASTER
OTHERWISE
CLEAR
@ 2.3 SAY "PLEASE ENTER VALUES BETWEEN 1 AND 10 ONLY"
@ 3.3 SAY "PRESS RETURN TO TRY AGAIN!"
ACCEPT TO hold
ENDCASE
ENDDO
RETURN
* 111100.PRG
* MODULE NAME : 1.1.1.1.0.0
* INPUT FILES : USER
* OUTPUT FILES: USER
* ROUTINES THAT CALL THE MODULE: 1.1.1.0.0.0
* ROUTINES THAT THE MODULE CALLS:1.1.1.0.0.0
* LOCAL VARIABLES USED:

* choice : CONTAINS THE NUMBER OF ACTION SELECTED. MAY ALSO CONTAIN THE
* CHARACTER STRING THAT IDENTIFIES THE RELATION BEING ADDED TO,
* MODIFIED, DELETED FROM OR OUTPUT.
* t : REPRESENTS THE BOOLEAN TRUE IS USED TO CREATE A CONTINUES
* LOOP.
* INPUT FILES: USER
* OUTPUT FILES: USER

* DESIGNED BY: ROBERT A. KIRSCH II
* WRITTEN BY: ROBERT A. KIRSCH II
* BASIC FUNCTION OF MODULE:
* THIS PROGRAM ALLOWS THE USER TO ENTER NEW TUPLES TO THE USER RELATION.

USE
do while .t.
CLEAR
@ 0,1 SAY "1.1.1.1.0.0"
@ 1,22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3,36 SAY "ADD USER"
@ 6,22 SAY "This program will allow you to enter"
@ 7,22 SAY "additional tuples to the USER relation."
@ 8,22 SAY "Instructions for entering data are"
@ 9,22 SAY "provided at top of entry screen."
@ 10,22 SAY "."
wait to choice
SET MENU ON
USE USER
APPEND
SET MENU OFF
RETURN
* 111200.PRG
* MODULE NAME: 1.1.1.2.0.0
* INPUT FILES: USER
* OUTPUT FILES: USER
* ROUTINES THAT CALL THE MODULE: 1.1.1.0.0.0
* ROUTINES THAT THE MODULE CALLS: 1.1.1.0.0.0
* LOCAL VARIABLES USED:
* choice: CONTAINS THE NUMBER OF ACTION SELECTED. MAY ALSO CONTAIN THE
* character string that identifies the relation being added to,
* modified, deleted from or output.
* t: REPRESENTS THE BOOLEAN TRUE IS USED TO CREATE A CONTINUES LOOP.
* INPUT FILES: SYSTEM.
* OUTPUT FILES: SYSTEM.
* DESIGNED BY: ROBERT A. KIRSCH II
* WRITTEN BY: ROBERT A. KIRSCH II
* BASIC FUNCTION OF MODULE:
* THIS PROGRAM ALLOWS THE USER TO ENTER NEW TUPLES TO THE SYSTEM RELATION.

USE
do while .t.
CLEAR
@ 0.1 SAY "1.1.1.2.0.0"
@ 1.22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3.35 SAY "ADD SYSTEM"
@ 6.22 SAY "This program will allow you to enter" additional tuples to the SYSTEM relation."
@ 8.22 SAY "Instructions for entering data are"
@ 9.22 SAY "provided at top of entry screen."
@ 10.22 SAY ""
wait to choice
SET MENU ON
USE SYSTEM
APPEND
SET MENU OFF
RETURN
* 111300.PRG
* MODULE NAME : 1.1.1.3.0.0
* INPUT FILES : USER
* OUTPUT FILES : USER
* ROUTINES THAT CALL THE MODLUE : 1.1.1.0.0.0
* ROUTINES THAT THE MODULE CALLS : 1.1.1.0.0.0
* choice : CONTAINS THE NUMBER OF ACTION SELECTED. MAY ALSO CONTAIN THE CHARACTER STRING THAT IDENTIFIES THE RELATION BEING ADDED TO, MODIFIED, DELETED FROM OR OUTPUT.
* t : REPRESENTS THE BOOLEAN TRUE IS USED TO CREATE A CONTINUES LOOP.
* INPUT FILES : PROGRAM.
* OUTPUT FILES : PROGRAM.
* DESIGNED BY : ROBERT A. KIRSCH II
* WRITTEN BY : ROBERT A. KIRSCH II
* BASIC FUNCTION OF MODULE :
* THIS PROGRAM ALLOWS THE USER TO ENTER NEW TUPLES TO THE PROGRAM RELATION.

USE
set color to 0/3.7/0.3
set talk off
do while .t.
  CLEAR
  @ 0.1 SAY "1.1.1.3.0.0"
  @ 1.22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
  @ 6.22 SAY "This program will allow you to enter"
  @ 7.22 SAY "additional tuples to the PROGRAM relation."
  @ 8.22 SAY "Instructions for entering data are"
  @ 9.22 SAY "provided at top of entry screen."
  wait to choice
  SET MENU ON
  USE PROGRAM
  APPEND
  SET MENU OFF
RETURN

139
* 111400.PRG
* MODULE NAME: 1.1.1.4.0.0
* INPUT FILES: USER
* OUTPUT FILES: USER
* ROUTINES THAT CALL THE MODULE: 1.1.1.0.0.0
* ROUTINES THAT THE MODULE CALLS: 1.1.1.0.0.0
* choice: CONTAINS THE NUMBER OF ACTION SELECTED. MAY ALSO CONTAIN THE
* CHARACTER STRING THAT IDENTIFIES THE RELATION BEING ADDED TO,
* MODIFIED, DELETED FROM OR OUTPUT.
* t: REPRESENTS THE BOOLEAN TRUE IS USED TO CREATE A CONTINUES
* LOOP.
* INPUT FILES: MODULE.
* OUTPUT FILES: MODULE.
* DESIGNED BY: ROBERT A. KIRSCH II
* WRITTEN BY: ROBERT A. KIRSCH II
* BASIC FUNCTION OF MODULE:
* THIS PROGRAM ALLOWS THE USER TO ENTER NEW TUPLES TO THE MODULE RELATION.
*
USE
do while .t.
CLEAR
@ 0,1 SAY "1.1.1.4.0.0"
@ 1.22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3.36 SAY "ADD MODULE"
@ 6.22 SAY "This program will allow you to enter"
@ 7.22 SAY "additional tuples to the MODULE relation."
@ 8.22 SAY "Instructions for entering data are"
@ 9.22 SAY "provided at top of entry screen."
@ 10.22 SAY ""
wait to choice
SET MENU ON
USE MODULE
APPEND
SET MENU OFF
RETURN
**111500.PRG**

**MODULE NAME**: 1.1.1.5.0.0

**INPUT FILES**: USER

**OUTPUT FILES**: USER

**ROUTINES THAT CALL THE MODULE**: 1.1.1.0.0.0

**ROUTINES THAT THE MODULE CALLS**: 1.1.1.0.0.0

**choice**: CONTAINS THE NUMBER OF ACTION SELECTED. MAY ALSO CONTAIN THE CHARACTER STRING THAT IDENTIFIES THE RELATION BEING ADDED TO, MODIFIED, DELETED FROM OR OUTPUT.

**t**: REPRESENTS THE BOOLEAN TRUE IS USED TO CREATE A CONTINUES LOOP.

**INPUT FILES**: DOCUMENT.

**OUTPUT FILES**: DOCUMENT.

**DESIGNED BY**: ROBERT A. KIRSCH II

**WRITTEN BY**: ROBERT A. KIRSCH II

**BASIC FUNCTION OF MODULE**: THIS PROGRAM ALLOWS THE USER TO ENTER NEW TUPLES TO THE DOCUMENT RELATION.

**USE**

set color to 0/3,7/0,3
set talk off
do while .t.
CLEAR
@ 0.1 SAY "1.1.1.5.0.0"
@ 1.22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3.36 SAY "ADD DOCUMENT"
@ 6.22 SAY "This program will allow you to enter"
@ 7.22 SAY "additional tuples to the DOCUMENT relation."
@ 8.22 SAY "Instructions for entering data are"
@ 9.22 SAY "provided at top of entry screen."
@ 10.22 SAY ""
wati to choice
SET MENU ON
USE DOCUMENT
APPEND
SET MENU OFF
RETURN
* module name: 1.1.1.6.0.0
* input files: USER
* output files: USER
* routines that call the module: 1.1.1.0.0.0
* routines that the module calls: 1.1.1.0.0.0
* choice: contains the number of action selected. may also contain the
* character string that identifies the relation being added to,
* modified, deleted from or output.
* t: represents the boolean true is used to create a continues
* loop.
* input files: FILE.
* output files: FILE.
* designed by: ROBERT A. KIRSCH II
* written by: ROBERT A. KIRSCH II
* basic function of module:
* this program allows the user to enter new tuples to the file relation.
* use
* do while .t.
* clear
* @ 0 1 say "1.1.1.6.0.0"
* @ 1 22 say "INFORMATION RESOURCE DICTIONARY SYSTEM"
* @ 3 36 say "ADD FILE"
* @ 6 22 say "This program will allow you to enter"
* @ 7 22 say "additional tuples to the FILE relation."
* @ 8 22 say "Instructions for entering data are"
* @ 9 22 say "provided at top of entry screen."
* @ 10 22 say ""
* wait to choice
* set menu on
* use file
* append
* set menu off
* return
**111700.PRG**

* MODULE NAME: 1.1.1.7.0.0
* INPUT FILES: USER
* OUTPUT FILES: USER
* ROUTINES THAT CALL THE MODULE: 1.1.1.0.0.0
* ROUTINES THAT THE MODULE CALLS: 1.1.1.0.0.0
* choice: CONTAINS THE NUMBER OF ACTION SELECTED. MAY ALSO CONTAIN THE
  CHARACTER STRING THAT IDENTIFIES THE RELATION BEING ADDED TO,
  MODIFIED, DELETED FROM OR OUTPUT.
* t: REPRESENTS THE BOOLEAN TRUE IS USED TO CREATE A CONTINUES
  LOOP.
* INPUT FILES: RECORD.
* OUTPUT FILES: RECORD.
* DESIGNED BY: ROBERT A. KIRSCH II
* WRITTEN BY: ROBERT A. KIRSCH II
* BASIC FUNCTION OF MODULE:
  THIS PROGRAM ALLOWS THE USER TO ENTER NEW TUPLES TO THE RECORD RELATION.

**USE**
do while .t.
clear
@ 0,1 say "1.1.1.7.0.0"
@ 1,22 say "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3,36 say "ADD RECORD"
@ 6,22 say "This program will allow you to enter".
@ 7,22 say "additional tuples to the RECORD relation."
@ 8,22 say "Instructions for entering data are"
@ 9,22 say "provided at top of entry screen."
@ 10,22 say ""
wait to choice
set menu on
use record
append
set menu off
return
111800.PRG

MODULE NAME: 1.1.1.8.0.0

INPUT FILES: USER

OUTPUT FILES: USER

ROUTINES THAT CALL THE MODULE: 1.1.1.0.0.0

ROUTINES THAT THE MODULE CALLS: 1.1.1.0.0.0

choice: CONTAINS THE NUMBER OF ACTION SELECTED. MAY ALSO CONTAIN THE

CHARACTER STRING THAT IDENTIFIES THE RELATION BEING ADDED TO,
MODIFIED, DELETED FROM OR OUTPUT.

* t: REPRESENTS THE BOOLEAN TRUE IS USED TO CREATE A CONTINUES LOOP.

INPUT FILES: ELEMENT.

OUTPUT FILES: ELEMENT.

DESIGNED BY: ROBERT A. KIRSCH II

WRITTEN BY: ROBERT A. KIRSCH II

BASIC FUNCTION OF MODULE:

THIS PROGRAM ALLOWS THE USER TO ENTER NEW TUPLES TO THE ELEMENT RELATION.

USE

do while .t.

CLEAR

@ 0,1 SAY "1.1.1.8.0.0"

@ 1.22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"

@ 3.36 SAY "ADD ELEMENT"

@ 6.22 SAY "This program will allow you to enter"

@ 7.22 SAY "additional tuples to the ELEMENT relation."

@ 8.22 SAY "Instructions for entering data are"

@ 9.22 SAY "provided at top of entry screen."

@ 10.22 SAY ""

wait to choice

SET MENU ON

USE ELEMENT

APPEND

SET MENU OFF

RETURN
MAIN
LOCAL VARIABLES USED: choice: CONTAINS THE NUMBER OF ACTION SELECTED.
hold: REPRESENTS NO VALUE AT ALL.
* INPUT FILE: MEM VAR.
* OUTPUT FILE: MEM VAR.
* DESIGNED BY: ROBERT A. KIRSCH II
* WRITTEN BY: ROBERT A. KIRSCH II
* BASIC FUNCTION OF MODULE:
* THIS MODULE ALLOW THE USER TO CHOOSE WHICH TYPE OF ENTITY RELATION
* TO MODIFY.
* do while .t.
  CLEAR
  @ 1,1 SAY "1.1.2.0.0.0"
  @ 2,2 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
  @ 4,35 SAY "MODIFY ENTITY"
  @ 7,15 SAY "1) USER 6) FILE"
  @ 9,15 SAY "2) SYSTEM 7) RECORD"
  @ 11,15 SAY "3) PROGRAM 8) ELEMENT"
  @ 13,15 SAY "4) MODULE 9) RETURN TO PREVIOUS MENU"
  @ 15,15 SAY "5) DOCUMENT 10) RETURN TO MAIN MENU"
  @ 16,22 SAY "ENTER YOUR CHOICE (1-10) FROM ABOVE: '
  ACCEPT choice
  DO CASE
  CASE choice = "1"
    store 'USER' to choice
    save to mem var
    do 112100
  CASE choice = "2"
    store 'SYSTEM' to choice
    save to mem var
    do 112100
  CASE choice = "3"
    store 'PROGRAM' to choice
    save to mem var
    do 112100
  CASE choice = "4"
    store 'MODULE' to choice
    save to mem var
    do 112100
  CASE choice = "5"
    store 'DOCUMENT' to choice
    save to mem var
    do 112100
  CASE choice = "6"
    store 'FILE' to choice
    save to mem var
    do 112100
  CASE choice = "7"
    store 'RECORD' to choice
    save to mem_var
    do 112100
  CASE choice = "8"
    store 'ELEMENT' to choice
    save to mem_var
    do 112100
  CASE choice = "9"
    RETURN
  CASE choice = "10"
    RETURN TO MASTER
OTHERWISE
CLEAR
@ 2.3 SAY "PLEASE ENTER VALUES BETWEEN 1 AND 10 ONLY"
@ 3.3 SAY "PRESS RETURN TO TRY AGAIN!"
ACCEPT TO hold
ENDCASE
ENDDO
RETURN
* 112100.PRG
* MODULE NAME: 1.1.2.0.0.0
* INPUT FILES: NONE
* OUTPUT FILES: NONE
* ROUTINES THAT CALL THE MODULE: 1.1.0.0.0
* ROUTINES THAT THE MODULE CALLS: 1.1.2.1.0.0, 1.1.2.2.0.0, 1.1.2.3.0.0,
* 1.1.2.4.0.0, 1.1.2.5.0.0, 1.1.2.6.0.0, 1.1.2.7.0.0, 1.1.2.8.0.0, 1.1.0.0.0.0
* MAIN

* LOCAL VARIABLES USED:
* choice : CONTAINS THE NUMBER OF ACTION SELECTED. MAY ALSO CONTAIN THE
* CHARACTER STRING THAT IDENTIFIES THE RELATION BEING ADDED TO,
* MODIFIED, DELETED FROM OR OUTPUT.
* hold : USED TO STOP ACTION FOR USER DECISION.
* rec_num : CONTAINS THE VALUE OF THE POINTER TO THE TUPLE TO BE CHANGED.
* stop : USED TO STOP ACTION FOR USER DECISION.
* t : REPRESENTS THE BOOLEAN TRUE IS USED TO CREATE A CONTINUES
* LOOP.
* INPUT FILES: MEM VAR, USER, SYSTEM, PROGRAM, MODULE, DOCUMENT, FILE, RECORD,
* ELEMENT.
* OUTPUT FILES: MEM VAR, USER, SYSTEM, PROGRAM, MODULE, DOCUMENT, FILE, RECORD,
* ELEMENT.
* DESIGNED BY: ROBERT A. KIRSCH II
* WRITTEN BY: ROBERT A. KIRSCH II
* BASIC FUNCTION OF MODULE:
* THIS MODULE ALLOW THE USER TO CHOOSE WHICH TYPE OF ENTITY RELATION
* TO MODIFY.

RESTORE FROM mem_var
STORE 0 TO rec_num, stop
CLEAR
STORE .t. TO TRUE
do while TRUE
CLEAR
@ 0,1 SAY "1.1.2.1.0.0"
@ 1.22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3.35 SAY "MODIFY ENTITY"
@ 7.24 SAY "ENTER TUPLE NUMBER OF THE"
@ 7.51 SAY choice
ACCEPT ' YOU WISH TO MODIFY ' TO rec_num
IF (rec_num <= '0') .OR. (rec_num > '99999')
CLEAR
@ 1.24 SAY rec_num
@ 1.32 SAY "IS NOT A VALID RESPONSE"
@ 2.23 SAY "TUPLE NUMBER MUST BE GREATER THAN 0"
@ 3.23 SAY "AND LESS THAN 99999."
WAIT TO stop
ELSE
STORE .F. TO TRUE
ENDIF
ENDDO
DO CASE
CASE choice = 'USER'
USE USER
EDIT(VAL( rec_num))
RETURN
CASE choice = 'SYSTEM'
USE SYSTEM
EDIT(VAL(rec_num))
RETURN
CASE choice = 'PROGRAM'
USE PROGRAM
EDIT(VAL(rec_num))
RETURN
CASE choice = 'MODULE'
USE MODULE

147
EDIT(VAL(rec_num))
RETURN
CASE choice = 'DOCUMENT'
USE DOCUMENT
EDIT(VAL(rec_num))
RETURN
CASE choice = 'FILE'
USE FILE
EDIT(VAL(rec_num))
RETURN
CASE choice = 'RECORD'
USE RECORD
EDIT(VAL(rec_num))
RETURN
CASE choice = 'ELEMENT'
USE ELEMENT
EDIT(VAL(rec_num))
RETURN
@ 42,1 SAY "RETURN]"
* 113000.PRG
* MODULE NAME: 1.1.3.0.0.0
* INPUT FILES: NONE
* OUTPUT FILES: NONE
* ROUTINES THAT CALL THE MODULE: 1.1.0.0.0.0
* ROUTINES THAT THE MODULE CALLS: 1.1.3.1.0.0, MAIN
* LOCAL VARIABLES USED:
  * choice: CONTAINS THE NUMBER OF ACTION SELECTED ALSO USED TO TRANSFER THE
  * RELATION NAME TO NEXT PROGRAM.
  * t: REPRENTS NO VALUE AT ALL.
  * hold: USED TO STOP ACTION FOR USER DECISION.
* INPUT FILE: MEM
* OUTPUT FILE: MEM VAR.
* DESIGNED BY: ROBERT A. KIRSCH II
* WRITTEN BY: ROBERT A. KIRSCH II
* BASIC FUNCTION OF MODULE:
  * THIS MODULE ALLOW THE USER TO CHOOSE WHICH TYPE OF ENTITY RELATION
  * TO MODIFY.
* SET EXACT
set color to 0/3,3
set talk off
CLEAR
do while .t.
   ERASE mem_var.mem
   CLEAR
   @ 1,1 SAY "1.1.3.0.0.0"
   @ 2,22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
   @ 4,35 SAY "DELETE ENTITY"
   @ 7,15 SAY "1) USER"
   @ 9,15 SAY "2) SYSTEM"
   @ 11,15 SAY "3) PROGRAM"
   @ 13,15 SAY "4) MODULE"
   @ 15,15 SAY "5) DOCUMENT"
   @ 16,22 SAY "6) FILE"
   @ 9,15 SAY "7) RECORD"
   @ 11,15 SAY "8) ELEMENT"
   @ 13,15 SAY "9) RETURN TO PREVIOUS MENU"
   @ 15,15 SAY "10) RETURN TO MAIN MENU"
   ACCEPT Enter your choice (1-10) from above: 'to choice
   DO CASE
      CASE choice = "1"
         store 'USER' to choice
         save to mem_var
         DO 113100
      CASE choice = "2"
         store 'SYSTEM' to choice
         save to mem_var
         DO 113100
      CASE choice = "3"
         store 'PROGRAM' to choice
         save to mem_var
         DO 113100
      CASE choice = "4"
         store 'MODULE' to choice
         save to mem_var
         DO 113100
      CASE choice = "5"
         store 'DOCUMENT' to choice
         save to mem_var
         DO 113100
      CASE choice = "6"
         store 'FILE' to choice
         save to mem_var
         DO 113100
      CASE choice = "7"
         store 'RECORD' to choice
         save to mem_var
         DO 113100
      CASE choice = "8"
         store 'ELEMENT' to choice
         save to mem_var
         DO 113100
   CASE
DO 113100
CASE choice = "9"
RETURN
CASE choice = "10"
RETURN TO MASTER
OTHERWISE
CLEAR
@ 2.18 SAY choice
@ 2.21 SAY "IS NOT A VALID CHOICE"
@ 3.18 SAY "PLEASE ENTER VALUES BETWEEN 1 AND 10 ONLY"
@ 4.21 SAY "PRESS RETURN TO TRY AGAIN!"
ACCEPT TO hold
ENDCASE
ENDDO
RETURN
RETURN
* 113100.PRG
* MODULE NAME: 1.1.3.1.0.0
* INPUT FILES: NONE
* OUTPUT FILES: NONE
* ROUTINES THAT CALL THE MODULE: 1.1.0.0.0.0
* ROUTINES THAT THE MODULE CALLS: 1.1.3.0.0.0
* LOCAL VARIABLES USED:
  * choice : CONTAINS THE NUMBER OF ACTION SELECTED.
  * t : REPRESENTS THE BOOLEAN TRUE IS USED TO CREATE A CONTINUES LOOP.
  * stop : USED TO STOP ACTION FOR USER DECISION.
  * true : USED AS A BOOLEAN VALUE IN LOOPS.
  * rec num : CONTAINS THE VALUE REPRESENTING THE RECORD CHANGED.
* INPUT FILES: USER, SYSTEM, PROGRAM, MODULE, DOCUMENT, FILE, RECORD, ELEMENT
* OUTPUT FILES: USER, SYSTEM, PROGRAM, MODULE, DOCUMENT, FILE, RECORD, ELEMENT
* mem var, mem
* DESIGNED BY: ROBERT A. KIRSCH II
* WRITTEN BY: ROBERT A. KIRSCH II
* BASIC FUNCTION OF MODULE:
  * THIS MODULE ALLOW THE USER TO CHOOSE WHICH TYPE OF ENTITY RELATION TO DELETE TUPLES FROM.
  *
SET MENU ON
RESTORE FROM mem_var
STORE 0 TO rec_num, stop
CLEAR
STORE .t. TO TRUE
DO WHILE TRUE
CLEAR
@ 0,1 SAY "1.1.3.1.0.0"
@ 1.22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3.35 SAY "DELETE ENTITY"
@ 7.24 SAY "ENTER TUPLE NUMBER OF THE"
@ 7.51 SAY choice
@ 8.24 SAY "TUPLE YOU WISH TO HAVE DELETED."
@ 9.24 SAY "THE RECORD WILL BE DISPLAYED"
@ 10.24 SAY "FOR YOU TO EXAMINE. IF YOU ARE"
@ 11.24 SAY "SURE THAT YOU ARE DELETING THE"
@ 12.24 SAY "RIGHT RECORD DEPRESS -U."
@ 14.24 SAY "IF YOU DO NOT WANT IT DELETED DEPRESS"
@ 16.24 SAY "'0' TO RETURN TO MAINTENANCE MENU."
ACCEPT 'ENTER THE TUPLE NUMBER NOW ' TO rec_num
IF rec_num > '99999'
CLEAR
@ 1.24 SAY rec_num
@ 1.32 SAY "IS NOT A VALID RESPONSE"
@ 2.23 SAY "TUPLE NUMBER MUST BE GREATER THAN 0"
@ 3.23 SAY "AND LESS THAN 99999."
WAIT TO stop
ELSE IF rec_num <= '0'
RETURN
STORE .F. TO TRUE
ENDIF
ENDDO
DO CASE
  CASE choice = 'USER'
    USE USER
    EDIT(VAL( rec_num))
    RETURN
  CASE choice = 'SYSTEM'
    USE SYSTEM
    EDIT(VAL(rec_num))
    RETURN
  CASE choice = 'PROGRAM'
    USE PROGRAM
    EDIT(VAL(rec_num))
    RETURN
ENDCASE

151
CASE choice = 'MODULE'
USE MODULE
EDIT(VAL(rec_num))
RETURN
CASE choice = 'DOCUMENT'
USE DOCUMENT
EDIT(VAL(rec_num))
RETURN
CASE choice = 'FILE'
USE FILE
EDIT(VAL(rec_num))
RETURN
CASE choice = 'RECORD'
USE RECORD
EDIT(VAL(rec_num))
RETURN
CASE choice = 'ELEMENT'
USE ELEMENT
EDIT(VAL(rec_num))
RETURN
ENDCASE
* 114000.PRG
* MODULE NAME: 1.1.4.0.0.0
* INPUT FILES: NONE
* OUTPUT FILES: NONE
* ROUTINES THAT CALL THE MODULE: 1.1.0.0.0.0, MAIN
* ROUTINES THAT THE MODULE CALLS: 1.1.4.1.0.0, MAIN
* LOCAL VARIABLES USED:
  * choice : CONTAINS THE NUMBER OF ACTION SELECTED. MAY ALSO CONTAIN THE
    CHARACTER STRING THAT IDENTIFIES THE RELATION BEING ADDED TO,
    MODIFIED, DELETED FROM OR OUTPUT.
  * hold : USED TO STOP THE SCREEN OUTPUT FOR A USER DECISION.
  * t : REPRESENTS THE BOOLEAN TRUE IS USED TO CREATE A CONTINUES LOOP.
  * title : CONTAINS THE CHARACTER STRING THAT DESCRIBES THE RELATIONSHIP
    BEING ADDED TO, DELETED FROM OR OUTPUT.
* INPUT FILES: MEM VAR
* OUTPUT FILES: MEM VAR
* DESIGNED BY: ROBERT A. KIRSCH II
* WRITTEN BY: ROBERT A. KIRSCH II
* BASIC FUNCTION OF MODULE:
  * THIS MODULE ALLOW THE USER TO CHOOSE WHICH RELATIONSHIP HE WOULD
    LIKE TO ADD TUPLES TO.
* CLEAR
  do while .t.
  ERASE mem.var.mem
  CLEAR
  @ 0,1  SAY "1.1.4.0.0.0"
  @ 1,22  SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
  @ 3,32  SAY "ADD TO RELATIONSHIP"
  @ 5,9  SAY "1) USER CONTAINS SYSTEM   8) FILE CONTAINS REC"
  @ 5,64  SAY "ORDS"
  @ 7,9  SAY "2) SYSTEM CONTAINS PROGRAM  9) RECORD CONTAINS E"
  @ 7,64  SAY "ELEMENT"
  @ 9,9  SAY "3) PROGRAM PROCESSES FILE 10) USER RESPONSIBLE"
  @ 9,64  SAY "FOR SYSTEM"
  @ 11,9  SAY "4) PROGRAM PROCESSES RECORD 11) USER RESPONSIBLE"
  @ 11,64  SAY "FOR FILE"
  @ 13,9  SAY "5) PROGRAM PROCESSES ELEMENT 12) PROGRAM PRODUCES"
  @ 13,64  SAY "DOCUMENT"
  @ 15,9  SAY "6) SYSTEM CONTAINS PROGRAM 13) RETURN TO PREVIOUS"
  @ 15,64  SAY "MENU"
  @ 17,9  SAY "7) PROGRAM CONTAINS MODULE 14) RETURN TO MAIN MENU"
  @ 17,64  SAY "NU"
  @ 18,22  SAY "-
  ACCEPT:
  "ENTER YOUR CHOICE (1-14) FROM ABOVE: ' TO choice
  DO CASE
  CASE choice = "1"
  store 'U PROC S' to choice
  store 'USER-PROCESSES-SYSTEM' to title
  save to mem_var
  do 114100
  CASE choice = "2"
  store 'S PROC P' to choice
  store 'SYSTEM-PROCESSES-PROGRAM' to title
  save to mem_var
  do 114100
  CASE choice = "3"
  store 'P PROC F' to choice
  store 'PROGRAM-PROCESSES-FILE' to title
  save to mem_var
  do 114100
  CASE choice = "4"
  store 'P PROC R' to choice
  store 'PROGRAM-PROCESSES-RECORD' to title
  save to mem_var
  do 114100
  CASE choice = "5"
store 'P PROC E' to choice
store 'PROGRAM-PROCESSES-ELEMENT' TO title
save to mem_var
do 114100
CASE choice = "6"
store 'S CONT P' to choice
store 'SYSTEM-CONTAINS-PROGRAM' TO title
save to mem_var
do 114100
CASE choice = "7"
store 'P CONT M' to choice
store 'PROGRAM-CONTAINS-MODULE' TO title
save to mem_var
do 114100
CASE choice = "8"
store 'F CONT R' to choice
store 'FILE-CONTAINS-RECORD' TO title
save to mem_var
do 114100
CASE choice = "9"
store 'R CONT E' to choice
store 'RECORD-CONTAINS-ELEMENT' TO title
save to mem_var
do 114100
CASE choice = "10"
store 'U RESP S' to choice
store 'USER-RESPONSIBLE-FOR-SYSTEM' TO title
save to mem_var
do 114100
CASE choice = "11"
store 'U RESP F' to choice
store 'USER-RESPONSIBLE-FOR-FILE' TO title
save to mem_var
do 114100
CASE choice = "12"
store 'P PROD D' to choice
store 'PROGRAM-PRODUCES-DOCUMENT' TO title
save to mem_var
do 114100
CASE choice = "13"
RETURN
CASE choice = "14"
RETURN TO MASTER
OTHERWISE
CLEAR
@ 1.21 SAY choice
@ 1.28 SAY "IS NOT A VALID CHOICE"
@ 2.20 SAY "PLEASE ENTER VALUES BETWEEN 1 AND 14 ONLY"
@ 3.20 SAY "PRESS RETURN TO TRY AGAIN!"
ACCEPT TO hold
ENDCASE
ENDDO
RETURN
do while t.
    RESTORE FROM mem_var
    CLEAR
    @ 0,1 SAY "1.1.4.1.0.0"
    @ 1,22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
    @ 3,32 SAY "ADD RELATIONSHIP"
    @ 8,22 SAY "YOU ARE ABOUT TO BEGIN ADDING TUPLES"
    @ 9,30 SAY "TO THE"
    @ 9,30 SAY TITLE
    @ 10,22 SAY "RELATION."
    WAIT TO STOP
    DO CASE
    CASE choice = "UPROC_S"
        USE UPROC_S
        APPEND
        RETURN
    CASE choice = "SPROC_P"
        USE SPROC_P
        APPEND
        RETURN
    CASE choice = "PPROC_F"
        USE PPROC_F
        APPEND
        RETURN
    CASE choice = "PPROC_R"
        USE PPROC_R
        APPEND
        RETURN
    CASE choice = "PPROC_E"
        USE PPROC_E
        APPEND
        RETURN
    CASE choice = "SCONT_P"
        USE SCONT_P
        APPEND
        RETURN
    CASE choice = "PCONT_M"
        USE PCONT_M
        APPEND
        RETURN
    DESIGNED BY: ROBERT A. KIRSCH II
    WRITTEN BY: ROBERT A. KIRSCH II
    BASIC FUNCTION OF MODULE:
    THIS MODULE ALLOW FOR THE ACTUAL INPUT OF ADDITIONAL TUPLES TO THE
    RELATIONSHIP RELATION SELECTED.
    CLEAR
    do while t.
    RESTORE FROM mem_var
    CLEAR
    @ 0,1 SAY "1.1.4.1.0.0"
    @ 1,22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
    @ 3,32 SAY "ADD RELATIONSHIP"
    @ 8,22 SAY "YOU ARE ABOUT TO BEGIN ADDING TUPLES"
    @ 9,30 SAY "TO THE"
    @ 9,30 SAY TITLE
    @ 10,22 SAY "RELATION."
    WAIT TO STOP
    DO CASE
    CASE choice = "UPROC_S"
        USE UPROC_S
        APPEND
        RETURN
    CASE choice = "SPROC_P"
        USE SPROC_P
        APPEND
        RETURN
    CASE choice = "PPROC_F"
        USE PPROC_F
        APPEND
        RETURN
    CASE choice = "PPROC_R"
        USE PPROC_R
        APPEND
        RETURN
    CASE choice = "PPROC_E"
        USE PPROC_E
        APPEND
        RETURN
    CASE choice = "SCONT_P"
        USE SCONT_P
        APPEND
        RETURN
    CASE choice = "PCONT_M"
        USE PCONT_M
        APPEND
        RETURN
    DESIGNED BY: ROBERT A. KIRSCH II
    WRITTEN BY: ROBERT A. KIRSCH II
    BASIC FUNCTION OF MODULE:
    THIS MODULE ALLOW FOR THE ACTUAL INPUT OF ADDITIONAL TUPLES TO THE
    RELATIONSHIP RELATION SELECTED.
    CLEAR
    do while t.
    RESTORE FROM mem_var
    CLEAR
    @ 0,1 SAY "1.1.4.1.0.0"
    @ 1,22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
    @ 3,32 SAY "ADD RELATIONSHIP"
    @ 8,22 SAY "YOU ARE ABOUT TO BEGIN ADDING TUPLES"
    @ 9,30 SAY "TO THE"
    @ 9,30 SAY TITLE
    @ 10,22 SAY "RELATION."
    WAIT TO STOP
    DO CASE
    CASE choice = "UPROC_S"
        USE UPROC_S
        APPEND
        RETURN
    CASE choice = "SPROC_P"
        USE SPROC_P
        APPEND
        RETURN
    CASE choice = "PPROC_F"
        USE PPROC_F
        APPEND
        RETURN
    CASE choice = "PPROC_R"
        USE PPROC_R
        APPEND
        RETURN
    CASE choice = "PPROC_E"
        USE PPROC_E
        APPEND
        RETURN
    CASE choice = "SCONT_P"
        USE SCONT_P
        APPEND
        RETURN
    CASE choice = "PCONT_M"
        USE PCONT_M
        APPEND
        RETURN
    DESIGNED BY: ROBERT A. KIRSCH II
    WRITTEN BY: ROBERT A. KIRSCH II
    BASIC FUNCTION OF MODULE:
    THIS MODULE ALLOW FOR THE ACTUAL INPUT OF ADDITIONAL TUPLES TO THE
    RELATIONSHIP RELATION SELECTED.
    CLEAR
    do while t.
    RESTORE FROM mem_var
    CLEAR
    @ 0,1 SAY "1.1.4.1.0.0"
    @ 1,22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
    @ 3,32 SAY "ADD RELATIONSHIP"
    @ 8,22 SAY "YOU ARE ABOUT TO BEGIN ADDING TUPLES"
    @ 9,30 SAY "TO THE"
    @ 9,30 SAY TITLE
    @ 10,22 SAY "RELATION."
    WAIT TO STOP
    DO CASE
    CASE choice = "UPROC_S"
        USE UPROC_S
        APPEND
        RETURN
    CASE choice = "SPROC_P"
        USE SPROC_P
        APPEND
        RETURN
    CASE choice = "PPROC_F"
        USE PPROC_F
        APPEND
        RETURN
    CASE choice = "PPROC_R"
        USE PPROC_R
        APPEND
        RETURN
    CASE choice = "PPROC_E"
        USE PPROC_E
        APPEND
        RETURN
    CASE choice = "SCONT_P"
        USE SCONT_P
        APPEND
        RETURN
    CASE choice = "PCONT_M"
        USE PCONT_M
        APPEND
        RETURN
    DESIGNED BY: ROBERT A. KIRSCH II
    WRITTEN BY: ROBERT A. KIRSCH II
    BASIC FUNCTION OF MODULE:
    THIS MODULE ALLOW FOR THE ACTUAL INPUT OF ADDITIONAL TUPLES TO THE
    RELATIONSHIP RELATION SELECTED.
USE P CONT_M
APPEND
RETURN
CASE choice = "F CONT_R"
USE F CONT_R
APPEND
RETURN
CASE choice = "R CONT_E"
USE R CONT_E
APPEND
RETURN
CASE choice = "U RESP_S"
USE U RESP_S
APPEND
RETURN
CASE choice = "U RESP_F"
USE U RESP_F
APPEND
RETURN
CASE choice = "P PROD_D"
USE P PROD_D
APPEND
RETURN
CASE choice = "13"
RETURN
CASE choice = "14"
RETURN TO MASTER
OTHERWISE
CLEAR
@ 1,21 SAY choice
@ 2,20 SAY "IS NOT A VALID CHOICE"
@ 2,20 SAY "PLEASE ENTER VALUES BETWEEN 1 AND 14 ONLY"
@ 3,20 SAY "PRESS RETURN TO TRY AGAIN!"
ACCEPT TO hold.
ENDCASE
ENDDO
RETURN
* 115000.PRG
* MODULE NAME: 1.1.5.0.0.0
* INPUT FILES: NONE
* OUTPUT FILES: NONE
* ROUTINES THAT CALL THE MODLUE: 1.1.0.0.0.0, MAIN
* LOCAL VARIABLES USED:
* choice : CONTAINS THE NUMBER OF ACTION SELECTED. MAY ALSO CONTAIN THE
* CHARACTER STRING THAT IDENTIFIES THE RELATION BEING ADDED TO,
* MODIFIED, DELETED FROM OR OUTPUT.
* hold : USED TO STOP ACTION FOR USER DECISION.
* t : REPRESENTS THE BOOLEAN TRUE IS USED TO CREATE A CONTINUES
* LOOP.
* title : CONTAINS THE CHARACTER STRING THAT DESCRIBES THE RELATIONSHIP
* BEING ADDED TO, DELETED FROM OR OUTPUT.
* INPUT FILE : MEM VAR.
* OUTPUT FILES: MEM VAR.
* DESIGNED BY: ROBERT A. KIRSCH II
* WRITTEN BY: ROBERT A. KIRSCH II
* BASIC FUNCTION OF MODULE:
* THIS MODULE ALLOW THE USER TO CHOOSE WHICH RELATIONSHIP HE WOULD
* LIKE TO DELETE TUPLES FROM.
* do while .t.
ERASE mem_var.mem
CLEAR
@ 0.1 SAY "1.1.5.0.0.0"
@ 1.2 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3.29 SAY "DELETE FROM RELATIONSHIP"
@ 5.9 SAY "1) USER CONTAINS SYSTEM"
@ 5.64 SAY "ORDS"
@ 7.9 SAY "2) SYSTEM CONTAINS PROGRAM"
@ 7.64 SAY "LEMENT"
@ 9.9 SAY "3) PROGRAM PROCESSES FILE"
@ 9.64 SAY "FOR SYSTEM"
@ 11.9 SAY "4) PROGRAM PROCESSES RECORD"
@ 11.64 SAY "FOR FILE"
@ 13.9 SAY "5) PROGRAM PROCESSES ELEMENT"
@ 13.64 SAY "DOCUMENT"
@ 15.6 SAY "6) SYSTEM CONTAINS PROGRAM"
@ 15.64 SAY "5 MENU"
@ 17.9 SAY "7) PROGRAM CONTAINS MODULE"
@ 17.64 SAY "NU"
@ 18.22 SAY "11 ACCEPT 'ENTER YOUR CHOICE (1-14) FROM ABOVE: ' TO choice
DO CASE
CASE choice = "1"
stroe 'U PROC S' to choice
store 'USER-PROCESSES-SYSTEM' to title
save to mem_var
do 115100
CASE choice = "2"
stroe 'S PROC P' to choice
store 'SYSTEM-PROCESSES-PROGRAM' to title
save to mem_var
do 115100
CASE choice = "3"
stroe 'P PROC F' to choice
store 'PROGRAM-PROCESSES-FILE' to title
save to mem_var
do 115100
CASE choice = "4"
stroe 'P PROC R' to choice
store 'PROGRAM-PROCESSES-RECORD' to title
save to mem_var
do 115100
CASE choice = "5"
stroe 'P PROC E' to choice

157
store 'PROGRAM-PROCESSES-ELEMENT' TO title
save to mem_var
do 115100
CASE choice = "6"
store 'S CONT P' to choice
store 'SYSTEM-CONTAINS-PROGRAM' to title
save to mem_var
do 115100
CASE choice = "7"
store 'P CONT M' to choice
store 'PROGRAM-CONTAINS-MODULE' to title
save to mem_var
do 115100
CASE choice = "8"
store 'F CONT R' to choice
store 'FILE-CONTAINS-RECORD' to title
save to mem_var
do 115100
CASE choice = "9"
store 'R CONT E' to choice
store 'RECORD-CONTAINS-ELEMENT' to title
save to mem_var
do 115100
CASE choice = "10"
store 'U RESP S' to choice
store 'USER-RESPONSIBLE-FOR-SYSTEM' to title
save to mem_var
do 115100
CASE choice = "11"
store 'U RESP F' to choice
store 'USER-RESPONSIBLE-FOR-FILE' to title
save to mem_var
do 115100
CASE choice = "12"
store 'P PROD D' to choice
store 'PROGRAM-PRODUCES-DOCUMENT' to title
save to mem_var
do 115100
CASE choice = "13"
RETURN
CASE choice = "14"
RETURN TO MASTER
OTHERWISE
CLEAR
@ 1,21 SAY choice
@ 1,28 SAY "IS NOT A VALID CHOICE"
@ 2,20 SAY "PLEASE ENTER VALUES BETWEEN 1 AND 14 ONLY"
@ 3,20 SAY "PRESS RETURN TO TRY AGAIN!"
ACCEPT TO hold
ENDCASE
ENDDO
RETURN
* 115100.PRG
* MODULE NAME: 1.1.5.1.0.0
* INPUT FILES: NONE
* OUTPUT FILES: NONE
* ROUTINES THAT CALL THE MODLUE: 1.1.5.0.0.0
* ROUTINES THAT THE MODULE CALLS: 1.1.5.0.0.0
* LOCAL VARIABLES USED:
* choice : CONTAINS THE NUMBER OF ACTION SELECTED. MAY ALSO CONTAIN THE
* CHARACTER STRING THAT IDENTIFIES THE RELATION BEING ADDED TO,
* MODIFIED, DELETED FROM OR OUTPUT.
* hold : USED TO STOP ACTION FOR USER DECISION.
* rec_num : CONTAINS THE VALUE OF THE POINTER TO THE TUPLE TO BE CHANGED.
* t : REPRESENTS THE BOOLEAN TRUE IS USED TO CREATE A CONTINUES
* LOOP.
* title : CONTAINS THE CHARACTER STRING THAT DESCRIBES THE RELATIONSHIP
* BEING ADDED TO, DELETED FROM OR OUTPUT.
* INPUT FILES: MEM VAR, U CONT S, U CONT_P, P PROC F, P_PROC R,
* P_PROC R, P_PROC E, S CONT_P, P_CONT M, F_CONT R, R_CONT E,
* U_RESP S, U_RESP F, P_PRED_D.
* OUTPUT FILES: MEM VAR, U CONT S, U CONT_P, P PROC F, P_PROC R,
* P_PROC R, P_PROC E, S CONT_P, P_CONT M, F_CONT R, R_CONT E,
* U_RESP S, U_RESP F, P_PRED_D.

* DESIGNED BY: ROBERT A. KIRSCH II
* WRITTEN BY: ROBERT A. KIRSCH II

* BASIC FUNCTION OF MODULE:
* THIS MODULE ALLOW FOR THE ACTUAL DELETION INPUT OF ADDITIONAL TUPLES
* FROM THE DESIGNATED RELATIONSHIP FILE SELECTED.

* do while .t.
RESTORE FROM mem_var
CLEAR @ 0.1 SAY "1.1.5.1.0.0"
@ 1.22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3.29 SAY "DELETE FROM RELATIONSHIP"
@ 8.21 SAY "ENTER TUPLE NUMBER OF THE"
@ 10.24 SAY TITLE
@ 12.21 SAY "TUPLE THAT YOU WISH TO HAVE DELETED."
@ 13.21 SAY "THE TUPLE WILL BE DISPLAYED FOR"
@ 14.21 SAY "YOU TO EXAMINE. IF YOU ARE"
@ 15.21 SAY "SURE THAT YOU ARE DELETING THE"
@ 16.21 SAY "RIGHT TUPLE, DEPRESS -U . IF"
@ 17.21 SAY "YOU DO NOT WANT IT DELETED,"
@ 18.21 SAY "TYPE 0 FOR TUPLE NUMBER"
@ 19.21 SAY "TO RETURN TO PREVIOUS MENU."
@ 20.21 SAY "" ACCEPT'
ENTER THE TUPLE NUMBER NOW ' TO rec_num
DO WHILE rec_num <> '0' DO CASE
CASE choice = "U_PROC_S"
USE U_PROC_S
GOTO (VAL(rec_num))
EDIT RETURN.
CASE choice = "S_PROC_P"
USE S_PROC_P
GOTO (VAL(rec_num))
EDIT RETURN.
CASE choice = "P_PROC_F"
USE P_PROC_F
GOTO (VAL(rec_num))
EDIT RETURN.
CASE choice = "P_PROC_R"
USE P_PROC_R
GOTO (VAL(rec_num))
EDIT
RETURN
CASE choice = "P_PROC_E"
USE P_PROC_E
GOTO (VAL(Fec_num))
EDIT
RETURN
CASE choice = "S_CONT_P"
USE S_CONT_P
GOTO (VAL(rec_num))
EDIT
RETURN
CASE choice = "P_CONT_M"
USE P_CONT_M
GOTO (VAL(Fec_num))
EDIT
RETURN
CASE choice = "F_CONT_R"
USE F_CONT_R
GOTO (VAL(Fec_num))
EDIT
RETURN
CASE choice = "R_CONT_E"
USE R_CONT_E
GOTO (VAL(Fec_num))
EDIT
RETURN
CASE choice = "U_RESP_S"
USE U_RESP_S
GOTO (VAL(Fec_num))
EDIT
RETURN
CASE choice = "U_RESP_F"
USE U_RESP_F
GOTO (VAL(Fec_num))
EDIT
RETURN
CASE choice = "P_PROD_D"
USE P_PROD_D
GOTO (VAL(Fec_num))
EDIT
RETURN
ENDCASE
ENDDO
RETURN
* 120000.PRG
* MODULE NAME: 1.2.0.0.0.0
* INPUT FILES: NONE
* OUTPUT FILES: NONE
* ROUTINES THAT CALL THE MODULE: MAIN
* ROUTINES THAT THE MODULE CALLS: 1.2.1.0.0.0, 1.2.2.0.0.0, MAIN.
* LOCAL VARIABLES USED: choice: CONTAINS THE NUMBER OF ACTION SELECTED.
  t: REPRESENTS NO VALUE AT ALL.
  hold: USED TO STOP ACTION FOR USER DECISION.
* DESIGNED BY: ROBERT A. KIRSCH II
* WRITTEN BY: ROBERT A. KIRSCH II
* BASIC FUNCTION OF MODULE:
  THIS PROGRAM ALLOWS THE USER TO SELECT ENTITY RELATIONS,
  AND RELATIONSHIP RELATIONS FOR OUTPUT.

do while .t.
CLEAR
@ 0,1 SAY "1.2.0.0.0.0"
@ 1,22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3,31 SAY "DICTIONARY OUTPUT"
@ 6,22 SAY "1) ENTITY"
@ 8,22 SAY "2) RELATIONSHIP"
@ 10,22 SAY "3) RETURN TO MAIN MENU"
@ 11,22 SAY " "
ACCEPT ' ENTER YOUR CHOICE (1-3) FROM ABOVE: ' TO choice
DO CASE
CASE choice = "1"
do 121000
CASE choice = "2"
do 122000
CASE choice = "3"
RETURN TO MASTER
OTHERWISE
CLEAR
@ 2,18 SAY choice
@ 2,21 SAY "IS NOT A VALID CHOICE"
@ 3,18 SAY "PLEASE ENTER VALUES BETWEEN 1 AND 3 ONLY"
@ 4,18 SAY "PRESS RETURN TO TRY AGAIN!"
ACCEPT TO hold
ENDCASE
ENDDO
RETURN
* 121000.PRG
* MODULE NAME: 1.2.1.0.0.0
* INPUT FILES: NONE
* OUTPUT FILES: NONE
* ROUTINES THAT CALL THE MODULE: 1.2.0.0.0.0
* ROUTINES THAT THE MODULE CALLS: 1.2.0.0.0.0, 1.2.1.0.0.0 MAIN
* LOCAL VARIABLES USED: choice; CONTAINS THE NUMBER OF ACTION SELECTED.
  t; REPRESENTS NO VALUE AT ALL.
  hold; USED TO STOP ACTION FOR USER DECISION.
* INPUT FILE : MEM VAR.
* OUTPUT FILE: MEM VAR
* DESIGNED BY: ROBERT A. KIRSCH II
* WRITTEN BY: ROBERT A. KIRSCH II
* BASIC FUNCTION OF MODULE:
* THIS MODULE ALLOW THE USER TO CHOOSE WHICH TYPE OF ENTITY RELATION
* TO OUTPUT.
* do while .t.
ERASE mem_var.mem
CLEAR
@ 0,1 SAY "1.2.1.0.0.0"
@ 1.22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3.34 SAY "ENTITY OUTPUT"
@ 6.15 SAY "1) USER 6) FILE"
@ 8.15 SAY "2) SYSTEM 7) RECORD"
@ 12.15 SAY "3) PROGRAM 8) ELEMENT"
@ 14.15 SAY "4) MODULE 9) RETURN TO PREVIOUS MENU"
@ 15.1 SAY "5) DOCUMENT 10) RETURN TO MAIN MENU"
ACCEPT ENTER YOUR CHOICE (1-10) FROM ABOVE: 'TO choice
DO CASE
CASE choice = "1"
store 'USER' to choice
save to mem_var
DO 121100
CASE choice = "2"
store 'SYSTEM' to choice
save to mem_var
DO 121100
CASE choice = "3"
store 'PROGRAM' to choice
save to mem_var
DO 121100
CASE choice = "4"
store 'MODULE' to choice
save to mem_var
DO 121100
CASE choice = "5"
store 'DOCUMENT' to choice
save to mem_var
DO 121100
CASE choice = "6"
store 'FILE' to choice
save to mem_var
DO 121100
CASE choice = "7"
store 'RECORD' to choice
save to mem_var
DO 121100
CASE choice = "8"
store 'ELEMENT' to choice
save to mem_var
DO 121100
CASE choice = "9"
RETURN
CASE choice = "10"
RETURN TO MASTER
OTHERWISE
CLEAR
@ 1.23 SAY choice
@ 1.31 SAY "IS NOT A VALID CHOICE"
@ 2.18 SAY "PLEASE ENTER VALUES BETWEEN 1 AND 10 ONLY"
@ 3.18 SAY "PRESS RETURN TO TRY AGAIN!"
ACCEPT TO hold
ENDCASE
ENDDO
RETURN
* 121100.PRG
* MODULE NAME: 1.2.1.1.0.0
* INPUT FILES: NONE
* OUTPUT FILES: NONE
* ROUTINES THAT CALL THE MODULE: 1.2.1.0.0.0
* ROUTINES THAT THE MODULE CALLS: 1.2.1.0.0.0
* LOCAL VARIABLES USED:
  * choice : CONTAINS THE NUMBER OF ACTION SELECTED. MAY ALSO CONTAIN THE
  * CHARACTER STRING THAT IDENTIFIES THE RELATION BEING ADDED TO,
  * MODIFIED, DELETED FROM OR OUTPUT.
  * hold : USED TO STOP ACTION FOR USER DECISION.
  * option : CONTAINS THE USER'S CHOICE ON WHETHER TO OUTPUT TO THE SCREEN
  * OR THE PRINTER.
  * t : REPRESENTS THE BOOLEAN TRUE IS USED TO CREATE A CONTINUES
  * LOOP.
  * true : USED AS A BOOLEAN VALUE IN LOOPS.
* INPUT FILES: MEM VAR
* OUTPUT FILES: MEM VAR
* DESIGNED BY: ROBERT A. KIRSCH II
* WRITTEN BY: ROBERT A. KIRSCH II
* BASIC FUNCTION OF MODULE:
  * THIS MODULE ALLOW THE USER TO CHOOSE WHETHER THE OUTPUT WILL BE
  * DISPLAYED ON THE SCREEN OR PRINTED.
* RESTORE FROM mem_var
STORE 0 TO rec_num, stop
STORE t. TO TRUE
DO while TRUE
  CLEAR
  @ 0.1 SAY "1.2.1.1.0.0"
  RESTORE FROM mem_var
  @ 2.22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
  @ 4.33 SAY "ENTITY OUTPUT"
  @ 8.23 SAY "LISTED BELOW ARE THE CHOICES FOR HOW"
  @ 9.23 SAY "YOU CAN HAVE THE RELATION"
  @ 9.50 SAY choice
  @ 10.23 SAY "DISPLAYED."
  @ 12.28 SAY "1) SCREEN OUTPUT"
  @ 14.28 SAY "2) PRINTER OUTPUT"
  @ 16.28 SAY "3) RETURN TO PREVIOUS MENU"
  @ 17.1 SAY " ACCEPT ENTER YOUR CHOICE (1-3) FROM ABOVE 'TO option
ERASE mem_var.mem
SAVE TO mem_var
DO CASE
  CASE option = '1'
  DO CASE
    CASE choice = 'USER'
      DO 121110
    CASE choice = 'SYSTEM'
      DO 121110
    CASE choice = 'PROGRAM'
      DO 121110
    CASE choice = 'DOCUMENT'
      DO 121110
    CASE choice = 'FILE'
      DO 121120
    CASE choice = 'RECORD'
      DO 121120
    CASE choice = 'ELEMENT'
      DO 121120
  END CASE
  CASE option = '2'
  DO CASE
  CASE option = 'USER'
  DO 121130

164
CASE choice = 'SYSTEM'
   DO 121130
CASE CHOICE = 'PROGRAM'
   DO 121130
CASE choice = 'MODULE'
   DO 121130
CASE CHOICE = 'DOCUMENT'
   DO 121140
CASE choice = 'FILE'
   DO 121140
CASE CHOICE = 'RECORD'
   DO 121140
CASE choice = 'ELEMENT'
   DO 121140
ENDCASE
CASE option = '3'
RETURN
OTHERWISE
CLEAR
@ 0,27 SAY option
@ 0,34 SAY "IS NOT A VALID CHOICE"
@ 1,26 SAY "PLEASE ENTER VALUES BETWEEN 1 AND 3 ONLY"
ACCEPT TO hold
ENDCASE
ENDDO
* MODULE NAME: 1.2.1.1.1.0
* INPUT FILES: NONE
* OUTPUT FILES: NONE
* ROUTINES THAT CALL THE MODLUE: 1.2.1.1.0.0
* ROUTINES THAT THE MODULE CALLS: 1.2.1.1.0.0
* LOCAL VARIABLES USED: choice: CONTAINS THE NUMBER OF ACTION SELECTED.
  * t: REPRESENTS NO VALUE AT ALL.
  * stop, hold: USED TO STOP ACTION FOR USER DECISION.
  * count: KEEPS TRACK OF ACCOUNT NUMBERS.
* INPUT FILE: MEM VAR.
* OUTPUT FILE: MEM VAR.
* DESIGNED BY: ROBERT A. KIRSCH II
* WRITTEN BY: ROBERT A. KIRSCH II
* BASIC FUNCTION OF MODULE:
  * THIS MODULE WILL DISPLAY ON THE SCREEN USER, SYSTEM,
  * PROGRAM AND MODULE RELATIONS
* RESTORE FROM mem_var
CLEAR
@ 0.1 SAY "1.2.1.1.1.0"
@ 1.22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3.30 SAY "ENTITY SCREEN OUTPUT"
@ 5.22 SAY "THIS MODULE WILL DISPLAY"
@ 5.48 SAY choice
@ 7.22 SAY "IF YOU DO NOT WISH TO DISPLAY"
@ 8.22 SAY "THIS ENTITY, TYPE '0' TO"
@ 9.22 SAY "RETURN TO THE PREVIOUS MENU."
WAIT TO stop
DO CASE
  CASE stop = '0'
    RETURN
  OTHERWISE
  ENDCASE
DO CASE
  CASE choice = 'USER'
    USE USER
    STORE 1 TO count
    SET HEADING OFF
    DO WHILE .NOT. EOF()
      CLEAR
      @ 0.33 SAY CHOICE
      @ 2.1 SAY "RECORD"
      @ 2.11 SAY count
      store count + 1 to count
      @ 4.1 SAY "ACCESS NAME:"
      @ 4.28 SAY ACC NAME
      @ 5.1 SAY "IDENTIFICATION NAME:"
      @ 5.28 SAY ID NAME
      @ 6.1 SAY "DATE TUPLE ADDED:"
      @ 6.28 SAY DATE ADDED
      @ 7.1 SAY "TUPLE ADDED BY:"
      @ 7.28 SAY ADDED BY
      @ 8.1 SAY "DATE TUPLE LAST MODIFIED:"
      @ 8.28 SAY LST MOD DT
      @ 9.1 SAY "TUPLE LAST MODIFIED BY:"
      @ 9.28 SAY LST MOD BY
      @ 10.1 SAY "NUMBER OF MODIFICATIONS:"
      @ 10.28 SAY NUM OF MOD
      @ 11.1 SAY "DESCRIPTION:"
      @ 11.28 SAY DESCRIPT
      @ 15.1 SAY "COMMENTS:"
      @ 15.28 SAY COMMENTS
      ACCEPT 'PRESS RETURN TO SEE NEXT TUPLE' TO hold
      SKIP
    ENDDO
RETURN
CASE choice = 'SYSTEM'
USE SYSTEM
STORE 1 TO count
SET HEADING OFF
DO WHILE .NOT. EOF()
CLEAR
@ 0.33 SAY CHOICE
@ 2,1 SAY "RECORD"
@ 2,11 SAY count
store count + 1 to count
@ 4,1 SAY "ACCESS NAME:"
@ 4,28 SAY ACC NAME
@ 5,1 SAY "IDENTIFICATION NAME:"
@ 5,28 SAY ID NAME
@ 6,1 SAY "DATE TUPLE ADDED:"
@ 6,28 SAY DATE ADDED
@ 7,1 SAY "TUPLE ADDED BY:"
@ 7,28 SAY ADDED BY
@ 8,1 SAY "DATE TUPLE LAST MODIFIED:"
@ 8,28 SAY LST MOD DT
@ 9,1 SAY "TUPLE LAST MODIFIED BY:"
@ 9,28 SAY LST MOD BY
@ 10,1 SAY "NUMBER OF MODIFICATIONS:"
@ 10,28 SAY NUM OF MOD
@ 11,1 SAY "DESCRIPTION:"
@ 11,28 SAY DESCRIPT
@ 15,1 SAY "COMMENTS:"
@ 15,28 SAY COMMENTS
ACCEPT 'PRESS RETURN TO SEE NEXT TUPLE' TO hold
SKIP
ENDDO
RETURN
CASE choice = 'PROGRAM'
USE PROGRAM
STORE 1 TO count
SET HEADING OFF
DO WHILE .NOT. EOF()
CLEAR
@ 0.33 SAY CHOICE
@ 2,1 SAY "RECORD"
@ 2,11 SAY count
store count + 1 to count
@ 4,1 SAY "ACCESS NAME:"
@ 4,28 SAY ACC NAME
@ 5,1 SAY "IDENTIFICATION NAME:"
@ 5,28 SAY ID NAME
@ 6,1 SAY "DATE TUPLE ADDED:"
@ 6,28 SAY DATE ADDED
@ 7,1 SAY "TUPLE ADDED BY:"
@ 7,28 SAY ADDED BY
@ 8,1 SAY "DATE TUPLE LAST MODIFIED:"
@ 8,28 SAY LST MOD DT
@ 9,1 SAY "TUPLE LAST MODIFIED BY:"
@ 9,28 SAY LST MOD BY
@ 10,1 SAY "NUMBER OF MODIFICATIONS:"
@ 10,28 SAY NUM OF MOD
@ 11,1 SAY "DESCRIPTION:"
@ 11,28 SAY DESCRIPT
@ 15,1 SAY "COMMENTS:"
@ 15,28 SAY COMMENTS
ACCEPT 'PRESS RETURN TO SEE NEXT TUPLE' TO hold
SKIP
ENDDO
RETURN
CASE choice = 'MODULE'
USE MODULE
STORE 1 TO count
SET HEADING OFF
DO WHILE .NOT. EOF()
CLEAR
@ 2.33 SAY CHOICE
@ 4.1 SAY "RECORD"
@ 4.11 SAY count
STORE count + 1 to count
@ 6.1 SAY "ACCESS NAME:"
@ 6.28 SAY ACC NAME
@ 7.1 SAY "IDENTIFICATION NAME:"
@ 7.28 SAY ID NAME
@ 8.1 SAY "DATE TUPLE ADDED:"
@ 8.28 SAY DATE ADDED
@ 9.1 SAY "TUPLE ADDED BY:"
@ 9.28 SAY ADDED BY
@ 10.1 SAY "DATE TUPLE LAST MODIFIED:"
@ 10.28 SAY LST MOD DT
@ 11.1 SAY "TUPLE LAST MODIFIED BY:"
@ 11.28 SAY LST MOD BY
@ 12.1 SAY "NUMBER OF MODIFICATIONS:"
@ 12.28 SAY NUM OF MOD
@ 13.1 SAY "DESCRIPTION:"
@ 11.28 SAY DESCRIPT
@ 17.1 SAY "COMMENTS:"
@ 15.28 SAY COMMENTS
ACCEPT 'PRESS RETURN TO SEE NEXT TUPLE' TO hold
SKIP
ENDDO
RETURN
ENDCASE
* 121120.PRG
* MODULE NAME: 1.2.1.1.2.0
* INPUT FILES: NONE
* OUTPUT FILES: NONE
* ROUTINES THAT CALL THE MODULE: 1.2.1.1.0.0
* ROUTINES THAT THE MODULE CALLS: 1.2.1.1.0.0
* LOCAL VARIABLES USED: choice: CONTAINS THE NUMBER OF ACTION SELECTED.
* t: REPRESENTS NO VALUE AT ALL.
* stop, hold: USED TO STOP ACTION FOR USER DECISION.
* count: KEEPS TRACK OF ACCOUNT NUMBERS.
* INPUT FILE: MEM VAR.
* OUTPUT FILE: NE VAR.
* DESIGNED BY: ROBERT A. KIRSCH II
* WRITTEN BY: ROBERT A. KIRSCH II
* BASIC FUNCTION OF MODULE:
* THIS MODULE WILL DISPLAY ON THE SCREEN DOCUMENT, FILE,
* RECORD, AND ELEMENT RELATIONS.

RESTORE FROM mem_var
STORE 0 TO rec_num, stop
CLEAR
@ 0.1 SAY "1.2.1.1.2.0"
@ 1.22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3.30 SAY "ENTITY SCREEN OUTPUT"
@ 5.22 SAY "THIS MODULE WILL DISPLAY"
@ 5.48 SAY choice
@ 7.22 SAY "IF YOU DO NOT WISH TO DISPLAY THIS"
@ 8.22 SAY "ENTITY, TYPE '0' TO RETURN TO"
@ 9.22 SAY "PREVIOUS MENU."
WAIT TO stop
DO CASE
CASE stop = '0'
RETURN
OTHERWISE
ENDCASE
DO CASE
CASE choice = 'DOCUMENT'
USE DOCUMENT
STORE 1 TO count
SET HEADING OFF
DO WHILE .NOT. EOF()
CLEAR
@ 0.33 SAY choice
@ 2.1 SAY "RECORD"
@ 2.11 SAY count
store count + 1 to count
@ 4.1 SAY "ACCESS NAME:"
@ 4.28 SAY ACC NAME
@ 5.1 SAY "IDENTIFICATION NAME:"
@ 5.28 SAY ID NAME
@ 6.1 SAY "DATE TUPLE ADDED:"
@ 6.28 SAY DATE ADDED
@ 7.1 SAY "TUPLE ADDED BY:"
@ 7.28 SAY ADDED BY
@ 8.1 SAY "DATE TUPLE LAST MODIFIED:"
@ 8.28 SAY LST MOD DT
@ 9.1 SAY "TUPLE LAST MODIFIED BY:"
@ 9.28 SAY LST MOD BY
@ 10.1 SAY "NUMBER OF MODIFICATIONS:"
@ 10.28 SAY NUM OF MOD
@ 11.1 SAY "DESCRIPTION:"
@ 11.28 DESCRIPT
@ 15.1 SAY "COMMENTS:"
@ 15.28 COMMENTS
ACCEPT 'PRESS RETURN TO SEE NEXT TUPLE': TO hold
SKIP
ENDDO

169
RETURN
CASE choice = 'FILE'
USE FILE
STORE 1 TO count
SET HEADING OFF
DO WHILE .NOT. EOF()
CLEAR
@ 0,33 SAY choice
@ 2,1 SAY "RECORD"
@ 2,11 SAY count
store count + 1 to count
@ 4,1 SAY "ACCESS NAME:"
@ 4,28 SAY ACC NAME
@ 5,1 SAY "IDENTIFICATION NAME:"
@ 5,26 SAY ID NAME
@ 6,1 SAY "DATE TUPLE ADDED:"
@ 6,28 SAY DATE ADDED
@ 7,1 SAY "TUPLE ADDED BY:"
@ 7,28 SAY ADDED BY
@ 8,1 SAY "DATE TUPLE LAST MODIFIED:"
@ 8,28 SAY LST MOD DT
@ 9,1 SAY "TUPLE LAST MODIFIED BY:"
@ 9,28 SAY LST MOD BY
@ 10,1 SAY "NUMBER OF MODIFICATIONS:"
@ 10,28 SAY NUM OF MOD
@ 11,1 SAY "DESCRIPTION:"
@ 11,28 DESCRIPT
@ 15,1 SAY "COMMENTS:"
@ 15,28 ACCEPT 'PRESS RETURN TO SEE NEXT TUPLE' TO hold
SKIP
ENDDO
RETURN
CASE choice = 'RECORD'
USE RECORD
STORE 1 TO count
SET HEADING OFF
DO WHILE .NOT. EOF()
CLEAR
@ 0,33 SAY choice
@ 2,1 SAY "RECORD"
@ 2,11 SAY count
store count + 1 to count
@ 4,1 SAY "ACCESS NAME:"
@ 4,28 SAY ACC NAME
@ 5,1 SAY "IDENTIFICATION NAME:"
@ 5,26 SAY ID NAME
@ 6,1 SAY "DATE TUPLE ADDED:"
@ 6,28 SAY DATE ADDED
@ 7,1 SAY "TUPLE ADDED BY:"
@ 7,28 SAY ADDED BY
@ 8,1 SAY "DATE TUPLE LAST MODIFIED:"
@ 8,28 SAY LST MOD DT
@ 9,1 SAY "TUPLE LAST MODIFIED BY:"
@ 9,28 SAY LST MOD BY
@ 10,1 SAY "NUMBER OF MODIFICATIONS:"
@ 10,28 SAY NUM OF MOD
@ 11,1 SAY "DESCRIPTION:"
@ 11,28 DESCRIPT
@ 15,1 SAY "COMMENTS:"
@ 15,28 ACCEPT 'PRESS RETURN TO SEE NEXT TUPLE' TO hold
SKIP
ENDDO
RETURN
CASE choice = 'ELEMENT'
USE ELEMENT
STORE 1 TO count
SET HEADING OFF
DO WHILE .NOT. EOF()
CLEAR
@ 0,33 SAY choice
@ 2,1 SAY "RECORD"
@ 2,11 SAY count
store count + 1 to count
@ 4,1 SAY "ACCESS NAME;"
@ 4,28 SAY ACC_NAME
@ 5,1 SAY "IDENTIFICATION NAME;"
@ 5,28 SAY ID_NAME
@ 6,1 SAY "DATE TUPLE ADDED;"
@ 6,28 SAY DATE_ADDED
@ 7,1 SAY "TUPLE ADDED BY;"
@ 7,28 SAY ADDED_BY
@ 8,1 SAY "DATE TUPLE LAST MODIFIED;"
@ 8,28 SAY LST_MOD_DT
@ 9,1 SAY "TUPLE LAST MODIFIED BY;"
@ 9,28 SAY LST_MOD_BY
@ 10,1 SAY "NUMBER OF MODIFICATIONS;"
@ 10,28 SAY NUM_OF_MOD
@ 11,1 SAY "DESCRIPTION;"
@ 11,28 DESCRIPT
@ 15,1 SAY "COMMENTS;"
@ 15,28 COMMENTS
ACCEPT 'PRESS RETURN TO SEE NEXT TUPLE'TO hold
SKIP
ENDDO
RETURN
ENDCASE
* MODULE NAME: 1.2.1.1.3.0
* INPUT FILES: NONE
* OUTPUT FILES: NONE
* ROUTINES THAT CALL THE MODULE: 1.2.1.1.0.0
* ROUTINES THAT THE MODULE CALLS: 1.2.1.1.0.0
* LOCAL VARIABLES USED: choice: CONTAINS THE NUMBER OF ACTION SELECTED.
  * t: REPRESENTS NO VALUE AT ALL.
  * count: KEEPS TRACK OF ACCOUNT NUMBERS.
* INPUT FILE: MEM VAR.
* OUTPUT FILE: MEM VAR.
* DESIGNED BY: ROBERT A. KIRSCH II
* WRITTEN BY: ROBERT A. KIRSCH II
* BASIC FUNCTION OF MODULE:
  * THIS MODULE WILL OUTPUT THE USER, SYSTEM, PROGRAM AND MODULE
  * RELATION FILES TO THE PRINTER.
* RESTORE FROM mem_var
STORE 0 TO rec_num, stop
CLEAR
@ 0,1 SAY "1.2.1.1.3.0"
@ 1,22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3,29 SAY "ENTITY PRINTER OUTPUT"
@ 6,23 SAY "THIS MODULE WILL PRINT"
@ 6,47 SAY choice
@ 8,23 SAY "PLEASE INSURE THAT YOUR PRINTER"
@ 9,23 SAY "IS TURNED ON AND IN THE ONLINE"
@ 10,23 SAY "MODE"
@ 12,23 SAY "IF YOU DO NOT WISH TO PRINT"
@ 13,23 SAY "THIS ENTITY, TYPE '0' TO"
@ 14,23 SAY "RETURN TO THE PREVIOUS MENU"
WAIT TO stop
DO CASE
CASE stop = '0'
RETURN
OTHERWISE
ENDCASE
SET DEVICE TO PRINT
SET CONSOLE OFF
DO CASE
CASE choice = 'USER'
USE USER
STORE 1 TO count
SET HEADING OFF
DO WHILE .NOT. EOF()
CLEAR
@ 0,33 SAY choice
@ 2,1 SAY "RECORD"
store count + 1 to count
@ 4,1 SAY "ACCESS NAME:"
@ 4,28 SAY ACC NAME
@ 5,1 SAY "IDENTIFICATION NAME:"
@ 5,28 SAY ID NAME
@ 6,1 SAY "DATE TUPLE ADDED:"
@ 6,28 SAY DATE ADDED
@ 7,1 SAY "TUPLE ADDED BY:"
@ 7,28 SAY ADDED BY
@ 8,1 SAY "DATE TUPLE LAST MODIFIED:"
@ 8,28 SAY LST MOD DT
@ 9,1 SAY "TUPLE LAST MODIFIED BY:"
@ 9,28 SAY LST MOD BY
@ 10,1 SAY "NUMBER OF MODIFICATIONS:"
@ 10,28 SAY NUM OF MOD
@ 11,1 SAY "DESCRIPTION:"
@ 11,28 SAY DESCRIPT
CASE choice = 'SYSTEM'
USE SYSTEM
STORE 1 TO count
SET HEADING OFF
DO WHILE .NOT. EOF()
CLEAR
@ 0.33 SAY choice
@ 2.1 SAY "RECORD"
@ 2.11 SAY count
store count + 1 to count
@ 4.1 SAY "ACCESS NAME:"
@ 4.28 SAY ACC NAME
@ 5.1 SAY "IDENTIFICATION NAME:"
@ 5.28 SAY ID NAME
@ 6.1 SAY "DATE TUPLE ADDED:"
@ 6.28 SAY DATE ADDED
@ 7.1 SAY "TUPLE ADDED BY:"
@ 7.28 SAY ADDED
@ 8.1 SAY "DATE TUPLE LAST MODIFIED:"
@ 8.28 SAY LST MOD DT
@ 9.1 SAY "TUPLE LAST MODIFIED BY:"
@ 9.28 SAY LST MOD BY
@ 10.1 SAY "NUMBER OF MODIFICATIONS:"
@ 10.28 SAY NUM OF MOD
@ 11.1 SAY "DESCRIPTION:"
@ 11.28 SAY DESCRIPT
@ 15.1 SAY "COMMENTS:"
@ 15.28 SAY COMMENTS
@ 18.1 SAY ""
SKIP
ENDDO
CASE choice = 'PROGRAM'
USE PROGRAM
STORE 1 TO count
SET HEADING OFF
DO WHILE .NOT. EOF()
CLEAR
@ 0.33 SAY choice
@ 2.1 SAY "RECORD"
@ 2.11 SAY count
store count + 1 to count
@ 4.1 SAY "ACCESS NAME:"
@ 4.28 SAY ACC NAME
@ 5.1 SAY "IDENTIFICATION NAME:"
@ 5.28 SAY ID NAME
@ 6.1 SAY "DATE TUPLE ADDED:"
@ 6.28 SAY DATE ADDED
@ 7.1 SAY "TUPLE ADDED BY:"
@ 7.28 SAY ADDED
@ 8.1 SAY "DATE TUPLE LAST MODIFIED:"
@ 8.28 SAY LST MOD DT
@ 9.1 SAY "TUPLE LAST MODIFIED BY:"
@ 9.28 SAY LST MOD BY
@ 10.1 SAY "NUMBER OF MODIFICATIONS:"
@ 10.28 SAY NUM OF MOD
@ 11.1 SAY "DESCRIPTION:"
@ 11.28 SAY DESCRIPT
@ 15.1 SAY "COMMENTS:"
@ 15.28 SAY COMMENTS
@ 18.1 SAY ""
SKIP
173
ENDDO
CASE choice = 'MODULE'
USE MODULE
STORE 1 to count
SET HEADING OFF
DO WHILE .NOT. EOF()
CLEAR
@ 0,33 SAY CHOICE
@ 2,1 SAY "RECORD"
@ 2,11 SAY count
store count + 1 to count
@ 4,1 SAY "ACCESS NAME:"
@ 4,28 SAY ACC NAME
@ 5,1 SAY "IDENTIFICATION NAME:"
@ 5,28 SAY ID NAME
@ 6,1 SAY "DATE TUPLE ADDED:"
@ 6,28 SAY DATE ADDED
@ 7,1 SAY "TUPLE ADDED BY:"
@ 7,28 SAY ADDED BY
@ 8,1 SAY "DATE TUPLE LAST MODIFIED:"
@ 8,28 SAY LST MOD DT
@ 9,1 SAY "TUPLE LAST MODIFIED BY:"
@ 9,28 SAY LST MOD BY
@ 10,1 SAY "NUMBER OF MODIFICATIONS:"
@ 10,28 SAY NUM OF MOD
@ 11,1 SAY "DESCRIPTION:"
@ 11,28 SAY DESCRIPT
@ 15,1 SAY "COMMENTS:"
@ 15,28 SAY COMMENTS
@ 18,1 SAY ""
SKIP
ENDDO
ENDCASE
SET DEVICE TO SCREEN
SET CONSOLE ON
RETURN
* 121140.PRG
* MODULE NAME: 1.2.1.1.4.0
* INPUT FILES: NONE
* OUTPUT FILES: NONE
* ROUTINES THAT CALL THE MODULE: 1.2.1.1.0.0
* ROUTINES THAT THE MODULE CALLS: 1.2.1.1.0.0
* LOCAL VARIABLES USED: choice: CONTAINS THE NUMBER OF ACTION SELECTED.
* t: REPRESENTS NO VALUE AT ALL.
* stop: hold: USED TO STOP ACTION FOR USER DECISION.
* count: KEEPS TRACK OF ACCOUNT NUMBERS.
* INPUT FILE: MEM VAR.
* OUTPUT FILE: MEM VAR.
* DESIGNED BY: ROBERT A. KIRSCH II
* WRITTEN BY: ROBERT A. KIRSCH II
* BASIC FUNCTION OF MODULE:
* THIS MODULE WILL OUTPUT THE FIRST FOUR RELATIONSHIP
* RELATION FILES TO THE PRINTER.
* RESTORE FROM mem_var
STORE 0 TO rec_num, stop
CLEAR
@ 0,1 SAY "1.2.1.1.4.0"
@ 1,22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3,29 SAY "ENTITY PRINTER OUTPUT"
@ 6,23 SAY "THIS MODULE WILL PRINT"
@ 6,47 SAY choice
@ 8,23 SAY "PLEASE INSURE THAT YOUR PRINTER"
@ 9,23 SAY "IS TURNED ON AND IN THE ONLINE"
@ 10,23 SAY "NOPE"
@ 12,23 SAY "IF YOU DO NOT WISH TO PRINT"
@ 13,23 SAY "THIS RELATION, TYPE '0' TO"
@ 14,23 SAY "RETURN TO THE PREVIOUS MENU"
WAIT TO stop
DO CASE
CASE stop = '0'
RETURN
OTHERWISE
ENDCASE
SET DEVICE TO PRINT
SET CONSOLE OFF
DO CASE
CASE choice = 'DOCUMENT'
USE DOCUMENT
STORE 1 TO count
SET HEADING OFF
DO WHILE .NOT. EOF()
CLEAR
@ 0,33 SAY CHOICE
@ 2,1 SAY "RECORD"
@ 2,11 SAY count
store count + 1 to count
@ 4,1 SAY "ACCESS NAME:"
@ 4,28 SAY ACC NAME
@ 5,1 SAY "IDENTIFICATION NAME:"
@ 5,28 SAY ID NAME
@ 6,1 SAY "DATE TUPLE ADDED:"
@ 6,28 SAY DATE_ADDED
@ 7,1 SAY "TUPLE ADDED BY:"
@ 7,28 SAY ADDED
@ 8,1 SAY "DATE TUPLE LAST MODIFIED:"
@ 8,28 SAY LST_MOD_DT
@ 9,1 SAY "TUPLE LAST MODIFIED BY:"
@ 9,28 SAY LST_MOD_BY
@ 10,1 SAY "NUMBER OF MODIFICATIONS:"
@ 10,28 SAY NUM_OF_MOD
@ 11,1 SAY "DESCRIPTION:"
@ 11,28 DESCRIPT

175
CASE choice = 'FILE'
USE FILE
STORE 1 TO count
SET HEADING OFF
DO WHILE .NOT. EOF()
CLEAR
@ 0,33 SAY CHOICE
@ 2,1 SAY "RECORD"
@ 2,11 SAY count
store count + 1 to count
@ 4,1 SAY "ACCESS NAME:"
@ 4,28 SAY ACC NAME
@ 5,1 SAY "IDENTIFICATION NAME:"
@ 5,28 SAY ID NAME
@ 6,1 SAY "DATE TUPLE ADDED:"
@ 6,28 SAY DATE ADDED
@ 7,1 SAY "TUPLE ADDED BY:"
@ 7,28 SAY ADDED BY
@ 8,1 SAY "DATE TUPLE LAST MODIFIED:"
@ 8,28 SAY LST MOD DT
@ 9,1 SAY "TUPLE LAST MODIFIED BY:"
@ 9,28 SAY LST MOD BY
@ 10,1 SAY "NUMBER OF MODIFICATIONS:"
@ 10,28 SAY NUM OF MOD
@ 11,1 SAY "DESCRIPTION:"
@ 11,28 DESCRIPT
@ 15,1 SAY "COMMENTS:"
@ 15,28 COMMENTS
@ 18,1 SAY ""
SKIP
ENDDO
CASE choice = 'RECORD'
USE RECORD
STORE 1 TO count
SET HEADING OFF
DO WHILE .NOT. EOF()
CLEAR
@ 0,33 SAY CHOICE
@ 2,1 SAY "RECORD"
@ 2,11 SAY count
store count + 1 to count
@ 4,1 SAY "ACCESS NAME:"
@ 4,28 SAY ACC NAME
@ 5,1 SAY "IDENTIFICATION NAME:"
@ 5,28 SAY ID NAME
@ 6,1 SAY "DATE TUPLE ADDED:"
@ 6,28 SAY DATE ADDED
@ 7,1 SAY "TUPLE ADDED BY:"
@ 7,28 SAY ADDED BY
@ 8,1 SAY "DATE TUPLE LAST MODIFIED:"
@ 8,28 SAY LST MOD DT
@ 9,1 SAY "TUPLE LAST MODIFIED BY:"
@ 9,28 SAY LST MOD BY
@ 10,1 SAY "NUMBER OF MODIFICATIONS:"
@ 10,28 SAY NUM OF MOD
@ 11,1 SAY "DESCRIPTION:"
@ 11,28 DESCRIPT
@ 15,1 SAY "COMMENTS:"
@ 15,28 COMMENTS
@ 18,1 SAY ""
SKIP
ENDDO

176
ENDDO
CASE choice = 'ELEMENT'
USE ELEMENT
STORE 1 TO count
SET HEADING OFF
DO WHILE .NOT. EOF()
CLEAR
@ 0,33 SAY CHOICE
@ 2,1 SAY "RECORD"
@ 2,11 SAY count
store count + 1 to count
@ 4,1 SAY "ACCESS NAME:"
@ 4,28 SAY ACC NAME
@ 5,1 SAY "IDENTIFICATION NAME:"
@ 5,28 SAY ID NAME
@ 6,1 SAY "DATE TUPLE ADDED:"
@ 6,28 SAY DATE ADDED
@ 7,1 SAY "TUPLE ADDED BY:"
@ 7,28 SAY ADDED BY
@ 8,1 SAY "DATE TUPLE LAST MODIFIED:"
@ 8,28 SAY LST MOD DT
@ 9,1 SAY "TUPLE LAST MODIFIED BY:"
@ 9,28 SAY LST MOD BY
@ 10,1 SAY "NUMBER OF MODIFICATIONS:"
@ 10,28 SAY NUM OF MOD
@ 11,1 SAY "DESCRIPTION:"
@ 11,28 DESCRIP
@ 15,1 SAY "COMMENTS:"
@ 15,28 COMMENTS
@ 18,1 SAY ""
SKIP
ENDDO
ENDCASE
SET DEVICE TO SCREEN
SET CONSOLE ON
RETURN
* 122100.PRG
* MODULE NAME: 1.2.2.1.0.0
* INPUT FILES: NONE
* OUTPUT FILES: NONE
* ROUTINES THAT CALL THE MODLUE: 1.2.2.0.0.0
* ROUTINES THAT THE MODLUE CALLS: 1.2.2.0.0.0
* LOCAL VARIABLES USED:
  * choice - contains the number of action selected. May also contain the character string that identifies the relation being added to, modified, deleted from or output.
  * hold - used to stop action for user decision.
  * option - contains the user's choice on whether to output to the screen or the printer.
  * t - represents the boolean true is used to create a continues loop.
* INPUT FILES: MEM VAR.
* OUTPUT FILES: MEM VAR.
* DESIGNED BY: ROBERT A. KIRSCH II
* WRITTEN BY: ROBERT A. KIRSCH II
* BASIC FUNCTION OF MODULE:
  * This module allow the user to choose whether the output will be displayed on the screen or printed.

RESTORE FROM mem_var
STORE 0 TO rec_num, stop
STORE t. TO TRUE
DO while TRUE
CLEAR
@ 0.1 SAY "1.2.2.1.0.0"
RESTORE FROM mem_var
@ 2.22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 4.29 SAY "RELATIONSHIP OUTPUT"
@ 8.23 SAY "LISTED BELOW ARE THE CHOICES FOR"
@ 9.23 SAY "HOW YOU CAN HAVE THE RELATIONSHIP"
@ 11.24 SAY TITLE
@ 13.23 SAY "DISPLAYED."
@ 15.28 SAY "1) SCREEN OUTPUT"
@ 17.28 SAY "2) PRINTER OUTPUT"
@ 19.28 SAY "3) RETURN TO PREVIOUS MENU"
@ 20.1 SAY ""
ACCEPT 'ENTER YOUR CHOICE (1-3) FROM ABOVE TO option
ERASE mem_var.mem
SAVE TO mem_var
DO CASE
CASE option = '1'
DO CASE
CASE choice = 'U_PROC_S'
DO 122110
CASE choice = 'S_PROC_P'
DO 122110
CASE choice = 'P_PROC_F'
DO 122110
CASE choice = 'P_PROC_R'
DO 122110
CASE choice = 'P_PROC_E'
DO 122120
CASE choice = 'S_CONT_P'
DO 122120
CASE choice = 'P_CONT_M'
DO 122120
CASE choice = 'F_CONT_R'
DO 122120
CASE choice = 'R_CONT_E'
DO 122120
CASE choice = 'U_RESP_S'
DO 122130
CASE choice = 'U_RESP_F'
DO 122130
178
CASE choice = 'P_PROD_D'
DO 122130
ENDCASE
CASE option = '2'
DO CASE
CASE choice = 'U_PROC_S'
DO 122140
CASE choice = 'S_PROC_P'
DO 122140
CASE choice = 'P_PROC_F'
DO 122140
CASE choice = 'P_PROC_R'
DO 122140
CASE choice = 'P_PROC_E'
DO 122150
CASE choice = 'S_CONT_P'
DO 122150
CASE choice = 'P_CONT_M'
DO 122150
CASE choice = 'F_CONT_R'
DO 122150
CASE choice = 'R_CONT_E'
DO 122160
CASE choice = 'U_RESP_S'
DO 122160
CASE choice = 'U_RESP_F'
DO 122160
CASE choice = 'P_PROD_D'
DO 122160
ENDCASE
CASE option = '3'
RETURN
OTHERWISE
CLEAR
@ 0.27 SAY option
@ 0.34 SAY "IS NOT A VALID CHOICE"
@ 1.26 SAY "PLEASE ENTER VALUES BETWEEN 1 AND 3 ONLY"
@ 2.26 SAY "PRESS RETURN AND TRY AGAIN!"
ACCEPT TO hold
ENDCASE
ENDDO

179
* 122110.PRG
* MODULE NAME: 1.2.2.1.1.0
* INPUT FILES: NONE
* OUTPUT FILES: NONE
* ROUTINES THAT CALL THE MODULE: 1.2.2.1.0.0
* ROUTINES THAT THE MODULE CALLS: 1.2.2.1.0.0
* LOCAL VARIABLES USED:
* choice : CONTAINS THE NUMBER OF ACTION SELECTED. MAY ALSO CONTAIN THE
*          CHARACTER STRING THAT IDENTIFIES THE RELATION BEING ADDED TO,
*          MODIFIED, DELETED FROM OR OUTPUT.
* count : USED TO KEEP TRACK OF THE RECORD NUMBER BEING DISPLAYED.
* hold : USED TO STOP ACTION FOR USER DECISION.
* option : CONTAINS THE USER'S CHOICE ON WHETHER TO OUTPUT TO THE SCREEN
*          OR THE PRINTER.
* t : REPRESENTS THE BOOLEAN TRUE IS USED TO CREATE A CONTINUOUS
*      LOOP.
* INPUT FILES: MEM VAR U_CONTS, S_CONT_P, P_PROC_F, P_PROC_R.
* OUTPUT FILES: MEM VAR.
* DESIGNED BY: ROBERT A. KIRSCH II
* WRITTEN BY: ROBERT A. KIRSCH II
* BASIC FUNCTION OF MODULE:
*   THIS MODULE WILL DISPLAY ON THE FIRST FOUR RELATIONSHIP TO THE SCREEN.

RESTORE FROM mem_var
CLEAR
@ 0,1 SAY "1.2.2.1.1.0"
@ 1,22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3,28 SAY "RELATIONSHIP SCREEN OUTPUT"
@ 5,22 SAY "THIS MODULE WILL DISPLAY"
@ 7,23 SAY TITLE
@ 9,22 SAY "IF YOU DO NOT WISH TO DISPLAY"
@ 10,22 SAY "THIS RELATIONSHIP, TYPE '0' TO"
@ 11,22 SAY "RETURN TO THE PREVIOUS MENU."
WAIT TO stop
DO CASE
CASE stop = '0'
RETURN
OTHERWISE
ENDCASE
DO CASE
CASE choice = 'U_PROC_S'
CLEAR
USE U_PROC_S
STORE 1 TO count
SET HEADING OFF
DO WHILE NOT. EOF()
CLEAR
@ 2,33 SAY TITLE
@ 4,1 SAY "RECORD"
@ 4,11 SAY count
store count + 1 to count
@ 6,1 SAY "ACCESS NAME:"
@ 6,28 SAY U_NAME
@ 7,1 SAY "IDENTIFICATION NAME:"
@ 7,28 SAY S_NAME
@ 9,1 SAY ""
ACCEPT 'PRESS RETURN TO SEE NEXT TUPLE' to hold
SKIP
ENDDO
RETURN
CASE choice = 'S_PROC_P'
USE S_PROC_P
STORE 1 TO count
SET HEADING OFF
DO WHILE NOT. EOF()
CLEAR
@ 2,33 SAY TITLE
@ 4,1 SAY "RECORD"
@ 4,11 SAY count
store count + 1 to count
@ 6,1 SAY "ACCESS NAME:"
@ 6,28 SAY $ NAME
@ 7,1 SAY "IDENTIFICATION NAME:"
@ 7,28 SAY P NAME
ACCEPT 'PRESS RETURN TO SEE NEXT TUPLE'TO hold
SKIP
ENDDO
RETURN
CASE choice = 'P_PROC_F'
USE P_PROC_F
STORE -1 TO count
SET HEADING OFF
DO WHILE .NOT. EOF()
CLEAR
@ 2,33 SAY TITLE
@ 4,1 SAY "RECORD"
@ 4,11 SAY count
store count + 1 to count
@ 6,1 SAY "ACCESS NAME:"
@ 6,28 SAY P NAME
@ 7,1 SAY "IDENTIFICATION NAME:"
@ 7,28 SAY F NAME
ACCEPT 'PRESS RETURN TO SEE NEXT TUPLE'TO hold
SKIP
ENDDO
RETURN
CASE choice = 'P_PROC_R'
USE P_PROC_R
STORE -1 TO count
SET HEADING OFF
DO WHILE .NOT. EOF()
CLEAR
@ 2,33 SAY TITLE
@ 4,1 SAY "RECORD"
@ 4,11 SAY count
store count + 1 to count
@ 6,1 SAY "ACCESS NAME:"
@ 6,28 SAY P NAME
@ 7,1 SAY "IDENTIFICATION NAME:"
@ 7,28 SAY R NAME
@ 8,1 SAY "COMMENTS:"
DISPLAY OFF COMMENTS
ACCEPT 'PRESS RETURN TO SEE NEXT TUPLE'TO hold
SKIP
ENDDO
RETURN
ENDCASE
* MODULE NAME: 1.2.2.0.0.0
* INPUT FILES: NONE
* OUTPUT FILES: NONE
* ROUTINES THAT CALL THE MODULE: 1.1.0.0.0.0
* ROUTINES THAT THE MODULE CALLS: TBD, MAIN
* LOCAL VARIABLES USED: choice: CONTAINS THE NUMBER OF ACTION SELECTED.
* hold: USED TO STOP ACTION FOR USER DECISION.
* INPUT FILE: MEM VAR.
* OUTPUT FILE: MEM VAR.
* DESIGNED BY: ROBERT A. KIRSCH II
* WRITTEN BY: ROBERT A. KIRSCH II
* BASIC FUNCTION OF MODULE: THIS MODULE ALLOW THE USER TO CHOOSE WHICH RELATIONSHIP HE WOULD LIKE TO DELETE TUPLES FROM.
* 
do while .t.
ERASE mem_var.mem
CLEAR
@ 0.1 SAY "1.2.2.0.0.0"
@ 1.22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3.29 SAY "RELATIONSHIP OUTPUT"
@ 5.9 SAY "1) USER CONTAINS SYSTEM  8)  FILE CONTAINS RECORD"
@ 5.64 SAY "ORDS"
@ 7.9 SAY "2)  SYSTEM CONTAINS PROGRAM  9)  RECORD CONTAINS ELEMENT"
@ 7.64 SAY "ELEMENT"
@ 9.9 SAY "3) PROGRAM PROCESSES FILE 10)  USER RESPONSIBLE"
@ 9.64 SAY "FOR SYSTEM"
@ 11.9 SAY "4) PROGRAM PROCESSES RECORD 11)  USER RESPONSIBLE"
@ 11.64 SAY "FOR FILE"
@ 13.9 SAY "5) PROGRAM PROCESSES ELEMENT 12)  PROGRAM PROCESSES"
@ 13.64 SAY "DOCUMENT"
@ 15.9 SAY "6) SYSTEM CONTAINS PROGRAM 13)  RETURN TO PREVIOUS"
@ 15.64 SAY "S MENU"
@ 17.9 SAY "7) PROGRAM CONTAINS MODULE 14)  RETURN TO MAIN MENU"
@ 17.64 SAY "NU"
@ 18.22 SAY " "
ACCEPT "ENTER YOUR CHOICE (1-14) FROM ABOVE:' TO choice
DO CASE
CASE choice = "1"
store 'U PROC S' to choice
store 'USER PROCESSES SYSTEM' TO title
save to mem_var
do 122100
CASE choice = "2"
store 'S PROC P' to choice
store 'SYSTEM PROCESSES PROGRAM' TO title
save to mem_var
do 122100
CASE choice = "3"
store 'P PROC F' to choice
store 'PROGRAM PROCESSES FILE' TO title
save to mem_var
do 122100
CASE choice = "4"
store 'P PROC R' to choice
store 'PROGRAM PROCESSES RECORD' TO title
save to mem_var
do 122100
CASE choice = "5"
store 'P PROC E' to choice
store 'PROGRAM PROCESSES ELEMENT' TO title
save to mem_var
do 122100
CASE choice = "6"
store 'S CONT P' to choice
store 'SYSTEM CONTAINS PROGRAM' TO title
save to mem_var
do 122100
CASE choice = "7"
store 'P CONT M' to choice
store 'PROGRAM CONTAINS MODULE' TO title
save to mem_var
do 122100
CASE choice = "8"
store 'P CONT R' to choice
store 'FILE CONTAINS RECORD' TO title
save to mem_var
do 122100
CASE choice = "9"
store 'R CONT E' to choice
store 'RECORD CONTAINS ELEMENT' TO title
save to mem_var
do 122100
CASE choice = "10"
store 'U RESP S' to choice
store 'USER RESPONSIBLE FOR SYSTEM' TO title
save to mem_var
do 122100
CASE choice = "11"
store 'U RESP F' to choice
store 'USER RESPONSIBLE FOR FILE' TO title
save to mem_var
do 122100
CASE choice = "12"
store 'P PROD D' to choice
store 'PROGRAM PRODUCES DOCUMENT' TO title
save to mem_var
do 122100
CASE choice = "13"
RETURN
CASE choice = "14"
RETURN TO MASTER
OTHERWISE
CLEAR
@ 1,21 SAY choice
@ 1,28 SAY "IS NOT A VALID CHOICE"
@ 2,20 SAY "PLEASE ENTER VALUES BETWEEN 1 AND 14 ONLY"
@ 3,20 SAY "PRESS RETURN TO TRY AGAIN!"
ACCEPT TO hold
ENDCASE
ENDDO
RETURN
* 122120.PRG
* MODULE NAME: 1.2.2.1.2.0
* INPUT FILES: NONE
* OUTPUT FILES: NONE
* ROUTINES THAT CALL THE MODULE: 1.2.2.1.0.0
* ROUTINES THAT THE MODULE CALLS: 1.2.2.1.0.0
* LOCAL VARIABLES USED:
  * choice : CONTAINS THE NUMBER OF ACTION SELECTED. MAY ALSO CONTAIN THE
    CHARACTER STRING THAT IDENTIFIES THE RELATION BEING ADDED TO,
    MODIFIED, DELETED FROM OR OUTPUT.
  * count : USED TO KEEP TRACK OF THE RECORD NUMBER BEING DISPLAYED.
  * hold : USED TO STOP ACTION FOR USER DECISION.
  * option : CONTAINS THE USER’S CHOICE ON WHETHER TO OUTPUT TO THE SCREEN
    OR THE PRINTER.
  * t : REPRESENTS THE BOOLEAN TRUE IS USED TO CREATE A CONTINUES LOOPS.
* INPUT FILES: MEM VAR P PROC_E, S_CONT_P, P_CONT_M, F_CONT_R.
* OUTPUT FILES: MEM VAR.
* DESIGNED BY: ROBERT A. KIRSCH II
* WRITTEN BY: ROBERT A. KIRSCH II
* BASIC FUNCTION OF MODULE:
  * THIS MODULE WILL DISPLAY THE NEXT FOUR RELATIONSHIP TO THE SCREEN.

RESTORE FROM mem-var
CLEAR
@ 0,1 SAY "1.2.2.1.2.0"
@ 1,22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3,28 SAY "RELATIONSHIP SCREEN OUTPUT"
@ 5,22 SAY "THIS MODULE WILL DISPLAY"
@ 7,23 SAY TITLE
@ 9,22 SAY "IF YOU DO NOT WISH TO DISPLAY"
@ 10,22 SAY "THIS RELATIONSHIP, TYPE '0' TO"
@ 11,22 SAY "RETURN TO THE PREVIOUS MENU."
WAIT TO stop
DO CASE
CASE stop = '0'
RETURN
OTHERWISE
ENDCASE
DO CASE
CASE choice = 'P_PROC_E'
CLEAR
USE P_PROC_E
STORE 1 TO count
SET HEADING OFF
DO WHILE .NOT. EOF()
CLEAR
@ 2,33 SAY TITLE
@ 4,1 SAY "RECORD"
@ 4,11 SAY count
store count + 1 to count
@ 6,1 SAY "ACCESS NAME:" @ 6,28 SAY P NAME
@ 7,1 SAY "IDENTIFICATION NAME:" @ 7,28 SAY E_NAME
ACCEPT 'PRESS RETURN TO SEE NEXT TUPLE' to hold SKIP
ENDDO
RETURN
CASE choice = 'S_CONT_P'
USE S_CONT_P
STORE 1 TO count
SET HEADING OFF
DO WHILE .NOT. EOF()
CLEAR
@ 2,33 SAY TITLE
@ 4,1 SAY "RECORD"
184
CASE choice = 'P_CONT_M'
USE P_CONT_M
STORE 1 TO count
SET HEADING OFF
DO WHILE .NOT. EOF()
CLEAR
@ 2.33 SAY TITLE
@ 4.1 SAY "RECORD"

@ 4.11 SAY count
store count + 1 to count
@ 6.1 SAY "ACCESS NAME:"
@ 6.28 SAY S_NAME
@ 7.1 SAY "IDENTIFICATION NAME:"
@ 7.28 SAY P_NAME
ACCEPT 'PRESS RETURN TO SEE NEXT TUPLE' TO hold
SKIP
ENDDO
RETURN
CASE choice = 'F_CONT_R'
USE F_CONT_R
STORE 1 TO count
SET HEADING OFF
DO WHILE .NOT. EOF()
CLEAR
@ 2.33 SAY TITLE
@ 4.1 SAY "RECORD"

@ 4.11 SAY count
store count + 1 to count
@ 6.1 SAY "ACCESS NAME:"
@ 6.28 SAY F_NAME
@ 7.1 SAY "IDENTIFICATION NAME:"
@ 7.28 SAY R_NAME
@ 8.1 SAY "COMMENTS:"
DISPLAY OFF COMMENTS
ACCEPT 'PRESS RETURN TO SEE NEXT TUPLE' TO hold
SKIP
ENDDO
RETURN
ENDCASE
* 122130.PRG
* MODULE NAME: 1.2.2.1.3.0
* INPUT FILES: NONE
* OUTPUT FILES: NONE
* ROUTINES THAT CALL THE MODULE: 1.2.2.1.0.0
* ROUTINES THAT THE MODULE CALLS: 1.2.2.1.0.0
* LOCAL VARIABLES USED: choice: CONTAINS THE NUMBER OF ACTION SELECTED.
* t: REPRESENTS NO VALUE AT ALL.
* hold: USED TO STOP ACTION FOR USER DECISION.
* count: KEEPS TRACK OF ACCOUNT NUMBERS.
* option:
* DESIGNED BY: ROBERT A. KIRSCH II
* WRITTEN BY: ROBERT A. KIRSCH II
* BASIC FUNCTION OF MODULE:
* THIS MODULE WILL DISPLAY ON THE FIRST THREE RELATIONSHIP
* RELATIONS
* SET EXACT ON
set color to 0/3,3
set talk off
set menu on
SET EXACT ON
RESTORE FROM mem_var
CLEAR
@ 0,1 SAY "1.2.2.1.3.0"
@ 1,22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3,22 SAY "RELATIONSHIP SCREEN OUTPUT"
@ 5,22 SAY "THIS MODULE WILL DISPLAY"
@ 7,23 SAY TITLE
@ 9,22 SAY "IF YOU DO NOT WISH TO DISPLAY"
@ 10,22 SAY "THIS RELATIONSHIP TYPE '0' TO"
@ 11,22 SAY "RETURN TO THE PREVIOUS MENU."
WAIT TO stop
DO CASE
CASE stop = '0'
RETURN
OTHERWISE
ENDCASE
DO CASE
CASE choice = 'R_CONT_E'
CLEAR
USE R_CONT_E
STORE 1 TO count
SET HEADING OFF
DO WHILE .NOT. EOF()
CLEAR
@ 2,33 SAY TITLE
@ 4,1 SAY "RECORD"
" 
@ 4,11 SAY count
store count + 1 to count
@ 6,1 SAY "ACCESS NAME:"
@ 6,28 SAY R_NAME
@ 7,1 SAY "IDENTIFICATION NAME:"
@ 7,28 SAY E_NAME
ACCEPT 'PRESS RETURN TO SEE NEXT TUPLE'TO hold
SKIP
ENDDO
RETURN
CASE choice = 'U_RESP_S'
USE U_RESP_S
STORE 1 TO count
SET HEADING OFF
DO WHILE .NOT. EOF()
CLEAR
@ 2,33 SAY TITLE
@ 4,1 SAY "RECORD"
"
@ 4,11 SAY count
store count + 1 to count
@ 6,1 SAY "ACCESS NAME:"
@ 6,28 SAY U NAME
@ 7,1 SAY "IDENTIFICATION NAME:"
@ 7,28 SAY S NAME
ACCEPT 'PRESS RETURN TO SEE NEXT TUPLE'TO hold SKIP
ENDDO
RETURN
CASE choice = 'U_RESP_F'
USE U_RESP_F
STORE 1 TO count
SET HEADING OFF
DO WHILE .NOT. EOF()
CLEAR
@ 2,33 SAY TITLE
@ 4,1 SAY "RECORD"
"
@ 4,11 SAY count
store count + 1 to count
@ 6,1 SAY "ACCESS NAME:"
@ 6,28 SAY U NAME
@ 7,1 SAY "IDENTIFICATION NAME:"
@ 7,28 SAY F NAME
ACCEPT 'PRESS RETURN TO SEE NEXT TUPLE'TO hold SKIP
ENDDO
RETURN
CASE choice = 'P_PROD_D'
USE P_PROD_D
STORE 1 TO count
SET HEADING OFF
DO WHILE .NOT. EOF()
CLEAR
@ 2,33 SAY TITLE
@ 4,1 SAY "RECORD"
"
@ 4,11 SAY count
store count + 1 to count
@ 6,1 SAY "ACCESS NAME:"
@ 6,28 SAY P NAME
@ 7,1 SAY "IDENTIFICATION NAME:"
@ 7,28 SAY D NAME
@ 8,1 SAY "COMMENTS:"
DISPLAY OFF COMMENTS
ACCEPT 'PRESS RETURN TO SEE NEXT TUPLE'TO hold SKIP
ENDDO
RETURN
ENDCASE
* 122140.PRG
* MODULE NAME: 1.2.1.1.4.0
* INPUT FILES: NONE
* OUTPUT FILES: NONE
* ROUTINES THAT CALL THE MODULE: 1.2.1.1.0.0
* ROUTINES THAT THE MODULE CALLS: 1.2.1.1.0.0
* LOCAL VARIABLES USED:
  * choice : CONTAINS THE NUMBER OF ACTION SELECTED. MAY ALSO CONTAIN THE
            CHARACTER STRING THAT IDENTIFIES THE RELATION BEING ADDED TO,
            MODIFIED, DELETED FROM OR OUTPUT.
  * count : USED TO KEEP TRACK OF THE RECORD NUMBER BEING DISPLAYED.
  * stop, hold : USED TO STOP ACTION FOR USER DECISION.
  * option : CONTAINS THE USER'S CHOICE ON WHETHER TO OUTPUT TO THE SCREEN
            OR THE PRINTER.
  * t : REPRESENTS THE BOOLEAN TRUE IS USED TO CREATE A CONTINUES
       LOOP.
* INPUT FILES: MEM VAR.
* OUTPUT FILES: MER VAR.
* DESIGNED BY: ROBERT A. KIRSCH II
* WRITTEN BY: ROBERT A. KIRSCH II
* BASIC FUNCTION OF MODULE:
  * THIS MODULE WILL OUTPUT THE USER, SYSTEM, PROGRAM AND MODULE
  * RELATION FILES TO THE PRINTER.
* RESTORE FROM mem var
STORE 0 TO rec_num, stop
CLEAR
@ 0,1 SAY "1.2.1.1.4.0"
@ 1.22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3.27 SAY "RELATIONSHIP PRINTER OUTPUT"
@ 6.23 SAY "THIS MODULE WILL PRINT"
@ 8.24 SAY TITLE
@ 10.23 SAY "PLEASE INSURE THAT YOUR PRINTER"
@ 11.23 SAY "IS TURNED ON AND IN THE ONLINE"
@ 12.23 SAY "MODE"
@ 14.23 SAY "IF YOU DO NOT WISH TO PRINT"
@ 15.23 SAY "THIS RELATIONSHIP, TYPE '0' TO"
@ 16.23 SAY "RETURN TO THE PREVIOUS MENU"
WAIT TO stop
DO CASE
CASE stop = '0'
RETURN
OTHERWISE
ENDCASE
SET DEVICE TO PRINT
SET CONSOLE OFF
DO CASE
CASE choice = 'U_PROC_S'
USE U_PROC_S
STORE -1 TO count
SET HEADING OFF
DO WHILE .NOT. EOF()
CLEAR
@ 0.33 SAY choice
@ 2.1 SAY "RECORD"
@ 2.11 SAY count
store count + 1 to count
@ 4.1 SAY "ACCESS NAME:"
@ 4.28 SAY U NAME
@ 5.1 SAY "IDENTIFICATION NAME:"
@ 5.28 SAY S_NAME
SKIP
ENDDO
CASE choice = 'S_PROC_P'
USE S_PROC_P
STORE -1 TO count
SET HEADING OFF

DO WHILE NOT EOF()
CLEAR
@ 0.33 SAY CHOICE
@ 2.1 SAY "RECORD"
@ 2.11 SAY count
store count + 1 to count
@ 4.1 SAY "ACCESS NAME:"
@ 4.28 SAY S_NAME
@ 5.1 SAY "IDENTIFICATION NAME:"
@ 5.28 SAY P_NAME
SKIP
ENDDO
CASE choice = 'P_PROC_F'
USE P_PROC_F
STORE 1 TO count
SET HEADING OFF
DO WHILE NOT EOF()
CLEAR
@ 0.33 SAY CHOICE
@ 2.1 SAY "RECORD"
@ 2.11 SAY count
store count + 1 to count
@ 4.1 SAY "ACCESS NAME:"
@ 4.28 SAY P_NAME
@ 5.1 SAY "IDENTIFICATION NAME:"
@ 5.28 SAY F_NAME
SKIP
ENDDO
CASE choice = 'P_PROC_R'
USE P_PROC_R
STORE 1 TO count
SET HEADING OFF
DO WHILE NOT EOF()
CLEAR
@ 0.33 SAY CHOICE
@ 2.1 SAY "RECORD"
@ 2.11 SAY count
store count + 1 to count
@ 4.1 SAY "ACCESS NAME:"
@ 4.28 SAY P_NAME
@ 5.1 SAY "IDENTIFICATION NAME:"
@ 5.28 SAY R_NAME
SKIP
ENDDO
ENDCASE
SET DEVICE TO SCREEN
SET CONSOLE ON
RETURN
* 122150.PRG
* MODULE NAME: 1.2.1.1.5.0
* INPUT FILES: NONE
* OUTPUT FILES: NONE
* ROUTINES THAT CALL THE MODULE: 1.2.1.1.0.0
* ROUTINES THAT THE MODULE CALLS: 1.2.1.1.0.0
* LOCAL VARIABLES USED:
  * choice: CONTAINS THE NUMBER OF ACTION SELECTED. MAY ALSO CONTAIN THE
    CHARACTER STRING THAT IDENTIFIES THE RELATION BEING ADDED TO,
    MODIFIED, DELETED FROM OR OUTPUT.
  * count: USED TO KEEP TRACK OF THE RECORD NUMBER BEING DISPLAYED.
  * hold: USED TO STOP ACTION FOR USER DECISION.
  * option: CONTAINS THE USER'S CHOICE ON WHETHER TO OUTPUT TO THE SCREEN
    OR THE PRINTER.
  * t: REPRESENTS THE BOOLEAN TRUE IS USED TO CREATE A CONTINUES
    LOOP.
* INPUT FILE: P_PROC_E, S_CONT_P, P_CONT_M, F_CONT_R.
* DESIGNED BY: ROBERT A. KIRSCH II
* WRITTEN BY: ROBERT A. KIRSCH II
* BASIC FUNCTION OF MODULE:
  * THIS MODULE WILL OUTPUT THE NEXT FOUR RELATIONSHIP
  * RELATION FILES TO THE PRINTER.
* RESTORE FROM mem_var
STORE 0 TO rec_num, stop
CLEAR
@ 0,1 SAY "1.2.1.1.5.0"
@ 1.22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3,27 SAY "RELATIONSHIP PRINTER OUTPUT"
@ 6,23 SAY "THIS MODULE WILL PRINT"
@ 8,24 SAY TITLE
@ 10,23 SAY "PLEASE INSURE THAT YOUR PRINTER"
@ 11,23 SAY "IS TURNED ON AND IN THE ONLINE"
@ 12,23 SAY "MODE"
@ 14,23 SAY "IF YOU DO NOT WISH TO PRINT"
@ 15,23 SAY "THIS RELATIONSHIP, TYPE '0' TO"
@ 16,23 SAY "RETURN TO THE PREVIOUS MENU"
WAIT TO stop
DO CASE
CASE stop = '0'
RETURN
OTHERWISE
ENDCASE
SET DEVICE TO PRINT
SET CONSOLE OFF
DO CASE
CASE choice = 'P_PROC_E'
USE P_PROC_E
STORE 1 TO count
SET HEADING OFF
DO WHILE .NOT. EOF()
CLEAR
@ 0,33 SAY choice
@ 2,1 SAY "RECORD"
@ 2,11 SAY count
STORE count + 1 TO count
@ 4,1 SAY "ACCESS NAME:"
@ 4,28 SAY P_NAME
@ 5,1 SAY "IDENTIFICATION NAME:"
@ 5,28 SAY E_NAME
SKIP
ENDDO
CASE choice = 'S_CONT_P'
USE S_CONT_P
STORE 1 TO count
SET HEADING OFF
DO WHILE .NOT. EOF()
CLEAR
@ 0,33 SAY CHOICE
@ 2,1 SAY "RECORD"
@ 2,11 SAY count
store count + 1 to count
@ 4,1 SAY "ACCESS NAME:"
@ 4,28 SAY S_NAME
@ 5,1 SAY "IDENTIFICATION NAME:"
@ 5,28 SAY P_NAME
SKIP
ENDDO
CASE choice = 'P_CONT_M'
USE P_CONT_M
STORE-1 to count
SET HEADING OFF
DO WHILE .NOT. EOF()
CLEAR
@ 0,33 SAY CHOICE
@ 2,1 SAY "RECORD"
@ 2,11 SAY count
store count + 1 to count
@ 4,1 SAY "ACCESS NAME:"
@ 4,28 SAY P_NAME
@ 5,1 SAY "IDENTIFICATION NAME:"
@ 5,28 SAY M_NAME
SKIP
ENDDO
CASE choice = 'F_CONT_R'
USE F_CONT_R
STORE-1 to count
SET HEADING OFF
DO WHILE .NOT. EOF()
CLEAR
@ 0,33 SAY CHOICE
@ 2,1 SAY "RECORD"
@ 2,11 SAY count
store count + 1 to count
@ 4,1 SAY "ACCESS NAME:"
@ 4,28 SAY F_NAME
@ 5,1 SAY "IDENTIFICATION NAME:"
@ 5,28 SAY R_NAME
SKIP
ENDDO
ENDCASE
SET DEVICE TO SCREEN
SET CONSOLE ON
RETURN
A RELATIONAL DATA DICTIONARY COMPATIBLE WITH THE NATIONAL BUREAU OF STAND. (U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA R A KIRSCH DEC 85

UNCLASSIFIED

END
* 122160.PRG
* MODULE NAME: 1.2.1.1.6.0
* INPUT FILES: NONE
* OUTPUT FILES: NONE
* ROUTINES THAT CALL THE MODULE: 1.2.1.1.0.0
* ROUTINES THAT THE MODULE CALLS: 1.2.1.1.0.0
* LOCAL VARIABLES USED:
  * choice: CONTAINS THE NUMBER OF ACTION SELECTED. MAY ALSO CONTAIN THE
    * CHARACTER STRING THAT IDENTIFIES THE RELATION BEING ADDED TO,
    * MODIFIED, DELETED FROM OR OUTPUT.
  * count: USED TO KEEP TRACK OF THE RECORD NUMBER BEING DISPLAYED.
  * hold: USED TO STOP ACTION FOR USER DECISION.
  * option: CONTAINS THE USER'S CHOICE ON WHETHER TO OUTPUT TO THE SCREEN
    OR THE PRINTER.
  * t: REPRESENTS THE BOOLEAN TRUE IS USED TO CREATE A CONTINUES LOOP.
* INPUT FILES: R_CONT_E, U_RESP_S, U_RESP_F, P_PRED_D.

DESIGNED BY: ROBERT A. KIRSCH II
WRITTEN BY: ROBERT A. KIRSCH II

BASIC FUNCTION OF MODULE:
THIS MODULE WILL OUTPUT THE LAST FOUR RELATIONSHIP FILES TO THE PRINTER.

RESTORE FROM mem_var
STORE 0 TO rec_num, stop
CLEAR
@ 0.1 SAY "1.2.1.1.6.0"
@ 1.22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3.27 SAY "RELATIONSHIP PRINTER OUTPUT"
@ 6.23 SAY "THIS MODULE WILL PRINT"
@ 8.24 SAY TITLE
@ 10.23 SAY "PLEASE INSURE THAT YOUR PRINTER"
@ 11.23 SAY "IS TURNED ON AND IN THE ONLINE"
@ 12.23 SAY "MODE"
@ 14.23 SAY "IF YOU DO NOT WISH TO PRINT"
@ 15.23 SAY "THIS RELATIONSHIP, TYPE '0' TO"
@ 16.23 SAY "RETURN TO THE PREVIOUS MENU"
WAIT TO stop
DO CASE
CASE stop = '0'
RETURN
OTHERWISE
ENDCASE
SET DEVICE TO PRINT
SET CONSOLE OFF
DO CASE
CASE choice = 'R_CONT_E'
USE R_CONT_E
STORE 1 TO count
SET HEADING OFF
DO WHILE .NOT. EOF()
CLEAR
@ 0.33 SAY choice
@ 2.1 SAY "RECORD"
@ 2.11 SAY count
store count + 1 to count
@ 4.1 SAY "ACCESS NAME:"
@ 4.28 SAY R_NAME
@ 5.1 SAY "IDENTIFICATION NAME:"
@ 5.28 SAY E_NAME
SKIP
ENDDO
CASE choice = 'U_RESP_S'
USE U_RESP_S
STORE 1 TO count
SET HEADING OFF
DO WHILE .NOT. EOF()
CLEAR
@ 0,33 SAY CHOICE
@ 2,1 SAY "RECORD"
@ 2,11 SAY count
store count + 1 to count
@ 4,1 SAY "ACCESS NAME:"
@ 4,28 SAY U_NAME
@ 5,1 SAY "IDENTIFICATION NAME:"
@ 5,28 SAY S_NAME
SKIP
ENDDO
CASE choice = 'U_RESP_F'
USE U_RESP_F
STORE 1 TO count
SET HEADING OFF
DO WHILE .NOT. EOF()
CLEAR
@ 0,33 SAY CHOICE
@ 2,1 SAY "RECORD"
@ 2,11 SAY count
store count + 1 to count
@ 4,1 SAY "ACCESS NAME:"
@ 4,28 SAY U_NAME
@ 5,1 SAY "IDENTIFICATION NAME:"
@ 5,28 SAY F_NAME
SKIP
ENDDO
CASE choice = 'P_PROD_D'
USE P_PROD_D
STORE 1 TO count
SET HEADING OFF
DO WHILE .NOT. EOF()
CLEAR
@ 0,33 SAY CHOICE
@ 2,1 SAY "RECORD"
@ 2,11 SAY count
store count + 1 to count
@ 4,1 SAY "ACCESS NAME:"
@ 4,28 SAY P_NAME
@ 5,1 SAY "IDENTIFICATION NAME:"
@ 5,28 SAY D_NAME
SKIP
ENDDO
ENDCASE
SET DEVICE TO SCREEN
SET CONSOLE ON
RETURN
* 130000.PRG
* MODULE NAME: 1.3.0.0.0.0
* ROUTINES THAT CALL THE MODLUE: 1.1.0.0.0.0
* ROUTINES THAT THE MODULE CALLS: 1.3.1.0.0.0, 1.3.2.0.0.0, 1.3.3.0.0.0
* 1.3.4.0.0.0, 1.3.5.0.0.0, 1.3.6.0.0.0, MAIN
* LOCAL VARIABLES USED:
* choice: CONTAINS THE NUMBER OF ACTION SELECTED.
* t: REPRESENTS NO VALUE AT ALL.
* hold: USED TO STOP ACTION FOR USER DECISION.
* name: CONTAINS THE ENTITY RELATION NAME.
* entity1: CONTAINS THE ACCESS-NAME FOR THE ENTITY RELATION BEING ?
* QUERIED.
* OUTPUT FILE: MEM VAR.
* DESIGNED BY: ROBERT A. KIRSCH II
* WRITTEN BY: ROBERT A. KIRSCH II
* BASIC FUNCTION OF MODULE:
* THIS MODULE ALLOW THE USER TO CHOOSE WHICH TYPE OF ENTITY RALATION
* AND ACCESS NAME VALUE THAT WILL BE USED IN THE QUERY

set color to 0/3,3
set talk off
ERASE mem_var.mem
CLEAR
STORE .t. TO true
do while true
CLEAR
@ 0.1 SAY "1.3.0.0.0.0"
@ 1.22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3.35 SAY "QUERY MENU"
@ 5.11 SAY "ENTITY-1 RELATIONSHIP ENTITY"
@ 5.66 SAY "-?"
@ 8.10 SAY "1) USER"
@ 9.10 SAY "2) SYSTEM"
@ 10.10 SAY "3) PROGRAM"
@ 11.10 SAY "4) MODULE"
@ 12.10 SAY "5) DOCUMENT"
@ 13.10 SAY "6) FILE"
@ 14.10 SAY "7) RECORD"
@ 15.10 SAY "8) ELEMENT"
@ 16.10 SAY "9) RETURN TO PREVIOUS MENU"
@ 17.9 SAY "10) RETURN TO MAIN MENU"
@ 18.4 SAY "-ACCEPT- ENTER YOUR CHOICE (1-10) FROM ABOVE: 'TO choice
STORE .f. TO true
DO CASE
CASE choice = "1"
STORE 'USER' TO name
CASE choice = "2"
STORE 'SYSTEM' TO name
CASE choice = "3"
STORE 'PROGRAM' TO name
CASE choice = "4"
STORE 'MODULE' TO name
CASE choice = "5"
STORE 'DOCUMENT' TO name
CASE choice = "6"
STORE 'FILE' TO name
CASE choice = "7"
STORE 'RECORD' TO name
CASE choice = "8"
STORE 'ELEMENT' TO name
CASE choice = "9"
RETURN
CASE choice = "10"
RETURN TO MASTER
OTHERWISE
CLEAR

195
@ 2.14 SAY "PLEASE ENTER VALUES BETWEEN 1 AND 10 ONLY"
@ 3.14 SAY "PRESS RETURN TO TRY AGAIN!"
ACCEPT TO hold
STORE .t. TO true
ENDCASE
ENDDO
STORE 'N' TO correct,
DO WHILE correct = 'N'
CLEAR
STORE ' ' TO entity1
@ 1.1 SAY "1.3.0.0.0.0.0"
@ 2.22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 4.35 SAY "QUERY MENU"
@ 6.12 SAY name
@ 6.31 SAY "RELATIONSHIP ENTITY-2"
@ 8.4 SAY "ENTER THE ACCESS-NAME FOR"
@ 8.31 SAY name
@ 9.4 SAY "YOU WISH TO QUERY ON"
@ 9.26 GET entity1
@ 10.4 SAY "AND PRESS RETURN"
READ
STORE 'Y' TO correct
@ 13.3 SAY "IS THIS THE ENTITY YOU WISH TO QUERY ON"
@ 13.44 SAY ENTITY1
@ 13.56 SAY "Y OR N"
@ 13.64 GET correct
READ
ENDDO
DO CASE
CASE choice = "1"
STORE 'USER' TO choice
STORE 100 TO selection
SAVE TO mem_var
do 131000
CASE choice = "2"
STORE 'SYSTEM' TO choice
STORE 200 TO selection
SAVE TO mem_var
do 132000
CASE choice = "3"
STORE 'PROGRAM' TO choice
STORE 300 TO selection
SAVE TO mem_var
do 133000
CASE choice = "4"
STORE 'MODULE' TO choice
STORE 400 TO selection
SAVE TO mem_var
do 134000
CASE choice = "5"
STORE 'DOCUMENT' TO choice
STORE 500 TO selection
SAVE TO mem_var
do 135000
CASE choice = "6"
STORE 'FILE' TO choice
STORE 600 TO selection
SAVE TO mem_var
do 136000
CASE choice = "7"
STORE 'RECORD' TO choice
STORE 700 TO selection
SAVE TO mem_var
do 137000
CASE choice = "8"
STORE 'ELEMENT' TO choice
STORE 800 TO selection
SAVE TO mem_var
do 138000
ENDCASE
* 131000.PRG
* MODULE NAME: 1.3.1.0.0.0
* INPUT FILES: NONE
* OUTPUT FILES: NONE
* ROUTINES THAT CALL THE MODULE: 1.3.0.0.0.0
* ROUTINES THAT THE MODULE CALLS: 1.3.1.1.0.0, 1.3.1.2.0.0
* LOCAL VARIABLES USED:
  * choice: CONTAINS THE NUMBER OF ACTION SELECTED.
  * hold: USED TO STOP ACTION FOR USER DECISION.
  * entity1: CONTAINS THE CHARACTER STRING THAT REPRESENTS THE FIRST VALUE IN A QUERY STRING.
  * rel_ship: CONTAINS THE CHARACTER STRING THAT REPRESENTS THE RELATIONSHIP VALUE IN A QUERY STRING.
  * true: USED AS A BOOLEAN VALUE IN LOOPS.
  * correct: USED AS TO HOLD USER'S CHOICE FOR LOOP TERMINATION.
  * selection: USED TO HOLD THE VALUE IDENTIFYING WHICH QUERY TO EXECUTE.
* INPUT FILES: MEM VAR
* OUTPUT FILES: MEM VAR
* mem_var.mem: USED TO TEMPORARILY STORE THE MEMORY VARIABLE VALUES.
* temp.dbf: USED TO STORE THE RESULT OF QUERY EXECUTION.
* DESIGNED BY: ROBERT A. KIRSCH II
* WRITTEN BY: ROBERT A. KIRSCH II
* BASIC FUNCTION OF MODULE:
  * THIS MODULE ALLOW THE USER TO CHOOSE WHICH TYPE OF RELATIONSHIP VALUE WILL BE USED IN THE QUERY
  * RESTORE FROM mem_var
  * ERASE mem_var.mem
  * CLEAR
  * STORE 'N' TO correct
  * DO WHILE correct = 'N'
  * STORE .t. TO true
  * do while true
  * CLEAR
  * @ 0,1 SAY "1.3.1.0.0.0"
  * @ 1,22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
  * @ 3,35 SAY "QUERY MENU"
  * @ 5,11 SAY entity1
  * @ 5,33 SAY "RELATIONSHIP ENTITY-2"
  * @ 8,32 SAY "1) CONTAINS"
  * @ 9,32 SAY "2) IS RESPONSIBLE FOR"
  * @ 10,32 SAY "3) RETURN TO PREVIOUS MENU"
  * @ 11,4 SAY ""
  * ACCEPT ENTER YOUR CHOICE (1-3) FROM ABOVE: 'TO choice
  * STORE .f. TO true
  * DO CASE
  * CASE choice = "1"
  * STORE 'CONTAINS' TO rel_ship
  * CASE choice = "2"
  * STORE 'IS RESPONSIBLE FOR' TO rel_ship
  * CASE choice = "3"
  * RETURN
  * OTHERWISE
  * CLEAR
  * @ 2,14 SAY "PLEASE ENTER VALUES BETWEEN 1 AND 3 ONLY"
  * @ 3,14 SAY "PRESS RETURN TO TRY AGAIN!"
  * ACCEPT TO hold
  * STORE .t. TO true
  * ENDCASE
  * ENDDO
  * CLEAR
  * @ 0,1 SAY "1.3.1.0.0.0"
  * @ 1,22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
  * @ 3,35 SAY "QUERY MENU"
  * @ 5,11 SAY entity1
  * @ 5,33 SAY rel_ship
  * @ 8,34 SAY "ENTITY-2"
  * STORE 'Y' TO correct

198
@ 10.3 SAY "IS THIS THE RELATIONSHIP THAT"
@ 11.3 SAY "YOU WISH TO QUERY ON"
@ 11.25 SAY rel ship
@ 12.3 SAY "Y OR N"
@ 12.11 GET correct
READ
ENDDO
DO CASE
CASE choice = "1"
STORE 'PROCESSES' TO choice
STORE selection + 10 TO selection
SAVE TO mem_var
do 131100
CASE choice = "2"
STORE 'IS RESPONSIBLE FOR' TO choice
STORE selection + 20 TO selection
SAVE TO mem_var
do 131200
ENDCASE
* 131100.PRG
* MODULE NAME: 1.3.1.1.0.0
* ROUTINES THAT CALL THE MODULE: 1.3.1.0.0.0
* ROUTINES THAT THE MODULE CALLS: 1.3.1.0.0.0
* LOCAL VARIABLES USED:
* choice : CONTAINS THE NUMBER OF ACTION SELECTED. MAY ALSO CONTAIN THE
*         CHARACTER STRING THAT IDENTIFIES THE RELATION BEING ADDED TO.
* entity1 : CONTAINS THE CHARACTER STRING THAT REPRESENTS THE FIRST VALUE
*         IN A QUERY STRING.
* entity2 : CONTAINS THE CHARACTER STRING THAT REPRESENTS THE SECOND VALUE
*          IN A QUERY STRING.
* hold : USED TO STOP ACTION FOR USER DECISION.
* rel_ship: CONTAINS THE CHARACTER STRING THAT REPRESENTS THE RELATIONSHIP
*          VALUE IN A QUERY STRING.
* t : REPRESENTS THE BOOLEAN TRUE IS USED TO CREATE A CONTINUES
*     LOOP.
* true : USED AS A BOOLEAN VALUE IN LOOPS.
* INPUT FILES: MEM VAR.
* OUTPUT FILES: MEM VAR.* DESIGNED BY: ROBERT A. KIRSCH II
* WRITTEN BY: ROBERT A. KIRSCH II
* BASIC FUNCTION OF MODULE:
* THIS MODULE ALLOW THE USER TO CHOOSE WHICH TYPE OF ENTITY RELATION
* AND ACCESS NAME VALUE THAT WILL BE USED IN THE QUERY
*
set color to 0/3,3
set talk off
SET EXACT ON
ERASE mem var.mem
CLEAR
STORE 'N' TO correct
DO WHILE correct = 'N'
STORE .t. TO true
DO while true
CLEAR
@ 0.1 SAY "1.3.1.1.0.0"
@ 1.22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3.35 SAY "QUERY MENU"
@ 5.12 SAY entity1
@ 5.34 SAY rel_ship
@ 5.60 SAY "ENTITY-2"
@ 8.54 SAY "1) SYSTEM"
@ 9.54 SAY "2) RETURN TO PREVIOUS"
@ 10.59 SAY "MENU"
@ 11.4 SAY " "
ACCEPT 'ENTER YOUR CHOICE (1-2) FROM ABOVE: 'TO choice
STORE .f. TO true
DO CASE
CASE choice = "1"
STORE 'SYSTEM' TO entity2
CASE choice = "2"
RETURN
OTHERWISE
CLEAR
@ 2.14 SAY "PLEASE ENTER VALUES BETWEEN 1 AND 2 ONLY"
@ 3.14 SAY "PRESS RETURN TO TRY AGAIN!"
ACCEPT TO hold
STORE .t. TO true
ENDCASE
ENDDO
CLEAR
@ 0.1 SAY "1.3.1.1.0.0"
@ 1.22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3.35 SAY "QUERY MENU"
@ 5.12 SAY entity1
@ 5.32 SAY rel_ship
@ 5.59 SAY entity2
STORE 'Y' TO correct
@ 8.3 SAY "IS THIS THE ENTITY YOU WISH TO QUERY ON"
@ 8.44 SAY entity2
@ 8.56 SAY "Y OR N"
@ 8.64 GET correct
READ
ENDDO
DO CASE
CASE choice = "1"
SAVE TO mem_var
SELECT 2
USE SYSTEM
SELECT 1
USE U PROC S
JOIN WITH SYSTEM TO TEMP FOR U_NAME = entity1 .AND. S_NAME = 
SYSTEM->ACC_NAME FIELDS ID_NAME, DESCRIPT
SELECT 2
USE
SELECT 1
USE
do 139000
ENDCASE
* 131200.PRG
* MODULE NAME: 1.3.1.2.0.0
* ROUTINES THAT CALL THE MODLUE: 1.3.1.0.0.0
* ROUTINES THAT THE MODULE CALLS: 1.3.1.0.0.0
* LOCAL VARIABLES USED:
  * choice : CONTAINS THE NUMBER OF ACTION SELECTED, MAY ALSO CONTAIN THE
  * CHARACTER STRING THAT IDENTIFIES THE RELATION BEING ADDED TO,
  * MODIFIED, DELETED FROM OR OUTPUT.
  * correct : CONTAINS USER RESPONSE AS TO WHETHER THE DISPLAYED VALUE IS
  * CORRECT OR NOT.
  * entity1 : CONTAINS THE CHARACTER STRING THAT REPRESENTS THE FIRST VALUE
  * IN A QUERY STRING.
  * entity2 : CONTAINS THE CHARACTER STRING THAT REPRESENTS THE SECOND VALUE
  * IN A QUERY STRING.
  * hold : USED TO STOP ACTION FOR USER DECISION.
  * rel_ship: CONTAINS THE CHARACTER STRING THAT REPRESENTS THE RELATIONSHIP
  * VALUE IN A QUERY STRING.
  * true : USED AS A BOOLEAN VALUE IN LOOPS.
* INPUT FILES: MEM VAR.
* OUTPUT FILES: MEM VAR.
* DESIGNED BY: ROBERT A. KIRSCH II
* WRITTEN BY: ROBERT A. KIRSCH II
* BASIC FUNCTION OF MODULE:
  * THIS MODULE ALLOW THE USER TO CHOOSE WHICH TYPE OF ENTITY RELATIONSHIP
  * THAT WILL BE USED IN THE QUERY
  * ERASE mem_var.mem
  CLEAR
STORE 'N' TO correct
DO WHILE correct = 'N'
STORE .t. TO true
do while true
CLEAR
@ 0,1 SAY "1.3.1.1.0.0"
@ 1.22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3.35 SAY "QUERY MENU"
@ 5.11 SAY "ENTITY-1" RELATIONSHIP ENTITY
@ 5.66 SAY "-2"
@ 8.54 SAY "1) SYSTEM"
@ 9.54 SAY "2) FILE"
@ 10.54 SAY "3) RETURN TO PREVIOUS"
@ 11.59 SAY "MENU"
@ 12.4 SAY ""
ACCEPT ENTER YOUR CHOICE (1-10) FROM ABOVE: 'TO choice
STORE .f. TO true
DO CASE
CASE choice = "1"
STORE 'SYSTEM' TO name
CASE choice = "2"
STORE 'FILE' TO name
CASE choice = "3"
RETURN
OTHERWISE
CLEAR
@ 2.14 SAY "PLEASE ENTER VALUES BETWEEN 1 AND 3 ONLY"
@ 3.14 SAY "PRESS RETURN TO TRY AGAIN!"
ACCEPT TO hold
STORE .t. TO true
ENDCASE
ENDDO
CLEAR
@ 0.1 SAY "1.3.1.2.0.0"
@ 1.22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3.35 SAY "QUERY MENU"
@ 5.12 SAY entity1
@ 5.32 SAY rel_ship
@ 5.59 SAY name
STORE 'Y' TO correct
202
@ 8.3 SAY "IS THIS THE ENTITY YOU WISH TO QUERY ON"
@ 8.44 SAY name
@ 8.56 SAY "Y OR N"
@ 8.64 GET correct READ ENDDO
DO CASE
CASE choice = "1"
SELECT 2
USE SYSTEM
SELECT 1
USE U_RESP_S
JOIN WITH SYSTEM TO TEMP FOR U_NAME = ENTITY1 .AND. S_NAME = SYSTEM-> ACC_NAME;
FIELDS ID_NAME, DESCRIP
SELECT 2
USE
SELECT 1
USE
DO 139000
CASE choice = "2"
SELECT 2
USE FILE
SELECT 1
USE U_RESP_F
JOIN WITH SYSTEM TO TEMP FOR U_NAME = ENTITY1 .AND. S_NAME = SYSTEM-> ACC_NAME;
FIELDS ID_NAME, DESCRIP
SELECT 2
USE
SELECT 1
USE
DO 139000
ENDCASE
**132000.PRG**

**MODULE NAME:** 1.3.2.0.0.0

**ROUTINES THAT CALL THE MODULE:** 1.3.0.0.0.0

**ROUTINES THAT THE MODULE CALLS:** MAIN

**LOCAL VARIABLES USED:**

- **choice:** CONTAINS THE NUMBER OF ACTION SELECTED. MAY ALSO CONTAIN THE CHARACTER STRING THAT IDENTIFIES THE RELATION BEING ADDED TO, MODIFIED, DELETED FROM OR OUTPUT.
- **correct:** CONTAINS USER RESPONSE AS TO WHETHER THE DISPLAYED VALUE IS CORRECT OR NOT.
- **entity1:** CONTAINS THE CHARACTER STRING THAT REPRESENTS THE FIRST VALUE IN A QUERY STRING.
- **rel_ship:** CONTAINS THE CHARACTER STRING THAT REPRESENTS THE RELATIONSHIP VALUE IN A QUERY STRING.
- **true:** USED AS A BOOLEAN VALUE IN LOOPS.

**INPUT FILES:** MEM VAR.

**OUTPUT FILES:** MEM VAR.

**DESIGNED BY:** ROBERT A. KIRSCH II

**WRITTEN BY:** ROBERT A. KIRSCH II

**BASIC FUNCTION OF MODULE:**

THIS MODULE ALLOW THE USER TO CHOOSE WHICH TYPE OF RELATIONSHIP THAT WILL BE USED IN THE QUERY

```plaintext
set color to 0/3,3
set talk off
SET EXACT ON
RESTORE FROM mem var
ERASE mem var.mem
CLEAR
STORE 'N' TO correct
DO WHILE correct = 'N'
STORE .t. TO true
DO CASE
CASE choice = "1"
STORE 'PROCESSES' TO rel_ship
CASE choice = "2"
STORE 'IS PROCESSED BY' TO rel_ship
CASE choice = "3"
STORE 'CONTAINS' TO rel_ship
CASE choice = "4"
RETURN
OTHERWISE
CLEAR
@ 2,14 SAY "PLEASE ENTER VALUES BETWEEN 1 AND 4 ONLY"
@ 3,14 SAY "PRESS RETURN TO TRY AGAIN!"
ACCEPT to hold
STORE .t. TO true
ENDCASE
ENDDO
CLEAR
@ 0.1 SAY "1.3.2.0.0.0"
@ 1,22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3,35 SAY "QUERY MENU"
@ 5,12 SAY entity1
```

204
@ 5.32 SAY rel_ship
@ 5.58 SAY "ENTITY-2"
@ 7.4 SAY "IS THIS THE RELATIONSHIP"
@ 8.4 SAY "THAT YOU WISH TO QUERY ON"
@ 8.31 SAY rel_ship
@ 9.4 SAY "Y OR N"
@ 9.12 GET correct
READ ENDDO
DO CASE
CASE choice = "1"
STORE selection + 10 TO selection
SAVE TO mem_var
do 132100
CASE choice = "2"
STORE selection + 20 TO selection
SAVE TO mem_var
do 132200
CASE choice = "3"
STORE selection + 30 TO selection
SAVE TO mem_var
do 132300
ENDCASE
RESTORE FROM mem_var
ERASE mem_var.mem
CLEAR
STORE 'N' TO correct
DO WHILE correct = 'N'
    STORE .t. TO true
    DO while true
    CLEAR
    @ 0.1 SAY "1.3.3.0.0.0"
    @ 1.22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
    @ 3.35 SAY "QUERY MENU"
    @ 5.11 SAY entity1
    @ 5.33 SAY "RELATIONSHIP ENTITY-2"
    @ 7.28 SAY "1) PROCESSES"
    @ 9.28 SAY "2) IS PROCESSED BY"
    @ 11.28 SAY "3) CONTAINS"
    @ 13.28 SAY "4) PRODUCES"
    @ 15.28 SAY "5) IS THE RESPONSIBILITY OF"
    @ 17.28 SAY "6) IS CONTAINED IN"
    @ 19.28 SAY "7) RETURN TO PREVIOUS MENU"
    @ 20.4 SAY "ACCEPT'
    ENTER YOUR CHOICE (1-7) FROM ABOVE: 'TO choice
    STORE .f. TO true
    DO CASE
    CASE choice = "1"
    STORE 'PROCESSES' TO rel_ship
    CASE choice = "2"
    STORE 'IS PROCESSED BY' TO rel_ship
    CASE choice = "3"
    STORE 'CONTAINS' TO rel_ship
    CASE choice = "4"
    STORE 'PRODUCES' TO rel_ship
    CASE choice = "5"
    STORE 'IS THE RESPONSIBILITY OF' TO rel_ship
    CASE choice = "6"
    STORE 'IS CONTAINED IN' TO rel_ship
    CASE choice = "7"
    RETURN
    OTHERWISE
    CLEAR
    @ 2.14 SAY "PLEASE ENTER VALUES BETWEEN 1 AND 7 ONLY"
    @ 3.14 SAY "PRESS RETURN TO TRY AGAIN!"
    ACCEPT TO hold
    STORE .t. TO true
    ENDCASE
CLEAR
@ 0.1 SAY "1.3.3.0.0.0"
@ 1.22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3.35 SAY "QUERY MENU"
@ 5.12 SAY entity1
@ 5.32 SAY rel ship
@ 5.58 SAY "ENTITY-2"
@ 7.4 SAY "IS THIS THE RELATIONSHIP"
@ 8.4 SAY "THAT YOU WISH TO QUERY ON"
@ 8.31 SAY rel ship
@ 9.4 SAY "Y OR N"
@ 9.12 GET correct
READ
ENDDO
DO CASE
CASE choice = "1"
STORE selection + 10 TO selection
SAVE TO mem_var
DO 133100
CASE choice = "2"
STORE selection + 20 TO selection
SAVE TO mem_var
DO 133200
CASE choice = "3"
STORE selection + 30 TO selection
SAVE TO mem_var
DO 133300
CASE choice = "4"
STORE selection + 40 TO selection
SAVE TO mem_var
DO 133400
CASE choice = "5"
STORE selection + 50 TO selection
SAVE TO mem_var
DO 133500
CASE choice = "6"
STORE selection + 60 TO selection
SAVE TO mem Var
DO 133600
ENDCASE
* 134000.PRG
* MODULE NAME: 1.3.4.0.0.0
* ROUTINES THAT CALL THE MODULE: 1.3.0.0.0.0
* ROUTINES THAT THE MODULE CALLS: MAIN
* LOCAL VARIABLES USED:
* choice : CONTAINS THE NUMBER OF ACTION SELECTED. MAY ALSO CONTAIN THE
* CHARACTER STRING THAT IDENTIFIES THE RELATION BEING ADDED TO,
* MODIFIED, DELETED FROM OR OUTPUT.
* correct : CONTAINS USER RESPONSE AS TO WHETHER THE DISPLAYED VALUE IS
* CORRECT OR NOT.
* entity1 : CONTAINS THE CHARACTER STRING THAT REPRESENTS THE FIRST VALUE
* IN A QUERY STRING.
* rel_ship: CONTAINS THE CHARACTER STRING THAT REPRESENTS THE RELATIONSHIP
* VALUE IN A QUERY STRING.
* true : USED AS A BOOLEAN VALUE IN LOOPS.
* INPUT FILES: MEM VAR.
* OUTPUT FILES: MEM VAR.
* THIS MODULE ALLOWS THE USER TO CHOOSE WHICH TYPE OF RELATIONSHIP WILL BE
* USED IN THE QUERY.

RESTORE FROM mem_var
ERASE mem_var.mem
CLEAR
STORE 'N' TO correct
DO WHILE correct = 'N'
STORE t. TO true
do while true
CLEAR
@ 0.1 SAY "1.3.4.0.0.0"
@ 1.22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3.35 SAY "QUERY MENU"
@ 5.11 SAY entity1
@ 5.33 SAY "RELATIONSHIP ENTITY-2"
@ 7.29 SAY "1) IS CONTAINED IN"
@ 9.29 SAY "2) IS PROCESSED BY"
@ 11.29 SAY "3) IS THE RESPONSIBILITY OF"
@ 13.29 SAY "4) RETURN TO PREVIOUS MENU"
@ 14.4 SAY ""
ACCEPT 'ENTER YOUR CHOICE (1-4) FROM ABOVE: ' TO choice
STORE .f. TO true
DO CASE
CASE choice = "1"
STORE 'IS CONTAINED IN' TO rel_ship
CASE choice = "2"
STORE 'IS PROCESSED BY' TO rel_ship
CASE choice = "3"
STORE 'IS THE RESPONSIBILITY OF' TO rel_ship
CASE choice = "4"
RETURN
OTHERWISE
CLEAR
@ 2.4 SAY "PLEASE ENTER VALUES BETWEEN 1 AND 4 ONLY"
@ 3.14 SAY "PRESS RETURN TO TRY AGAIN!"
ACCEPT TO hold
STORE .t. TO true
ENDCASE
ENDDO
CLEAR
@ 0.1 SAY "1.3.4.0.0.0"
@ 1.22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3.35 SAY "QUERY MENU"
@ 5.12 SAY entity1
@ 5.32 SAY rel_ship
@ 5.58 SAY "ENTITY-2"
@ 7.4 SAY "IS THIS THE RELATIONSHIP"
@ 8.4 SAY "THAT YOU WISH TO QUERY ON"
@ 8.31 SAY rel_ship
@ 9.4 SAY "Y OR N"

208
@ 9.12 GET correct
READ
ENDDO
DO CASE
CASE choice = "1"
STORE selection + 10 TO selection
SAVE TO mem_var
do 134100
CASE choice = "2"
STORE selection + 20 TO selection
SAVE TO mem_var
do 134200
CASE choice = "3"
STORE selection + 30 TO selection
SAVE TO mem_var
do 134300
ENDCASE
VARIABLES USED:
choice: CONTAINS THE NUMBER OF ACTION SELECTED. MAY ALSO CONTAIN THE
CHARACTER STRING THAT IDENTIFIES THE RELATION BEING ADDED TO,
MODIFIED, DELETED FROM OR OUTPUT.
correct: CONTAINS USER RESPONSE AS TO WHETHER THE DISPLAYED VALUE IS
CORRECT OR NOT.
etyl: CONTAINS THE CHARACTER STRING THAT REPRESENTS THE FIRST VALUE
IN A QUERY STRING.
rel_ship: CONTAINS THE CHARACTER STRING THAT REPRESENTS THE RELATIONSHIP
VALUE IN A QUERY STRING.
true: USED AS A BOOLEAN VALUE IN LOOPS.
INPUT FILES: MEM VAR.
OUTPUT FILES: MEM VAR.
THIS MODULE ALLOW THE USER TO CHOOSE WHICH TYPE OF RELATIONSHIP WILL BE
USED IN THE QUERY.

```
set color to 0/3,3
set talk off
SET EXACT ON
'RESTORE FROM mem_var
ERASE mem_var.mem
CLEAR
STORE 'N' TO correct
DO WHILE correct = 'N'
STORE .t. TO true
do while true
CLEAR
@ 0.1 SAY "1.3.5.0.0.0"
@ 1.22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3.35 SAY "QUERY MENU"
@ 5.11 SAY entityl
@ 5.33 SAY "RELATIONSHIP ENTITY-2"
@ 8.28 SAY "1) IS PRODUCED BY"
@ 10.28 SAY "2) RETURN TO PREVIOUS MENU"
@ 11.4 SAY ""
ACCEPT ENTER YOUR CHOICE (1-2) FROM ABOVE: 'TO choice
STORE .f. TO true
DO CASE
CASE choice = "1"
STORE 'IS PRODUCED BY' TO rel_ship
CASE choice = "2"
RETURN
OTHERWISE
CLEAR
@ 2.14 SAY "PLEASE ENTER VALUES BETWEEN 1 AND 2 ONLY"
@ 3.14 SAY "PRESS RETURN TO TRY AGAIN!"
ACCEPT TO hold
STORE .t. TO true
ENDCASE
ENDDO
CLEAR
@ 0.1 SAY "1.3.5.0.0.0"
@ 1.22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3.35 SAY "QUERY MENU"
@ 5.12 SAY entityl
@ 5.32 SAY rel_ship
@ 5.58 SAY "ENTITY-2"
@ 7.4 SAY "IS THIS THE RELATIONSHIP"
@ 8.4 SAY "THAT YOU WISH TO QUERY ON"
@ 8.31 SAY rel_ship
@ 9.4 SAY "Y OR N"
@ 9.12 GET correct
READ
ENDDO
DO CASE
```

210
CASE choice = "1"
STORE selection + 10 TO selection
SAVE TO mem_var
do 135100
ENDCASE
* 136000.PRG
* MODULE NAME: 1.3.6.0.0.0
* ROUTINES THAT CALL THE MODULE: 1.3.0.0.0.0
* ROUTINES THAT THE MODULE CALLS: MAIN
* LOCAL VARIABLES USED:
  * choice : CONTAINS THE NUMBER OF ACTION SELECTED. MAY ALSO CONTAIN THE
    * CHARACTER STRING THAT IDENTIFIES THE RELATION BEING ADDED TO,
    * MODIFIED, DELETED FROM OR OUTPUT.
  * correct : CONTAINS USER RESPONSE AS TO WHETHER THE DISPLAYED VALUE IS
    * CORRECT OR NOT.
  * entity1 : CONTAINS THE CHARACTER STRING THAT REPRESENTS THE FIRST VALUE
    * IN A QUERY STRING.
  * rel_ship: CONTAINS THE CHARACTER STRING THAT REPRESENTS THE RELATIONSHIP
    * VALUE IN A QUERY STRING.
  * true : USED AS A BOOLEAN VALUE IN LOOPS.
* INPUT FILES: MEM VAR.
* OUTPUT FILES: MEM VAR.
* THIS MODULE ALLOWS THE USER TO CHOOSE WHICH TYPE OF RELATIONSHIP WILL BE
  * USED IN THE QUERY.
* RESTORE FROM mem var
ERASE mem_var_mem
CLEAR
STORE 'N' TO correct
DO WHILE correct = 'N'
STORE .t. TO true
DO while true
CLEAR
@ 0.1 SAY "1.3.6.0.0.0"
@ 1.22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3.35 SAY "QUERY MENU"
@ 5.11 SAY entity1
@ 5.33 SAY "RELATIONSHIP ENTITY-2"
@ 7.28 SAY "1) CONTAINS"
@ 9.28 SAY "2) IS PROCESSED BY"
@ 11.28 SAY "3) IS THE RESPONSIBILITY OF"
@ 13.28 SAY "4) RETURN TO PREVIOUS MENU"
@ 14.4 SAY ""
ACCEPT 'Enter your choice (1-4) from above': 'TO choice
STORE .f. TO true
DO CASE
CASE choice = "1"
STORE 'CONTAINS' TO rel_ship
CASE choice = "2"
STORE 'IS PROCESSED BY' TO rel_ship
CASE choice = "3"
STORE 'IS THE RESPONSIBILITY OF' TO rel_ship
CASE choice = "4"
RETURN
OTHERWISE
CLEAR
@ 2.14 SAY "PLEASE ENTER VALUES BETWEEN 1 AND 4 ONLY"
@ 3.14 SAY "PRESS RETURN TO TRY AGAIN!"
ACCEPT TO hold
STORE .t. TO true
ENDCASE
ENDDO
CLEAR
@ 0.1 SAY "1.3.6.0.0.0"
@ 1.22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3.35 SAY "QUERY MENU"
@ 5.12 SAY entity1
@ 5.32 SAY rel_ship
@ 5.58 SAY "ENTITY-2"
@ 7.4 SAY "IS THIS THE RELATIONSHIP"
@ 8.4 SAY "THAT YOU WISH TO QUERY ON"
@ 8.31 SAY rel_ship
@ 9.4 SAY "Y OR N"

212
@ 9,12 GET correct
READ
ENDDO
DO CASE
CASE choice = "1"
STORE selection + 10 TO selection
SAVE TO mem_var
do 136100
CASE choice = "2"
STORE selection + 20 TO selection
SAVE TO mem_var
do 136200
CASE choice = "3"
STORE selection + 30 TO selection
SAVE TO mem_var
do 136300
ENDCASE
* 137000.PRG
* MODULE NAME: 1.3.7.0.0.0
* ROUTINES THAT CALL THE MODULE: 1.3.0.0.0.0
* ROUTINES THAT THE MODULE CALLS: MAIN
* LOCAL VARIABLES USED:
  * choice : CONTAINS THE NUMBER OF ACTION SELECTED. MAY ALSO CONTAIN THE
  * correct : CONTAINS USER RESPONSE AS TO WHETHER THE DISPLAYED 'VALUE IS
  * entity1 : CONTAINS THE CHARACTER STRING THAT REPRESENTS THE FIRST VALUE
  * rel_ship: CONTAINS THE CHARACTER STRING THAT REPRESENTS THE RELATIONSHIP
  * true : USED AS A BOOLEAN VALUE IN LOOPS.
* INPUT FILES: MEN VAR.
* OUTPUT FILES: MEN VAR.
* THIS MODULE ALLOW THE USER TO CHOOSE WHICH TYPE OF RELATIONSHIP WILL BE
  * USED IN THE QUERY.
* RESTORE FROM mem_var
  ERASE mem_var.mem
CLEAR
STORE 'N' TO correct
DO WHILE correct = 'N'
STORE .t. TO true
do while true
CLEAR
@ 0.1 SAY "1.3.7.0.0.0"
@ 1.22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3.35 SAY "QUERY MENU"
@ 5.11 SAY entity1
@ 5.33 SAY "RELATIONSHIP ENTITY-2"
@ 7.27 SAY "1) CONTAINS"
@ 9.27 SAY "2) IS CONTAINED IN"
@ 11.27 SAY "3) IS PROCESSED BY"
@ 13.27 SAY "4) IS THE RESPONSIBILITY OF"
@ 15.27 SAY "5) RETURN TO PREVIOUS MENU"
@ 16.1 SAY "ENTER YOUR CHOICE (1-5) FROM ABOVE:'TO choice
STORE .f. TO true
DO CASE
CASE choice = "1"
STORE 'CONTAINS' TO rel_ship
CASE choice = "2"
STORE 'IS CONTAINED IN' TO rel_ship
CASE choice = "3"
STORE 'IS PROCESSED BY' TO rel_ship
CASE choice = "4"
STORE 'IS THE RESPONSIBILITY OF' TO rel_ship
CASE choice = "5"
RETURN
OTHERWISE
CLEAR
@ 2.14 SAY "PLEASE ENTER VALUES BETWEEN 1 AND 5 ONLY"
@ 3.14 SAY "PRESS RETURN TO TRY AGAIN!'" ACCEPT TO hold
STORE .t. TO true
ENDCASE
ENDDO
CLEAR
@ 0.1 SAY "1.3.7.0.0.0"
@ 1.22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3.35 SAY "QUERY MENU"
@ 5.12 SAY entity1
@ 5.32 SAY rel_ship
@ 5.58 SAY "ENTITY-2"
@ 7.4 SAY "IS THIS THE RELATIONSHIP"
@ 8.4 SAY "THAT YOU WISH TO QUERY ON"
@ 8.31 SAY rel_ship
@ 9.4 SAY "Y OR N"
@ 9.12 GET correct
READ
ENDDO
DO CASE
CASE choice = "1"
STORE selection + 10 TO selection
SAVE TO mem_var
do 137100
CASE choice = "2"
STORE selection + 20 TO selection
SAVE TO mem_var
do 137200
CASE choice = "3"
STORE selection + 30 TO selection
SAVE TO mem_var
do 137300
CASE choice = "4"
STORE selection + 40 TO selection
SAVE TO mem_var
do 137400
ENDCASE

*
* 138000.PRG
* MODULE NAME: 1.3.8.0.0.0
* ROUTINES THAT CALL THE MODLUE: 1.3.0.0.0.0
* ROUTINES THAT THE MODULE CALLS: MAIN
* LOCAL VARIABLES USED:
* choice : CONTAINS THE NUMBER OF ACTION SELECTED. MAY ALSO CONTAIN THE
* CHARACTER STRING THAT IDENTIFIES THE RELATION BEING ADDED TO,
* MODIFIED, DELETED FROM OR OUTPUT.
* correct : CONTAINS USER RESPONSE AS TO WHETHER THE DISPLAYED VALUE IS
* CORRECT OR NOT.
* IN A QUERY STRING.
* IN A QUERY STRING.
* rel_ship: CONTAINS THE CHARACTER STRING THAT REPRESENTS THE RELATIONSHIP
* VALUE IN A QUERY STRING.
* true : USED AS A BOOLEAN VALUE IN LOOPS.
* INPUT FILES: MEM VAR.
* OUTPUT FILES: MEM VAR.
* THIS MODULE ALLOW THE USER TO CHOOSE WHICH TYPE OF RELATIONSHIP WILL BE
* USED IN THE QUERY.
* RESTORE FROM mem_var
* ERASE mem_var.mem
* CLEAR
* STORE 'N' TO correct
* DO WHILE correct = 'N'
* STORE .t. TO true
* do while true
* CLEAR
* @ 0.1 SAY "1.3.8.0.0.0"
* @ 1.22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
* @ 3.35 SAY "QUERY MENU"
* @ 5.11 SAY entity1
* @ 5.33 SAY "RELATIONSHIP ENTITY-2"
* @ 7.27 SAY "1) IS CONTAINED IN"
* @ 9.27 SAY "2) IS PROCESSED BY"
* @ 11.27 SAY "3) IS THE RESPONSIBILITY OF"
* @ 13.27 SAY "4) RETURN TO THE PREVIOUS MENU"
* @ 14.4 SAY ""
* ACCEPT ENTER YOUR CHOICE (1-4) FROM ABOVE: 'TO choice
* STORE .f. TO true
* DO CASE
* CASE choice = "1"
* STORE 'IS CONTAINED IN' TO rel_ship
* CASE choice = "2"
* STORE 'IS PROCESSED BY' TO rel_ship
* CASE choice = "3"
* STORE 'IS THE RESPONSIBILITY OF' TO rel_ship
* CASE choice = "4"
* RETURN
* OTHERWISE
* CLEAR
* @ 2.14 SAY "PLEASE ENTER VALUES BETWEEN 1 AND 4 ONLY"
* @ 3.14 SAY "PRESS RETURN TO TRY AGAIN!"
* ACCEPT TO hold
* STORE .t. TO true
* END CASE
* ENDDO
* CLEAR
* @ 0.1 SAY "1.3.8.0.0.0"
* @ 1.22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
* @ 3.35 SAY "QUERY MENU"
* @ 5.12 SAY entity1
* @ 5.32 SAY rel_ship
* @ 5.38 SAY "ENTITY-2"
* @ 7.4 SAY "IS THIS THE RELATIONSHIP"
* @ 8.4 SAY "THAT YOU WISH TO QUERY ON"

216
@ 8.31 SAY rel ship
@ 9.4 SAY "Y OR N"
@ 9.12 GET correct
READ
ENDDO
DO CASE
CASE choice = "1"
STORE selection + 10 TO selection
SAVE TO mem_var
do 138100
CASE choice = "2"
STORE selection + 20 TO selection
SAVE TO mem_var
do 138200
CASE choice = "3"
STORE selection + 30 TO selection
SAVE TO mem_var
do 138300
ENDCASE
* 139000.PRG
* MODULE NAME: 1.3.9.0.0.0
* ROUTINES THAT CALL THE MODULE: 1.3.1.0.0 THRU 1.3.8.3.0.0
* ROUTINES THAT THE MODULE CALLS: 1.3.1.1.0.0 THRU 1.3.8.3.0.0
* LOCAL VARIABLES USED:
  * choice : CONTAINS THE NUMBER OF ACTION SELECTED. MAY ALSO CONTAIN THE
            CHARACTER STRING THAT IDENTIFIES THE RELATION BEING ADDED TO,
            MODIFIED, DELETED FROM OR OUTPUT.
  * true : USED AS A BOOLEAN VALUE IN LOOPS.
  * option : USED TO HOLD THE VALUE REPRESENTING THE CHOICE OF PRINTER OR
            SCREEN OUTPUT.
* INPUT FILES: MEM VAR.
* OUTPUT FILES: MEM VAR.
* THIS MODULE ALLOW THE USER TO CHOOSE WHICH TYPE OF RELATIONSHIP WILL BE
  USED IN THE QUERY.
* DESIGNED BY: ROBERT A. KIRSCH II
* WRITTEN BY: ROBERT A. KIRSCH II
* BASIC FUNCTION OF MODULE:
  * THIS MODULE ALLOW THE USER TO CHOOSE WHETHER THE OUTPUT WILL BE
  DISPLAYED ON THE SCREEN OR PRINTED.
* RESTORE FROM mem_var
STORE 0 TO rec_num, stop
CLEAR
STORE t. TO TRUE
do while TRUE
  CLEAR
  @ 0,1 SAY "1.3.9.0.0.0"
  RESTORE FROM mem_var
  @ 2,22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
  @ 4,34 SAY "QUERY OUTPUT"
  @ 8,23 SAY "LISTED BELOW ARE THE CHOICES FOR HOW"
  @ 9,23 SAY "YOU CAN HAVE THE QUERY"
  @ 11,24 SAY entity1
  @ 11,38 SAY rel_ship
  @ 11,57 SAY entity2
  @ 13,23 SAY "DISPLAYED."
  @ 15,28 SAY "1) SCREEN OUTPUT"
  @ 17,28 SAY "2) PRINTER OUTPUT"
  @ 19,28 SAY "3) RETURN TO PREVIOUS MENU"
  @ 20,1 SAY ""
  ACCEPT ENTER YOUR CHOICE (1-3) FROM ABOVE 'TO option
ERASE mem_var.mem
SAVE TO mem_var
DO CASE
  CASE option = '1'
    DO 139100
  CASE option = '2'
    DO 139200
  CASE option = '3'
    RETURN
  OTHERWISE
    CLEAR
    @ 0,27 SAY option
    @ 0,34 SAY "IS NOT A VALID CHOICE"
    @ 1,26 SAY "PLEASE ENTER VALUES BETWEEN 1 AND 3 ONLY"
    @ 2,26 SAY "PRESS RETURN AND TRY AGAIN!"
    ACCEPT TO hold
ENDCASE
ENDDO
* MODULE NAME: 1.3.9.1.0.0
* ROUTINES THAT CALL THE MODULE: 1.3.9.0.0.0
* ROUTINES THAT THE MODULE CALLS: 1.3.9.0.0.0
* LOCAL VARIABLES USED:
  * choice : CONTAINS THE NUMBER OF ACTION SELECTED. MAY ALSO CONTAIN THE
            CHARACTER STRING THAT IDENTIFIES THE RELATION BEING ADDED TO,
            MODIFIED, DELETED FROM OR OUTPUT.
  * hold : USED TO STOP ACTION FOR USER DECISION.
  * option : CONTAINS THE USER'S CHOICE ON WHETHER TO OUTPUT TO THE SCREEN
            OR THE PRINTER.
  * stop : USED TO STOP ACTION FOR USER DECISION.
  * t : REPRESENTS THE BOOLEAN TRUE IS USED TO CREATE A CONTINES
       LOOP.
  * entity1 : CONTAINS THE CHARACTER STRING THAT REPRESENTS THE FIRST VALUE
             IN A QUERY STRING.
  * entity2 : CONTAINS THE CHARACTER STRING THAT REPRESENTS THE SECOND VALUE
             IN A QUERY STRING.
  * rel_ship: CONTAINS THE CHARACTER STRING THAT REPRESENTS THE RELATIONSHIP
             VALUE IN A QUERY STRING.
* INPUT FILES: MEM VAR.
* OUTPUT FILES: MEM VAR.
* THIS MODULE WILL DISPLAY THE RESULTS OF THE QUERY ON THE SCREEN.
* DESIGNED BY: ROBERT A. KIRSCH II
* WRITTEN BY: ROBERT A. KIRSCH II
* BASIC FUNCTION OF MODULE:
* THIS MODULE WILL DISPLAY THE RESULTS OF THE QUERY
* ON THE SCREEN.
* PROGRAM AND MODULE RELATIONS
* RESTORE FROM mem var
CLEAR
@ 0,1 SAY "1.3.9.1.0.0"
@ 1,22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3,30 SAY "QUERY SCREEN OUTPUT"
@ 5,22 SAY "THIS MODULE WILL DISPLAY THE RESULTS OF"
@ 7,21 SAY entity1
@ 7,38 SAY rel_ship
@ 7,59 SAY entity2
@ 9,22 SAY "IF YOU DO NOT WISH TO DISPLAY THIS RELATION,"
@ 10,22 SAY "TYPE '0' TO RETURN TO THE PREVIOUS MENU."
WAIT TO stop
DO CASE
CASE stop = '0'
RETURN
OTHERWISE
ENDCASE
CLEAR
USE TEMP
STORE 1 TO count
SET HEADING OFF
DO WHILE .NOT. EOF()
CLEAR
@ 0,1 SAY "1.3.9.1.0.0"
@ 1,22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3,32 SAY "QUERY RESULTS FOR"
@ 5,21 SAY entity1
@ 5,38 SAY rel_ship
@ 5,59 SAY entity2
USE TEMP
STORE 1 TO count
SET HEADING OFF
DO WHILE .NOT. EOF()
STORE count + 1 to count
@ 7,9 SAY count
@ 9,1 SAY ""
store count + 1 to count
@ 10,4 SAY "IDENTIFICATION NAME:"
@ 10,31 SAY ID NAME
@ 12,4 SAY "DESCRIPTION:"
@ 12,21 SAY descript
@ 17,4 SAY " "
WAIT TO hold
SKIP
ENDDO
RETURN
* MODULE NAME: 1.3.9.2.0.0
* INPUT FILES: NONE
* OUTPUT FILES: NONE
* ROUTINES THAT CALL THE MODULE: 1.3.9.0.0.0
* ROUTINES THAT THE MODULE CALLS: 1.3.9.0.0.0
* LOCAL VARIABLES USED:
  * choice : CONTAINS THE NUMBER OF ACTION SELECTED. MAY ALSO CONTAIN THE CHARACTER STRING THAT IDENTIFIES THE RELATION BEING ADDED TO, MODIFIED, DELETED FROM OR OUTPUT.
  * correct : CONTAINS USER RESPONSE AS TO WHETHER THE DISPLAYED VALUE IS CORRECT OR NOT.
  * entity1 : CONTAINS THE CHARACTER STRING THAT REPRESENTS THE FIRST VALUE IN A QUERY STRING.
  * entity2 : CONTAINS THE CHARACTER STRING THAT REPRESENTS THE SECOND VALUE IN A QUERY STRING.
  * rel_ship: CONTAINS THE CHARACTER STRING THAT REPRESENTS THE RELATIONSHIP VALUE IN A QUERY STRING.
  * true : USED AS A BOOLEAN VALUE IN LOOPS.
* INPUT FILES: HEM VAR.
* OUTPUT FILES: MER VAR.
* THIS MODULE ALLOW THE USER TO CHOOSE WHICH TYPE OF RELATIONSHIP WILL BE USED IN THE QUERY.
* DESIGNED BY: ROBERT A. KIRSCH II
* WRITTEN BY: ROBERT A. KIRSCH II
* BASIC FUNCTION OF MODULE:
* THIS MODULE WILL OUTPUT THE QUERY TO THE PRINTER.

SET EXACT ON
set color to 0/3,3
set talk off
set menu on
SET EXACT ON
RESTORE FROM mem_var
STORE 0 TO rec_num, stop
CLEAR
@ 0,1 SAY "1.3.9.2.0.0"
@ 1,22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3,29 SAY "QUERY PRINTER OUTPUT"
@ 6,23 SAY "THIS MODULE WILL PRINT QUERY"
@ 8,20 SAY entity1
@ 8,37 SAY rel_ship
@ 8,56 SAY name
@ 10,23 SAY "PLEASE INSURE THAT YOUR PRINTER"
@ 11,23 SAY "IS TURNED ON AND IN THE ONLINE"
@ 12,23 SAY "MODE"
@ 14,23 SAY "IF YOU DO NOT WISH TO PRINT"
@ 15,23 SAY "THIS RELATION, TYPE '0' TO"
@ 16,23 SAY "RETURN TO THE PREVIOUS MENU"
WAIT TO stop
DO CASE
CASE stop = '0'
RETURN
OTHERWISE
ENDCASE
SET DEVICE TO PRINT
SET CONSOLE OFF
USE TEMP
STORE 1 TO count
DO WHILE ..NOT. EOF()
  @ 29.1 SAY "1.3.9.2.0.0"
  @ 30.22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
  @ 32.32 SAY "RESULTS FOR QUERY"
  @ 34.20 SAY entity1
  @ 34.37 SAY rel_ship
  @ 34.56 SAY name
  @ 40.1 SAY "RECORD #"
  @ 40,11 SAY count

221
store count + 1 to count
@ 42,3 SAY "IDENTIFICATION NAME:"
@ 42,30 SAY ID NAME
@ 44,3 SAY "DESCRIPTION:"
@ 44,19 SAY descript
SKIP
ENDDO
SET DEVICE TO SCREEN
SET CONSOLE ON
RETURN
* 140000.PRG
* MODULE NAME: 1.4.0.0.0.0
* INPUT FILES: NONE
* OUTPUT FILES: NONE
* ROUTINES THAT CALL THE MODULE: MAIN
* ROUTINES THAT THE MODULE CALLS: 1.1.1.0.0.0, 1.1.2.0.0.0, 1.1.3.0.0.0, 1.1.4.0.0.0, 1.1.5.0.0.0, MAIN.
* LOCAL VARIABLES USED: choice: CONTAINS THE NUMBER OF ACTION SELECTED.
* t: REPRESENTS NO VALUE AT ALL.
* hold: USED TO STOP ACTION FOR USER DECISION.
* DESIGNED BY: ROBERT A. KIRSCH II
* WRITTEN BY: ROBERT A. KIRSCH II
* BASIC FUNCTION OF MODULE:
* THIS PROGRAM ALLOWS FOR THE MAINTENANCE OF ENTITY SCHEMA,
* AND RELATIONSHIP SCHEMA.
*
* do while .t.
CLEAR
@ 0,1 SAY "1.4.0.0.0.0"
@ 1,22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3,31 SAY "MAINTENANCE MENU"
@ 6,22 SAY "1) MODIFY ENTITY SCHEMA"
@ 8,22 SAY "2) MODIFY,RELATIONSHIP SCHEMA"
@ 10,22 SAY "3) RETURN TO MAIN MENU"
@ 11,1 SAY "" ACCEPT ' ENTER YOUR CHOICE (1-3) FROM ABOVE: ' TO choice
DO CASE
CASE choice = "1"
do 141000
CASE choice = "2"
DO 142000
CASE choice = "3"
RETURN TO MASTER
OTHERWISE
CLEAR
@ 2,20 SAY "PLEASE ENTER VALUES BETWEEN 1 AND 3 ONLY"
@ 3,20 SAY "PRESS RETURN TO TRY AGAIN!"
ACCEPT TO hold
ENDCASE
ENDDO
RETURN

223
* 141000.PRG
* MODULE NAME: 1.4.1.0.0.0
* INPUT FILES: NONE
* OUTPUT FILES: NONE
* ROUTINES THAT CALL THE MODLUE: 1.4.0.0.0.0
* ROUTINES THAT THE MODULE CALLS: MAIN
* LOCAL VARIABLES USED: choice: CONTAINS THE NUMBER OF ACTION SELECTED.
  * t: REPRESTENTS NO VALUE AT ALL.
  * hold: USED TO STOP ACTION FOR USER DECISION.
* INPUT FILES: MEM_VAR, USER, SYSTEM, PROGRAM, MODULE, DOCUMENT, FILE, RECORD,
  * ELEMENT.
* OUTPUT FILES: MEM_VAR, USER, SYSTEM, PROGRAM, MODULE, DOCUMENT, FILE, RECORD,
  * ELEMENT
* DESIGNED BY: ROBERT A. KIRSCH II
* WRITTEN BY: ROBERT A. KIRSCH II
* BASIC FUNCTION OF MODULE:
  * THIS MODULE ALLOW THE USER TO CHOOSE WHICH TYPE OF ENTITY RELATION
  * TO MODIFY.
  *
  do while .t.
  CLEAR
  @ 0,1 SAY "1.4.1.0.0.0"
  @ 1,22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
  @ 3,31 SAY "MODIFY ENTITY SCHEMA"
  @ 6,15 SAY "1) USER 6) FILE"
  @ 8,15 SAY "2) SYSTEM 7) RECORD"
  @ 10,15 SAY "3) PROGRAM 8) ELEMENT"
  @ 12,15 SAY "4) MODULE 9) RETURN TO PREVIOUS MENU"
  @ 14,15 SAY "5) DOCUMENT 10) RETURN TO MAIN MENU"
  @ 16,1 SAY ""
  ACCEPT T' ENTER YOUR CHOICE (1-10) FROM ABOVE: 'TO choice
  DO CASE
  CASE choice = "1"
  USE USER
  MODIFY STRUCTURE
  CASE choice = "2"
  USE SYSTEM
  MODIFY STRUCTURE
  CASE choice = "3"
  USE PROGRAM
  MODIFY STRUCTURE
  CASE choice = "4"
  USE MODULE
  MODIFY STRUCTURE
  CASE choice = "5"
  USE DOCUMENT
MODIFY STRUCTURE
CASE choice = "6"
USE FILE
MODIFY STRUCTURE
CASE choice = "7"
USE RECORD
MODIFY STRUCTURE
CASE choice = "8"
USE ELEMENT
MODIFY STRUCTURE
CASE choice = "9"
RETURN
CASE choice = "10"
RETURN TO MASTER
OTHERWISE
CLEAR
@ 2,20 SAY "PLEASE ENTER VALUES BETWEEN 1 AND 10 ONLY"
@ 3,20 SAY "PRESS RETURN TO TRY AGAIN!"
ACCEPT TO hold
ENDCASE
ENDDO
RETURN

225
* 142000.PRG
* MODULE NAME: 1.4.2.0.0.0
* ROUTINES THAT CALL THE MODULE: 1.4.0.0.0.0
* ROUTINES THAT THE MODULE CALLS: 1.4.0.0.0, MAIN
* LOCAL VARIABLES USED: choice: CONTAINS THE NUMBER OF ACTION SELECTED.
  * t: REPRESENTS NO VALUE AT ALL.
  * hold: USED TO STOP ACTION FOR USER DECISION.
* INPUT FILES: MEM_VAR, U_CONTS, U_CONT_S,
  * U_CONT_P, P_PROC_F, P_PROC_R,
  * P_PROC_R, P_PROC_E, S_CONT_P, P_CONT_M, F_CONT_R, R_CONT_E,
  * U_RESP_S, U_RESP_F, P_PRED_D.
* OUTPUT FILES: MEM_VAR, ELEMENT, U_CONTS, U_CONT_S,
  * U_CONT_P, P_PROC_F, P_PROC_R,
  * P_PROC_R, P_PROC_E, S_CONT_P, P_CONT_M, F_CONT_R, R_CONT_E,
  * U_RESP_S, U_RESP_F, P_PRED_D.

* DESIGNED BY: ROBERT A. KIRSCH II
* WRITTEN BY: ROBERT A. KIRSCH II
* BASIC FUNCTION OF MODULE:
* THIS MODULE ALLOW THE USER TO CHOOSE WHICH RELATIONSHIP
* SCHEMA HE WOULD LIKE TO MODIFY.
*
do while .t.
  CLEAR
  SET MENU ON
  1,1 SAY "1.4.2.0.0.0"
  @  2,22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
  @  4,25 SAY "RELATIONSHIP SCHEMA MAINTENANCE"
  @  6,9 SAY "1) USER CONTAINS SYSTEM  8) FILE CONTAINS REC"
  @  6,64 SAY "ORDS"
  @  8,9 SAY "2) SYSTEM CONTAINS PROGRAM  9) RECORD CONTAINS E"
  @  8,64 SAY "ELEMENT"
  @ 10,9 SAY "3) PROGRAM PROCESSES FILE 10) USER RESPONSIBLE"
  @ 10,64 SAY "FOR SYSTEM"
  @ 12,9 SAY "4) PROGRAM PROCESSES RECORD 11) USER RESPONSIBLE"
  @ 12,64 SAY "FOR FILE"
  @ 14,9 SAY "5) PROGRAM PROCESSES ELEMENT 12) PROGRAM PRODUCES"
  @ 14,64 SAY "DOCUMENT"
  @ 16,9 SAY "6) SYSTEM CONTAINS PROGRAM 13) RETURN TO PREVIOUS"
  @ 16,64 SAY "MENU"
  @ 18,9 SAY "7) PROGRAM CONTAINS MODULE 14) RETURN TO MAIN ME"
  @ 18,64 SAY "NU"
  @ 19,1 SAY ""
  ACCEPT ' ENTER YOUR CHOICE (1-10) FROM ABOVE: ' TO choice
  DO CASE
  CASE choice = "1"
USE U_PROC_S
MODIFY STRUCTURE
CASE choice = "2"
USE S_PROC_P
MODIFY STRUCTURE
CASE choice = "3"
USE P_PROC_F
MODIFY STRUCTURE
CASE choice = "4"
USE P_PROC_R
MODIFY STRUCTURE
CASE choice = "5"
USE P_PROC_E
MODIFY STRUCTURE
CASE choice = "6"
USE S_CONT_P
MODIFY STRUCTURE
CASE choice = "7"
USE P_CONT_M
MODIFY STRUCTURE
CASE choice = "8"
USE F_CONT_R
MODIFY STRUCTURE
CASE choice = "9"
USE R_CONT_E
MODIFY STRUCTURE
CASE choice = "10"
USE U_RESP_S
MODIFY STRUCTURE
CASE choice = "11"
USE U_RESP_F
MODIFY STRUCTURE
CASE choice = "12"
USE P_PROD_D
MODIFY STRUCTURE
CASE choice = "13"
RETURN
CASE choice = "14"
RETURN TO MASTER
OTHERWISE
CLEAR
@ 1,21 SAY choice
@ 1,28 SAY "IS NOT A VALID CHOICE"
@ 2,20 SAY "PLEASE ENTER VALUES BETWEEN 1 AND 14 ONLY"
@ 3,20 SAY "PRESS RETURN TO TRY AGAIN!"
ACCEPT TO hold
ENDCASE
ENDDO
RETURN
* 150000.prg
* MODULE NAME: 1.5.0.0.0.0
* INPUT FILES: NONE
* OUTPUT FILES: NONE
* ROUTINES THAT CALL THE MODULE: MAIN
* ROUTINES THAT THE MODULE CALLS: MAIN.
* LOCAL VARIABLES USED: choice: CONTAINS THE NUMBER OF ACTION SELECTED.
  * t: REPRESENTS NO VALUE AT ALL.
  * hold: USED TO STOP ACTION FOR USER DECISION.
* DESIGNED BY: ROBERT A. KIRSCH II
* WRITTEN BY: ROBERT A. KIRSCH II
* BASIC FUNCTION OF MODULE:
  * THIS PROGRAM ALLOWS FOR THE THE SELECTION OF WHICH TYPE OF
  * SCHEMA WILL BE OUTPUT.
* do while .t.
CLEAR
@ 0,1 SAY "1.5.0.0.0.0"
@ 1,22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3,34 SAY "SCHEMA OUTPUT"
@ 6,22 SAY "1) ENTITY"
@ 8,22 SAY "2) RELATIONSHIP"
@ 10,22 SAY "3) RETURN TO MAIN MENU"
@ 11,22 SAY ""
ACCEPT ' ENTER YOUR CHOICE (1-3) FROM ABOVE: ' TO choice
DO CASE
CASE choice = "1"
do 151000
CASE choice = "2"
do 152000
CASE choice = "3"
RETURN TO MASTER
OTHERWISE
CLEAR
@ 2,20 SAY "PLEASE ENTER VALUES BETWEEN 1 AND 3 ONLY"
@ 3,20 SAY "PRESS RETURN TO TRY AGAIN!"
ACCEPT TO hold
ENDCASE
ENDDO
RETURN

229
* 121000.PRG
* MODULE NAME: 1.5.1.0.0.0
* ROUTINES THAT CALL THE MODULE: 1.5.0.0.0.0
* ROUTINES THAT THE MODULE CALLS: 1.5.0.0.0.0, 1.5.1.1.0.0 MAIN
* LOCAL VARIABLES USED: choice: CONTAINS THE NUMBER OF ACTION SELECTED.
  * t: REPRESENTS THE NUMBER OF VALUE AT ALL.
  * hold: USED TO STOP ACTION FOR USER DECISION.
* INPUT FILES: MEM_VAR.
* OUTPUT FILES: MEM_VAR.
* THIS MODULE ALLOW THE USER TO CHOOSE WHICH TYPE OF RELATIONSHIP WILL BE
  * USED IN THE QUERY.
* DESIGNED BY: ROBERT A. KIRSCH II
* WRITTEN BY: ROBERT A. KIRSCH II
* BASIC FUNCTION OF MODULE:
  * THIS MODULE ALLOW THE USER TO CHOOSE WHICH TYPE OF ENTITY RELATION
  * TO OUTPUT.
  *
SET EXACT ON
set color to 0/3,3
set talk off
CLEAR
do while .t.
ERASE mem_var.mem
CLEAR
@ 0,1 SAY "1.5.1.0.0.0"
@ 1,22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3,29 SAY "ENTITY SCHEMA OUTPUT"
@ 6,15 SAY "1) USER 6) FILE"
@ 8,15 SAY "2) SYSTEM 7) RECORD"
@ 10,15 SAY "3) PROGRAM 8) ELEMENT"
@ 12,15 SAY "4) MODULE 9) RETURN TO PREVIOUS MENU"
@ 14,15 SAY "5) DOCUMENT 10) RETURN TO MAIN MENU"
@ 15,1 SAY " "
ACCEPT' ENTER YOUR CHOICE (1-10) FROM ABOVE: 'TO choice
DO CASE
CASE choice = "1"
store 'USER' to choice
save to mem_var
do 151100
CASE choice = "2"
store 'SYSTEM' to choice
save to mem_var
DO 151100
CASE choice = "3"
store 'PROGRAM' to choice
save to mem_var

DO 151100
CASE choice = "4"
store 'MODULE' to choice
save to mem_var
DO 151100
CASE choice = "5"
store 'DOCUMENT' to choice
save to mem_var
DO 151100
CASE choice = "6"
store 'FILE' to choice
save to mem_var
DO 151100
CASE choice = "7"
store 'RECORD' to choice
save to mem_var
DO 151100
CASE choice = "8"
store 'ELEMENT' to choice
save to mem_var
DO 151100
CASE choice = "9"
RETURN
CASE choice = "10"
RETURN TO MASTER
OTHERWISE
CLEAR
@ 1,23 SAY choice
@ 1,31 SAY "IS NOT A VALID CHOICE"
@ 2,18 SAY "PLEASE ENTER VALUES BETWEEN 1 AND 10 ONLY"
@ 3,18 SAY "PRESS RETURN TO TRY AGAIN!"
ACCEPT TO hold
ENDCASE
ENDDO
RETURN
* 151100.PRG
* MODULE NAME: 1.5.1.1.0.0
* INPUT FILES: NONE
* OUTPUT FILES: NONE
* ROUTINES THAT CALL THE MODULE: 1.5.1.0.0.0
* ROUTINES THAT THE MODULE CALLS: 1.5.1.0.0.0
* LOCAL VARIABLES USED: choice: CONTAINS THE NUMBER OF ACTION SELECTED.
*               t: REPRESENTS NO VALUE AT ALL.
*               hold: USED TO STOP ACTION FOR USER DECISION.
*            count: KEEPS TRACK OF ACCOUNT NUMBERS.
*            option: USED TO SELECT PRINTER OR SCREEN.
* INPUT FILES: MEM VAR.
* OUTPUT FILES: MEM VAR.
* THIS MODULE ALLOW THE USER TO CHOOSE WHICH TYPE OF RELATIONSHIP WILL BE
* USED IN THE QUERY.
* DESIGNED BY: ROBERT A. KIRSCH II
* WRITTEN BY: ROBERT A. KIRSCH II
* BASIC FUNCTION OF MODULE:
* THIS MODULE ALLOW THE USER TO CHOOSE WHETHER THE OUTPUT WILL BE
* DISPLAYED ON THE SCREEN OR PRINTED.
*    RESTORE FROM mem_var
*    STORE 0 TO rec_num, stop
*    CLEAR
*    STORE .t. TO TRUE
*    do while TRUE
*    CLEAR
*    @ 0,1 SAY "1.5.1.1.0.0"
*    RESTORE FROM mem_var
*    @ 2,22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
*    @ 4,29 SAY "ENTITY SCHEMA OUTPUT"
*    @ 8,23 SAY "LISTED BELOW ARE THE CHOICES FOR HOW"
*    @ 9,23 SAY "YOU CAN HAVE THE RELATION"
*    @ 9,50 SAY CHOICE
*    @ 10,23 SAY "DISPLAYED."
*    @ 12,28 SAY "1) SCREEN OUTPUT"
*    @ 14,28 SAY "2) PRINTER OUTPUT"
*    @ 16,28 SAY "3) RETURN TO PREVIOUS MENU"
*    @ 17,1 SAY ""
*    ACCEPT' ENTER YOUR CHOICE (1-3) FROM ABOVE 'TO option
*    ERASE mem_var.mem
*    SAVE TO mem_var
*    DO CASE
*    CASE option = '1'
*    DO CASE
*    CASE CHOICE = 'USER'
DO 151110
CASE choice = 'SYSTEM'
DO 151110
CASE CHOICE = 'PROGRAM'
DO 151110
CASE choice = 'MODULE'
DO 151110
CASE CHOICE = 'DOUCMENT'
DO 151110
CASE choice = 'FILE'
DO 151110
CASE CHOICE = 'RECORD'
DO 151110
CASE choice = 'ELEMENT'
DO 151110
ENDCASE
CASE option = '2'
DO CASE
CASE CHOICE = 'USER'
DO 151120
CASE choice = 'SYSTEM'
DO 151120
CASE CHOICE = 'PROGRAM'
DO 151120
CASE choice = 'MODULE'
DO 151120
CASE CHOICE = 'DOUCMENT'
DO 151120
CASE choice = 'FILE'
DO 151120
CASE CHOICE = 'RECORD'
DO 151120
CASE choice = 'ELEMENT'
DO 151120
ENDCASE
CASE option = '3'
RETURN
OTHERWISE
CLEAR
@ 0,27 SAY option
@ 0,34 SAY "IS NOT A VALID CHOICE"
@ 1,26 SAY "PLEASE ENTER VALUES BETWEEN 1 AND 3 ONLY"
WAIT TO stop
ENDCASE
ENDDO
* 151110.PRG
* MODULE NAME: 1.5.1.1.0
* ROUTINES THAT CALL THE MODULE: 1.5.1.1.0.0
* ROUTINES THAT THE MODULE CALLS: 1.5.1.1.0.0
* LOCAL VARIABLES USED:
* choice : CONTAINS THE NUMBER OF ACTION SELECTED. MAY ALSO CONTAIN THE
*          CHARACTER STRING THAT IDENTIFIES THE RELATION BEING ADDED TO,
*          MODIFIED, DELETED FROM OR OUTPUT.
* count : USED TO KEEP TRACK OF THE RECORD NUMBER BEING DISPLAYED.
* stop : USED TO STOP ACTION FOR USER DECISION.
* t : REPRESENTS THE BOOLEAN VALUE TRUE IS USED TO CREATE A CONTINUES
*     LOOP.
* INPUT FILES: MEM_VAR.
* OUTPUT FILES: MEM_VAR.
* THIS MODULE ALLOW THE USER TO CHOOSE WHICH TYPE OF RELATIONSHIP WILL BE
* USED IN THE QUERY.
* DESIGNED BY: ROBERT A. KIRSCH II
* WRITTEN BY: ROBERT A. KIRSCH II
* BASIC FUNCTION OF MODULE:
* THIS MODULE WILL DISPLAY ON THE SCREEN ENTITY RELATION SCHEMA.
*
SET EXACT ON
set color to 0/3,3
set talk off
set menu on
SET EXACT ON
RESTORE FROM mem_var
CLEAR
@ 0,1 SAY "1.5.1.1.1.0"
@ 1,22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3,26 SAY "ENTITY SCHEMA SCREEN OUTPUT"
@ 5,22 SAY "THIS MODULE WILL DISPLAY"
@ 5,48 SAY choice
@ 7,22 SAY "IF YOU DO NOT WISH TO DISPLAY"
@ 8,22 SAY "THIS SCHEMA, TYPE 'O' TO"
@ 9,22 SAY "RETURN TO THE PREVIOUS MENU."
WAIT TO stop
DO CASE
CASE stop = '0'
RETURN
OTHERWISE
ENDCASE
@ 1,1 SAY "1.5.1.1.1.0"
@ 2,22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 4,30 SAY "RELATION SCHEMA FOR"
@ 6,37 SAY choice
DO CASE
CASE choice = 'USER'
CLEAR
USE USER
DISPLAY STRUCTURE
WAIT TO stop
RETURN
CASE choice = 'SYSTEM'
CLEAR
USE SYSTEM
DISPLAY STRUCTURE
WAIT TO stop
RETURN
CASE choice = 'PROGRAM'
CLEAR
USE PROGRAM
DISPLAY STRUCTURE
WAIT TO stop
RETURN
CASE choice = 'MODULE'
CLEAR
USE MODULE
DISPLAY STRUCTURE
WAIT TO stop
RETURN
CASE choice = 'DOCUMENT'
CLEAR
USE DOCUMENT
DISPLAY STRUCTURE
WAIT TO stop
RETURN
CASE choice = 'FILE'
CLEAR
USE FILE
DISPLAY STRUCTURE
WAIT TO stop
RETURN
CASE choice = 'RECORD'
CLEAR
USE RECORD
DISPLAY STRUCTURE
WAIT TO stop
RETURN
CASE choice = 'ELEMENT'
CLEAR
USE ELEMENT
DISPLAY STRUCTURE
WAIT TO stop
RETURN
ENDCASE
* 151120.PRG
  * MODULE NAME: 1.5.1.1.2.0
  * ROUTINES THAT CALL THE MODULE: 1.5.1.1.0.0
  * ROUTINES THAT THE MODULE CALLS: 1.5.1.1.0.0
  * LOCAL VARIABLES USED:
  * choice : CONTAINS THE NUMBER OF ACTION SELECTED. MAY ALSO CONTAIN THE
  *          CHARACTER STRING THAT IDENTIFIES THE RELATION BEING ADDED TO,
  *          MODIFIED, DELETED FROM OR OUTPUT.
  * count : USED TO KEEP TRACK OF THE RECORD NUMBER BEING DISPLAYED.
  * hold : USED TO STOP ACTION FOR USER DECISION.
  * option : CONTAINS THE USER'S CHOICE ON WHETHER TO OUTPUT TO THE SCREEN
  *           OR THE PRINTER.
  * t : REPRESENTS THE BOOLEAN VALUE TRUE IS USED TO CREATE A CONTINUES
  *           LOOP.
  * INPUT FILES: MEM_VAR, USER, SYSTEM, PROGRAM, MODULE, DOCUMENT, FILE, RECORD,
  * OUTPUT FILES: MEM_VAR USER, SYSTEM, PROGRAM, MODULE, DOCUMENT, FILE, RECORD,
  * THIS MODULE ALLOW THE USER TO CHOOSE WHICH TYPE OF RELATIONSHIP WILL BE
  * USED IN THE QUERY.
  * DESIGNED BY: ROBERT A. KIRSCH II
  * WRITTEN BY: ROBERT A. KIRSCH II
  * BASIC FUNCTION OF MODULE:
  * THIS MODULE WILL OUTPUT THE USER, SYSTEM, PROGRAM AND MODULE
  * RELATION FILES TO THE PRINTER.

SET EXACT ON
set color to 0/3,3
set talk off
set menu on
SET EXACT ON
RESTORE FROM mem_var
STORE 0 TO rec_num, stop
CLEAR
@ 0,1 SAY "1.5.1.1.2.0"
@ 1.22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3.27 SAY "ENTITY SCHEMA PRINTER OUTPUT"
@ 5.23 SAY "THIS MODULE WILL PRINT"
@ 5.47 SAY choice
@ 7.23 SAY "PLEASE INSURE THAT YOUR PRINTER"
@ 8.23 SAY "IS TURNED ON AND IN THE ONLINE"
@ 9.23 SAY "MODE"
@ 11.23 SAY "IF YOU DO NOT WISH TO PRINT"
@ 12.23 SAY "THIS SCHEMA, TYPE '0' TO"
@ 13.23 SAY "RETURN TO THE PREVIOUS MENU"
WAIT TO stop
DO CASE
CASE stop = '0'
RETURN
OTHERWISE
ENDCASE
SET DEVICE TO PRINT
SET CONSOLE OFF
@ 1,1 SAY "1.5.1.1.2.0"
@ 2,22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 4,31 SAY "RELATION SCHEMA FOR"
@ 6,35 SAY choice
@ 8,1 SAY ""
DO CASE
CASE choice = 'USER'
USE USER
DISPLAY STRUCTURE TO PRINT
CASE choice = 'SYSTEM'
USE SYSTEM
DISPLAY STRUCTURE TO PRINT
CASE choice = 'PROGRAM'
USE PROGRAM
DISPLAY STRUCTURE TO PRINT
CASE choice = 'MODULE'
USE MODULE
DISPLAY STRUCTURE TO PRINT
CASE choice = 'DOCUMENT'
USE DOCUMENT
DISPLAY STRUCTURE TO PRINT
CASE choice = 'FILE'
USE FILE
DISPLAY STRUCTURE TO PRINT
CASE choice = 'RECORD'
USE RECORD
DISPLAY STRUCTURE TO PRINT
CASE choice = 'ELEMENT'
USE ELEMENT
DISPLAY STRUCTURE TO PRINT
ENDCASE
SET DEVICE TO SCREEN
SET CONSOLE ON
RETURN
* MODULE NAME: 1.5.2.0.0.0
* ROUTINES THAT CALL THE MODULE: 1.1.0.0.0.0
* ROUTINES THAT THE MODULE CALLS: TBD, MAIN

* LOCAL VARIABLES USED:

* choice : CONTAINS THE NUMBER OF ACTION SELECTED. MAY ALSO CONTAIN THE
          CHARACTER STRING THAT IDENTIFIES THE RELATION BEING ADDED TO,
          MODIFIED, DELETED FROM OR OUTPUT.
* hold : USED TO STOP ACTION FOR USER DECISION.
* t : REPRESENTS THE BOOLEAN VALUE TRUE IS USED TO CREATE A CONTINUES
      LOOP.
* title : CONTAINS THE CHARACTER STRING THAT DESCRIBES THE RELATIONSHIP
          BEING ADDED TO, DELETED FROM OR OUTPUT.
* INPUT FILES: MEM_VAR.
* OUTPUT FILES: MEM_VAR.
* THIS MODULE ALLOW THE USER TO CHOOSE WHICH TYPE OF RELATIONSHIP WILL BE
  USED IN THE QUERY.
* DESIGNED BY: ROBERT A. KIRSCH II
* WRITTEN BY: ROBERT A. KIRSCH II
* BASIC FUNCTION OF MODULE:
  THIS MODULE ALLOW THE USER TO CHOOSE WHICH RELATIONSHIP HE WOULD
  LIKE TO DISPLAY THE SCHEMA OF.

SET EXACT ON
set color to 0/3,3
set talk off
CLEAR
do while .t.
ERASE mem_var.mem
CLEAR
@ 0,1 SAY "1.5.2.0.0.0"
@ 1,22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3,27 SAY "RELATIONSHIP SCHEMA OUTPUT"
@ 5,9 SAY "1) USER CONTAINS SYSTEM 8) FILE CONTAINS REC"
@ 5,64 SAY "ORDS"
@ 7,9 SAY "2) SYSTEM CONTAINS PROGRAM 9) RECORD CONTAINS E"
@ 7,64 SAY "LEMENT"
@ 9,9 SAY "3) PROGRAM PROCESSES FILE 10) USER RESPONSIBLE"
@ 9,64 SAY "FOR SYSTEM"
@ 11,9 SAY "4) PROGRAM PROCESSES RECORD 11) USER RESPONSIBLE"
@ 11,64 SAY "FOR FILE"
@ 13,9 SAY "5) PROGRAM PROCESSES ELEMENT 12) PROGRAM PRODUCES"
@ 13,64 SAY "DOCUMENT"
@ 15,9 SAY "6) SYSTEM CONTAINS PROGRAM 13) RETURN TO PREVIOUS"
(15,64) SAY "S MENU"
(17,9) SAY "7) PROGRAM CONTAINS MODULE  14) RETURN TO MAIN ME"
(17,64) SAY "NU"
(18,22) SAY ""
ACCEPT ' ' ENTER YOUR CHOICE (1-14) FROM ABOVE:'TO choice
DO CASE
CASE choice = "1"
store 'U_PROC_S' to choice
store 'USER CONTAINS SYSTEM' TO title
save to mem_var
do 152100
CASE choice = "2"
store 'S_PROC_P' to choice
store 'SYSTEM CONTAINS PROGRAM' TO title
save to mem_var
do 152100
CASE choice = "3"
store 'P_PROC_F' to choice
store 'PROGRAM PROCESSES FILE' TO title
save to mem_var
do 152100
CASE choice = "4"
store 'P_PROC_R' to choice
store 'PROGRAM PROCESSES RECORD' TO title
save to mem_var
do 152100
CASE choice = "5"
store 'P_PROC_E' to choice
store 'PROGRAM PROCESSES ELEMENT' TO title
save to mem_var
do 152100
CASE choice = "6"
store 'S_CONT_P' to choice
store 'SYSTEM CONTAINS PROGRAM' TO title
save to mem_var
do 152100
CASE choice = "7"
store 'P_CONT_M' to choice
store 'PROGRAM CONTAINS MODULE' TO title
save to mem_var
do 152100
CASE choice = "8"
store 'F_CONT_R' to choice
store 'FILE CONTAINS RECORD' TO title
save to mem_var
do 152100

240
CASE choice = "9"
store 'R_CONT_E' to choice
store 'RECORD CONTAINS ELEMENT' to title
save to mem_var
do 152100
CASE choice = "10"
store 'U_RESP_S' to choice
store 'USER RESPONSIBLE FOR SYSTEM' to title
save to mem_var
do 152100
CASE choice = "11"
store 'U_RESP_F' to choice
store 'USER RESPONSIBLE FOR FILE' to title
save to mem_var
do 152100
CASE choice = "12"
store 'P_PROD_D' to choice
store 'PROGRAM PRODUCES DOCUMENT' to title
save to mem_var
do 152100
CASE choice = "13"
RETURN
CASE choice = "14"
RETURN TO MASTER
OTHERWISE
CLEAR
@ 1,21 SAY choice
@ 1,28 SAY "IS NOT A VALID CHOICE"
@ 2,20 SAY "PLEASE ENTER VALUES BETWEEN 1 AND 14 ONLY"
@ 3,20 SAY "PRESS RETURN TO TRY AGAIN!"
ACCEPT TO hold
ENDCASE
ENDDO
RETURN
* 152100.PRG
* MODULE NAME: 1.5.2.1.0.0
* ROUTINES THAT CALL THE MODULE: 1.5.2.0.0.0
* ROUTINES THAT THE MODULE CALLS: 1.5.2.0.0.0
* LOCAL VARIABLES USED: choice: CONTAINS THE NUMBER OF ACTION SELECTED.
* t: REPRESENTS NO VALUE AT ALL.
* hold: USED TO STOP ACTION FOR USER DECISION.
* count: KEEPS TRACK OF ACCOUNT NUMBERS.
* option:
* INPUT FILES: MEM_VAR.
* OUTPUT FILES: MEM_VAR.
* THIS MODULE ALLOW THE USER TO CHOOSE WHICH TYPE OF RELATIONSHIP WILL BE
* USED IN THE QUERY.
* DESIGNED BY: ROBERT A. KIRSCH II
* WRITTEN BY: ROBERT A. KIRSCH II
* BASIC FUNCTION OF MODULE:
* THIS MODULE ALLOW THE USER TO CHOOSE WHETHER THE OUTPUT WILL BE
* DISPLAYED ON THE SCREEN OR PRINTED.

RESTORE FROM mem_var
STORE 0 TO rec_num, stop
CLEAR
STORE .t. TO TRUE
do while TRUE
CLEAR
@ 0,1 SAY "1.5.2.1.0.0"
RESTORE FROM mem_var
@ 2,22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 4,22 SAY "RELATIONSHIP SCHEMA OUTPUT"
@ 8,23 SAY "LISTED BELOW ARE THE CHOICES FOR"
@ 9,23 SAY "HOW YOU CAN HAVE THE SCHEMA FOR"
@ 10,24 SAY TITLE
@ 11,23 SAY "DISPLAYED."
@ 13,28 SAY "1) SCREEN OUTPUT"
@ 15,28 SAY "2) PRINTER OUTPUT"
@ 17,28 SAY "3) RETURN TO PREVIOUS MENU"
@ 18,1 SAY ""
ACCEPT' ENTER YOUR CHOICE (1-3) FROM ABOVE 'TO option
ERASE mem_var.mem
SAVE TO mem_var
DO CASE
CASE option = '1'
DO CASE
CASE choice = 'U_PROC_S'
DO 152110
CASE choice = 'S_PROC_P'
DO 152110
  CASE CHOICE = 'P_PROC_F'
  DO 152110
  CASE choice = 'P_PROC_R'
  DO 152110
  CASE choice = 'P_PROC_F'
  DO 152110
  CASE CHOICE = 'S_CONT_P'
  DO 152110
  CASE CHOICE = 'P_CONT_M'
  DO 152110
  CASE choice = 'F_CONT_R'
  DO 152110
  CASE CHOICE = 'R_CONT_E'
  DO 152110
  CASE choice = 'U_RESP_S'
  DO 152110
  CASE CHOICE = 'U_RESP_F'
  DO 152110
  CASE choice = 'P_PROD_D'
  DO 152110
ENDCASE
CASE option = '2'
DO CASE
  CASE CHOICE = 'U_PROC_S'
  DO 152120
  CASE choice = 'S_PROC_P'
  DO 152120
  CASE CHOICE = 'P_PROC_F'
  DO 152120
  CASE choice = 'P_PROC_R'
  DO 152120
  CASE CHOICE = 'P_PROC_E'
  DO 152120
  CASE choice = 'S_CONT_P'
  DO 152120
  CASE CHOICE = 'P_CONT_M'
  DO 152120
  CASE choice = 'F_CONT_R'
  DO 152120
  CASE CHOICE = 'R_CONT_E'
  DO 152120
  CASE choice = 'U_RESP_S'
  DO 152120
  CASE CHOICE = 'U_RESP_F'
  DO 152120
ENDCASE
CASE choice = 'P_PROD_D'
DO 152120
ENDCASE
CASE option = '3'
RETURN
OTHERWISE
CLEAR
@ 0.27 SAY option
@ 0.34 SAY "IS NOT A VALID CHOICE"
@ 1.26 SAY "PLEASE ENTER VALUES BETWEEN 1 AND 3 ONLY"
@ 2.26 SAY "PRESS RETURN AND TRY AGAIN!"
ACCEPT TO hold
ENDCASE
ENDCASE
ENDDO
* 152110.PRG
* MODULE NAME: 1.5.2.1.1.0
* ROUTINES THAT CALL THE MODULE: 1.5.2.1.0.0
* ROUTINES THAT THE MODULE CALLS: 1.5.2.1.0.0
* LOCAL VARIABLES USED: choice: CONTAINS THE NUMBER OF ACTION SELECTED.
* hold: USED TO STOP ACTION FOR USER DECISION.
* INPUT FILES: MEM_VAR U_CONTS, U_CONT_P, P_PROC_F, P_PROC_R,
* P_PROC_R. P_PROC_E, S_CONT_P, P_CONT_M, F_CONT_R, R_CONT_E,
* U_RESP_S, U_RESP_F, P_PRED_D.
* OUTPUT FILES: MEM_VAR
* U_CONTS, U_CONT_P, P_PROC_F, P_PROC_R,
* P_PROC_R. P_PROC_E, S_CONT_P, P_CONT_M, F_CONT_R, R_CONT_E,
* U_RESP_S, U_RESP_F, P_PRED_D.
* DESIGNED BY: ROBERT A. KIRSCH II
* WRITTEN BY: ROBERT A. KIRSCH II
* BASIC FUNCTION OF MODULE:
* THIS MODULE WILL DISPLAY ON THE RELATIONSHIP SCHEMAS
* RESTORE FROM mem_var
CLEAR
@ 0,1 SAY "1.5.2.1.1.0"
@ 1,22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3,25 SAY "RELATIONSHIP SCHEMA SCREEN OUTPUT"
@ 5,22 SAY "THIS MODULE WILL DISPLAY"
@ 7,23 SAY TITLE
@ 9,22 SAY "IF YOU DO NOT WISH TO DISPLAY"
@ 10,22 SAY "THIS SCHEMA, TYPE '0' TO"
@ 11,22 SAY "RETURN TO THE PREVIOUS MENU."
WAIT TO stop
DO CASE
CASE stop = '0'
RETURN
OTHERWISE
ENDCASE
CLEAR
@ 0,1 SAY "1.5.2.1.1.0"
@ 1,22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3,28 SAY "RELATIONSHIP SCHEMA FOR"
@ 5,27 SAY title
@ 7,1 SAY " "
DO CASE
CASE choice = 'U_PROC_S'
USE U_PROC_S
DISPLAY STRUCTURE
WAIT TO hold
RETURN
CASE choice = 'S_PROC_P'
USE S_PROC_P
DISPLAY STRUCTURE
WAIT TO hold
RETURN
CASE choice = 'P_PROC_F'
USE P_PROC_F
DISPLAY STRUCTURE
WAIT TO hold
RETURN
CASE choice = 'P_PROC_R'
USE P_PROC_R
DISPLAY STRUCTURE
WAIT TO hold
RETURN
CASE choice = 'P_PROC_E'
USE P_PROC_E
DISPLAY STRUCTURE
WAIT TO hold
RETURN
CASE choice = 'S_CONT_P'
USE S_CONT_P
DISPLAY STRUCTURE
WAIT TO hold
RETURN
CASE choice = 'P_CONT_M'
USE P_CONT_M
DISPLAY STRUCTURE
WAIT TO hold
RETURN
CASE choice = 'F_CONT_R'
USE F_CONT_R
DISPLAY STRUCTURE
WAIT TO hold
RETURN
CASE choice = 'R_CONT_E'
USE R_CONT_E
DISPLAY STRUCTURE
WAIT TO hold
RETURN
CASE choice = 'U_RESP_S'
USE U_RESP_S
DISPLAY STRUCTURE
WAIT TO hold
RETURN
CASE choice = 'U_RESP_F'
USE U_RESP_F
DISPLAY STRUCTURE
WAIT TO hold
RETURN
CASE choice = 'P_PROD_D'
USE P_PROD_D
DISPLAY STRUCTURE
WAIT TO hold
RETURN
* 152120.PRG
* MODULE NAME: 1.5.2.1.2.0
* ROUTINES THAT CALL THE MODULE: 1.5.2.1.0.0
* ROUTINES THAT THE MODULE CALLS: 1.5.2.1.0.0
* LOCAL VARIABLES USED: choice: CONTAINS THE NUMBER OF ACTION SELECTED.
*    hold: USED TO STOP ACTION FOR USER DECISION.
* INPUT FILES: MEM_VAR, U_CONTS, U_CONT, P_PROC, P_PROC_R,
*    P_PROC_R, P_PROC_E, S_CONT, P_CONT, F_CONT, R_CONT_E,
* OUTPUT FILES: MEM_VAR
*    U_CONTS, U_CONT, P_PROC, P_PROC_R,
*    P_PROC_R, P_PROC_E, S_CONT, P_CONT, F_CONT, R_CONT_E,
*    U_RESP_S, U_RESP, P_PROD_D.
* DESIGNED BY: ROBERT A. KIRCH II
* WRITTEN BY: ROBERT A. KIRCH II
* BASIC FUNCTION OF MODULE:
* THIS MODULE WILL OUTPUT THE USER, SYSTEM, PROGRAM AND MODULE
* RELATION FILES TO THE PRINTER.
* RESTORE FROM mem_var
STORE 0 TO rec_num, stop
CLEAR
@ 0,1 SAY "1.5.1.2.0"
@ 1,22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 3,27 SAY "RELATIONSHIP PRINTER OUTPUT"
@ 6,23 SAY "THIS MODULE WILL PRINT"
@ 8,24 SAY TITLE
@ 10,23 SAY "PLEASE INSURE THAT YOUR PRINTER"
@ 11,23 SAY "IS TURNED ON AND IN THE ONLINE"
@ 12,23 SAY "MODE"
@ 14,23 SAY "IF YOU DO NOT WISH TO PRINT"
@ 15,23 SAY "THIS RELATION, TYPE '0' TO"
@ 16,23 SAY "RETURN TO THE PREVIOUS MENU"
WAIT TO stop
DO CASE
CASE stop = '0'
RETURN
OTHERWISE
ENDCASE
SET DEVICE TO PRINT
SET CONSOLE OFF
@ 1,1 SAY "1.5.1.1.2.0"
@ 2,22 SAY "INFORMATION RESOURCE DICTIONARY SYSTEM"
@ 4,36 SAY "SCHEMA FOR"
@ 6,28 SAY title
@ 9,1 SAY " "
DO CASE
CASE choice = 'U_PROC_S'
USE U_PROC_S
DISPLAY STRUCTURE TO PRINT
CASE choice = 'S_PROC_P'
USE S_PROC_P
DISPLAY STRUCTURE TO PRINT
CASE choice = 'P_PROC_R'
USE P_PROC_R
DISPLAY STRUCTURE TO PRINT
CASE choice = 'P_PROC_E'
USE P_PROC_E
DISPLAY STRUCTURE TO PRINT
CASE choice = 'S_CONT_P'
USE S_CONT_P
DISPLAY STRUCTURE TO PRINT
CASE choice = 'P_CONT_M'
USE P_CONT_M
DISPLAY STRUCTURE TO PRINT
CASE choice = 'F_CONT_R'
USE F_CONT_R
DISPLAY STRUCTURE TO PRINT
CASE choice = 'R_CONT_E'
USE R_CONT_E
DISPLAY STRUCTURE TO PRINT
CASE choice = 'U_RESP_S'
USE U_RESP_S
DISPLAY STRUCTURE TO PRINT
CASE choice = 'U_RESP_F'
USE U_RESP_F
DISPLAY STRUCTURE TO PRINT
CASE choice = 'P_PROD_D'
USE P_PROD_D
DISPLAY STRUCTURE TO PRINT
ENDCASE
SET DEVICE TO SCREEN
SET CONSOLE ON
RETURN
LIST OF REFERENCES


9. Ibid, pp. 564-565


22. Ibid, p. 18.


32. Ibid, pp. 578-600.


34. Ibid, pp. 686-743.


<table>
<thead>
<tr>
<th>No.</th>
<th>Name and Code</th>
<th>Address and City, State and Zip Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>MAJ Robert A. Kirsch II</td>
<td>5458 Suwanee Circle, Mobile, Alabama 36608</td>
</tr>
<tr>
<td>2.</td>
<td>Professor Daniel R. Dolk, Code 54Dk</td>
<td>Naval Postgraduate School, Monterey, California 93943-5004</td>
</tr>
<tr>
<td>3.</td>
<td>LCDR Paul W. Callahan, Code 52Cs</td>
<td>Naval Postgraduate School, Monterey, California 93943-5004</td>
</tr>
<tr>
<td>4.</td>
<td>Computer Technology Programs, Code 37</td>
<td>Naval Postgraduate School, Monterey, California 93943-5004</td>
</tr>
<tr>
<td>5.</td>
<td>Library, Code 8142</td>
<td>Naval Postgraduate School, Monterey, California 93943-5002</td>
</tr>
<tr>
<td>6.</td>
<td>Defense Technical Information Center</td>
<td>Cameron Station, Alexandria, Virginia 22304-6145</td>
</tr>
</tbody>
</table>
END

FILMED

DTIC