

310
AN IMPLEMENTATION OF A SUBSET
OF PSL/PSA

by

FRANCIS B. HAJEK

B.A., Peru State College, 1961
M.S., Oklahoma State University, 1966
Ed.D., Oklahoma State University, 1970

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

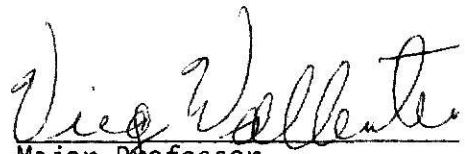
MASTER OF SCIENCE

Department of Computer Science

Kansas State University
Manhattan, Kansas

1984

Approved by:


Major Professor

LD
2668
R4
1984
H34
c. 2

ACKNOWLEDGEMENT

I wish to express my sincere appreciation to Dr. David Gustafson for his advice and encouragement in regard to this project and during my program of graduate study here at Kansas State University. I also wish to thank Tim Moore of the computing center staff for his help with IDMS in CMS. A very special acknowledgement is due to my wife Gretchen and daughter Angela for their love and patience during this project.

TABLE OF CONTENTS

| | |
|--|-----|
| Table of contents | ii |
| List of Illustrations | iii |
| Chapter 1 Introduction..... | 1 |
| 1.1 Overview..... | 1 |
| 1.2 Background | 2 |
| 1.21 Software Life Cycle | 2 |
| 1.22 Methodologies | 4 |
| 1.3 PSL/PSA | 5 |
| 1.4 General Requirements | 8 |
| 1.5 Organization | 10 |
| Chapter 2 System Design..... | 11 |
| Chapter 3 Restrictions and Limitations..... | 16 |
| Chapter 4 System Implementation | 22 |
| 4.1 Introduction | 23 |
| 4.2 IDMS Schema | 25 |
| 4.3 Program for PSL Entry | 28 |
| Chapter 5 Conclusions and Enhancements | 33 |
| 5.1 Conclusions | 33 |
| 5.2 Enhancements | 35 |
| Selected Bibliography | 37 |
| Appendix A: User's Manual | |
| Appendix B: EXECs and Procedures | |
| Appendix C: Sample Session | |
| Appendix D: Source Code Listings | |

LIST OF ILLUSTRATIONS

| | |
|---|----|
| 1. Table of PSL objects and Relationships | 17 |
| 2. Junction Records | 20 |
| 3. IDMS Schema Structure | 24 |
| 4. System Menu | A3 |

AN IMPLEMENTATION OF A SUBSET OF PSL/PSA

by

Frank Hajek

Chapter One

INTRODUCTION

1.1 OVERVIEW

This project implements some of the capabilities of PSL/PSA and makes use of the IDMS data base management system in the implementation. PSL/PSA is a software tool used for collection and analysis of data about information processing systems. The data is stored in a data base and can be analyzed and organized by PSA to define requirements specifications. The project is an experiment to determine if a workable subset of PSL/PSA could be implemented on the National Advanced Systems computer at Kansas State University and if the power of a large data base management system (IDMS) could be utilized to facilitate the implementation. A previous experiment implementing PSL with the relational data base Query-by-Example (QBE) provides a precedent for this type of project. The QBE experiment, described in a paper by Perriens(9), was successful in showing that a general-purpose relational data base system could provide the analytical support functions of PSA and thus could be used for implementation of PSL/PSA. The current project will attempt to employ the data base management system IDMS similar to the way in which QBE was used in the previous experiment. IDMS is based on a network

model and PSL, because it is modeled in terms of objects and relations among the objects, has a relational orientation and so the advantage of the theoretical similarities contributing to the success of the previous experiment will not be present in this project.

1.2 BACKGROUND

1.2.1 SOFTWARE LIFE CYCLE

It is generally agreed that software passes through several stages from its inception to its use in a production environment.(16) These stages are collectively called the software life cycle.

The first stage, called the requirements analysis stage, defines the requirements for an acceptable solution to a problem. All aspects of the new system including resources, limitations, necessary features, and optional features must be considered at this stage so that the best choices between conflicting constraints can be made.

The second stage is called the specification stage. In this stage inputs, outputs, and general algorithms are determined. It is here that PSL/PSA can be used to identify, record, and analyze the relationships between the many elements of the system. PSA reports can help coordinate the efforts of the various teams involved in the project.

In the next stage, called the design stage, specific algorithms are developed and modules are defined. A stepwise decomposition is used to subdivide the modules so

that they could be developed by individuals or small teams.

The fourth stage is called the coding stage. The algorithms are implemented in a computer language in this stage. If the previous steps were done well, this stage is reasonably easy and leads to more correct results.

In the fifth stage, called the testing stage, the performance of the computer code is compared with some known results. The test values for this stage should have been specified in the design stage. For best results modules should first be tested independently, then small groups of components should be tested together and finally the complete system should be tested.

The sixth and last stage is the operation and maintenance stage. In this stage errors found by users of the system are corrected and changes requested by the users are made. More than 60% of the total effort expended on a software project is often represented in this last stage.

It is important that all stages of the software life cycle receive careful consideration. Problems at one stage may not be detected until a later stage. For example, the design may reveal flaws in the specifications and weakness in the design may be revealed in the coding, testing, or operations stages. The earlier that the problems are found and corrected, the less costly and disruptive will be their solution. Thus any methodology to improve the quality of the early stages of software development can prove to be very beneficial.

1.22 METHODOLOGIES

One of the attempts to utilize a computer in the design and development of large software systems was the ISDOS Project. The ISDOS Project was developed by the Department of Industrial Engineering at the University of Michigan during the 1960's(4). The ISDOS project identified several problems associated with the building of large software systems. Some of these problems are:

- There are inconsistent levels of detail on different parts of the system because of different abilities of communication of the personnel involved with the project.

- There are incomplete specifications because of confusion about who actually writes specifications for some parts of the system.

- There is inconsistency in the naming of variables and even in the meanings associated with some of the words used.

- There is inefficient data storage because of different views of the relationships between the data.

- There is often a communication problem between designers and future users of the system.

A major result of the ISDOS project in response to these problems was the development of PSL/PSA. PSL/PSA, a subset of which is to be implemented in the current project, is discussed in section 1.3.

Several other methodologies have been developed, some automated and some manual, to aid in the requirements

specification stage. Among these are Software Engineering Requirements Methodologies (SREM) (1), Higher Order Software (HOS) (5), Structured Analysis (SA) (12), and the methods used on the A-7 project (6) .

1.3 PSL/PSA

Problem Statement Language-Problem Statement Analyzer (PSL/PSA) is a software tool that can be used to analyze systems and develop specifications. PSL is a language designed primarily for use during the requirements specifications phase of the system life cycle. PSA is the tool designed to analyze and interpret the PSL language.

The PSL language consists of statements which describe a software system in terms of system objects and how these objects are related to one another. These statements allow high-level design specifications but they must employ specific constructs and obey certain laws of syntax. Besides these formal statements, there is also some provision for brief narrative descriptions.

PSA is a software package which accepts the language statements, processes them, and stores them in a data base. Information from the data base can then be used by PSA to provide diagnostic support in terms of error and consistency checks and also to promote design analysis by extensive report generation.

There are definite advantages to be gained by the use of PSL/PSA. PSL/PSA promotes better communication between all

groups involved with the development of the new software system. Consistency is enhanced because of the ability to check the documentation from the early stages by everyone involved in the project. Checks for completeness can also be provided by PSL/PSA at any point in the project. These advantages are realized largely because the system is automated but there are also other advantages inherent in PSL/PSA. The unambiguous syntax provides for a better definition of the problem and allows for consistent documentation of the requirements by different teams. PSA outputs such as the Formatted Problem Statement report give a clear presentation of the overall requirements specification.(9) The PSL language is designed to state the requirements and not the procedures to solve the problem and so is less constricting on the designers. Although there would be an expected time and cost savings because of automation of clerical tasks, the major benefit claimed for PSL/PSA is that the quality of the documentation is improved and thus the cost of design, implementation, and maintenance is reduced.

The PSL/PSA system is characterized by the following capabilities.

- (a) A special language used to describe software systems.
- (b) A translator for putting the formal language statements in an internally processable form.
- (c) A data base for storing the information from step(b).
- (d) A report generator for retrieval, analysis, and formatting of information from the data base.
- (e) An update facility for maintaining

consistent, and accurate information in the data base.

The PSL language specifies objects, properties of the objects, and relationships among the objects. The objects may be physical or conceptual things in the proposed software system such as inputs, outputs, processes, conditions, events, or logical collections of data such as sets, entities, or groups. Each object must have a name and be classified by type. There are approximately 30 different object types in PSL.

Relationships describe the connections and interrelations between objects, such as PROCESS-object RECEIVES INPUT-object or PROCESS-object GENERATES OUTPUT-object. Relationships may be used to describe the interaction between a system and its environment, to indicate the size and behavior of the system, to describe the data flow of the system, to define logical collections of data, to indicate subdivisions of system objects, and to allow for presenting many characteristics of system objects. There are approximately 75 different relationships available in PSL.

Properties of PSL objects are special relationships that allow for more complete descriptions and presentation of the of the characteristics of objects. While most relationships represent the connection between two different object types, properties are used to represent associations between objects of the same type. Examples of Properties relationships are SUBPARTS ARE, and ATTRIBUTES ARE.

PSL/PSA can be used for writing better requirements specifications. The PSL language allows description of proposed software systems. PSA provides analysis and

documentation support. PSL/PSA, if applied carefully in the specification stage can result in reduced costs for later stages of the software life cycle.

1.4 General Requirements.

Kansas State University students of computer Science do not have access to a PSL/PSA system and so this project is intended to provide a software system which will simulate some of the capabilities of PSL/PSA for use by students studying software engineering. The PSL/PSA system contains many more objects and relationships than can be effectively utilized by the novice user of the system. A small but workable subset of PSL/PSA-like capabilities will allow the user to gain some familiarity and competence with PSL/PSA and gain experience, in at least a limited way, with an automated requirements specification system.

In order to provide a worthwhile implementation of a subset of PSL/PSA, the following goals were deemed important. These goals define general requirements for the system and each goal leads to several specific decisions on how best the goal can be achieved. The goals are described in the remainder of this section while the choices and considerations for achieving these goals are discussed in the next chapter.

The first goal is that of a non-trivial implementation. The system should include enough PSL objects and relationships to allow implementation of simple but non-trivial examples. Some basis for determining how many objects and relationships to implement is given in 'A PSL Primer', by Eldon Wig(15), The paper is a beginner's manual

for PSL/PSA. Its purpose is to provide the novice user of PSL/PSA with a basic familiarity and understanding of the system without having to struggle through the considerable documentation usually required. It explains the use of a subset of PSL/PSA in describing a simple payroll processing system. The current project should attempt to implement most of the PSL/PSA capability discussed in this PSL/PSA primer.

The second goal is that of user-friendliness. The system should be user friendly to the extent that learning the mechanics of the system can be accomplished quickly. The user should not have to memorize a large set of new terms to use the system. He/She should not have to be concerned with syntax such as using commas, semi-colons, and other marks of punctuation. The user should be reminded of his options or be prompted for particular information each time he/she is required to enter data for the computer.

The third goal is that of being educational. The system should aid the student in learning about PSL/PSA by guiding him/her to make valid choices about PSL objects and the relationships between these objects. The student should be able to use the system even with a very meager knowledge of PSL/PSA. When the user selects the option of entering a PSL object in the data base he should be reminded of the different objects types available in this system. When he intends to insert relationships between pairs of these objects he should be reminded of the possible relationships available in this system and the particular object types that

may be associated with each relationship. Of course to be an aid to learning PSL/PSA the information presented must represent valid PSL/PSA constructs and be repeated each time the user attempts to enter objects or relationships.

The fourth goal is that of convenient access.

A student should be able to access the system with a minimum of JCL and commands. The student should not have to keep track of details such as opening and closing files and saving updated files of information. Ideally the user should be able, after logging on the computer, to issue a single command and have the software become available for use. A single command from within the system should allow the user to exit the system with all the changes saved.

Thus a non-trivial implementation, user friendliness, instructiveness, and convenient access are the four goals identified for this project. The design and implementation of the project was conducted with these goals in mind.

1.5 Report Organization.

The rest of this report is organized as follows. Chapter two presents some of the design considerations and tradeoff possibilities. The third chapter gives more specific restrictions and limitations of the project. Programs and procedures to be implemented are discussed in chapter four, and a summary of the project and future enhancements are presented in chapter five.

Chapter 2

System Design

In an attempt to accomplish goal number one, eight (8) PSL objects and 24 relationships between these objects were included. This includes the objects (except for SET, INTERFACE, INTERVAL, KEYWORD, and SYSTEM-PARAMETER) mentioned in the PSL Primer(15).

The eight objects are INPUT, OUTPUT, PROCES, ENTITY, GROUP, ELEMENT, EVENT, and CONDITION. PSL/PSA is to be used to model a software system and thus the above named objects have the following meanings. INPUT - data that enters the software system from outside the system. OUTPUT - information generated within the system and delivered to the external world. PROCESS - the part of the system that converts input to output. ENTITY - a compound data item internal to the system. ELEMENT - analogous to a field of data. GROUP - a collection of fields within a record of data. EVENT - a discrete occurrence in time. CONDITION - a status or state.

These eight objects allow for approximately 29(excluding inverse relationships) relationships and twenty four of these relationships were implemented. Several different relationships may exist between objects of the same type and since there may be several instances of each object, these eight objects and twenty four relationships provide an opportunity to exercise many of the capabilities of PSL/PSA. Thus the user should be able to generate a non-trivial requirements specification.

In order to achieve the second goal, to make the system as user friendly as possible it was decided to have the system in an interactive rather than a batch mode. Although batch mode might involve less computer cost, the ease of data entry and immediate response of interactive mode was considered to be more important. PSA has an associated data base to store the PSL information. To provide this function, this project uses an existing data base and associated data base management system, IDMS, available on the AS 5 computer. IDMS provides the usual advantages of a data base management system such as sharability, nonredundancy of data, ease of use, flexibility and integrity. For this project, the capabilities of IDMS to locate and retrieve data quickly will be utilized. Also the structure of the system will be used to help to correctly organize and maintain the PSL data. Further, the use of IDMS will provide some information about the advantages and disadvantages of a network data base model being used to implement PSL/PSA.

There are many ways to map the PSL objects and relations to the structures in IDMS. The least restrictive mapping is to map information into the data base in a strictly textual format. That is, let each record consist of only one large field which would contain the object name, object type, a description of the object, and a statement about relationships in which the object participates. This would allow the user the freedom to include all of the PSL objects and relationships. The user would also have less restrictions on how the information should be organized,

however this lack of specific organization would make reports similar to that of PSL/PSA very difficult if not impossible to generate. There would be no key on which to sort and no way to pick out particular pieces of information because the information would not necessarily be in the same position in each record. Also it would not be using the power of IDMS to enforce the PSL/PSA rules. The user could inadvertently insert incorrect object types or establish relationships that would not be valid in PSL/PSA. Another approach might be to have one record for each object with a field for the object type, a field for the object name, a field for a description of the object, and one or more fields for listing relationships in which the object participates. This would allow for sorting on various keys or fields of the records but still the power of IDMS would not be utilized. A simple file with these fields could perhaps serve as well as the data base.

Because of the limitations of these approaches, a more structured approach to mapping of PSL information into the data base was employed.

By letting the objects of PSL/PSA be associated with the records of IDMS and letting the relationships of PSL/PSA be associated with sets of IDMS, a more restrictive mapping of PSL/PSA into IDMS was obtained. This mapping imposes a certain organizational structure on the information entered, and is therefore much more restrictive than if only textual information were used. However, making use of the network structure of the data base greatly enhances navigation through the data base for searching and report generation.

Letting the PSL objects be represented by records in the data base encourages more accurate and consistent use of names and terminology. Thus, the structure and capabilities of IDMS could be used to advantage in enforcing the rules of PSL/PSA.

In order to achieve the third goal of being educational, it was considered appropriate to write the application program for the IDMS data base in such a way that the user would not need to deal with the complexities of IDMS, but to simply respond to prompts from the program. The user is therefore presented menus from which he/she may choose items. He/She is then prompted for appropriate names and data needed to ensure correct processing of the selected item. This kind of organization and presentation thus helps to fulfill goal number three. The user has the opportunity to learn PSL/PSA by using a subset of it and is guided in making valid choices regarding PSL objects and relationships.

In an effort to achieve the fourth goal, to make the system convenient to access, an EXEC file is used to carry out the many commands and JCL required to run an IDMS application in CMS, the interactive system on the computer. The necessary commands are quite extensive and would present a formidable obstacle if not contained in an EXEC. The commands and EXEC are documented in the appendix.

In this chapter methods of achieving the four goals of a non-trivial implementation, user friendliness, being educational, and convenience of access, have been considered. The plan to achieve these goals includes

mapping PSL data into the IDMS data base by letting PSL objects be IDMS records and PSL relationships be IDMS sets, letting the IDMS data base management system perform the function normally provided by PSA, and the use of the IDMS system in an interactive mode by a menu-driven application program initiated by an EXEC.

Chapter 3

Restrictions and Limitations.

In this chapter some restrictions and limitations of the project are discussed. These restrictions and limitations involve the separate sections of code required to manipulate each object and relationship, implementation of inverse relationships, implementation of the ATTRIBUTES ARE relationship, implementation of relationships of an object to objects of the same type, and implementation of many to many relationships.

Table I indicates the objects and relationships to which this project is restricted. Each unique relationship requires a separate section of code because when the application program is compiled the IDMS schema (actually subschema) is checked and the actual IDMS record type (PSL object type) and IDMS set type (PSL relationship) is expected by the compiler. This prevents using a general procedure which could be called many times to insert the different relationships and so the application program is difficult to modularize.

Table I is symmetric in the sense that if a relationship exists between object A and object B, then an inverse relationship exists between object B and object A. For example with objects PROCESS and INPUT, the relationship PROCESS USES INPUT has an inverse relationship INPUT is USED BY PROCESS.

IDMS allows for owner pointers, that is, pointers from

| PSL OBJECTS | INPUT | OUTPUT | PROCESS | ENTITY | GROUP | ELEMENT | EVENT | CONDITION | ATTRIBUTE |
|-------------|--|--|--|---|---|---|---|--|-----------------------|
| INPUT | PARTS OF (SS) SUBPARTS ARE (SS) | | RECEIVED BY (SF) USED BY (DD) | | CONSISTS OF (DS) | CONSISTS OF (DS) | | | ATTRIBUTES ARE (P) |
| OUTPUT | | PARTS OF (SS) SUBPARTS ARE (SS) | GENERATED BY (SF) DERIVED BY (DD) | | CONSISTS OF (DS) | CONSISTS OF (DS) | | | ATTRIBUTES ARE (P) |
| PROCESS | USES (DD) RECEIVES (SF) | GENERATES (SF) DERIVES (DD) | UTILIZES (SS) SUBPARTS ARE (SS) | USES (DD) UPDATES (DD) DERIVES (DD) | USES (DD) UPDATES (DD) DERIVES (DD) | USES (DD) UPDATES (DD) DERIVES (DD) | INCEPTION- CAUSES (SD) TERMINATION CAUSES (SD) TRIGGERED BY (SD) | | ATTRIBUTES ARE (P) |
| ENTITY | | | USED BY (DD) UPDATED BY (DD) DERIVED BY (DD) | | CONSISTS OF (DS) | CONSISTS OF (DS) | | | ATTRIBUTES ARE (P) |
| GROUP | CONTAINED IN (DS) | CONTAINED IN (DS) | USED BY (DD) UPDATED BY (DD) DERIVED BY (DD) | CONTAINED IN (DS) | CONSISTS OF (DS) | CONSISTS OF (DS) | | | ATTRIBUTES ARE (P) |
| ELEMENT | CONTAINED IN (DS) | CONTAINED IN (DS) | USED BY (DD) UPDATED BY (DD) DERIVED BY (DD) | CONTAINED IN (DS) | CONTAINED IN (DS) | | | | ATTRIBUTES ARE (P) |
| EVENT | | | ON- INCEP- TION (SD) ON- TERMIN- ATION (SD) TRIGGERS (SD) | | | | | WHEN A CONDITION BECOMES T OR F (SD) | ATTRIBUTES ARE (P) |
| CONDITION | | | | | | | BECOMING T OR F (SD) | | ATTRIBUTES ARE (P) |

SF SYSTEM FLOW
SS SYSTEM STRUCTURE
DS DATA STRUCTURE
DD DATA DERIVATION
SD SYSTEM DYNAMICS
P PROPERTIES

TABLE 1

each member in a set directly back to the owner of the set. The use of these pointers would allow for the inverse relationships to be printed out in reports even though they were not actually entered as such. This corresponds to the "double entry bookkeeping" of PSL/PSA. (In PSL/PSA when a relationship is entered, its inverse is automatically entered.) For example, to simulate INPUT USED BY PROCESS, which is the inverse of PROCESS USES INPUT, locate the INPUT-OWNER and traverse the INPUT-OWNER SUBPARTS ARE INPUT set. For each member record of this set use the owner pointer of the set PROCESS USES INPUT to find and print each PROCESS so that this particular INPUT is USED BY PROCESS relationship is displayed.

Although the owner pointers are implemented in the schema, the implementation of the inverse relationships in this way would require another 24 sections of code in the application program. In order to allow timely completion of the project the inverse relationships were omitted.

Each PSL object can be characterized by one or more attributes. Some of these attributes could actually be represented by ATTRIBUTE-VALUE relationships. However this would involve implementing many more records and sets in IDMS and so to keep the project manageable it was decided not to implement this relationship but rather to allow for a field in the IDMS record to list the particular attributes and descriptions of each object.

PSL/PSA allows certain objects to be related to themselves, for example PROCESS UTILIZES PROCESS, but IDMS does not allow for sets where the owner object type is the

same as the member object type and so these types of relationships could not be implemented.

IDMS does not directly allow for many to many relationships. Only one to n relationships are allowed. That is, one owner record to n (n greater than or equal to one) member records (relationships) for each set. However, one record type can be a member of more than one set type in IDMS. If an owner record of a first set type is related to many members and each of these members is related to an owner of a second set type, the owner record of the first set type will be related to many records of the second set type. Since this situation can also be reversed, a many to many relationship is possible. These records which are members of two or more set types are called "junction" records. (10) In this project the junction records are duplicates of the (object name)_owner record and related to the (object name)_owner record by the relationship SUBPARTS ARE. For example to allow for a Process to be associated with several inputs and an input to be associated with several processes the INPUT record type is used as a junction record. When an input is entered it is actually stored as an INPUT_OWNER record type. Then when the set PROCESS_USES_INPUT is entered a copy of the INPUT_OWNER record is made and stored as an INPUT record type. This record is then linked to the correct PROCESS record to complete the PROCESS_USES_INPUT set and also linked to the INPUT_OWNER record type of which it is a copy so that it is a member record of the INPUT_OWNER_SUBPARTS_ ARE_INPUT set. This same procedure is carried out for the sets that involve

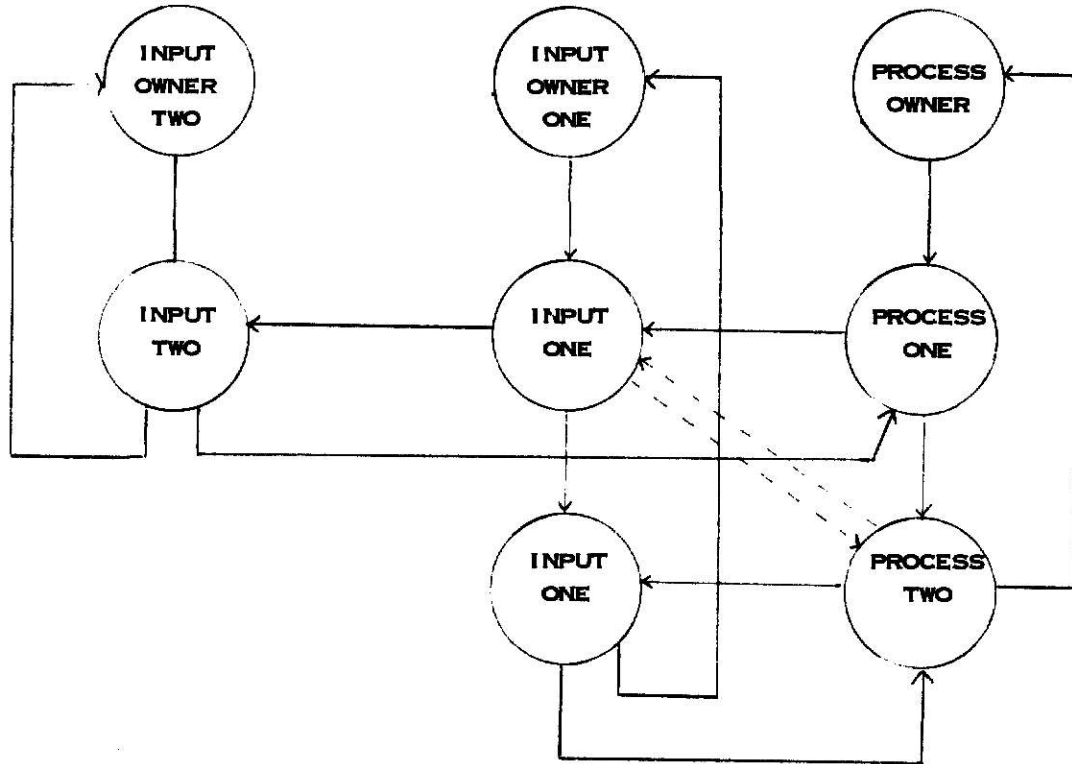


FIGURE 1

The "junction" records INPUT-ONE and INPUT-TWO allow PROCESS-ONE to be associated with two INPUTs, and INPUT-ONE to be associated with two PROCESSES. Junction records are necessary since the relationship represented by the dashed line is not allowed in IDMS.

Process and Entity, Process and Group, Process and Element, and Process and Output. The use of the SUBPARTS ARE relationship is not used in exactly the way it would be used in PSL/PSA and is not available as such for the user of the system. It is used here only to facilitate implementation of the possible many to many relationships between PROCESS objects and several of the other object types in the system.

Only a minimum amount of the report generating capabilities of PSL/PSA was provided. The PSL objects that have been entered are printed, followed by any relationships between these objects, as defined by the user. This information printed is not actually given as it would be in any of the several PSA reports , but it most closely resembles that of the PSA report called the Formatted Problem Statement. Other reports could be generated by selective searches of the data base but the report generated by this project gives a complete picture of the PSL/PSA information that has been entered. The procedures used to generate this report demonstrate the basic procedures for searching the data base.

The program consists of several "case" statements, (implemented by the PL/I Select statement) one for each menu. Each "case" statement is followed by a series of "procedures", (not actually a PL/I procedure, but a separate code section accessed by GOTO statements), one for each option of the case statement, to perform the operations selected. There are for instance, 24 such "procedures" to enter the 24 relationships implemented in this project, and 24 "procedures" to delete the relationships. Although the

codes for these "procedures" are very similar they must each have the specific IDMS names and so cannot be replaced by a single module.

Thus several special considerations were made and techniques employed to facilitate the completion of this project. The project is limited to the implementation of only objects and relationships listed in Table 1. Not all of these possible relationships were implemented. Some relationships, for example UTILIZES, were not implemented because of the restrictions of IDMS and others, for example the inverse relationships and the ATTRIBUTES-ARE relationships, because they would greatly increase the scope of the project. Several (object name)-OWNER records had to be created to allow the many to many relationships of PSL/PSA to be represented in IDMS. The report generating facility was limited to printing the contents of the data base in a form similar to that of the PSA Formatted Problem Statement. The implementation of the system under these constraints is discussed in the next chapter.

Chapter Four

System Implementation.

4.1 Introduction.

In this chapter some of the details of the implementation for this project are given. The implementation includes a schema and an application program. The schema is written using DDL(Data Description Language) with the host language being COBOL. The application program is written using DML (Data Manipulation Language) with the host language being PL/I. The programs were implemented on the National Advanced Systems computer at Kansas State University. Some of the JCL is given and described in the appendix.

The first step in the implementation was to design and code an IDMS schema to represent the PSA data base. The schema is a complete description of the data base and must include the names of all data items, records, sets, and areas in the data base. A data structure diagram given by Figure 2 was used to represent the PSL objects and relationships of Table 1. The schema for this project is discussed further in section 4.2.

The second step of the project was to design and code an application program that would allow entry of PSL-like information into the IDMS data base. The application program is a PL/I program that allows for entry of the PSL objects

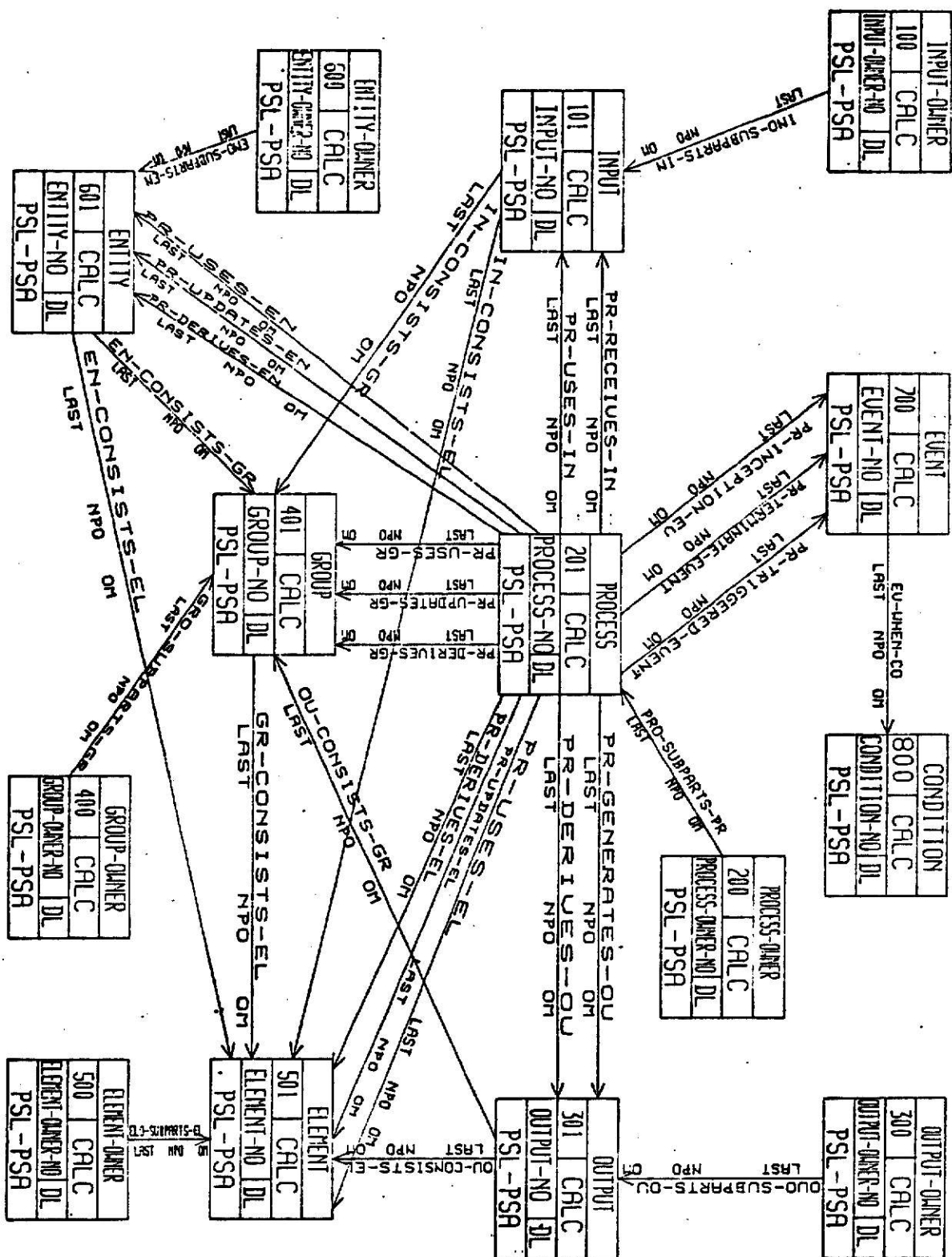


FIGURE 2

and relationships. The use of DML commands then allows storage of these objects and relationships in the IDMS data base. This program is designed to aid the novice user of PSL/PSA by suggesting only legitimate choices for IDMS set relationships. The application program for this project is discussed in section 4.3.

The third step of the project was to design and code a program to generate reports on the contents of the data base. This program must allow for navigation through the entire data base so that the information pertinent to a particular report could be retrieved. The information must then be compiled and printed in the desired report format. The report generation for this project is accomplished by the application program and is discussed further in section 4.4.

4.2 IDMS Schema

The schema for the PSL/PSA data base defines all record types, set types, areas, and files used in the data base. Figure 2 is a graphical representation of the complete data base. For this project a record type was defined for each of eight PSL objects, INPUT, OUTPUT, PROCESS, ENTITY, ELEMENT, EVENT, and CONDITION. Other record types were defined, those with the suffix OWNER, to facilitate location of records and navigation through the data base. For example, for records of type PROCESS a record PROCESS_OWNER was defined and used as the owner of the set PROCESS_OWNER SUBPARTS ARE PROCESS. This facilitates location of PROCESS records because once this

owner record is located by an IDMS command, one can traverse the set by use of the IDMS links between members of a set. An IDMS set type was defined for each of twenty four (24) PSL relationships to be represented, plus six (6) sets to help facilitate location and retrieval of records from the data base. There may be more than one relationship, thus IDMS set, between two record types. For instance if 'Customer-Orders' and 'Pay-Amounts' are INPUT objects and 'Order-Process' and 'Bill-Process' are PROCESS objects, some of the relationships that may exist are: Order-Process USES Customer-Orders; Bill-Process USES Pay-Amounts; and Bill-Process USES Customer-Orders. The schema gives the basic structure for the PSL/PSA data base. The record types and set types which are defined, such as PROCESS USES INPUT, indicate which relationships are possible, but the actual entry of specific instances of these records and sets is accomplished by an application program.

The schema code is comprised of five main subdivisions: the schema description statements, the file description statements, area description statements, record description statements, and set description statements. The complete code for the IDMS Schema program is given in the appendix but a brief description is as follows.

The schema description statements consist of the schema name and version number, date, installation information, and a section entitled remarks for further documentation. Only the schema name is mandatory.

The file description statements indicate the number and location of the data base files. There may be several files

used with the data base. A usual convention is to have each data base Area be stored on a separate file. It is also necessary to have one file, called the Journal file, for keeping track of data base transactions for purposes of backup. Since just one Area was defined for this project, only one data base file, besides the Journal file, is used.

The Area description indicates that the data base be divided into Areas and Pages within these Areas. The subdivision of a data base into different Areas allows for situations where certain portions of the data base are not made available to some of the users. This provides for some security and a means of control of access to the data. Since these concerns are not relevant to this project only one Area, named PSL-PSA, was used for this schema.

The record description statements give the name and field descriptions of each of the record types used in the data base. Each record type had a four character field for a record number, a sixteen character field for a record name, and an eighty character field to be used to indicate attributes or a description of the PSL object represented by the record.

Set description statements consist of Name, Order, Owner, and Member sentences for each set in the data base. The set names are made up of the first two characters of the owner record name concatenated with the name of the PSL relationship connecting the two PSL objects, concatenated with the first two characters of the member record name. The ORDER sentence for each of the sets in the schema

indicates the order in which new records will be positioned in the set. This has been arbitrarily specified as LAST for each of the sets. The MODE sentence specifies NEXT and PRIOR linkage so that the set can be traversed either forward or backward. The OWNER sentence gives the owner name and specifies the position of the sets NEXT and PRIOR pointers in the record prefix. The MEMBER sentence gives the member recrd name and specifies the position of the NEXT, PRIOR, and OWNER pointers in the record prefix. The disconnect option of the MEMBER sentence was set to OPTIONAL MANUAL so that records could be disconnected(or connected to) from a set without affecting the other sets in which the record participates.

4.3 Program for PSL entry.

The application program is the vehicle which allows entry, deletion, or modification of data, in this case the PSL objects and relationships, into the data base and the manipulation and retrieval of the data. For this project the application program first requires the entry of each PSL object name that is to be a record instance in the IDMS data base. To enter a PSL object, the name and description of the object must be entered and assigned to the proper fields of the IDMS record. The count for that object type is incremented and assigned to the count field of the record and then the record is stored in the data base with the IDMS STORE command. If the object is of type PROCESS it is then connected to the PROCESS_OWNER record that is set up when

the program begins. If the object is of type INPUT it is stored as an INPUT_OWNER record. OBJECTS of type OUTPUT, ENTITY, GROUP, and ELEMENT are also stored as (object_name)_OWNER. If the object is of type EVENT or CONDITION it is simply stored as such.

To modify a PSL object, the record must first be located, the new information must then be entered in the record, and the record stored again in the data base with the IDMS command MODIFY.

To delete a PSL object from the data base, first the record must be located ,then disconnected from any PSL relationships it may have participated in, if any, and then ERASED. The count for that object type must be decremented and the count field in each succeeding record of that type must also be adjusted accordingly.

Once the proper PSL objects have been entered in the data base then relationships between these objects can be defined. In a PSL relationship such as INPUT CONSISTS OF ELEMENT, INPUT is considered the owner and ELEMENT is considered a member for purposes of inserting the relationship into an IDMS set. First the owner object which would be of type INPUT_OWNER in the above relationship, must be located. Then if no copy of INPUT_OWNER exists, a copy is made and STORED as an INPUT record. This sets up the pointers for the IDMS set. The member object, which would be ELEMENT_OWNER in the above relationship, is then located and a copy is made and STORED as an object of type ELEMENT. The two objects are then entered in the IDMS set relationship by a CONNECT command. If the owner object is of type PROCESS

then no copy of it is made since PROCESS is not a junction record as is INPUT for instance.

To delete relationships the location process is followed as above, of course no new records are created, and a DISCONNECT comand is issued instead of the CONNECT command.

When entry of all instances of records and sets has been made the data base has the data necessary to generate some of the information to be included in PSL/PSA reports. A separate procedure may then be used to locate certain records and sets in the data base, to retrieve the required data, and compile the information to be included in the printed report. The procedure to print a report on the contents of the data base operates as follows. First each object named for each of the eight objects types is located by the IDMS CALC command and the name and attribute information for the object is printed. Then each relationship is taken in turn beginning with the relationship PROCESS RECEIVES INPUT. The owner_object type is searched and when an owner of the relationship is found all member names are printed. A similar process is followed for each of the 24 relationships implemented in this project so that every instance of each of the relationships is printed.

Since the application program is written in PL/I but uses IDMS, the PL/I compiler must recognize the IDMS DML commands and names. This is achieved by the use of several IDMS EXECs. The Exec SUBSCHEMA_DESCRIPTION relieves the user from having to declare all the variables of the subschema

again in the PL/I application program. This Exec, along with the Exec SUBSCHEMA_BINDS generate the communications block, IDMS names, and IDMS records, and the necessary BIND functions for the application program. The Exec IDMS_STATUS is used to check for error conditions and should be called after each DML command. In this program, however, many times the IDMS-STATUS is not called after DML commands so that conditions like end-of-set and end-of-area could be used to terminate loops in the program.

The naming conventions of IDMS allow only sixteen characters for a name so that set names such as PROCESS_USES_INPUT and INPUT_OWNER_SUBPARTS_ARE_INPUT would be too long. As indicated in section 4.2 the set name will consist of the first two characters of the owner record type name, an underscore, the the PSL relationship name, an underscore, and the first two characters of the member record type name. An exception to this is that record type names with the suffix OWNER are written as the first two characters of the record type followed by an O. Thus PROCESS_USES_INPUT is written as PR_USES_IN and INPUT_OWNER_SUBPARTS_ARE_INPUT is written as INO_SUBPARTS_IN.

Incorporation of the above procedures, Execs and naming conventions provides an application program which allows interactive storage, modification, and retrieval of PSL-like information from the IDMS data base without the user having to be concerned about the details of IDMS. In fact the user may assume that he/she is working in a strictly PSL/PSA type system and need not be aware that IDMS is even involved.

This is made possible by the menu-driven nature of the PL/I application program. The user only makes choices from menus or responds to prompts from the program. A user manual is provided which gives more complete details on use of the program.

CHAPTER 5

5.1 Conclusions.

The goals of the project, as listed in Chapter Two were all achieved. In fact, many of the capabilities included in the PSL Primer by Wig(15) are contained in this project. The system is convenient to access, user friendly, and allows for some learning about PSL/PSA.

The eight PSL objects and twenty four PSL relationships allow for simple but non-trivial examples to be implemented. The number of different instances of each of these eight objects, and therefore the number of relationships connecting pairs of objects, is limited only by the size of the memory allocated for the data base (which can be increased by minor changes in the schema). More PSL objects and relationships can be incorporated into the system as will be discussed in the enhancements section of this chapter. Thus, the potential for realization of the first goal is considerable.

The second goal, the system must be user friendly, was also realized. The student will have an EXEC to run the program so that the many details of JCL and preparation of the data base will be transparent to the user. Once the program has begun the user is presented with some choices in a menu format. Eventually the user is required to enter names and properties of PSL objects, but then is returned to a menu. Each menu requires only the entry of a single letter or digit. The use of menus eliminates the need to

remember large sets of commands, and thus one should be competent in use of the system within a few minutes of use.

Use of the project will help a student become familiar with some of the terms and concepts of PSL/PSA. A student is guided to make correct choices for relationships between PSL objects by the options available in the menu. An incorrect relationship such as OUTPUT DERIVES INPUT is not possible since that choice is not available in the menu. The object types are included in the list of relationships available and the student need only supply the names of the objects to be included in the particular relationship. In this way goal three, the system should aid the student learn about PSL/PSA, is achieved.

The fourth goal, the system should be convenient, is realized since it was possible to incorporate the entire system in a module to run under CMS. The system can be run from any terminal and, as was stated previously, only the name of an EXEC must be typed in to get the system started. A printed report of the results of a session can be obtained by running the system at a hard-copy terminal or by having the history of a session sent to a file at the users virtual reader and later retrieved at a hard-copy device.

Thus it can be concluded that at least some of the capabilities of PSL/PSL can be implemented using IDMS in an interactive mode with a menu -driven application program. For the PSL objects and relationships that are implemented the system works well and allows insertions, deletions, and modifications. Additional information about each object can be included in the description field of each object record.

This information is available in a data base and need only be retrieved in various combinations and formats to resemble the reports generated by PSA. This project is perhaps too limited in scope to make any further claims of its applicability or value but if the suggestions for enhancements in the next section are carried out there will be an improved basis for comparisons and evaluations.

5.2 Enhancements.

There are many possible enhancements to be made that might improve or enlarge this system. Some of these would require changes in only the application program and others would require changes in the schema and perhaps the DMCL as well as in the application program. The report generation capabilities could be accomplished by modifications to the application program. The report generated by this project most closely resembles the report called the Formatted Problem Statement of PSL/PSA, but other reports could be obtained by retrieving and organizing the information stored in the data base in suitable ways.

Completeness checks could be included as possible outputs from the system. This would involve checking to see if all INPUTS consist of GROUPS and ELEMENTS and that all GROUPS are reducible to ELEMENTS. These kinds of checks could be accomplished by modifications in the application program.

Enhancements that would involve changes in the schema for the data base include increasing the number and/or size

of fields in the records of the data base and incorporating more of the PSL objects and relationships into the schema. INTERFACE, which might represent a department which interacts with some part of a software system, and SET, which might correspond to a file (such as a master payroll file) are two objects which could be incorporated into the system. These two objects would make possible several more relationships and thus increase the generality and applicability of the system.

The organization of the system could perhaps be patterned after that of PSL/PSA. The PSL objects are grouped in classes based on the aspect of the target system which they describe. Those classes are System Flow, System Structure, Data Structure, Data Derivation, System Size and Volume, System Dynamics, System Properties, and Project Management. Each of the above classes of objects could perhaps be stored in a different AREA of the data base to provide a greater degree of autonomy.

Other enhancements might be to include the complementary relationships in such a way that the user need not be concerned about entering both relationships. For example if the relationship PROCESS RECEIVES INPUT is entered then the relationship INPUT IS RECEIVED BY PROCESS is automatically entered. A scheme for doing this was discussed in chapter 3.

Bibliography

1. Bell, Thomas E., Bixler, David C., Dyer, Margaret E., "An Extendable Approach to Computer-Aided Software Requirements Engineering", IEEE Transactions on Software Engineering, Vol. SE-3, No. 1, January 1977
2. Computing Center, "User's Guide to IDMS in CMS", Kansas State University, July 1, 1983
3. Conrow, Kenneth, The CMS Cookbook, Computing Center, Kansas State University, February 1980
- Deveaux, D., "Users Guide to PSL-PSA", ISDOS Working Paper No. 52, Department of Industrial Engineering, The University of Michigan, November 1971
5. Hamilton, Margaret, Zellin, Saydean, "Higher Order Software - A Methodology for Defining Software", IEEE Transactions on Software Engineering, March 1976
6. Heninger, Kathryn L., "Specifying Software Requirements for Complex Systems: New Techniques and Their Application", IEEE Transactions on Software Engineering, Vol. SE-6, No. 1, January 1980
7. ISDOS Project, "Problem Statement Language (PSL) Introduction And Users Manuel, Department of Industrial

And Operations Engineering, University of Michigan, May 1977

8. Lamie, Edwards L. PL/1 Programing, Wadsworth Publishing Company, Belmont, California, 1982
9. Perrins, Matt, "Software System Description and Analysis With QBE - An Experiment Based on PSL/PSA, Software Engineering Exchange, Vol. 2, No. 3, April 1980
10. Perron, Bob, et al, "Concepts and Facilities", Cullinane Corporation, Wellesby, Mass. 1977
11. Ross, Douglas T. and Schoman, Kenneth C. Jr., "Structured Analysis for Requirements Definition", IEEE Transactions on Sofrware Engineering, Vol. SE-3, No. 1, January 1977
12. Ross, Douglas T., "Structured Analysis (SA): A Language for Communicating Ideas, IEEE Transactions on Software Engineering, Vol. SE-3, No. 1, January 1977
13. Tiechrow, Daniel, and Sayani, Hasan, "Automation of System Building", Datamation, August 15, 1971
14. Tiechroew, Daniel, and Hershey, Ernest A. III, "PSL/PSA: A Computer-Aided Technique for Structured Documentation And Analysis of Information Processing Systems", IEEE Transactions on Software Engineering, Vol. SE-3, No. 1,

January 1977

15. Wig, Eldon D., "PSL/PSA Primer", Department of Computational Science, University of Saskatchewan, November 1978
16. Zelkowitz, Marvin V., "Perspectives on Software Engineering", Computing Surveys, Vol. 10, No. 2, June 1978

APPENDIX A
User's Manual

APPENDIX A

User Manual

Before the program is used the first time the data base files must be initialized using the EXEC PSLSTART. If the program "crashes" during execution the EXEC PSLSTART can be used to reinitialize the data base files and start over.

To restart the program type PSLPSA. An EXEC readies the environment for use by the system and starts the program. This process requires some time and several messages are printed which should be ignored. See the sample session in Appendix C. The program begins with a welcome message and then a menu is presented as shown in figure 3. Choices 1,2,and 3 lead to sub-menus. Each sub-menu has an option to return to the main menu. Choice number 4 prints a report on the contents of the data base and then returns to the main menu. Choice number 5 terminates the program. The usual sequence for use of the program would thus be:

1. Make a choice from the main menu.
2. Make a choice from the sub-menu.
3. Enter the required information.
4. Repeat steps 2 and 3 as necessary.
5. Return to the main menu.
6. Repeat steps 1 through 4 as necessary.
7. Print the report.
8. Exit the program.

All choices and responses typed in by the user must be followed by pressing the RETURN(or ENTER) key.

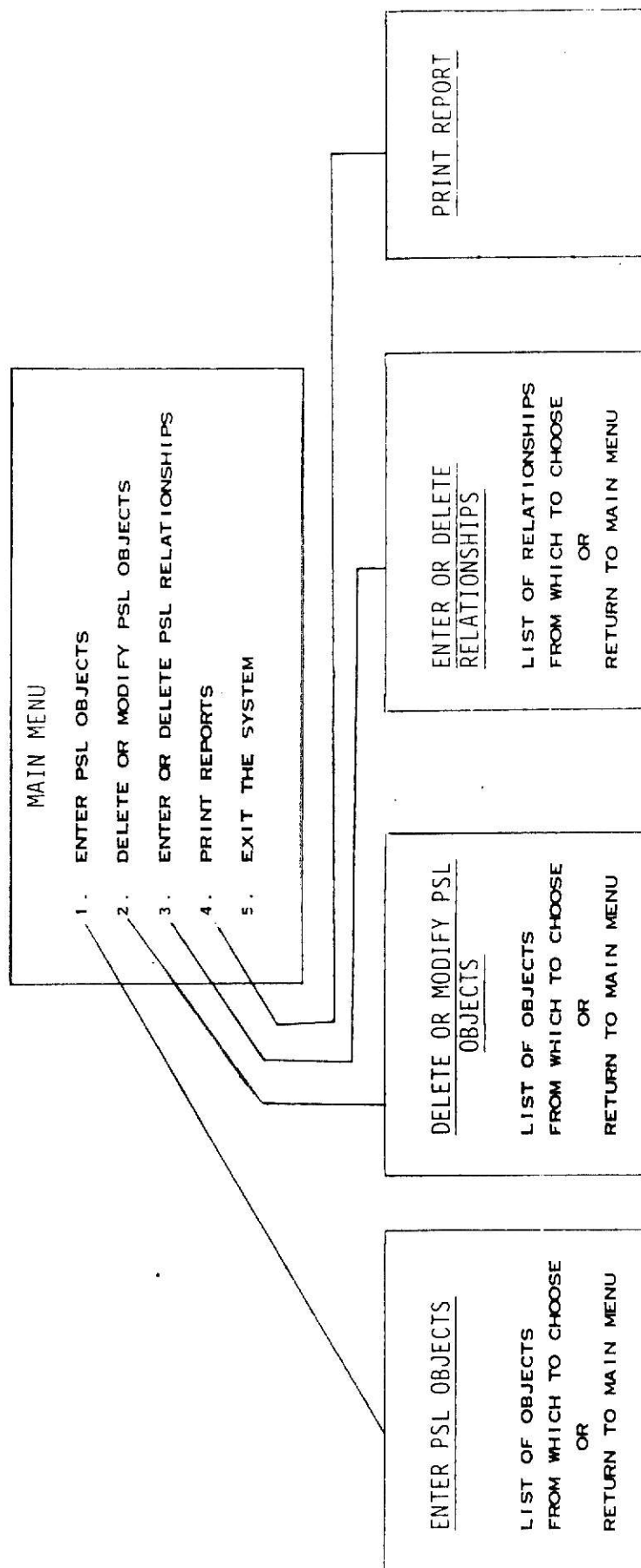


FIGURE 3

In the user's first session with the system, the first step would be to enter some PSL objects, choice number 1. Otherwise the first step should be to print the contents of the data base. This is necessary to set the counters for the data base but also lets the user know what objects and relationships are present.

The "Enter PSL Objects" menu allows the entry of each of the eight types of PSL objects into the data base. To enter an object, say of type INPUT, choose number 1. You are then prompted to enter the name of the INPUT object. After the name is entered, you will be prompted to enter a description of or attributes of the INPUT object. This must be done with 72 characters or less. When finished, you are given the option of entering another object of the same type or returning to the "Enter PSL Objects" menu by typing NONE when prompted for the name of an INPUT. From this menu you may choose to enter objects of other types using the same procedure as above or you may return to the main menu by entering a "9".

The "Delete or Modify PSL Objects" menu allows for corrections of mistakes and/or changing object names and descriptions. You are asked to select the type and then enter the name of the object to be modified or deleted. The next choice is D to delete the object or M to modify the object. If the object is to be modified you are required to enter the new name and description of the object. The new name can of course be the old name if only the description

needs to be changed, but it must nevertheless be typed in again. You are then returned to the "Delete or Modify PSL Objects" menu. When all modifications have been made you can return to the main menu by entering a "9".

The "Enter or Delete Relationships" menu allows for entry or deletion of PSL relationships. You are presented with a menu of 24 possible relationships. Suppose you choose number 1, PROCESS RECEIVES INPUT. After entering your choice you are asked if you wish to I insert or D delete the relationship. You are then prompted for the name of the PROCESS and for the name of the INPUT. Upon entry of the input name you are returned to the menu to choose another relationship or you may choose to return to the main menu by typing a "25".

Once you have entered all objects and relationships you may choose number 4, Print Report, a report of the data or number 5, Exit the Program.

APPENDIX B

EXECs and Procedures

EXECs and Procedures

The compilation steps and initialization of the data dictionary and data base files is accomplished by following the directions given in the User's Guide to IDMS in CMS (2). The processes require much virtual memory and A DISK space and this can be acquired by the following log on.

```
LOGON VMxxx 800K NOIPL#DEF T3350 191 15#IPL VMSTART
```

The next step is to access the IDMS EXECs on the I DISK by the command LINKIDMS.

The data dictionary is initialized by using the command IDMSINIT DDICT.

The schema program SCHEMA1 SCHMA is compiled by the EXEC IDMSCHMA.

The DMCL program SAMPBASE DMCL is compiled by the EXEC IDMSDMCL.

The subschema SSHEMA SUBSC is compiled by the EXEC IDMSUBSC.

The above steps each require only a few minutes each to be completed, however the next step of compiling the PL/I application program APLIPROG DMLP requires about one and one half hour and is accomplished by the EXEC IDMSDMLP.

The data base file DATABASE FILE1 is created and initialized by the EXEC IDMSINIT DBASE. There are three questions for which

answers are required before this step can be completed. The first question requires the name of the DMCL. For this project the answer is SAMPBASE. The second question asks for the number of database files in use. For this project the answer is 1. The third question requires the size of the data base file in pages. For this project the answer is 50.

The system is then started with the EXEC IDMSRUN. Several preliminary messages are printed which may be ignored. No filedefs need to be entered and in fact no response is required of the user until he/she is asked to make a choice from the first menu.

There is a demonstration IDMS package given by the User's Guide to IDMS in CMS(2). Studying the various steps given by this demo package along with the source programs for this project should allow one to replicate or expand upon this project.

The EXEC PSLPSA allows a student to use this system for a PSL/PSA example provided that the following files exist on the students A DISK.

DICTDB DB - The data dictionary created by the EXEC IDMSINIT DDICT
DMSGDB DB - A file created by the EXEC IDMSINIT DDICT
DLOddb DB - A file created by the EXEC IDMSINIT DDICT
SSCHEMA TEXT - Compiled version of SSHEMA SUBSC
SAMPBASE TEXT - Compiled version of SAMPBASE DMCL
DATABASE FILE1 - The data base file created by the
IDMSINIT DBASE step.

PSLPSA EXEC

The files with filetype DB are expanded and compressed by the EXEC PSLPSA but if the other files are stored in compressed form they must first be expanded. Type PSLPSA and the system begins. Several messages are printed and some time elapses, as indicated by the sample session in Appendix C, before the user must respond by making a choice from the first menu.

The EXEC PSLSTART does all that the PSLPSA EXEC does plus initialize the file DATABASE FILE1 for new starts or restarts. It should only be used for starting the system for a new example or restarting an old example from the beginning. The code for these EXECs follows.

PSLSTART EXEC

EXEC LINKIDMS

EXPAND * DB

&STACK SAMPBASE

&STACK 1

&STACK 50

EXEC IDMSINIT DBASE

&STACK

EXEC IDMSRUN APLIPROG

COMPRESS * DB

&EXIT

PSLPSA EXEC

EXEC LINKIDMS

EXPAND * DB

&STACK

EXEC IDMSRUN APLIPROG

COMPRESS * DB

&EXIT

APPENDIX C

Sample Session

SAMPLE SESSION

PSLPSA

EXEC LINKIDMS

I (29B) R/O

EXPAND * DB

\$

\$

\$

\$

\$

\$

\$

\$

\$

EXEC IDMSRUN APLIPROG

ENTER FILEDEFS FOR APPLICATION PROGRAM

EXECUTION BEGINS...

THE FOLLOWING NAMES ARE UNDEFINED:

IDMSCLCK IDMSTRAC IDMSJLRX

THE FOLLOWING NAMES ARE UNDEFINED:

IDMSJNL2 IDMSIOXT IDMSJLRX

WELCOME TO THE KSU IMPLEMENTATION OF PSL/PSA.

YOU HAVE THE FOLLOWING OPTIONS

1 ENTER PSL OBJECTS

2 DELETE OR MODIFY PSL OBJECTS

3 ENTER OR DELETE PSL RELATIONSHIPS

4 PRINT REPORT ON THE CONTENTS OF THE PSL/PSA DATA BASE

5 EXIT THIS PROGRAM

IF YOU ARE BEGINNING YOUR SECOND OR SUBSEQUENT SESSION

YOU MUST CHOOSE NO. 4, PRINT REPORT, TO SET THE COUNTERS
FOR THE DATA BASE

ENTER THE NUMBER FOR YOUR CHOICE

4

PSL/PSA OBJECTS AND RELATIONSHIPS REPORT

OBJECTS OF TYPE PROCESS

1 PAYROLL_PROCESSI

A PAYROLL PROCESSING SYSTEM EXAMPLE

2 MASTER_UPDATE

RECEIVES EMPLOYEE INFORMATION AND UPDATES MASTER FILE

3 CHEQUE_WRITER

RECEIVES TIME CARDS AND GENERATES A CHEQUE

OBJECTS OF TYPE INPUT

1 EMPLOYEE_INFO

EMPLOYEE INFORMATION TO BE USED BY PROCESS MASTER_UPDATE

2 TIME_CARDS

CONTAINS EMPLOYEE NUMBE AND HOURSWORKED

OBJECTS OF TYPE OUTPUT

1 CHEQUE

THE EMPLOYEES PAYCHECK

OBJECTS OF TYPE ENTITY

1 MASTER_FILE

RECORD OF THE MASTER FILE

OBJECTS OF TYPE GROUP

OBJECTS OF TYPE ELEMENT

1 OPERATION_CODE

VALUES ARE 1 THRU 3

2 HOURLY_WAGE

VALUES ARE 5 THRU 20

3 EMPLOYEE_NUMBER

EMPLOYEES IDENTIFICATION NUMBER

4 LAST_NAME

EMPLOYEES LAST NAME

5 INITIALS

EMPLOYEES FIRST AND MIDDLE INITIALS

6 TAX_CODE

7 REGULAR_HOURS

HOURS LESS THAN OR EQUAL TO 40

8 OVERTIME_HOURS

HOUR OVER 40

9 GROSS_PAY

10 INCOME_TAX

11 INSURANCE

UNEMPLOYMENT INSURANCE

12 CANADA_PENSION

13 UNION_DUES

14 NET_PAY

15 RECORD_KEY

16 PROVINCE

OBJECTS OF TYPE EVENT

OBJECTS OF TYPE CONDITION

RELATIONSHIP(S) PROCESS RECEIVES INPUT

MASTER_UPDATE RECEIVES EMPLOYEE_INFO

RELATIONSHIP(S) PROCESS RECEIVES INPUT

CHEQUE_WRITER RECEIVES TIME_CARDS

RELATIONSHIP(S) PROCESS GENERATES OUTPUT

CHEQUE_WRITER GENERATES CHEQUE

PRESS 'ENTER' TO CONTINUE

YOU HAVE THE FOLLOWING OPTIONS

1 ENTER PSL OBJECTS

2 DELETE OR MODIFY PSL OBJECTS

3 ENTER OR DELETE PSL RELATIONSHIPS

4 PRINT REPORT ON THE CONTENTS OF THE PSL/PSA DATA BASE

5 EXIT THIS PROGRAM

IF YOU ARE BEGINNING YOUR SECOND OR SUBSEQUENT SESSION
YOU MUST CHOOSE NO. 4, PRINT REPORT, TO SET THE COUNTERS
FOR THE DATA BASE

ENTER THE NUMBER FOR YOUR CHOICE

3

THE FOLLOWING RELATIONSHIPS MAY BE DEFINED

| | |
|-------------------------------------|-----------------------------------|
| 1-PROCESS_RECEIVES_INPUT | 2-PROCESS_USES_INPUT |
| 3-INPUT_CONSISTS_OF_ELEMENT | 4-INPUT_CONSISTS_OF_GROUP |
| 5-PROCESS_GENERATES_OUTPUT | 6-PROCESS_DERIVES_OUTPUT |
| 7-PROCESS_USES_ELEMENT | 8-PROCESS_UPDATES_ELEMENT |
| 9-PROCESS_DERIVES_ELEMENT | 10-PROCESS_DERIVES_GROUP |
| 11-PROCESS_UPDATES_GROUP | 12-PROCESS_USES_GROUP |
| 13-PROCESS_DERIVES_ENTITY | 14-PROCESS_UPDATES_ENTITY |
| 15-PROCESS_USES_ENTITY | 16-PROCESS_INCEPTION_CAUSES_EVENT |
| 17-PROCESS_TERMINATION_CAUSES_EVENT | |
| | 18-PROCESS_TRIGGERED_BY_EVENT |
| 19-EVENT_WHEN_CONDITION | 20-ENTITY_CONSISTS_OF_GROUP |
| 21-GROUP_CONSISTS_OF_ELEMENT | 22-ENTITY_CONSISTS_OF_ELEMENT |
| 23-OUTPUT_CONSISTS_OF_ELEMENT | 24-OUTPUT_CONSISTS_OF_GROUP |

ENTER THE NUMBER CORRESPONDING TO THE DESIRED
RELATIONSHIP OR 25 TO RETURN TO MENU

3

YOU HAVE CHOSEN THE RELATIONSHIP
INPUT_CONSISTS_OF_ELEMENT.

ENTER THE INPUT NAME

employee_info

ENTER THE ELEMENT NAME

operation_code

DO YOU WISH TO (I) INSERT,

OR (D) DELETE THIS RELATIONSHIP? ENTER YOUR CHOICE

OF I, D

i

RELATIONSHIP EMPLOYEE_INFO CONSISTS OPERATION_CODE HAS BEEN INSERTED

THE FOLLOWING RELATIONSHIPS MAY BE DEFINED

1-PROCESS_RECEIVES_INPUT

2-PROCESS_USES_INPUT

3-INPUT_CONSISTS_OF_ELEMENT

4-INPUT_CONSISTS_OF_GROUP

5-PROCESS_GENERATES_OUTPUT

6-PROCESS_DERIVES_OUTPUT

7-PROCESS_USES_ELEMENT

8-PROCESS_UPDATES_ELEMENT

9-PROCESS_DERIVES_ELEMENT

10-PROCESS_DERIVES_GROUP

11-PROCESS_UPDATES_GROUP

12-PROCESS_USES_GROUP

13-PROCESS_DERIVES_ENTITY

14-PROCESS_UPDATES_ENTITY

15-PROCESS_USES_ENTITY

16-PROCESS_INCEPTION_CAUSES_EVENT

17-PROCESS_TERMINATION_CAUSES_EVENT

18-PROCESS_TRIGGERED_BY_EVENT

19-EVENT_WHEN_CONDITION

20-ENTITY_CONSISTS_OF_GROUP

21-GROUP_CONSISTS_OF_ELEMENT

22-ENTITY_CONSISTS_OF_ELEMENT

23-OUTPUT_CONSISTS_OF_ELEMENT

24-OUTPUT_CONSISTS_OF_GROUP

ENTER THE NUMBER CORRESPONDING TO THE DESIRED

RELATIONSHIP OR 25 TO RETURN TO MENU

4

YOU HAVE CHOSEN THE RELATIONSHIP

INPUT_CONSISTS_OF_GROUP.

ENTER THE INPUT NAME

employee_info

ENTER THE GROUP NAME

required_info

DO YOU WISH TO (I) INSERT,

OR (D) DELETE THIS RELATIONSHIP? ENTER YOUR CHOICE

OF I, D

i

THE FOLLOWING RELATIONSHIPS MAY BE DEFINED

| | |
|-------------------------------------|-----------------------------------|
| 1-PROCESS_RECEIVES_INPUT | 2-PROCESS_USES_INPUT |
| 3-INPUT_CONSISTS_OF_ELEMENT | 4-INPUT_CONSISTS_OF_GROUP |
| 5-PROCESS_GENERATES_OUTPUT | 6-PROCESS_DERIVES_OUTPUT |
| 7-PROCESS_USES_ELEMENT | 8-PROCESS_UPDATES_ELEMENT |
| 9-PROCESS_DERIVES_ELEMENT | 10-PROCESS_DERIVES_GROUP |
| 11-PROCESS_UPDATES_GROUP | 12-PROCESS_USES_GROUP |
| 13-PROCESS_DERIVES_ENTITY | 14-PROCESS_UPDATES_ENTITY |
| 15-PROCESS_USES_ENTITY | 16-PROCESS_INCEPTION_CAUSES_EVENT |
| 17-PROCESS_TERMINATION_CAUSES_EVENT | |
| 18-PROCESS_TRIGGERED_BY_EVENT | |
| 19-EVENT_WHEN_CONDITION | 20-ENTITY_CONSISTS_OF_GROUP |
| 21-GROUP_CONSISTS_OF_ELEMENT | 22-ENTITY_CONSISTS_OF_ELEMENT |
| 23-OUTPUT_CONSISTS_OF_ELEMENT | 24-OUTPUT_CONSISTS_OF_GROUP |

ENTER THE NUMBER CORRESPONDING TO THE DESIRED
RELATIONSHIP OR 25 TO RETURN TO MENU

25

YOU HAVE THE FOLLOWING OPTIONS

1 ENTER PSL OBJECTS

- 2 DELETE OR MODIFY PSL OBJECTS
- 3 ENTER OR DELETE PSL RELATIONSHIPS
- 4 PRINT REPORT ON THE CONTENTS OF THE PSL/PSA DATA BASE
- 5 EXIT THIS PROGRAM

IF YOU ARE BEGINNING YOUR SECOND OR SUBSEQUENT SESSION
YOU MUST CHOOSE NO. 4, PRINT REPORT, TO SET THE COUNTERS
FOR THE DATA BASE

ENTER THE NUMBER FOR YOUR CHOICE

1

THE FOLLOWING PSL OBJECTS MAY BE ENTERED

- | | |
|----------|-------------|
| 1 INPUT | 2 PROCESS |
| 3 OUTPUT | 4 ENTITY |
| 5 GROUP | 6 ELEMENT |
| 7 EVENT | 8 CONDITION |

ENTER THE NUMBER FOR YOUR CHOICE

OR A 9 TO RETURN TO THE MAIN MENU

5

ENTER THE NAME FOR AN GROUP OR 'NONE' IF THERE ARE
NO (OR NO MORE) OBJECTS OF TYPE GROUP

required_info

STATE IN 72 CHARACTERS OR LESS, A DESCRIPTION

OR ANY ATTRIBUTES OF REQUIRED_INFO

GROUP REQUIRED_INFO HAS BEEN ENTERED

ENTER THE NAME FOR AN GROUP OR 'NONE' IF THERE ARE

NO (OR NO MORE) OBJECTS OF TYPE GROUP

a_time_card

STATE IN 72 CHARACTERS OR LESS, A DESCRIPTION

OR ANY ATTRIBUTES OF A_TIME_CARD

consists of employee_name, regular hours and overtime hours

GROUP A_TIME_CARD HAS BEEN ENTERED

ENTER THE NAME FOR AN GROUP OR 'NONE' IF THERE ARE

NO (OR NO MORE) OBJECTS OF TYPE GROUP

employee_name

STATE IN 72 CHARACTERS OR LESS, A DESCRIPTION

OR ANY ATTRIBUTES OF EMPLOYEE_NAME

consists of last name and initials

GROUP EMPLOYEE_NAME HAS BEEN ENTERED

ENTER THE NAME FOR AN GROUP OR 'NONE' IF THERE ARE

NO (OR NO MORE) OBJECTS OF TYPE GROUP

deduction_stub

STATE IN 72 CHARACTERS OR LESS, A DESCRIPTION

OR ANY ATTRIBUTES OF DEDUCTION_STUB

identified by record key

GROUP DEDUCTION_STUB HAS BEEN ENTERED

ENTER THE NAME FOR AN GROUP OR 'NONE' IF THERE ARE

NO (OR NO MORE) OBJECTS OF TYPE GROUP

none

THE FOLLOWING PSL OBJECTS MAY BE ENTERED

| | |
|----------|-------------|
| 1 INPUT | 2 PROCESS |
| 3 OUTPUT | 4 ENTITY |
| 5 GROUP | 6 ELEMENT |
| 7 EVENT | 8 CONDITION |

ENTER THE NUMBER FOR YOUR CHOICE
OR A 9 TO RETURN TO THE MAIN MENU

9

YOU HAVE THE FOLLOWING OPTIONS

- 1 ENTER PSL OBJECTS
- 2 DELETE OR MODIFY PSL OBJECTS
- 3 ENTER OR DELETE PSL RELATIONSHIPS
- 4 PRINT REPORT ON THE CONTENTS OF THE PSL/PSA DATA BASE
- 5 EXIT THIS PROGRAM

3

IF YOU ARE BEGINNING YOUR SECOND OR SUBSEQUENT SESSION
YOU MUST CHOOSE NO. 4, PRINT REPORT, TO SET THE COUNTERS
FOR THE DATA BASE

ENTER THE NUMBER FOR YOUR CHOICE

THE FOLLOWING RELATIONSHIPS MAY BE DEFINED

- | | |
|-------------------------------------|-----------------------------------|
| 1-PROCESS_RECEIVES_INPUT | 2-PROCESS_USES_INPUT |
| 3-INPUT_CONSISTS_OF_ELEMENT | 4-INPUT_CONSISTS_OF_GROUP |
| 5-PROCESS_GENERATES_OUTPUT | 6-PROCESS_DERIVES_OUTPUT |
| 7-PROCESS_USES_ELEMENT | 8-PROCESS_UPDATES_ELEMENT |
| 9-PROCESS_DERIVES_ELEMENT | 10-PROCESS_DERIVES_GROUP |
| 11-PROCESS_UPDATES_GROUP | 12-PROCESS_USES_GROUP |
| 13-PROCESS_DERIVES_ENTITY | 14-PROCESS_UPDATES_ENTITY |
| 15-PROCESS_USES_ENTITY | 16-PROCESS_INCEPTION_CAUSES_EVENT |
| 17-PROCESS_TERMINATION_CAUSES_EVENT | |
| | 18-PROCESS_TRIGGERED_BY_EVENT |
| 19-EVENT_WHEN_CONDITION | 20-ENTITY_CONSISTS_OF_GROUP |

21-GROUP_CONSISTS_OF_ELEMENT 22-ENTITY_CONSISTS_OF_ELEMENT
23-OUTPUT_CONSISTS_OF_ELEMENT 24-OUTPUT_CONSISTS_OF_GROUP
ENTER THE NUMBER CORRESPONDING TO THE DESIRED
RELATIONSHIP OR 25 TO RETURN TO MENU

4

YOU HAVE CHOSEN THE RELATIONSHIP
INPUT_CONSISTS_OF_GROUP.

ENTER THE INPUT NAME

employee_info

ENTER THE GROUP NAME

required_info

DO YOU WISH TO (I) INSERT,

OR (D) DELETE THIS RELATIONSHIP? ENTER YOUR CHOICE

OF I, D

i

RELATIONSHIP EMPLOYEE_INFO CONSISTS REQUIRED_INFO HAS BEEN INSERTED
THE FOLLOWING RELATIONSHIPS MAY BE DEFINED

| | |
|-------------------------------------|-----------------------------------|
| 1-PROCESS_RECEIVES_INPUT | 2-PROCESS_USES_INPUT |
| 3-INPUT_CONSISTS_OF_ELEMENT | 4-INPUT_CONSISTS_OF_GROUP |
| 5-PROCESS_GENERATES_OUTPUT | 6-PROCESS_DERIVES_OUTPUT |
| 7-PROCESS_USES_ELEMENT | 8-PROCESS_UPDATES_ELEMENT |
| 9-PROCESS_DERIVES_ELEMENT | 10-PROCESS_DERIVES_GROUP |
| 11-PROCESS_UPDATES_GROUP | 12-PROCESS_USES_GROUP |
| 13-PROCESS_DERIVES_ENTITY | 14-PROCESS_UPDATES_ENTITY |
| 15-PROCESS_USES_ENTITY | 16-PROCESS_INCEPTION_CAUSES_EVENT |
| 17-PROCESS_TERMINATION_CAUSES_EVENT | |

4

18-PROCESS_TRIGGERED_BY_EVENT

19-EVENT_WHEN_CONDITION 20-ENTITY_CONSISTS_OF_GROUP
21-GROUP_CONSISTS_OF_ELEMENT 22-ENTITY_CONSISTS_OF_ELEMENT
23-OUTPUT_CONSISTS_OF_ELEMENT 24-OUTPUT_CONSISTS_OF_GROUP
ENTER THE NUMBER CORRESPONDING TO THE DESIRED
RELATIONSHIP OR 25 TO RETURN TO MENU
YOU HAVE CHOSEN THE RELATIONSHIP
INPUT_CONSISTS_OF_GROUP.

ENTER THE INPUT NAME

time_cards

ENTER THE GROUP NAME

a_time_card

DO YOU WISH TO (I) INSERT,

OR (D) DELETE THIS RELATIONSHIP? ENTER YOUR CHOICE
OF I, D

i

RELATIONSHIP TIME_CARDS CONSISTS A_TIME_CARD HAS BEEN INSERTED
THE FOLLOWING RELATIONSHIPS MAY BE DEFINED

| | |
|-------------------------------------|-----------------------------------|
| 1-PROCESS_RECEIVES_INPUT | 2-PROCESS_USES_INPUT |
| 3-INPUT_CONSISTS_OF_ELEMENT | 4-INPUT_CONSISTS_OF_GROUP |
| 5-PROCESS_GENERATES_OUTPUT | 6-PROCESS_DERIVES_OUTPUT |
| 7-PROCESS_USES_ELEMENT | 8-PROCESS_UPDATES_ELEMENT |
| 9-PROCESS_DERIVES_ELEMENT | 10-PROCESS_DERIVES_GROUP |
| 11-PROCESS_UPDATES_GROUP | 12-PROCESS_USES_GROUP |
| 13-PROCESS_DERIVES_ENTITY | 14-PROCESS_UPDATES_ENTITY |
| 15-PROCESS_USES_ENTITY | 16-PROCESS_INCEPTION_CAUSES_EVENT |
| 17-PROCESS_TERMINATION_CAUSES_EVENT | |

18-PROCESS_TRIGGERED_BY_EVENT

19-EVENT_WHEN_CONDITION 20-ENTITY_CONSISTS_OF_GROUP
21-GROUP_CONSISTS_OF_ELEMENT 22-ENTITY_CONSISTS_OF_ELEMENT
23-OUTPUT_CONSISTS_OF_ELEMENT 24-OUTPUT_CONSISTS_OF_GROUP
ENTER THE NUMBER CORRESPONDING TO THE DESIRED
RELATIONSHIP OR 25 TO RETURN TO MENU

25

YOU HAVE THE FOLLOWING OPTIONS

1 ENTER PSL OBJECTS
2 DELETE OR MODIFY PSL OBJECTS
3 ENTER OR DELETE PSL RELATIONSHIPS
4 PRINT REPORT ON THE CONTENTS OF THE PSL/PSA DATA BASE
5 EXIT THIS PROGRAM

IF YOU ARE BEGINNING YOUR SECOND OR SUBSEQUENT SESSION
YOU MUST CHOOSE NO. 4, PRINT REPORT, TO SET THE COUNTERS
FOR THE DATA BASE

ENTER THE NUMBER FOR YOUR CHOICE

2

THE FOLLOWING OBJECT TYPES MAY BE MODIFIED
OR DELETED.

WARNING!!!

DO NOT ATTEMPT TO DELETE AN OBJECT IF YOU HAVE ALREADY
ENTERED A RELATIONSHIP USING THIS OBJECT UNLESS YOU
FIRST DELETE THE RELATIONSHIP

1 INPUT 2 PROCESS
3 OUTPUT 4 ENTITY

5 GROUP

6 ELEMENT

7 EVENT

8 CONDITION

ENTER THE NUMBER FOR YOUR CHOICE OR '9' TO GET BACK
TO MAIN MENU

2

ENTER THE NAME OF THE PROCESS TO MODIFIED
OR DELETED

PAYROLL_PROCESSI

DO YOU WISH TO DELETE (D) OR MODIFY (M) THIS OBJECT?

M

ENTER THE (NEW) NAME FOR THE PROCESS

PAYROLL_PROCESS

STATE IN 72 CHARACTERS OR LESS, A DESCRIPTION OR
ANY ATTRIBUTES OF PAYROLL_PROCESS

THIS IS THE MAIN PROCESS FOR A PAYROLL PROCESSING SYSTEM

PROCESS PAYROLL_PROCESS HAS BEEN MODIFIED
THE FOLLOWING OBJECT TYPES MAY BE MODIFIED
OR DELETED.

WARNING!!!

DO NOT ATTEMPT TO DELETE AN OBJECT IF YOU HAVE ALREADY
ENTERED A RELATIONSHIP USING THIS OBJECT UNLESS YOU
FIRST DELETE THE RELATIONSHIP

1 INPUT

2 PROCESS

3 OUTPUT

4 ENTITY

5 GROUP

6 ELEMENT

7 EVENT

8 CONDITION

ENTER THE NUMBER FOR YOUR CHOICE OR '9' TO GET BACK
TO MAIN MENU

9

YOU HAVE THE FOLLOWING OPTIONS

1 ENTER PSL OBJECTS

2 DELETE OR MODIFY PSL OBJECTS

3 ENTER OR DELETE PSL RELATIONSHIPS

4 PRINT REPORT ON THE CONTENTS OF THE PSL/PSA DATA BASE

5 EXIT THIS PROGRAM

IF YOU ARE BEGINNING YOUR SECOND OR SUBSEQUENT SESSION
YOU MUST CHOOSE NO. 4, PRINT REPORT, TO SET THE COUNTERS
FOR THE DATA BASE

ENTER THE NUMBER FOR YOUR CHOICE

3

THE FOLLOWING RELATIONSHIPS MAY BE DEFINED

| | |
|-------------------------------------|-----------------------------------|
| 1-PROCESS_RECEIVES_INPUT | 2-PROCESS_USES_INPUT |
| 3-INPUT_CONSISTS_OF_ELEMENT | 4-INPUT_CONSISTS_OF_GROUP |
| 5-PROCESS_GENERATES_OUTPUT | 6-PROCESS_DERIVES_OUTPUT |
| 7-PROCESS_USES_ELEMENT | 8-PROCESS_UPDATES_ELEMENT |
| 9-PROCESS_DERIVES_ELEMENT | 10-PROCESS_DERIVES_GROUP |
| 11-PROCESS_UPDATES_GROUP | 12-PROCESS_USES_GROUP |
| 13-PROCESS_DERIVES_ENTITY | 14-PROCESS_UPDATES_ENTITY |
| 15-PROCESS_USES_ENTITY | 16-PROCESS_INCEPTION_CAUSES_EVENT |
| 17-PROCESS_TERMINATION_CAUSES_EVENT | |
| | 18-PROCESS_TRIGGERED_BY_EVENT |
| 19-EVENT_WHEN_CONDITION | 20-ENTITY_CONSISTS_OF_GROUP |

21-GROUP_CONSISTS_OF_ELEMENT 22-ENTITY_CONSISTS_OF_ELEMENT
23-OUTPUT_CONSISTS_OF_ELEMENT 24-OUTPUT_CONSISTS_OF_GROUP
ENTER THE NUMBER CORRESPONDING TO THE DESIRED
RELATIONSHIP OR 25 TO RETURN TO MENU

22

YOU HAVE CHOSEN THE RELATIONSHIP

ENTITY_CONSISTS_OF_ELEMENT.

ENTER THE ENTITY NAME

MASTER_FILE

ENTER THE ELEMENT NAME

RECORD_KEY

DO YOU WISH TO (I) INSERT,

OR (D) DELETE THIS RELATIONSHIP? ENTER YOUR CHOICE

OF I, D

D

RELATIONSHIP MASTER_FILE CONSISTS RECORD_KEY HAS BEEN DELETED
THE FOLLOWING RELATIONSHIPS MAY BE DEFINED

| | |
|-------------------------------------|-----------------------------------|
| 1-PROCESS_RECEIVES_INPUT | 2-PROCESS_USES_INPUT |
| 3-INPUT_CONSISTS_OF_ELEMENT | 4-INPUT_CONSISTS_OF_GROUP |
| 5-PROCESS_GENERATES_OUTPUT | 6-PROCESS_DERIVES_OUTPUT |
| 7-PROCESS_USES_ELEMENT | 8-PROCESS_UPDATES_ELEMENT |
| 9-PROCESS_DERIVES_ELEMENT | 10-PROCESS_DERIVES_GROUP |
| 11-PROCESS_UPDATES_GROUP | 12-PROCESS_USES_GROUP |
| 13-PROCESS_DERIVES_ENTITY | 14-PROCESS_UPDATES_ENTITY |
| 15-PROCESS_USES_ENTITY | 16-PROCESS_INCEPTION_CAUSES_EVENT |
| 17-PROCESS_TERMINATION_CAUSES_EVENT | |
| | 18-PROCESS_TRIGGERED_BY_EVENT |

19-EVENT_WHEN_CONDITION 20-ENTITY_CONSISTS_OF_GROUP
21-GROUP_CONSISTS_OF_ELEMENT 22-ENTITY_CONSISTS_OF_ELEMENT
23-OUTPUT_CONSISTS_OF_ELEMENT 24-OUTPUT_CONSISTS_OF_GROUP

ENTER THE NUMBER CORRESPONDING TO THE DESIRED
RELATIONSHIP OR 25 TO RETURN TO MENU

25

YOU HAVE THE FOLLOWING OPTIONS

- 1 ENTER PSL OBJECTS
- 2 DELETE OR MODIFY PSL OBJECTS
- 3 ENTER OR DELETE PSL RELATIONSHIPS
- 4 PRINT REPORT ON THE CONTENTS OF THE PSL/PSA DATA BASE
- 5 EXIT THIS PROGRAM

IF YOU ARE BEGINNING YOUR SECOND OR SUBSEQUENT SESSION
YOU MUST CHOOSE NO. 4, PRINT REPORT, TO SET THE COUNTERS
FOR THE DATA BASE

ENTER THE NUMBER FOR YOUR CHOICE

2

THE FOLLOWING OBJECT TYPES MAY BE MODIFIED
OR DELETED.

WARNING!!!

DO NOT ATTEMPT TO DELETE AN OBJECT IF YOU HAVE ALREADY
ENTERED A RELATIONSHIP USING THIS OBJECT UNLESS YOU
FIRST DELETE THE RELATIONSHIP

- | | |
|----------|-----------|
| 1 INPUT | 2 PROCESS |
| 3 OUTPUT | 4 ENTITY |
| 5 GROUP | 6 ELEMENT |

7 EVENT

8 CONDITION

ENTER THE NUMBER FOR YOUR CHOICE OR '9' TO GET BACK
TO MAIN MENU

6

ENTER THE NAME OF THE ELEMENT TO BE MODIFIED
OR DELETED

RECORD_KEY

DO YOU WISH TO DELETE(D) OR MODIFY(M)
THIS OBJECT

D

ELEMENT RECORD_KEY HAS BEEN DELETED

THE FOLLOWING OBJECT TYPES MAY BE MODIFIED
OR DELETED.

WARNING!!!

DO NOT ATTEMPT TO DELETE AN OBJECT IF YOU HAVE ALREADY
ENTERED A RELATIONSHIP USING THIS OBJECT UNLESS YOU
FIRST DELETE THE RELATIONSHIP

1 INPUT

2 PROCESS

3 OUTPUT

4 ENTITY

5 GROUP

6 ELEMENT

7 EVENT

8 CONDITION

ENTER THE NUMBER FOR YOUR CHOICE OR '9' TO GET BACK
TO MAIN MENU

9

YOU HAVE THE FOLLOWING OPTIONS

1 ENTER PSL OBJECTS

2 DELETE OR MODIFY PSL OBJECTS

3 ENTER OR DELETE PSL RELATIONSHIPS

4 PRINT REPORT ON THE CONTENTS OF THE PSL/PSA DATA BASE

5 EXIT THIS PROGRAM

IF YOU ARE BEGINNING YOUR SECOND OR SUBSEQUENT SESSION

YOU MUST CHOOSE NO. 4, PRINT REPORT, TO SET THE COUNTERS
FOR THE DATA BASE

ENTER THE NUMBER FOR YOUR CHOICE

4

PSL/PSA OBJECTS AND RELATIONSHIPS REPORT

OBJECTS OF TYPE PROCESS

1 PAYROLL_PROCESS

THIS IS THE MAIN PROCESS FOR A PAYROLL PROCESSING SYSTEM

2 MASTER_UPDATE

RECEIVES EMPLOYEE INFORMATION AND UPDATES MASTER FILE

3 CHEQUE_WRITER

RECEIVES TIME CARDS AND GENERATES A CHEQUE

OBJECTS OF TYPE INPUT

1 EMPLOYEE_INFO

EMPLOYEE INFORMATION TO BE USED BY PROCESS MASTER_UPDATE

2 TIME_CARDS

CONTAINS EMPLOYEE NUMBE AND HOURSWORKED

OBJECTS OF TYPE OUTPUT

1 CHEQUE

THE EMPLOYEES PAYCHECK

OBJECTS OF TYPE ENTITY

1 MASTER_FILE

RECORD OF THE MASTER FILE

OBJECTS OF TYPE GROUP

1 REQUIRED_INFO

2 A_TIME_CARD

CONSISTS OF EMPLOYEE_NAME, REGULAR HOURS AND OVERTIME HOURS

3 EMPLOYEE_NAME

CONSISTS OF LAST NAME AND INITIALS

4 DEDUCTION_STUB

IDENTIFIED BY RECORD KEY

OBJECTS OF TYPE ELEMENT

1 OPERATION_CODE

VALUES ARE 1 THRU 3

2 HOURLY_WAGE

VALUES ARE 5 THRU 20

3 EMPLOYEE_NUMBER

EMPLOYEES IDENTIFICATION NUMBER

4 LAST_NAME

EMPLOYEES LAST NAME

5 INITIALS

EMPLOYEES FIRST AND MIDDLE INITIALS

6 TAX_CODE

7 REGULAR_HOURS
HOURS LESS THAN OR EQUAL TO 40

8 OVERTIME_HOURS
HOUR OVER 40

9 GROSS_PAY

10 INCOME_TAX

11 INSURANCE
UNEMPLOYMENT INSURANCE

12 CANADA_PENSION

13 UNION_DUES

14 NET_PAY

15 PROVINCE

OBJECTS OF TYPE EVENT

OBJECTS OF TYPE CONDITION

RELATIONSHIP(S) PROCESS RECEIVES INPUT

MASTER_UPDATE RECEIVES EMPLOYEE_INFO

RELATIONSHIP(S) PROCESS RECEIVES INPUT

CHEQUE_WRITER RECEIVES TIME_CARDS

RELATIONSHIP(S) PROCESS GENERATES OUTPUT

CHEQUE_WRITER GENERATES CHEQUE

RELATIONSHIP(S) INPUT CONSISTS OF ELEMENT

EMPLOYEE_INFO CONSISTS OF OPERATION_CODE

RELATIONSHIP(S) INPUT CONSISTS OF GROUP
 EMPLOYEE_INFO CONSISTS OF REQUIRED_INFO
 RELATIONSHIP(S) INPUT CONSISTS OF GROUP
 TIME_CARDS CONSISTS OF A_TIME_CARD
 RELATIONSHIP(S) OUTPUT CONSISTS OF ELEMENT
 CHEQUE CONSISTS OF NET_PAY
 RELATIONSHIP(S) GROUP CONSISTS OF ELEMENT
 A_TIME_CARD CONSISTS OF EMPLOYEE_NUMBER
 A_TIME_CARD CONSISTS OF REGULAR_HOURS
 A_TIME_CARD CONSISTS OF OVERTIME_HOURS
 RELATIONSHIP(S) GROUP CONSISTS OF ELEMENT
 EMPLOYEE_NAME CONSISTS OF LAST_NAME
 EMPLOYEE_NAME CONSISTS OF INITIALS
 RELATIONSHIP(S) GROUP CONSISTS OF ELEMENT
 DEDUCTION_STUB CONSISTS OF GROSS_PAY
 DEDUCTION_STUB CONSISTS OF INCOME_TAX
 DEDUCTION_STUB CONSISTS OF INSURANCE
 DEDUCTION_STUB CONSISTS OF CANADA_PENSION
 DEDUCTION_STUB CONSISTS OF UNION_DUES
 DEDUCTION_STUB CONSISTS OF NET_PAY
 RELATIONSHIP(S) ENTITY CONSISTS OF ELEMENT
 MASTER_FILE CONSISTS OF EMPLOYEE_NUMBER
 MASTER_FILE CONSISTS OF HOURLY_WAGE
 MASTER_FILE CONSISTS OF TAX_CODE
 RELATIONSHIP(S) ENTITY CONSISTS OF GROUP
 MASTER_FILE CONSISTS OF EMPLOYEE_NAME
 RELATIONSHIP(S) OUTPUT CONSISTS OF GROUP

CHEQUE CONSISTS OF EMPLOYEE_NAME
CHEQUE CONSISTS OF DEDUCTION_STUB
PRESS 'ENTER' TO CONTINUE

YOU HAVE THE FOLLOWING OPTIONS

- 1 ENTER PSL OBJECTS
- 2 DELETE OR MODIFY PSL OBJECTS
- 3 ENTER OR DELETE PSL RELATIONSHIPS
- 4 PRINT REPORT ON THE CONTENTS OF THE PSL/PSA DATA BASE
- 5 EXIT THIS PROGRAM

IF YOU ARE BEGINNING YOUR SECOND OR SUBSEQUENT SESSION
YOU MUST CHOOSE NO. 4, PRINT REPORT, TO SET THE COUNTERS
FOR THE DATA BASE

ENTER THE NUMBER FOR YOUR CHOICE

5

END OF PROGRAM

COMPRESS * DB

\$

\$

\$

\$

\$

\$

\$

\$

\$

\$

\$

\$

\$

R;

APPENDIX D
Source Code Listings

SCHEMA1 SCHMA

* * SCHEMA DESCRIPTION STATEMENTS *

* *****

*

SCHEMA DESCRIPTION.

SCHEMA NAME IS SCHEMA1.

DATE. 10/06/82.

INSTALLATION. CULLINANE CORPORATION
3250 WEST MARKET STREET
AKRON, OHIO 44313

REMARKS. THIS IS THE IDMS SCHEMA FOR PSL-PSA SIMULATED IN IDMS.

*

* *****

* * FILE DESCRIPTION STATEMENTS *

* *****

*

FILE DESCRIPTION.

FILE NAME IS IDMS-FILE1 ASSIGN TO SYS010

DEVICE TYPE IS 3330.

FILE NAME IS JOURNAL ASSIGN TO SYS009

DEVICE TYPE IS 2400.

* * AREA DESCRIPTION STATEMENTS *

AREA DESCRIPTION.

AREA NAME IS PSL-PSA

RANGE IS 1002 THRU 1051

WITHIN FILE IDMS-FILE1 FROM 1 THRU 50.

* * RECORD DESCRIPTION STATEMENTS *

RECORD DESCRIPTION.

| RECORD | NAME | INPUT-OWNER | |
|--------|------|-------------|--|
|--------|------|-------------|--|

RECORD ID 100 .

LOCATION MODE CALC USING INPUT-OWNER-NO DUPLICATES LAST.

WITHIN PSL-PSA AREA.

05 INPUT-OWNER-NO PIC X(4).

05 INPUT-OWNER-NAME PIC X(16).

05 INO-ATTRIBUTES PIC X (72) .

RECORD NAME INPUT .

RECORD ID 101.

LOCATION MODE CALC USING INPUT-NO DUPLICATES LAST.

WITHIN PSL-PSA AREA.

05 INPUT-NO PIC X(4).

05 INPUT-NAME PIC X(16).

05 IN-ATTRIBUTES PIC X(72).

RECORD NAME PROCESS-OWNER .

RECORD ID 200.

LOCATION MODE CALC USING PROCESS-OWNER-NO DUPLICATES LAST.

WITHIN PSL-PSA AREA.

05 PROCESS-OWNER-NO PIC X(4).

05 PROCESSOWNER-NAME PIC X(16).

05 PRO-ATTRIBUTES PIC X(72).

RECORD NAME PROCESS .

RECORD ID 201 .

LOCATION MODE CALC USING PROCESS-NO DUPLICATES LAST.

WITHIN PSL-PSA AREA .

05 PROCESS-NO PIC X(4).

05 PROCESS-NAME PIC X(16).
05 PR-ATTRIBUTES PIC X(72).

RECORD NAME OUTPUT-OWNER .
RECORD ID 300 .
LOCATION MODE CALC USING OUTPUT-OWNER-NO DUPLICATES LAST.
WITHIN PSL-PSA AREA .

05 OUTPUT-OWNER-NO PIC X(4).
05 OUTPUTOWNER-NAME PIC X(16).
05 OUO-ATTRIBUTES PIC X(72).

RECORD NAME OUTPUT .
RECORD ID 301.
LOCATION MODE CALC USING OUTPUT-NO DUPLICATES LAST.
WITHIN PSL-PSA AREA .

05 OUTPUT-NO PIC X(4).
05 OUTPUT-NAME PIC X(16).
05 OU-ATTRIBUTES PIC X(72).

RECORD NAME GROUP-OWNER .
RECORD ID 400 .
LOCATION MODE CALC USING GROUP-OWNER-NO DUPLICATES LAST.
WITHIN PSL-PSA AREA .

05 GROUP-OWNER-NO PIC X(4).
05 GROUP-OWNER-NAME PIC X(16).
05 GRO-ATTRIBUTES PIC X(72).

RECORD NAME GROUP .
RECORD ID 401 .

LOCATION MODE IS CALC USING GROUP-NO DUPLICATES LAST.
WITHIN PSL-PSA AREA .

05 GROUP-NO PIC X(4).
05 GROUP-NAME PIC X(16).
05 GR-ATTRIBUTES PIC X(72).

RECORD NAME ELEMENT .
RECORD ID 501 .

LOCATION MODE CALC USING ELEMENT-NO DUPLICATES LAST.
WITHIN PSL-PSA AREA .

05 ELEMENT-NO PIC X(4).
05 ELEMENT-NAME PIC X(16).
05 EL-ATTRIBUTES PIC X(72).

RECORD NAME ENTITY .
RECORD ID 601 .

LOCATION MODE IS CALC USING ENTITY-NO DUPLICATES LAST.
WITHIN PSL-PSA AREA .

05 ENTITY-NO PIC X(4) .
05 ENTITY-NAME PIC X(16) .
05 EN-ATTRIBUTES PIC X(72) .

RECORD NAME EVENT .
RECORD ID 700 .
LOCATION MODE CALC USING EVENT-NO DUPLICATES LAST.
WITHIN PSL-PSA AREA .

05 EVENT-NO PIC X(4) .
05 EVENT-NAME PIC X(16) .
05 EV-ATTRIBUTES PIC X(72) .

RECORD NAME CONDITION .
RECORD ID 800 .
LOCATION MODE CALC USING CONDITION-NO DUPLICATES LAST.
WITHIN PSL-PSA AREA .

05 CONDITION-NO PIC X(4) .
05 CONDITION-NAME PIC X(16) .
05 CO-ATTRIBUTES PIC X(72) .

RECORD NAME ENTITY-OWNER.
RECORD ID 600.

LOCATION MODE IS CALC USING ENTITY-OWNER-NO DUPLICATES LAST.

WITHIN PSL-PSA AREA.

| | |
|----------------------|------------|
| 05 ENTITY-OWNER-NO | PIC X(4). |
| 05 ENTITY-OWNER-NAME | PIC X(16). |
| 05 ENO-ATTRIBUTES | PIC X(72). |

RECORD NAME ELEMENT-OWNER.

RECORD ID 500.

LOCATION MODE CALC USING ELEMENT-OWNER-NO DUPLICATES LAST.

WITHIN PSL-PSA AREA.

| | |
|-----------------------|------------|
| 05 ELEMENT-OWNER-NO | PIC X(4). |
| 05 ELEMENT-OWNER-NAME | PIC X(16). |
| 05 ELO-ATTRIBUTES | PIC X(72). |

*

* *****

* * SET DESCRIPTION STATEMENTS *

* *****

*

SET DESCRIPTION.

SET NAME INO-SUBPARTS-IN .

ORDER LAST.

MODE CHAIN LINKED TO PRIOR.
OWNER INPUT-OWNER NEXT POSITION 1
 PRIOR POSITION 2.

MEMBER INPUT NEXT POSITION 7
 PRIOR POSITION 8
 LINKED TO OWNER
 OWNER POSITION 9

OPTIONAL MANUAL.

SET NAME PR-RECEIVES-IN .

ORDER LAST.

MODE CHAIN LINKED TO PRIOR.
OWNER PROCESS NEXT POSITION 28 PRIOR POSITION 29.
MEMBER INPUT NEXT POSITION 1 PRIOR POSITION 2
 LINKED TO OWNER
 OWNER POSITION 3

OPTIONAL MANUAL.

SET NAME IS PR-USES-IN.

ORDER IS LAST.

MODE IS CHAIN LINKED TO PRIOR.
OWNER IS PROCESS NEXT POSITION IS 26

MEMBER IS INPUT

PRIOR POSITION IS 27.
NEXT POSITION IS 4
PRIOR POSITION IS 5
LINKED TO OWNER
OWNER POSITION IS 6
OPTIONAL MANUAL.

SET NAME IS IN-CONSISTS-EL.
ORDER IS LAST.

MODE IS CHAIN

OWNER IS INPUT

MEMBER IS ELEMENT

LINKED TO PRIOR.
NEXT POSITION IS 10
PRIOR POSITION IS 11.
NEXT POSITION IS 13
PRIOR POSITION IS 14
LINKED TO OWNER
OWNER POSITION IS 15
OPTIONAL MANUAL.

SET NAME IS IN-CONSISTS-GR.
ORDER IS LAST.

| | |
|-----------------|-----------------------|
| MODE IS CHAIN | LINKED TO PRIOR. |
| OWNER IS INPUT | NEXT POSITION IS 12 |
| | PRIOR POSITION IS 13. |
| MEMBER IS GROUP | NEXT POSITION IS 19 |
| | PRIOR POSITION IS 20 |
| | LINKED TO OWNER |
| | OWNER POSITION IS 21 |
| | OPTIONAL MANUAL. |

SET NAME IS PRO-SUBPARTS-PR.

ORDER IS LAST.

| | |
|------------------------|----------------------|
| MODE IS CHAIN | LINKED TO PRIOR. |
| OWNER IS PROCESS-OWNER | NEXT POSITION IS 1 |
| | PRIOR POSITION IS 2. |
| MEMBER IS PROCESS | NEXT POSITION IS 1 |
| | PRIOR POSITION IS 2 |
| | LINKED TO OWNER |
| | OWNER POSITION IS 3 |
| | OPTIONAL MANUAL. |

SET NAME IS PR-GENERATES-OU.

ORDER IS LAST.

MODE IS CHAIN

LINKED TO PRIOR.

OWNER IS PROCESS

NEXT POSITION IS 4

PRIOR POSITION IS 5.

MEMBER IS OUTPUT

NEXT POSITION IS 7

PRIOR POSITION IS 8

LINKED TO OWNER

OWNER POSITION IS 9

OPTIONAL MANUAL.

SET NAME IS PR-DERIVES-OU.

ORDER IS LAST.

MODE IS CHAIN

LINKED TO PRIOR.

OWNER IS PROCESS

NEXT POSITION IS 6

PRIOR POSITION IS 7.

MEMBER IS OUTPUT

NEXT POSITION IS 4

PRIOR POSITION IS 5

LINKED TO OWNER

OWNER POSITION IS 6

OPTIONAL MANUAL.

SET NAME IS PR-USES-EL.

ORDER IS LAST.

MODE IS CHAIN

LINKED TO PRIOR.

OWNER IS PROCESS

NEXT POSITION IS 8

PRIOR POSITION IS 9.

MEMBER IS ELEMENT

NEXT POSITION IS 22

PRIOR POSITION IS 23

LINKED TO OWNER

OWNER POSITION IS 24

OPTIONAL MANUAL.

SET NAME IS PR-UPDATES-EL.

ORDER IS LAST.

MODE IS CHAIN

LINKED TO PRIOR.

OWNER IS PROCESS

NEXT POSITION IS 10

PRIOR POSITION IS 11.

MEMBER IS ELEMENT

NEXT POSITION IS 19

PRIOR POSITION IS 20

LINKED TO OWNER

OWNER POSITION IS 21

OPTIONAL MANUAL.

SET NAME IS PR-DERIVES-EL.

ORDER IS LAST.

MODE IS CHAIN

LINKED TO PRIOR.

OWNER IS PROCESS

NEXT POSITION IS 12

PRIOR POSITION IS 13.

MEMBER IS ELEMENT

NEXT POSITION IS 16

PRIOR POSITION IS 17

LINKED TO OWNER

OWNER POSITION IS 18

OPTIONAL MANUAL.

SET NAME IS PR-DERIVES-GR.

ORDER IS LAST.

MODE IS CHAIN

LINKED TO PRIOR.

OWNER IS PROCESS

NEXT POSITION IS 14

PRIOR POSITION IS 15.

MEMBER IS GROUP

NEXT POSITION IS 7

PRIOR POSITION IS 8

LINKED TO OWNER
OWNER POSITION IS 9
OPTIONAL MANUAL.

SET NAME IS PR-UPDATES-GR.

ORDER IS LAST.

MODE IS CHAIN

LINKED TO PRIOR.

OWNER IS PROCESS

NEXT POSITION IS 16

PRIOR POSITION IS 17.

MEMBER IS GROUP

NEXT POSITION IS 4

PRIOR POSITION IS 5

LINKED TO OWNER

OWNER POSITION IS 6

OPTIONAL MANUAL.

SET NAME IS PR-USES-GR.

ORDER IS LAST.

MODE IS CHAIN

LINKED TO PRIOR.

OWNER IS PROCESS

NEXT POSITION IS 18

PRIOR POSITION IS 19.

MEMBER IS GROUP NEXT POSITION IS 1
PRIOR POSITION IS 2
LINKED TO OWNER
OWNER POSITION IS 3
OPTIONAL MANUAL.

SET NAME IS PR-DERIVES-EN.

ORDER IS LAST.

MODE IS CHAIN LINKED TO PRIOR.
OWNER IS PROCESS NEXT POSITION IS 20
PRIOR POSITION IS 21.
MEMBER IS ENTITY NEXT POSITION IS 7
PRIOR POSITION IS 8
LINKED TO OWNER
OWNER POSITION IS 9
OPTIONAL MANUAL.

SET NAME IS PR-UPDATES-EN.

ORDER IS LAST.

MODE IS CHAIN LINKED TO PRIOR.

OWNER IS PROCESS

NEXT POSITION IS 22

PRIOR POSITION IS 23.

MEMBER IS ENTITY

NEXT POSITION IS 4

PRIOR POSITION IS 5

LINKED TO OWNER

OWNER POSITION IS 6

OPTIONAL MANUAL.

SET NAME IS PR-USES-EN.

ORDER IS LAST.

MODE IS CHAIN

LINKED TO PRIOR.

OWNER IS PROCESS

NEXT POSITION IS 24

PRIOR POSITION IS 25.

MEMBER IS ENTITY

NEXT POSITION IS 1

PRIOR POSITION IS 2

LINKED TO OWNER

OWNER POSITION IS 3

OPTIONAL MANUAL.

SET NAME IS PR-INCEPTION-EV.

ORDER IS LAST.

MODE IS CHAIN

OWNER IS PROCESS

MEMBER IS EVENT

LINKED TO PRIOR.

NEXT POSITION IS 30

PRIOR POSITION IS 31.

NEXT POSITION IS 7

PRIOR POSITION IS 8

LINKED TO OWNER

OWNER POSITION IS 9

OPTIONAL MANUAL.

SET NAME IS PR-TERMINATE-EV.

ORDER IS LAST.

MODE IS CHAIN

OWNER IS PROCESS

MEMBER IS EVENT

LINKED TO PRIOR.

NEXT POSITION IS 32

PRIOR POSITION IS 33.

NEXT POSITION IS 4

PRIOR POSITION IS 5

LINKED TO OWNER

OWNER POSITION IS 6

OPTIONAL MANUAL.

SET NAME IS PR-TRIGGERED-EV.

ORDER IS LAST.

MODE IS CHAIN

LINKED TO PRIOR.

OWNER IS PROCESS

NEXT POSITION IS 34

PRIOR POSITION IS 35.

MEMBER IS EVENT

NEXT POSITION IS 1

PRIOR POSITION IS 2

LINKED TO OWNER

OWNER POSITION IS 3

OPTIONAL MANUAL.

SET NAME IS EV-WHEN-CO.

ORDER IS LAST.

MODE IS CHAIN

LINKED TO PRIOR.

OWNER IS EVENT

NEXT POSITION IS 10

PRIOR POSITION IS 11.

MEMBER IS CONDITION

NEXT POSITION IS 1

PRIOR POSITION IS 2

LINKED TO OWNER

OWNER POSITION IS 3

OPTIONAL MANUAL.

SET NAME IS OUC-SUBPARTS-OU.

ORDER IS LAST.

MODE IS CHAIN

LINKED TO PRIOR.

OWNER IS OUTPUT-OWNER

NEXT POSITION IS 1

PRIOR POSITION IS 2.

MEMBER IS OUTPUT

NEXT POSITION IS 1

PRIOR POSITION IS 2

LINKED TO OWNER

OWNER POSITION IS 3

OPTIONAL MANUAL.

SET NAME IS GRO-SUBPARTS-GR.

ORDER IS LAST.

MODE IS CHAIN

LINKED TO PRIOR.

OWNER IS GROUP-OWNER

NEXT POSITION IS 1

PRIOR POSITION IS 2.

MEMBER IS GROUP

NEXT POSITION IS 13

PRIOR POSITION IS 14

LINKED TO OWNER

OWNER POSITION IS 15

OPTIONAL MANUAL.

SET NAME IS EN-CONSISTS-GR.

ORDER IS LAST.

MODE IS CHAIN

LINKED TO PRIOR.

OWNER IS ENTITY

NEXT POSITION IS 13

PRIOR POSITION IS 14.

MEMBER IS GROUP

NEXT POSITION IS 16

PRIOR POSITION IS 17

LINKED TO OWNER

OWNER POSITION IS 18

OPTIONAL MANUAL.

SET NAME IS GR-CONSISTS-EL.

ORDER IS LAST.

MODE IS CHAIN

LINKED TO PRIOR.

OWNER IS GROUP

NEXT POSITION IS 22

PRIOR POSITION IS 23.

MEMBER IS ELEMENT

NEXT POSITION IS 10

PRIOR POSITION IS 11
LINKED TO OWNER
OWNER POSITION IS 12
OPTIONAL MANUAL.

SET NAME IS EN-CONSISTS-EL.

ORDER IS LAST.

| | |
|-------------------|-----------------------|
| MODE IS CHAIN | LINKED TO PRIOR. |
| OWNER IS ENTITY | NEXT POSITION IS 15 |
| | PRIOR POSITION IS 16. |
| MEMBER IS ELEMENT | NEXT POSITION IS 7 |
| | PRIOR POSITION IS 8 |
| | LINKED TO OWNER |
| | OWNER POSITION IS 9 |
| | OPTIONAL MANUAL. |

SET NAME IS OU-CONSISTS-EL.

ORDER IS LAST.

| | |
|-----------------|---------------------|
| MODE IS CHAIN | LINKED TO PRIOR. |
| OWNER IS OUTPUT | NEXT POSITION IS 10 |

MEMBER IS ELEMENT PRIOR POSITION IS 11.
NEXT POSITION IS 1
PRIOR POSITION IS 2
LINKED TO OWNER
OWNER POSITION IS 3
OPTIONAL MANUAL.

SET NAME IS OU-CONSISTS-GR.

ORDER IS LAST.

MODE IS CHAIN LINKED TO PRIOR.
OWNER IS OUTPUT NEXT POSITION IS 12
PRIOR POSITION IS 13.
MEMBER IS GROUP NEXT POSITION IS 10
PRIOR POSITION IS 11
LINKED TO OWNER
OWNER POSITION IS 12
OPTIONAL MANUAL.

SET NAME IS ENO-SUBPARTS-EN.

ORDER IS LAST.

MODE IS CHAIN LINKED TO PRIOR.
OWNER IS ENTITY-OWNER NEXT POSITION IS 1
 PRIOR POSITION IS 2.
MEMBER IS ENTITY NEXT POSITION IS 10
 PRIOR POSITION IS 11
 LINKED TO OWNER
 OWNER POSITION IS 12
 OPTIONAL MANUAL.

SET NAME IS ELO-SUBPARTS-EL.

ORDER IS LAST.

MODE IS CHAIN LINKED TO PRIOR.
OWNER IS ELEMENT-OWNER NEXT POSITION IS 1
 PRIOR POSITION IS 2.
MEMBER IS ELEMENT NEXT POSITION IS 4
 PRIOR POSITION IS 5
 LINKED TO OWNER
 OWNER POSITION IS 6
 OPTIONAL MANUAL.

SSHEMA SUBSC

ADD SUBSCHEMA NAME IS SSHEMA
OF SCHEMA NAME IS SCHEMA1
DMCL NAME IS SAMPBASE
OF SCHEMA NAME IS SCHEMA1 VERSION LOW.
ADD AREA PSL-PSA.
ADD RECORD INPUT-OWNER.
ADD RECORD INPUT.
ADD RECORD PROCESS-OWNER.
ADD RECORD PROCESS.
ADD RECORD OUTPUT-OWNER.
ADD RECORD OUTPUT.
ADD RECORD GROUP-OWNER.
ADD RECORD GROUP.
ADD RECORD ELEMENT.
ADD RECORD ENTITY.
ADD RECORD EVENT.
ADD RECORD CONDITION.
ADD RECORD ENTITY-OWNER.
ADD RECORD ELEMENT-OWNER.
ADD SET INO-SUBPARTS-IN.
ADD SET PR-RECEIVES-IN.
ADD SET PR-USES-IN.
ADD SET IN-CONSISTS-EL.
ADD SET IN-CONSISTS-GR.

ADD SET PRO-SUBPARTS-PR.
ADD SET PR-GENERATES-OU.
ADD SET PR-DERIVES-OU.
ADD SET PR-USES-EL.
ADD SET PR-UPDATES-EL.
ADD SET PR-DERIVES-EL.
ADD SET PR-DERIVES-GR.
ADD SET PR-UPDATES-GR.
ADD SET PR-USES-GR.
ADD SET PR-DERIVES-EN.
ADD SET PR-UPDATES-EN.
ADD SET PR-USES-EN.
ADD SET PR-INCEPTION-EV.
ADD SET PR-TERMINATE-EV.
ADD SET PR-TRIGGERED-EV.
ADD SET EV-WHEN-CO.
ADD SET OUO-SUBPARTS-OU.
ADD SET OU-CONSISTS-EL.
ADD SET OU-CONSISTS-GR.
ADD SET GRO-SUBPARTS-GR.
ADD SET EN-CONSISTS-GR.
ADD SET GR-CONSISTS-EL.
ADD SET EN-CONSISTS-EL.
ADD SET ENO-SUBPARTS-EN.
ADD SET ELO-SUBPARTS-EL.
GENERATE.

SAMPBASE DMCL

DEVICE-MEDIA DESCRIPTION.

DEVICE-MEDIA NAME IS SAMPBASE OF SCHEMA NAME SCHEMA1.

AUTHOR. FRANK HAJEK

DATE. 10/08/82

INSTALLATION. KANSAS STATE UNIVERSITY

REMARKS. THIS IS THE DMCL FOR THE
PSL-PSA SCHEMA.

BUFFER SECTION.

BUFFER NAME IS IDMS-BUFFER

PAGE CONTAINS 496 CHARACTERS

BUFFER CONTAINS 50 PAGES.

AREA SECTION.

COPY PSL-PSA AREA

PAGE-RESERVE CONTAINS 100 CHARACTERS.

APLIPROG DMLP

```

/* AN IMPLEMENTATION OF PSL/PSA */
PSLPROG:PROC OPTIONS(MAIN);
DCL REC_NUM CHAR(4) VAR,
     REC_NAME CHAR(16) VAR,
     REC_TYPE CHAR(16) VAR,
     (I,J) FIXED DECIMAL,
     (COCNT,PRCNT,OUCNT,ENCNT,ELCNT,GRCNT,EVCNT,INCNT) FIXED DECIMAL,
     ATTRIBUTES CHAR(72) VAR,
     CHOICE CHAR(2) VAR,
     OWNER_NAME CHAR(16) VAR,
     SET_NAME CHAR(16) VAR,
     MEMBER_NAME CHAR(16) VAR,
     END_OF_SET CHAR(4) INIT('0307'),
     END_OF_AREA CHAR(4) INIT('0307'),
     OK CHAR(4) INIT('0000');
DCL (IDMS,ABORT) OPTIONS (INTER,ASM) ENTRY;
DCL (SSHEMA SUBSCHEMA, SCHEMA1 SCHEMA VERSION 1)
     MODE (BATCH);
INCLUDE IDMS(SUBSCHEMA_DESCRIPTION);
INCLUDE IDMS(SUBSCHEMA_BINDS);
INCLUDE IDMS(IDMS_STATUS);
READY EXCLUSIVE UPDATE;
CALL IDMS_STATUS;
COCNT = 0;

```

```

PRCNT = 0;
OUCNT = 0;
ENCNT = 0;
GRCNT = 0;
ELCNT = 0;
EVCNT = 0;
INCNT = 0;

PROCESS_OWNER_NO = ' 1';

PROCESSOWNER_NAME = 'PROCESS_OWNER';

PRO_ATTRIBUTES = 'THIS IS A DUMMY RECORD WHICH OWNS ALL PROCESSES';

STORE RECORD(PROCESS_OWNER);

CALL IDMS_STATUS;

DISPLAY('WELCOME TO THE KSU IMPLEMENTATION OF PSL/PSA. ');

MENU:  DISPLAY('YOU HAVE THE FOLLOWING OPTIONS');

DISPLAY('1 ENTER PSL OBJECTS ');

DISPLAY('2 DELETE OR MODIFY PSL OBJECTS');

DISPLAY('3 ENTER OR DELETE  PSL RELATIONSHIPS');

DISPLAY('4 PRINT REPORT ON THE CONTENTS OF THE PSL/PSA DATA BASE');

DISPLAY('5 EXIT THIS PROGRAM');

DISPLAY(' ');

DISPLAY('IF YOU ARE BEGINNING YOUR SECOND OR SUBSEQUENT SESSION ');

DISPLAY('YOU MUST CHOOSE NO. 4, PRINT REPORT, TO SET THE COUNTERS');

DISPLAY(' FOR THE DATA BASE');

DISPLAY(' ');

DISPLAY('ENTER THE NUMBER FOR YOUR CHOICE')REPLY (CHOICE);

SELECT;

    WHEN (CHOICE= '1' ) GO TO ENTER_RECORDS;

```

```

    WHEN (CHOICE= '2' ) GO TO DELETE_RECORDS;
    WHEN (CHOICE= '3' ) GO TO ENTER_SETS;
    WHEN (CHOICE= '4' ) GO TO FIND_RECORDS;
    WHEN (CHOICE= '5' ) GO TO EXIT_PROGRAM;
    OTHERWISE DISPLAY('INVALID CHOICE, TRY AGAIN');
END; /* OF SELECT */

    GO TO MENU;

ENTER_RECORDS:DISPLAY(' THE FOLLOWING PSL OBJECTS MAY BE ENTERED');
DISPLAY('1 INPUT                2 PROCESS ');
DISPLAY('3 OUTPUT                4 ENTITY ');
DISPLAY('5 GROUP                6 ELEMENT ');
DISPLAY('7 EVENT                8 CONDITION');
DISPLAY(' ENTER THE NUMBER FOR YOUR CHOICE');
DISPLAY(' OR A 9 TO RETURN TO THE MAIN MENU')REPLY(CHOICE);
SELECT;

    WHEN (CHOICE = '1' ) GO TO ENTER_INPUT;
    WHEN (CHOICE = '2' ) GO TO ENTER_PROCESS;
    WHEN (CHOICE = '3' ) GO TO ENTER_OUTPUT;
    WHEN (CHOICE = '4' ) GO TO ENTER_ENTITY;
    WHEN (CHOICE = '5' ) GO TO ENTER_GROUP;
    WHEN (CHOICE = '6' ) GO TO ENTER_ELEMENT;
    WHEN (CHOICE = '7' ) GO TO ENTER_EVENT;
    WHEN (CHOICE = '8' ) GO TO ENTER_CONDITION;
    OTHERWISE GO TO MENU;
END; /* OF SELECT */

ENTER_INPUT :
DISPLAY('ENTER THE NAME FOR AN INPUT OR ''NONE'' IF THERE ARE');

```

```

DISPLAY(' NO (OR NO MORE) OBJECTS OF TYPE INPUT')REPLY(REC_NAME);
IF (REC_NAME = 'NONE') THEN GO TO ENTER_RECORDS;

    INCNT = INCNT + 1;

    PUT STRING(INPUT_OWNER_NO) EDIT(INCNT) (F(4));

    INPUT_OWNER_NAME = REC_NAME;

    DISPLAY('STATE IN 72 CHARACTERS OR LESS, A DESCRIPTION  ');
    DISPLAY('OR ANY ATTRIBUTES OF ' || REC_NAME)REPLY(ATTRIBUTES);
    INO_ATTRIBUTES = ATTRIBUTES;

    STORE RECORD (INPUT_OWNER);

    CALL IDMS_STATUS;

    DISPLAY(' INPUT ' || REC_NAME || ' HAS BEEN ENTERED');

    GO TO ENTER_INPUT;

ENTER_PROCESS:

DISPLAY('ENTER THE NAME FOR A PROCESS OR ''NONE'' IF THERE ARE');
DISPLAY(' NO (OR NO MORE) OBJECTS OF TYPE PROCESS')REPLY(REC_NAME);
IF (REC_NAME = 'NONE') THEN GO TO ENTER_RECORDS;

    PRCNT = PRCNT + 1;

    PUT STRING(PROCESS_NO) EDIT(PRCNT) (F(4));

    PROCESS_NAME = REC_NAME;

    DISPLAY('STATE IN 72 CHARACTERS OR LESS, A DESCRIPTION  ');
    DISPLAY('OR ANY ATTRIBUTES OF ' || REC_NAME)REPLY(ATTRIBUTES);
    PR_ATTRIBUTES = ATTRIBUTES;

    STORE RECORD (PROCESS);

    CALL IDMS_STATUS;

    CONNECT RECORD (PROCESS) SET(PRO_SUBPARTS_PR);

    CALL IDMS_STATUS;

    DISPLAY(' PROCESS ' || REC_NAME || ' HAS BEEN ENTERED');

```

```

        GO TO ENTER_PROCESS;

ENTER_OUTPUT :

DISPLAY('ENTER THE NAME FOR AN OUTPUT OR ''NONE'' IF THERE ARE');
DISPLAY(' NO (OR NO MORE) OBJECTS OF TYPE OUTPUT')REPLY(REC_NAME);
IF (REC_NAME = 'NONE') THEN GO TO ENTER_RECORDS;

        OUCNT = OUCNT + 1;

        PUT STRING(OUTPUT_OWNER_NO) EDIT(OUCNT) (F(4));

        OUTPUTOWNER_NAME = REC_NAME;

        DISPLAY('STATE IN 72 CHARACTERS OR LESS, A DESCRIPTION  ');
        DISPLAY('OR ANY ATTRIBUTES OF ' || REC_NAME)REPLY(ATTRIBUTES);
        OUO_ATTRIBUTES = ATTRIBUTES;

        STORE RECORD (OUTPUT_OWNER);

        CALL IDMS_STATUS;

        DISPLAY(' OUTPUT ' || REC_NAME || ' HAS BEEN ENTERED');

        GO TO ENTER_OUTPUT;

ENTER_ENTITY :

DISPLAY('ENTER THE NAME FOR AN ENTITY OR ''NONE'' IF THERE ARE');
DISPLAY(' NO (OR NO MORE) OBJECTS OF TYPE ENTITY')REPLY(REC_NAME);
IF (REC_NAME = 'NONE') THEN GO TO ENTER_RECORDS;

        ENCNT = ENCNT + 1;

        PUT STRING(ENTITY_OWNER_NO) EDIT(ENCNT) (F(4));

        ENTITY_OWNER_NAME = REC_NAME;

        DISPLAY('STATE IN 72 CHARACTERS OR LESS, A DESCRIPTION  ');
        DISPLAY('OR ANY ATTRIBUTES OF ' || REC_NAME)REPLY(ATTRIBUTES);
        ENO_ATTRIBUTES = ATTRIBUTES;

        STORE RECORD (ENTITY_OWNER);

        CALL IDMS_STATUS;

```

```

        DISPLAY(' ENTITY ' || REC_NAME || ' HAS BEEN ENTERED');
        GO TO ENTER_ENTITY;

ENTER_GROUP :
DISPLAY('ENTER THE NAME FOR AN GROUP OR ''NONE'' IF THERE ARE');
DISPLAY(' NO (OR NO MORE) OBJECTS OF TYPE GROUP')REPLY(REC_NAME);
IF (REC_NAME = 'NONE') THEN GO TO ENTER_RECORDS;

        GRCNT = GRCNT + 1;

        PUT STRING(GROUP_OWNER_NO) EDIT(GRCNT) (F(4));

        GROUP_OWNER_NAME = REC_NAME;

        DISPLAY('STATE IN 72 CHARACTERS OR LESS, A DESCRIPTION ');
        DISPLAY('OR ANY ATTRIBUTES OF ' || REC_NAME)REPLY(ATTRIBUTES);
        GRO_ATTRIBUTES = ATTRIBUTES;

        STORE RECORD (GROUP_OWNER);

        CALL IDMS_STATUS;

        DISPLAY(' GROUP ' || REC_NAME || ' HAS BEEN ENTERED');

        GO TO ENTER_GROUP;

ENTER_ELEMENT :
DISPLAY('ENTER THE NAME FOR AN ELEMENT OR ''NONE'' IF THERE ARE');
DISPLAY(' NO (OR NO MORE) OBJECTS OF TYPE ELEMENT')REPLY(REC_NAME);
IF (REC_NAME = 'NONE') THEN GO TO ENTER_RECORDS;

        ELCNT = ELCNT + 1;

        PUT STRING(ELEMENT_OWNER_NO) EDIT(ELCNT) (F(4));

        ELEMENT_OWNER_NAME = REC_NAME;

        DISPLAY('STATE IN 72 CHARACTERS OR LESS, A DESCRIPTION ');
        DISPLAY('OR ANY ATTRIBUTES OF ' || REC_NAME)REPLY(ATTRIBUTES);
        ELO_ATTRIBUTES = ATTRIBUTES;

        STORE RECORD (ELEMENT_OWNER);

```



```

        CALL IDMS_STATUS;

        DISPLAY(' ELEMENT ' || REC_NAME || ' HAS BEEN ENTERED');

        GO TO ENTER_ELEMENT;

ENTER_EVENT :

DISPLAY('ENTER THE NAME FOR AN EVENT OR ''NONE'' IF THERE ARE');
DISPLAY(' NO (OR NO MORE) OBJECTS OF TYPE EVENT')REPLY(REC_NAME);
IF (REC_NAME = 'NONE') THEN GO TO ENTER_RECORDS;

        EVCNT = EVCNT + 1;

        PUT STRING(EVENT_NO) EDIT(EVCNT) (F(4));

        EVENT_NAME = REC_NAME;

        DISPLAY('STATE IN 72 CHARACTERS OR LESS, A DESCRIPTION ');
        DISPLAY('OR ANY ATTRIBUTES OF ' || REC_NAME)REPLY(ATTRIBUTES);
        EV_ATTRIBUTES = ATTRIBUTES;

        STORE RECORD (EVENT);

        CALL IDMS_STATUS;

        DISPLAY(' EVENT ' || REC_NAME || ' HAS BEEN ENTERED');

        GO TO ENTER_EVENT;

ENTER_CONDITION :

DISPLAY('ENTER THE NAME FOR AN CONDITION OR ''NONE'' IF THERE ARE');
DISPLAY('NO (OR NO MORE) OBJECTS OF TYPE CONDITION')REPLY(REC_NAME);
IF (REC_NAME = 'NONE') THEN GO TO ENTER_RECORDS;

        COCNT = COCNT + 1;

        PUT STRING(CONDITION_NO) EDIT(COCNT) (F(4));

        CONDITION_NAME = REC_NAME;

        DISPLAY('STATE IN 72 CHARACTERS OR LESS, A DESCRIPTION ');
        DISPLAY('OR ANY ATTRIBUTES OF ' || REC_NAME)REPLY(ATTRIBUTES);
        CO_ATTRIBUTES = ATTRIBUTES;

```

```

        STORE RECORD (CONDITION);

        CALL IDMS_STATUS;

        DISPLAY(' CONDITION ' || REC_NAME || ' HAS BEEN ENTERED');

        GO TO ENTER_CONDITION;

DELETE_RECORDS: DISPLAY('THE FOLLOWING OBJECT TYPES MAY BE ');

DISPLAY(' MODIFIED OR DELETED. ');

DISPLAY(' WARNING!!! ');

DISPLAY('DO NOT ATTEMPT TO DELETE AN OBJECT IF YOU HAVE ALREADY ');

DISPLAY('ENTERED A RELATIONSHIP USING THIS OBJECT UNLESS YOU');

DISPLAY('FIRST DELETE THE RELATIONSHIP');

DISPLAY('1 INPUT                                2 PROCESS');

DISPLAY('3 OUTPUT                                4 ENTITY ');

DISPLAY('5 GROUP                                6 ELEMENT');

DISPLAY('7 EVENT                                8 CONDITION');

DISPLAY('ENTER THE NUMBER FOR YOUR CHOICE OR ''9'' TO GET BACK ');

DISPLAY('TO MAIN MENU') REPLY(CHOICE);

SELECT;

    WHEN (CHOICE = '1') GO TO DELETE_INPUT;

    WHEN (CHOICE = '2') GO TO DELETE_PROCESS;

    WHEN (CHOICE = '3') GO TO DELETE_OUTPUT;

    WHEN (CHOICE = '4') GO TO DELETE_ENTITY;

    WHEN (CHOICE = '5') GO TO DELETE_GROUP;

    WHEN (CHOICE = '6') GO TO DELETE_ELEMENT;

    WHEN (CHOICE = '7') GO TO DELETE_EVENT;

    WHEN (CHOICE = '8') GO TO DELETE_CONDITION;

    OTHERWISE GO TO MENU;

END; /* OF SELECT */

```

```

DELETE_PROCESS: DISPLAY ('ENTER THE NAME OF THE PROCESS TO MODIFIED');
DISPLAY ('OR DELETED ') REPLY (REC_NAME);
PROCESS_OWNER_NO = ' 1';
OBTAIN CALC RECORD (PROCESS_OWNER);
CALL IDMS_STATUS;
OBTAIN FIRST RECORD (PROCESS) SET (PRO_SUBPARTS_PR);
DO WHILE (ERROR_STATUS = OK);
    IF (PROCESS_NAME = REC_NAME) THEN DO;
        DISPLAY ('DO YOU WISH TO DELETE(D) OR MODIFY(M) THIS OBJECT?');
        DISPLAY (' ') REPLY (CHOICE);
        IF (CHOICE = 'D') THEN DO;
            DISCONNECT RECORD (PROCESS) SET (PRO_SUBPARTS_PR);
            ERASE RECORD (PROCESS);
            DISPLAY (' PROCESS ' || REC_NAME || ' HAS BEEN DELETED');
            /* RESET RECORD COUNT IN REMAINING RECORDS */
            OBTAIN NEXT RECORD (PROCESS) SET (PRO_SUBPARTS_PR);
            DO WHILE (ERROR_STATUS = OK);
                REC_NUM = PROCESS_NO;
                CALL DEC (REC_NUM);
                PROCESS_NO = REC_NUM;
                MODIFY RECORD (PROCESS);
                CALL IDMS_STATUS;
                OBTAIN NEXT RECORD (PROCESS) SET (PRO_SUBPARTS_PR);
            END; /* OF WHILE */
            PRCNT = PRCNT - 1;
        END; /* OF IF CHOICE */
    ELSE DO;

```

```

    DISPLAY('ENTER THE (NEW) NAME FOR THE PROCESS') REPLY(REC_NAME);
    PROCESS_NAME = REC_NAME;
    DISPLAY('STATE IN 72 CHARACTERS OR LESS, A DESCRIPTION OR ');
    DISPLAY('ANY ATTRIBUTES OF ' || REC_NAME) REPLY(ATTRIBUTES);
    PR_ATTRIBUTES = ATTRIBUTES;
    MODIFY RECORD(PROCESS);
    CALL IDMS_STATUS;
    DISPLAY(' PROCESS ' || REC_NAME || ' HAS BEEN MODIFIED');
    ERROR_STATUS = END_OF_SET;

END; /* OF ELSE */

END; /* OF IF PROCESS_NAME */

ELSE DO;

    OBTAIN NEXT RECORD(PROCESS) SET(PRO_SUBPARTS_PR);

    END; /* OF ELSE */

END; /* OF WHILE */

GO TO DELETE_RECORDS;

DELETE_INPUT:  INPUT_OWNER_NO = ' 1';

DISPLAY('ENTER THE NAME OF THE INPUT TO BE MODIFIED ');
DISPLAY('OR DELETED ') REPLY(REC_NAME);

J = 1;

OBTAIN CALC RECORD(INPUT_OWNER);

DO WHILE (J<= INCNT);

IF (REC_NAME = INPUT_OWNER_NAME) THEN DO;

DISPLAY('DO YOU WISH TO DELETE(D) OR MODIFY(M) ');

DISPLAY('THIS OBJECT ') REPLY(CHOICE);

IF (CHOICE = 'D') THEN DO;

```

```

OBTAIN FIRST RECORD(INPUT) SET(INO_SUBPARTS_IN);
DO WHILE (ERROR_STATUS = OK);
    ERASE RECORD(INPUT);
    OBTAIN NEXT RECORD(INPUT) SET(INO_SUBPARTS_IN);
END;
ERASE RECORD(INPUT_OWNER);
    DISPLAY(' INPUT ' || REC_NAME || ' HAS BEEN DELETED');
J = J + 1;
DO WHILE (J<= INCNT);
    PUT STRING(INPUT_OWNER_NO) EDIT(J) (F(4));
    OBTAIN CALC RECORD(INPUT_OWNER);
    CALL IDMS_STATUS;
    J = J + 1;
    REC_NUM=INPUT_OWNER_NO;
    CALL DEC(REC_NUM);
    INPUT_OWNER_NO = REC_NUM;
    MODIFY RECORD(INPUT_OWNER);
    CALL IDMS_STATUS;
END; /* OF WHILE */
    INCNT = INCNT -1;
END; /* OF THEN */
ELSE DO;
    DISPLAY('ENTER THE NAME FOR THE (NEW) INPUT ')REPLY(REC_NAME);
    INPUT_OWNER_NAME = REC_NAME;
    DISPLAY('STATE IN 72 CHARACTERS OR LESS, A DESCRIPTION');
    DISPLAY(' OR ANY ATTRIBUTES OF ' || REC_NAME)REPLY(ATTRIBUTES);
    INO_ATTRIBUTES = ATTRIBUTES;

```

```

MODIFY RECORD (INPUT_OWNER) ;

CALL IDMS_STATUS;

    DISPLAY (' INPUT ' || REC_NAME || ' HAS BEEN MODIFIED');

J = INCNT + 1;

END; /* OF ELSE */

END; /* OF IF INPUT */

ELSE DO;

    J = J + 1;

    PUT STRING (INPUT_OWNER_NO) EDIT (J) (F(4));

    OBTAIN CALC RECORD (INPUT_OWNER) ;

END; /* OF ELSE */

END; /* OF WHILE */

GO TO DELETE_RECORDS;

DELETE_OUTPUT:  OUTPUT_OWNER_NO = '    1';

DISPLAY ('ENTER THE NAME OF THE OUTPUT TO BE MODIFIED ');

DISPLAY ('OR DELETED ') REPLY (REC_NAME) ;

J = 1;

OBTAIN CALC RECORD (OUTPUT_OWNER) ;

DO WHILE (J<= OUCNT) ;

IF (REC_NAME = OUTPUTOWNER_NAME) THEN DO;

DISPLAY ('DO YOU WISH TO DELETE(D) OR MODIFY(M) ');

DISPLAY ('THIS OBJECT ') REPLY (CHOICE) ;

IF (CHOICE = 'D') THEN DO;

    OBTAIN FIRST RECORD (OUTPUT) SET (OUO_SUBPARTS_OU) ;

    DO WHILE (ERROR_STATUS = OK) ;

        ERASE RECORD (OUTPUT) ;

        OBTAIN NEXT RECORD (OUTPUT) SET (OUO_SUBPARTS_OU) ;

```

```

END;

ERASE RECORD(OUTPUT_OWNER);

    DISPLAY(' OUTPUT ' || REC_NAME || ' HAS BEEN DELETED');

J = J + 1;

DO WHILE (J<= OUCNT);

    PUT STRING(OUTPUT_OWNER_NO) EDIT(J) (F(4));

    OBTAIN CALC RECORD(OUTPUT_OWNER);

    CALL IDMS_STATUS;

    REC_NUM = OUTPUT_OWNER_NO;

    CALL DEC(REC_NUM);

    OUTPUT_OWNER_NO = REC_NUM;

    MODIFY RECORD(OUTPUT_OWNER);

    CALL IDMS_STATUS;

    J = J + 1;

END; /* OF WHILE */

OUCNT = OUCNT - 1;

END; /* OF THEN */

ELSE DO;

DISPLAY('ENTER THE NAME FOR THE (NEW) OUTPUT ' ) REPLY(REC_NAME);

OUTPUTOWNER_NAME = REC_NAME;

DISPLAY('STATE IN 72 CHARACTERS OR LESS, A DESCRIPTION');

DISPLAY(' OR ANY ATTRIBUTES OF ' || REC_NAME) REPLY(ATTRIBUTES);

OUO_ATTRIBUTES = ATTRIBUTES;

MODIFY RECORD(OUTPUT_OWNER);

CALL IDMS_STATUS;

    DISPLAY(' OUTPUT ' || REC_NAME || ' HAS BEEN MODIFIED');

J = OUCNT + 1;

```

```

END; /* OF ELSE */
END; /* OF IF OUTPUT */
ELSE DO;
    J = J + 1;
    PUT STRING (OUTPUT_OWNER_NO) EDIT(J) (F(4));
    OBTAIN CALC RECORD(OUTPUT_OWNER);
END; /* OF ELSE */
END; /* OF WHILE */
GO TO DELETE_RECORDS;
DELETE_GROUP:  GROUP_OWNER_NO = '    1';
DISPLAY('ENTER THE NAME OF THE GROUP TO BE MODIFIED ');
DISPLAY('OR DELETED ')REPLY(REC_NAME);
J = 1;
OBTAIN CALC RECORD(GROUP_OWNER);
DO WHILE (J<= GRCNT);
IF (REC_NAME = GROUP_OWNER_NAME) THEN DO;
DISPLAY('DO YOU WISH TO DELETE(D) OR MODIFY(M) ');
DISPLAY('THIS OBJECT ')REPLY(CHOICE);
IF (CHOICE = 'D') THEN DO;
    OBTAIN FIRST RECORD(GROUP) SET(GRO_SUBPARTS_GR);
    DO WHILE (ERROR_STATUS = OK);
        ERASE RECORD(GROUP);
        OBTAIN NEXT RECORD(GROUP) SET(GRO_SUBPARTS_GR);
    END;
    ERASE RECORD(GROUP_OWNER);
    DISPLAY(' GROUP ' || REC_NAME || ' HAS BEEN DELETED');
J = J + 1;

```



```

DO WHILE (J<= GRCNT);

    PUT STRING(GROUP_OWNER_NO) EDIT(J) (F(4));

    OBTAIN CALC RECORD(GROUP_OWNER);

    CALL IDMS_STATUS;

    REC_NUM = GROUP_OWNER_NO;

    CALL DEC(REC_NUM);

    GROUP_OWNER_NO = REC_NUM;

    MODIFY RECORD(GROUP_OWNER);

    CALL IDMS_STATUS;

    J = J + 1;

END; /* OF WHILE */

GRCNT = GRCNT - 1;

END; /* OF THEN */

ELSE DO;

    DISPLAY('ENTER THE NAME FOR THE (NEW) GROUP ')REPLY(REC_NAME);

    GROUP_OWNER_NAME = REC_NAME;

    DISPLAY('STATE IN 72 CHARACTERS OR LESS, A DESCRIPTION');

    DISPLAY(' OR ANY ATTRIBUTES OF ' || REC_NAME)REPLY(ATTRIBUTES);

    GRO_ATTRIBUTES = ATTRIBUTES;

    MODIFY RECORD(GROUP_OWNER);

    CALL IDMS_STATUS;

        DISPLAY(' GROUP ' || REC_NAME || ' HAS BEEN MODIFIED');

    J = GRCNT + 1;

END; /* OF ELSE */

END; /* OF IF GROUP */

ELSE DO;

    J = J + 1;

```

```

    PUT STRING (GROUP_OWNER_NO) EDIT (J) (F(4));

    OBTAIN CALC RECORD (GROUP_OWNER);

END; /* OF ELSE */

END; /* OF WHILE */

GO TO DELETE_RECORDS;

DELETE_ENTITY:  ENTITY_OWNER_NO = '    1';

DISPLAY('ENTER THE NAME OF THE ENTITY TO BE MODIFIED ');

DISPLAY('OR DELETED ')REPLY(REC_NAME);

J = 1;

OBTAIN CALC RECORD (ENTITY_OWNER);

DO WHILE (J<= ENCNT);

IF (REC_NAME = ENTITY_OWNER_NAME) THEN DO;

DISPLAY('DO YOU WISH TO DELETE(D) OR MODIFY(M) ');

DISPLAY('THIS OBJECT ')REPLY(CHOICE);

IF (CHOICE = 'D') THEN DO;

    OBTAIN FIRST RECORD (ENTITY) SET (ENO_SUBPARTS_EN);

    DO WHILE (ERROR_STATUS = OK);

        ERASE RECORD (ENTITY);

        OBTAIN NEXT RECORD (ENTITY) SET (ENO_SUBPARTS_EN);

    END;

    ERASE RECORD (ENTITY_OWNER);

        DISPLAY(' ENTITY ' || REC_NAME || ' HAS BEEN DELETED');

J = J + 1;

DO WHILE (J<= ENCNT);

    PUT STRING (ENTITY_OWNER_NO) EDIT (J) (F(4));

    OBTAIN CALC RECORD (ENTITY_OWNER);

    CALL IDMS_STATUS;

```

```

    REC_NUM = ENTITY_OWNER_NO;
    CALL DEC(REC_NUM);
    ENTITY_OWNER_NO = REC_NUM;
    MODIFY RECORD(ENTITY_OWNER);
    CALL IDMS_STATUS;

    J = J + 1;

END; /* OF WHILE */

    ENCNT = ENCNT - 1;

END; /* OF THEN */

ELSE DO;

    DISPLAY('ENTER THE NAME FOR THE (NEW) ENTITY  ') REPLY(REC_NAME);
    ENTITY_OWNER_NAME = REC_NAME;
    DISPLAY('STATE IN 72 CHARACTERS OR LESS, A DESCRIPTION');
    DISPLAY(' OR ANY ATTRIBUTES OF ' || REC_NAME) REPLY(ATTRIBUTES);
    ENO_ATTRIBUTES = ATTRIBUTES;
    MODIFY RECORD(ENTITY_OWNER);
    CALL IDMS_STATUS;

        DISPLAY(' ENTITY ' || REC_NAME || ' HAS BEEN MODIFIED');

    J = ENCNT + 1;

END; /* OF ELSE */

END; /* OF IF ENTITY */

ELSE DO;

    J = J + 1;

    PUT STRING (ENTITY_OWNER_NO) EDIT(J) (F(4));

    OBTAIN CALC RECORD(ENTITY_OWNER);

END; /* OF ELSE */

END; /* OF WHILE */

```

```

GO TO DELETE_RECORDS;

DELETE_ELEMENT:  ELEMENT_OWNER_NO = '  1';

DISPLAY('ENTER THE NAME OF THE ELEMENT TO BE MODIFIED ');

DISPLAY('OR DELETED ') REPLY (REC_NAME);

J = 1;

OBTAIN CALC RECORD (ELEMENT_OWNER);

DO WHILE (J<= ELCNT);

IF (REC_NAME = ELEMENT_OWNER_NAME) THEN DO;

DISPLAY('DO YOU WISH TO DELETE(D) OR MODIFY(M) ');

DISPLAY('THIS OBJECT ') REPLY (CHOICE);

IF (CHOICE = 'D') THEN DO;

    ERASE RECORD (ELEMENT_OWNER);

        DISPLAY(' ELEMENT ' || REC_NAME || ' HAS BEEN DELETED');

J = J + 1;

DO WHILE (J<= ELCNT);

    PUT STRING (ELEMENT_OWNER_NO) EDIT (J) (F(4));

    OBTAIN CALC RECORD (ELEMENT_OWNER);

    CALL IDMS_STATUS;

    REC_NUM = ELEMENT_OWNER_NO;

    CALL DEC (REC_NUM);

    ELEMENT_OWNER_NO = REC_NUM;

    MODIFY RECORD (ELEMENT_OWNER);

    CALL IDMS_STATUS;

    J = J + 1;

END; /* OF WHILE */

ELCNT = ELCNT - 1;

END; /* OF THEN */

```

```

ELSE DO;

DISPLAY('ENTER THE NAME FOR THE (NEW) ELEMENT ')REPLY(REC_NAME);

ELEMENT_OWNER_NAME = REC_NAME;

DISPLAY('STATE IN 72 CHARACTERS OR LESS, A DESCRIPTION');

DISPLAY(' OR ANY ATTRIBUTES OF ' || REC_NAME)REPLY(ATTRIBUTES);

ELO_ATTRIBUTES = ATTRIBUTES;

MODIFY RECORD(ELEMENT_OWNER);

CALL IDMS_STATUS;

        DISPLAY(' ELEMENT ' || REC_NAME || ' HAS BEEN MODIFIED');

J = ELCNT + 1;

END; /* OF ELSE */

END; /* OF IF ELEMENT */

ELSE DO;

        J = J + 1;

        PUT STRING (ELEMENT_OWNER_NO) EDIT(J) (F(4));

        OBTAIN CALC RECORD(ELEMENT_OWNER);

END; /* OF ELSE */

END; /* OF WHILE */

GO TO DELETE_RECORDS;

DELETE_EVENT:  EVENT_NO = '    1';

DISPLAY('ENTER THE NAME OF THE EVENT TO BE MODIFIED ');

DISPLAY('OR DELETED ')REPLY(REC_NAME);

J = 1;

OBTAIN CALC RECORD(EVENT);

DO WHILE (J<= EVCNT);

IF (REC_NAME = EVENT_NAME) THEN DO;

DISPLAY('DO YOU WISH TO DELETE(D) OR MODIFY(M)');

```

```

DISPLAY('THIS OBJECT ')REPLY(CHOICE);
IF (CHOICE = 'D') THEN DO;
    ERASE RECORD(EVENT);
        DISPLAY(' EVENT ' || REC_NAME || ' HAS BEEN DELETED');
J = J + 1;
DO WHILE (J<= EVCNT);
    PUT STRING(EVENT_NO) EDIT(J) (F(4));
    OBTAIN CALC RECORD(EVENT);
    CALL IDMS_STATUS;
    REC_NUM = EVENT_NO;
    CALL DEC(REC_NUM);
    EVENT_NO = REC_NUM;
    MODIFY RECORD(EVENT);
    CALL IDMS_STATUS;
    J = J + 1;
END; /* OF WHILE */
EVCNT = EVCNT -1;
END; /* OF THEN */
ELSE DO;
DISPLAY('ENTER THE NAME FOR THE (NEW) EVENT ')REPLY(REC_NAME);
EVENT_NAME = REC_NAME;
DISPLAY('STATE IN 72 CHARACTERS OR LESS, A DESCRIPTION');
DISPLAY(' OR ANY ATTRIBUTES OF ' || REC_NAME)REPLY(ATTRIBUTES);
EV_ATTRIBUTES = ATTRIBUTES;
MODIFY RECORD(EVENT);
CALL IDMS_STATUS;
    DISPLAY(' EVENT ' || REC_NAME || ' HAS BEEN MODIFIED');

```

```

J = EVCNT + 1;
END; /* OF ELSE */
END; /* OF IF EVENT */
ELSE DO;
    J = J + 1;
    PUT STRING (EVENT_NO) EDIT(J) (F(4));
    OBTAIN CALC RECORD(EVENT);
END; /* OF ELSE */
END; /* OF WHILE */
GO TO DELETE_RECORDS;
DELETE_CONDITION:  CONDITION_NO = '    1';
DISPLAY('ENTER THE NAME OF THE CONDITION TO BE MODIFIED ');
DISPLAY('OR DELETED ')REPLY(REC_NAME);
J = 1;
OBTAIN CALC RECORD(CONDITION);
DO WHILE (J<= COCNT);
IF (REC_NAME = CONDITION_NAME) THEN DO;
DISPLAY('DO YOU WISH TO DELETE(D) OR MODIFY(M)');
DISPLAY('THIS OBJECT ')REPLY(CHOICE);
IF (CHOICE = 'D') THEN DO;
    ERASE RECORD(CONDITION);
    DISPLAY(' CONDITION ' || REC_NAME || ' HAS BEEN DELETED');
J = J + 1;
DO WHILE (J<= COCNT);
    PUT STRING(CONDITION_NO) EDIT(J) (F(4));
    OBTAIN CALC RECORD(CONDITION);
    CALL IDMS_STATUS;

```

```

    REC_NUM = CONDITION_NO;
    CALL DEC (REC_NUM);
    CONDITION_NO = REC_NUM;
    MODIFY RECORD (CONDITION);
    CALL IDMS_STATUS;

    J = J + 1;

END; /* OF WHILE */

COCNT = COCNT - 1;

END; /* OF THEN */

ELSE DO;

DISPLAY('ENTER THE NAME FOR THE (NEW) CONDITION  ') REPLY (REC_NAME);
CONDITION_NAME = REC_NAME;
DISPLAY('STATE IN 72 CHARACTERS OR LESS, A DESCRIPTION');
DISPLAY(' OR ANY ATTRIBUTES OF ' || REC_NAME) REPLY (ATTRIBUTES);
CO_ATTRIBUTES = ATTRIBUTES;
MODIFY RECORD (CONDITION);
CALL IDMS_STATUS;

    DISPLAY(' CONDITION ' || REC_NAME || ' HAS BEEN MODIFIED');

J = COCNT + 1;

END; /* OF ELSE */

END; /* OF IF CONDITION */

ELSE DO;

    J = J + 1;

    PUT STRING (CONDITION_NO) EDIT (J) (F(4));

    OBTAIN CALC RECORD (CONDITION);

END; /* OF ELSE */

END; /* OF WHILE */

```



```

GO TO DELETE_RECORDS;

ENTER_SETS: DISPLAY('THE FOLLOWING RELATIONSHIPS MAY BE DEFINED');
DISPLAY(' 1-PROCESS_RECEIVES_INPUT      2-PROCESS_USES_INPUT');
DISPLAY(' 3-INPUT_CONSISTS_OF_ELEMENT  4-INPUT_CONSISTS_OF_GROUP');
DISPLAY(' 5-PROCESS_GENERATES_OUTPUT    6-PROCESS_DERIVES_OUTPUT');
DISPLAY(' 7-PROCESS_USES_ELEMENT        8-PROCESS_UPDATES_ELEMENT');
DISPLAY(' 9-PROCESS_DERIVES_ELEMENT    10-PROCESS_DERIVES_GROUP');
DISPLAY('11-PROCESS_UPDATES_GROUP       12-PROCESS_USES_GROUP');
DISPLAY('13-PROCESS_DERIVES_ENTITY      14-PROCESS_UPDATES_ENTITY');
DISPLAY('15-PROCESS_USES_ENTITY 16-PROCESS_INCEPTION_CAUSES_EVENT');
DISPLAY('17-PROCESS_TERMINATION_CAUSES_EVENT ');
DISPLAY('                                18-PROCESS_TRIGGERED_BY_EVENT');
DISPLAY('19-EVENT_WHEN_CONDITION      20-ENTITY_CONSISTS_OF_GROUP');
DISPLAY('21-GROUP_CONSISTS_OF_ELEMENT  22-ENTITY_CONSISTS_ELEMENT');
DISPLAY('23-OUTPUT_CONSISTS_OF_ELEMENT  24-OUTPUT_CONSISTS_GROUP');
DISPLAY('ENTER THE NUMBER CORRESPONDING TO THE DESIRED ');
DISPLAY('RELATIONSHIP OR 25 TO RETURN TO MENU')REPLY(CHOICE);
SELECT;

WHEN ( CHOICE='1' ) GO TO PR_REC_IN;
WHEN ( CHOICE='2' ) GO TO PR_USE_IN;
WHEN ( CHOICE='3' ) GO TO IN_CON_EL;
WHEN ( CHOICE='4' ) GO TO IN_CON_GR;
WHEN ( CHOICE='5' ) GO TO PR_GEN_OU;
WHEN ( CHOICE='6' ) GO TO PR_DER_OU;
WHEN ( CHOICE='7' ) GO TO PR_USE_EL;
WHEN ( CHOICE='8' ) GO TO PR_UPD_EL;
WHEN ( CHOICE='9' ) GO TO PR_DER_EL;

```

```

WHEN ( CHOICE='10' ) GO TO PR_DER_GR;
WHEN ( CHOICE='11' ) GO TO PR_UPD_GR;
WHEN ( CHOICE='12' ) GO TO PR_USE_GR;
WHEN ( CHOICE='13' ) GO TO PR_DER_EN;
WHEN ( CHOICE='14' ) GO TO PR_UPD_EN;
WHEN ( CHOICE='15' ) GO TO PR_USE_EN;
WHEN ( CHOICE='16' ) GO TO PR_INC_EV;
WHEN ( CHOICE='17' ) GO TO PR_TER_EV;
WHEN ( CHOICE='18' ) GO TO PR_TRI_EV;
WHEN ( CHOICE='19' ) GO TO EV_WHE_CO;
WHEN ( CHOICE='20' ) GO TO EN_CON_GR;
WHEN ( CHOICE='21' ) GO TO GR_CON_EL;
WHEN ( CHOICE='22' ) GO TO EN_CON_EL;
WHEN ( CHOICE='23' ) GO TO OU_CON_EL;
WHEN ( CHOICE='24' ) GO TO OU_CON_GR;
OTHERWISE      GO TO MENU;

END;

PR_REC_IN :   DISPLAY('YOU HAVE CHOSEN THE RELATIONSHIP ');
DISPLAY('PROCESS_RECEIVES_INPUT. ');
DISPLAY('ENTER THE PROCESS NAME ') REPLY(OWNER_NAME);
DISPLAY('ENTER THE INPUT NAME ') REPLY(MEMBER_NAME);
DISPLAY('DO YOU WISH TO (I) INSERT, ');
DISPLAY(' OR (D) DELETE THIS RELATIONSHIP?  ENTER YOUR CHOICE ');
DISPLAY(' OF I, D') REPLY(CHOICE);
IF CHOICE = 'I' THEN GO TO INSERT_PR_REC_IN;
IF CHOICE = 'D' THEN GO TO DELETE_PR_REC_IN;
DISPLAY('INVALID CHOICE, TRY AGAIN');

```

```

GO TO PR_REC_IN;
INSERT_PR_REC_IN :
PROCESS_OWNER_NO = ' 1';
J = 1;
OBTAIN CALC RECORD(PROCESS_OWNER);
OBTAIN FIRST RECORD(PROCESS) SET(PRO_SUBPARTS_PR);
DO WHILE (J <= PRCNT);
    IF (PROCESS_NAME = OWNER_NAME) THEN DO;
        INPUT_OWNER_NO = ' 1';
        I = 1;
        OBTAIN CALC RECORD(INPUT_OWNER);
        DO WHILE (I <= INCNT);
            IF INPUT_OWNER_NAME = MEMBER_NAME THEN DO;
                INPUT_NAME = INPUT_OWNER_NAME;
                INPUT_NO = INPUT_OWNER_NO;
                IN_ATTRIBUTES = INO_ATTRIBUTES;
                STORE RECORD(INPUT);
                CALL IDMS_STATUS;
                CONNECT RECORD(INPUT) SET(INO_SUBPARTS_IN);
                CALL IDMS_STATUS;
                CONNECT RECORD(INPUT) SET(PR_RECEIVES_IN);
                CALL IDMS_STATUS;
                DISPLAY('RELATIONSHIP ' || OWNER_NAME || ' RECEIVES ' ||
MEMBER_NAME || ' HAS BEEN INSERTED ');
                I = INCNT + 1;
                J = PRCNT + 1;
            END; /* OF IF */
        END;
    END;

```

```

ELSE DO;

    FIND CURRENT RECORD(INPUT_OWNER);

    PUT STRING(INPUT_OWNER_NO) EDIT(I+1) (F(4));

    OBTAIN CALC RECORD(INPUT_OWNER);

    END; /* ELSE */

    I = I + 1;

    END; /* OF WHILE */

END; /* OF THEN */

ELSE DO;

    OBTAIN NEXT RECORD(PROCESS) SET(PRO_SUBPARTS_PR);

    END; /* OF ELSE */

    J = J + 1;

    END; /* OF WHILE */

GO TO ENTER_SETS;

DELETE_PR_REC_IN: PROCESS_OWNER_NO = '    1';

J = 1;

OBTAIN CALC RECORD(PROCESS_OWNER);

OBTAIN FIRST RECORD(PROCESS) SET(PRO_SUBPARTS_PR);

DO WHILE (J<=PRCNT);

    IF (OWNER_NAME = PROCESS_NAME) THEN DO;

        OBTAIN FIRST RECORD(INPUT) SET(PR_RECEIVES_IN);

        DO WHILE (ERROR_STATUS = OK);

            IF (INPUT_NAME = MEMBER_NAME) THEN DO;

                DISCONNECT RECORD(INPUT) SET(PR_RECEIVES_IN);

                CALL IDMS_STATUS;

                DISPLAY(' RELATIONSHIP ' || OWNER_NAME || ' RECEIVES ' ||

```

```

MEMBER_NAME || ' HAS BEEN DELETED ');

    J = PRCNT + 1;

    ERROR_STATUS = END_OF_SET;

END; /* OF THEN */

ELSE DO;

    OBTAIN NEXT RECORD (INPUT) SET (PR_RECEIVES_IN);

END; /* OF ELSE */

END; /* OF WHILE ERROR_STATUS */

END; /* OF IF */

ELSE DO;

    OBTAIN NEXT RECORD (PROCESS) SET (PRO_SUBPARTS_PR);

END; /* OF ELSE */

    J = J + 1;

END; /* OF WHILE */

GO TO ENTER_SETS;

PR_USE_IN :   DISPLAY ('YOU HAVE CHOSEN THE RELATIONSHIP ');

DISPLAY ('PROCESS_USES_INPUT. ');

DISPLAY ('ENTER THE PROCESS NAME ') REPLY (OWNER_NAME);

DISPLAY ('ENTER THE INPUT NAME ') REPLY (MEMBER_NAME);

DISPLAY ('DO YOU WISH TO (I) INSERT, ');

DISPLAY (' OR (D) DELETE THIS RELATIONSHIP?  ENTER YOUR CHOICE ');

DISPLAY (' OF I, D') REPLY (CHOICE);

IF CHOICE = 'I' THEN GO TO INSERT_PR_USE_IN;

IF CHOICE = 'D' THEN GO TO DELETE_PR_USE_IN;

DISPLAY ('INVALID CHOICE, TRY AGAIN');

GO TO PR_USE_IN;

```

```

INSERT_PR_USE_IN :
PROCESS_OWNER_NO = '  1';

J = 1;

OBTAIN CALC RECORD (PROCESS_OWNER) ;

OBTAIN FIRST RECORD (PROCESS) SET (PRO_SUBPARTS_PR) ;

DO WHILE (J <= PRCNT) ;

  IF (PROCESS_NAME = OWNER_NAME) THEN DO ;

    INPUT_OWNER_NO = '  1';

    I = 1;

    OBTAIN CALC RECORD (INPUT_OWNER) ;

    DO WHILE (I <= INCNT) ;

      IF INPUT_OWNER_NAME = MEMBER_NAME THEN DO;

        INPUT_NAME = INPUT_OWNER_NAME;

        INPUT_NO = INPUT_OWNER_NO;

        IN_ATTRIBUTES = INO_ATTRIBUTES;

        STORE RECORD (INPUT) ;

        CALL IDMS_STATUS;

        CONNECT RECORD (INPUT) SET (INO_SUBPARTS_IN) ;

        CONNECT RECORD (INPUT) SET (PR_USES_IN) ;

        CALL IDMS_STATUS;

        DISPLAY('RELATIONSHIP ' || OWNER_NAME || '  USES  ' ||
MEMBER_NAME || ' HAS BEEN INSERTED ');

        I = INCNT +1;

        J = PRCNT + 1;

        END; /* OF IF */

      ELSE DO;

        FIND CURRENT RECORD (INPUT_OWNER) ;

```

```

        PUT STRING (INPUT_OWNER_NO) EDIT (I+1) (F(4));

        OBTAIN CALC RECORD (INPUT_OWNER);

        END; /* ELSE */

        I = I + 1;

        END; /* OF WHILE */

        END; /* OF IF */

ELSE DO;

        OBTAIN NEXT RECORD (PROCESS) SET (PRO_SUBPARTS_PR);

        END; /* OF ELSE */

        J = J + 1;

        END; /* OF WHILE */

GO TO ENTER_SETS;

DELETE_PR_USE_IN: PROCESS_OWNER_NO = '    1';

J = 1;

OBTAIN CALC RECORD (PROCESS_OWNER);

OBTAIN FIRST RECORD (PROCESS) SET (PRO_SUBPARTS_PR);

DO WHILE (J<=PRCNT);

    IF (OWNER_NAME = PROCESS_NAME) THEN DO;

        OBTAIN FIRST RECORD (INPUT) SET (PR_USES_IN);

        DO WHILE (ERROR_STATUS = OK);

            IF (INPUT_NAME = MEMBER_NAME) THEN DO;

                DISCONNECT RECORD (INPUT) SET (PR_USES_IN);

                CALL IDMS_STATUS;

                DISPLAY ('RELATIONSHIP ' || OWNER_NAME || '    USES    ' ||

MEMBER_NAME || ' HAS BEEN DELETED ');

                J = PRCNT + 1;

```

```

        ERROR_STATUS = END_OF_SET;
    END; /* OF THEN */

    ELSE DO;

        OBTAIN NEXT RECORD (INPUT) SET (PR_USES_IN);

        END; /* OF ELSE */

    END; /* OF WHILE ERROR_STATUS */

END; /* OF IF */

ELSE DO;

    OBTAIN NEXT RECORD (PROCESS) SET (PRO_SUBPARTS_PR);

    END; /* OF ELSE */

    J = J + 1;

END; /* OF WHILE */

GO TO ENTER_SETS;

PR_GEN_OU :   DISPLAY ('YOU HAVE CHOSEN THE RELATIONSHIP ');

DISPLAY ('PROCESS_GENERATES_OUTPUT. ');

DISPLAY ('ENTER THE PROCESS NAME ') REPLY (OWNER_NAME);

DISPLAY ('ENTER THE OUTPUT NAME ') REPLY (MEMBER_NAME);

DISPLAY ('DO YOU WISH TO (I) INSERT, ');

DISPLAY (' OR (D) DELETE THIS RELATIONSHIP?  ENTER YOUR CHOICE ');

DISPLAY (' OF I, D') REPLY (CHOICE);

IF CHOICE = 'I' THEN GO TO INSERT_PR_GEN_OU;

IF CHOICE = 'D' THEN GO TO DELETE_PR_GEN_OU;

DISPLAY ('INVALID CHOICE, TRY AGAIN');

GO TO PR_GEN_OU;

INSERT_PR_GEN_OU :

PROCESS_OWNER_NO = '    1';

```



```

J = 1;

OBTAIN CALC RECORD (PROCESS_OWNER);

OBTAIN FIRST RECORD (PROCESS) SET (PRO_SUBPARTS_PR);

DO WHILE (J <= PRCNT);

    IF (PROCESS_NAME = OWNER_NAME) THEN DO;

        OUTPUT_OWNER_NO = '    1';

        I = 1;

        OBTAIN CALC RECORD (OUTPUT_OWNER);

        DO WHILE (I <= OUCNT);

            IF OUTPUTOWNER_NAME = MEMBER_NAME THEN DO;

                OUTPUT_NAME = OUTPUTOWNER_NAME;

                OUTPUT_NO = OUTPUT_OWNER_NO;

                OU_ATTRIBUTES = OUO_ATTRIBUTES;

                STORE RECORD (OUTPUT);

                CALL IDMS_STATUS;

                CONNECT RECORD (OUTPUT) SET (OUO_SUBPARTS_OU);

                CONNECT RECORD (OUTPUT) SET (PR_GENERATES_OU);

                CALL IDMS_STATUS;

                DISPLAY ('RELATIONSHIP ' || OWNER_NAME || ' GENERATES '
|| MEMBER_NAME || ' HAS BEEN INSERTED ');

                I = OUCNT + 1;

                J = PRCNT + 1;

                END; /* OF IF */

            ELSE DO;

                FIND CURRENT RECORD (OUTPUT_OWNER);

                PUT STRING (OUTPUT_OWNER_NO) EDIT (I+1) (F(4));

                OBTAIN CALC RECORD (OUTPUT_OWNER);

```

```

        END; /* ELSE */

        I = I + 1;

        END; /* OF WHILE */

        END; /* OF THEN */

    ELSE DO;

        OBTAIN NEXT RECORD (PROCESS) SET (PRO_SUBPARTS_PR);

        END; /* OF ELSE */

        J = J + 1;

        END; /* OF WHILE */

GO TO ENTER_SETS;

DELETE_PR_GEN_OU: PROCESS_OWNER_NO = '    1';

J = 1;

OBTAIN CALC RECORD (PROCESS_OWNER);

OBTAIN FIRST RECORD (PROCESS) SET (PRO_SUBPARTS_PR);

DO WHILE (J<=PRCNT);

    IF (OWNER_NAME = PROCESS_NAME) THEN DO;

        OBTAIN FIRST RECORD (OUTPUT) SET (PR_GENERATES_OU);

        DO WHILE (ERROR_STATUS = OK);

            IF (OUTPUT_NAME = MEMBER_NAME) THEN DO;

                DISCONNECT RECORD (OUTPUT) SET (PR_GENERATES_OU);

                CALL IDMS_STATUS;

                DISPLAY ('RELATIONSHIP ' || OWNER_NAME || ' GENERATES '

|| MEMBER_NAME || ' HAS BEEN DELETED ');

                J = PRCNT + 1;

                ERROR_STATUS = END_OF_SET;

            END; /* OF THEN */

```

```

        ELSE DO;

            OBTAIN NEXT RECORD(OUTPUT) SET(PR_GENERATES_OU);

            END; /* OF ELSE */

        END; /* OF WHILE ERROR_STATUS */

    END; /* OF IF */

    ELSE DO;

        OBTAIN NEXT RECORD(PROCESS) SET(PRO_SUBPARTS_PR);

    END; /* OF ELSE */

    J = J + 1;

END; /* OF WHILE */

GO TO ENTER_SETS;

PR_DER_OU :   DISPLAY('YOU HAVE CHOSEN THE RELATIONSHIP ');

DISPLAY('PROCESS_DERIVES_OUPUT. ');

DISPLAY('ENTER THE PROCESS NAME ')REPLY(OWNER_NAME);

DISPLAY('ENTER THE OUTPUT NAME ')REPLY(MEMBER_NAME);

DISPLAY('DO YOU WISH TO (I) INSERT, ');

DISPLAY(' OR (D) DELETE THIS RELATIONSHIP?  ENTER YOUR CHOICE ');

DISPLAY(' OF I, D')REPLY(CHOICE);

IF CHOICE = 'I' THEN GO TO INSERT_PR_DER_OU;

IF CHOICE = 'D' THEN GO TO DELETE_PR_DER_OU;

DISPLAY('INVALID CHOICE, TRY AGAIN');

GO TO PR_DER_OU;

INSERT_PR_DER_OU :

PROCESS_OWNER_NO = '    1';

J = 1;

OBTAIN CALC RECORD(PROCESS_OWNER);

```

```

OBTAIN FIRST RECORD(PROCESS) SET (PRO_SUBPARTS_PR);
DO WHILE (J <= PRCNT);
    IF (PROCESS_NAME = OWNER_NAME) THEN DO;
        OUTPUT_OWNER_NO = '    1';
        I = 1;
        OBTAIN CALC RECORD (OUTPUT_OWNER);
        DO WHILE (I <= OUCNT);
            IF OUTPUTOWNER_NAME = MEMBER_NAME THEN DO;
                OUTPUT_NAME = OUTPUTOWNER_NAME;
                OUTPUT_NO = OUTPUT_OWNER_NO;
                OU_ATTRIBUTES = OUO_ATTRIBUTES;
                STORE RECORD (OUTPUT);
                CALL IDMS_STATUS;
                CONNECT RECORD (OUTPUT) SET (OUO_SUBPARTS_OU);
                CONNECT RECORD (OUTPUT) SET (PR_DERIVES_OU);
                CALL IDMS_STATUS;
                DISPLAY(' RELATIONSHIP ' || OWNER_NAME || ' DERIVES ' ||
MEMBER_NAME || ' HAS BEEN INSERTED ');
                I = OUCNT + 1;
                J = PRCNT + 1;
                END; /* OF IF */
            ELSE DO;
                FIND CURRENT RECORD (OUTPUT_OWNER);
                PUT STRING (OUTPUT_OWNER_NO) EDIT (I+1) (F(4));
                OBTAIN CALC RECORD (OUTPUT_OWNER);
                END; /* ELSE */
                I = I + 1;

```

```

        END; /* OF WHILE */

    END; /* OF THEN */

ELSE DO;

    OBTAIN NEXT RECORD (PROCESS) SET (PRO_SUBPARTS_PR);

    END; /* OF ELSE */

    J = J + 1;

END; /* OF WHILE */

GO TO ENTER_SETS;

DELETE_PR_DER_OU: PROCESS_OWNER_NO = '    1';

J = 1;

OBTAIN CALC RECORD (PROCESS_OWNER);

OBTAIN FIRST RECORD (PROCESS) SET (PRO_SUBPARTS_PR);

DO WHILE (J<=PRCNT);

    IF (OWNER_NAME = PROCESS_NAME) THEN DO;

        OBTAIN FIRST RECORD (OUTPUT) SET (PR_DERIVES_OU);

        DO WHILE (ERROR_STATUS = OK);

            IF (OUTPUT_NAME = MEMBER_NAME) THEN DO;

                DISCONNECT RECORD (OUTPUT) SET (PR_DERIVES_OU);

                CALL IDMS_STATUS;

                DISPLAY(' RELATIONSHIP ' || OWNER_NAME || ' DERIVES ' ||
MEMBER_NAME || ' HAS BEEN DELETED ');

                J = PRCNT + 1;

                ERROR_STATUS = END_OF_SET;

            END; /* OF THEN */

        ELSE DO;

            OBTAIN NEXT RECORD (OUTPUT) SET (PR_DERIVES_OU);

```

```

        END; /* OF ELSE */

    END; /* OF WHILE ERROR_STATUS */

    END; /* OF IF */

    ELSE DO;

        OBTAIN NEXT RECORD (PROCESS) SET (PRO_SUBPARTS_PR);

    END; /* OF ELSE */

    J = J + 1;

    END; /* OF WHILE */

GO TO ENTER_SETS;

PR_USE_EL :   DISPLAY ('YOU HAVE CHOSEN THE RELATIONSHIP ');

DISPLAY ('PROCESS_USES_ELEMENT. ');

DISPLAY ('ENTER THE PROCESS NAME ') REPLY (OWNER_NAME);

DISPLAY ('ENTER THE ELEMENT NAME ') REPLY (MEMBER_NAME);

DISPLAY ('DO YOU WISH TO (I) INSERT, ');

DISPLAY (' OR (D) DELETE THIS RELATIONSHIP? ENTER YOUR CHOICE ');

DISPLAY (' OF I, D') REPLY (CHOICE);

IF CHOICE = 'I' THEN GO TO INSERT_PR_USE_EL;

IF CHOICE = 'D' THEN GO TO DELETE_PR_USE_EL;

DISPLAY ('INVALID CHOICE, TRY AGAIN');

GO TO PR_USE_EL;

INSERT_PR_USE_EL :

PROCESS_OWNER_NO = '    1';

J = 1;

OBTAIN CALC RECORD (PROCESS_OWNER);

OBTAIN FIRST RECORD (PROCESS) SET (PRO_SUBPARTS_PR);

DO WHILE (J <= PRCNT);

```

```

IF (PROCESS_NAME = OWNER_NAME) THEN DO;
    ELEMENT_OWNER_NO = '    1';
    I = 1;
    OBTAIN CALC RECORD(ELEMENT_OWNER);
    DO WHILE (I <= ELCNT);
        IF ELEMENT_OWNER_NAME = MEMBER_NAME THEN DO;
            ELEMENT_NAME = ELEMENT_OWNER_NAME;
            ELEMENT_NO = ELEMENT_OWNER_NO;
            EL_ATTRIBUTES = ELO_ATTRIBUTES;
            STORE RECORD(ELEMENT);
            CALL IDMS_STATUS;
            CONNECT RECORD(ELEMENT) SET(ELO_SUBPARTS_EL);
            CONNECT RECORD(ELEMENT) SET(PR_USES_EL);
            CALL IDMS_STATUS;
            DISPLAY('RELATIONSHIP ' || OWNER_NAME || '    USES    ' ||
MEMBER_NAME || ' HAS BEEN INSERTED ');
            I = ELCNT + 1;
            J = PRCNT + 1;
            END; /* OF IF */
        ELSE DO;
            FIND CURRENT RECORD(ELEMENT_OWNER);
            PUT STRING(ELEMENT_OWNER_NO) EDIT(I+1) (F(4));
            OBTAIN CALC RECORD(ELEMENT_OWNER);
            END; /* ELSE */
            I = I + 1;
        END; /* OF WHILE */
    END; /* OF THEN */

```

```

ELSE DO;

    OBTAIN NEXT RECORD(PROCESS) SET (PRO_SUBPARTS_PR);

END; /* OF ELSE */

J = J + 1;

END; /* OF WHILE */

GO TO ENTER_SETS;

DELETE_PR_USE_EL: PROCESS_OWNER_NO = '    1';

J = 1;

OBTAIN CALC RECORD (PROCESS_OWNER);

OBTAIN FIRST RECORD (PROCESS) SET (PRO_SUBPARTS_PR);

DO WHILE (J<=PRCNT);

    IF (OWNER_NAME = PROCESS_NAME) THEN DO;

        OBTAIN FIRST RECORD (ELEMENT) SET (PR_USES_EL);

        DO WHILE (ERROR_STATUS = OK);

            IF (ELEMENT_NAME = MEMBER_NAME) THEN DO;

                DISCONNECT RECORD (ELEMENT) SET (PR_USES_EL);

                CALL IDMS_STATUS;

                DISPLAY ('RELATIONSHIP ' || OWNER_NAME || '    USES    ' ||
MEMBER_NAME || ' HAS BEEN DELETED ');

                J = PRCNT + 1;

                ERROR_STATUS = END_OF_SET;

            END; /* OF THEN */

        ELSE DO;

            OBTAIN NEXT RECORD (ELEMENT) SET (PR_USES_EL);

        END; /* OF ELSE */

    END; /* OF WHILE ERROR_STATUS */

```



```

    END; /* OF IF */

    ELSE DO;

        OBTAIN NEXT RECORD (PROCESS) SET (PRO_SUBPARTS_PR);

    END; /* OF ELSE */

    J = J + 1;

END; /* OF WHILE */


GO TO ENTER_SETS;

PR_UPD_EL :   DISPLAY ('YOU HAVE CHOSEN THE RELATIONSHIP ');

DISPLAY ('PROCESS_UPDATES_ELEMENT. ');

DISPLAY ('ENTER THE PROCESS NAME ') REPLY (OWNER_NAME);

DISPLAY ('ENTER THE ELEMENT NAME ') REPLY (MEMBER_NAME);

DISPLAY ('DO YOU WISH TO (I) INSERT, ');

DISPLAY (' OR (D) DELETE THIS RELATIONSHIP?  ENTER YOUR CHOICE ');

DISPLAY (' OF I, D') REPLY (CHOICE);

IF CHOICE = 'I' THEN GO TO INSERT_PR_UPD_EL;

IF CHOICE = 'D' THEN GO TO DELETE_PR_UPD_EL;

DISPLAY ('INVALID CHOICE, TRY AGAIN');

GO TO PR_UPD_EL;

INSERT_PR_UPD_EL :

PROCESS_OWNER_NO = '    1';

J = 1;

OBTAIN CALC RECORD (PROCESS_OWNER);

OBTAIN FIRST RECORD (PROCESS) SET (PRO_SUBPARTS_PR);

DO WHILE (J <= PRCNT);

    IF (PROCESS_NAME = OWNER_NAME) THEN DO;

        ELEMENT_OWNER_NO = '    1';

```

```

I = 1;
OBTAIN CALC RECORD(ELEMENT_OWNER);
DO WHILE (I <= ELCNT);
    IF ELEMENT_OWNER_NAME = MEMBER_NAME THEN DO;
        ELEMENT_NAME = ELEMENT_OWNER_NAME;
        ELEMENT_NO = ELEMENT_OWNER_NO;
        EL_ATTRIBUTES = ELO_ATTRIBUTES;
        STORE RECORD(ELEMENT);
        CALL IDMS_STATUS;
        CONNECT RECORD(ELEMENT) SET(ELO_SUBPARTS_EL);
        CONNECT RECORD(ELEMENT) SET(PR_UPDATES_EL);
        CALL IDMS_STATUS;
        DISPLAY('RELATIONSHIP ' || OWNER_NAME || ' UPDATES ' ||
MEMBER_NAME || ' HAS BEEN INSERTED ');
        I = ELCNT + 1;
        J = PRCNT + 1;
        END; /* OF IF */
    ELSE DO;
        FIND CURRENT RECORD(ELEMENT_OWNER);
        PUT STRING(ELEMENT_OWNER_NO) EDIT(I+1) (F(4));
        OBTAIN CALC RECORD(ELEMENT_OWNER);
        END; /* ELSE */
        I = I + 1;
    END; /* OF WHILE */
END; /* OF THEN */
ELSE DO;
    OBTAIN NEXT RECORD(PROCESS) SET(PRO_SUBPARTS_PR);

```

```

        END; /* OF ELSE */

        J = J + 1;

    END; /* OF WHILE */

GO TO ENTER_SETS;

DELETE_PR_UPD_EL: PROCESS_OWNER_NO = '    1';

J = 1;

OBTAIN CALC RECORD (PROCESS_OWNER);

OBTAIN FIRST RECORD (PROCESS) SET (PRO_SUBPARTS_PR);

DO WHILE (J<=PRCNT);

    IF (OWNER_NAME = PROCESS_NAME) THEN DO;

        OBTAIN FIRST RECORD (ELEMENT) SET (PR_UPDATES_EL);

        DO WHILE (ERROR_STATUS = OK);

            IF (ELEMENT_NAME = MEMBER_NAME) THEN DO;

                DISCONNECT RECORD (ELEMENT) SET (PR_UPDATES_EL);

                CALL IDMS_STATUS;

                DISPLAY ('RELATIONSHIP ' || OWNER_NAME || '  UPDATES ' ||
MEMBER_NAME || ' HAS BEEN DELETED ');

                J = PRCNT + 1;

                ERROR_STATUS = END_OF_SET;

            END; /* OF THEN */

            ELSE DO;

                OBTAIN NEXT RECORD (ELEMENT) SET (PR_UPDATES_EL);

            END; /* OF ELSE */

        END; /* OF WHILE ERROR_STATUS */

    END; /* OF IF */

ELSE DO;

```

```

    OBTAIN NEXT RECORD (PROCESS) SET (PRO_SUBPARTS_PR);
END; /* OF ELSE */

J = J + 1;
END; /* OF WHILE */

GO TO ENTER_SETS;

PR_DER_EL :   DISPLAY ('YOU HAVE CHOSEN THE RELATIONSHIP ');
DISPLAY ('PROCESS_DERIVES_ELEMENT. ');
DISPLAY ('ENTER THE PROCESS NAME ') REPLY (OWNER_NAME);
DISPLAY ('ENTER THE ELEMENT NAME ') REPLY (MEMBER_NAME);
DISPLAY ('DO YOU WISH TO (I) INSERT, ');
DISPLAY (' OR (D) DELETE THIS RELATIONSHIP?  ENTER YOUR CHOICE ');
DISPLAY (' OF I, D') REPLY (CHOICE);
IF CHOICE = 'I' THEN GO TO INSERT_PR_DER_EL;
IF CHOICE = 'D' THEN GO TO DELETE_PR_DER_EL;
DISPLAY ('INVALID CHOICE, TRY AGAIN');
GO TO PR_DER_EL;

INSERT_PR_DER_EL :
PROCESS_OWNER_NO = '    1';
J = 1;
OBTAIN CALC RECORD (PROCESS_OWNER);
OBTAIN FIRST RECORD (PROCESS) SET (PRO_SUBPARTS_PR);
DO WHILE (J <= PRCNT);
    IF (PROCESS_NAME = OWNER_NAME) THEN DO;
        ELEMENT_OWNER_NO = '    1';
        I = 1;
        OBTAIN CALC RECORD (ELEMENT_OWNER);

```

```

DO WHILE (I <= ELCNT);
  IF (ELEMENT_OWNER_NAME = MEMBER_NAME) THEN DO;
    ELEMENT_NAME = ELEMENT_OWNER_NAME;
    ELEMENT_NO = ELEMENT_OWNER_NO;
    EL_ATTRIBUTES = ELO_ATTRIBUTES;
    STORE RECORD(ELEMENT);
    CALL IDMS_STATUS;
    CONNECT RECORD(ELEMENT) SET(ELO_SUBPARTS_EL);
    CONNECT RECORD(ELEMENT) SET(PR_DERIVES_EL);
    CALL IDMS_STATUS;
    DISPLAY('RELATIONSHIP ' || OWNER_NAME || ' DERIVES ' ||
MEMBER_NAME || ' HAS BEEN INSERTED ');
    I = ELCNT + 1;
    J = PRCNT + 1;
    END; /* OF IF */
  ELSE DO;
    FIND CURRENT RECORD(ELEMENT_OWNER);
    PUT STRING (ELEMENT_OWNER_NO) EDIT (I+1) (F(4));
    OBTAIN CALC RECORD(ELEMENT_OWNER);
    END; /* ELSE */
    I = I + 1;
  END; /* OF WHILE */
END; /* OF THEN */
ELSE DO;
  OBTAIN NEXT RECORD(PROCESS) SET (PRO_SUBPARTS_PR);
END; /* OF ELSE */
J = J + 1;

```

```

END; /* OF WHILE */

GO TO ENTER_SETS;

DELETE_PR_DER_EL: PROCESS_OWNER_NO = ' 1';

J = 1;

OBTAIN CALC RECORD (PROCESS_OWNER) ;

OBTAIN FIRST RECORD (PROCESS) SET (PRO_SUBPARTS_PR) ;

DO WHILE (J<=PRCNT) ;

    IF (OWNER_NAME = PROCESS_NAME) THEN DO ;

        OBTAIN FIRST RECORD (ELEMENT) SET (PR_DERIVES_EL) ;

        DO WHILE (ERROR_STATUS = OK) ;

            IF (ELEMENT_NAME = MEMBER_NAME) THEN DO ;

                DISCONNECT RECORD (ELEMENT) SET (PR_DERIVES_EL) ;

                CALL IDMS_STATUS ;

                DISPLAY ('RELATIONSHIP ' || OWNER_NAME || ' DERIVES ' ||
MEMBER_NAME || ' HAS BEEN DELETED ');

                J = PRCNT + 1;

                ERROR_STATUS = END_OF_SET;

            END; /* OF THEN */

            ELSE DO ;

                OBTAIN NEXT RECORD (ELEMENT) SET (PR_DERIVES_EL) ;

            END; /* OF ELSE */

        END; /* OF WHILE ERROR_STATUS */

    END; /* OF IF */

    ELSE DO ;

        OBTAIN NEXT RECORD (PROCESS) SET (PRO_SUBPARTS_PR) ;

    END; /* OF ELSE */

```

```

    J = J + 1;
END; /* OF WHILE */

GO TO ENTER_SETS;

PR_DER_GR :   DISPLAY('YOU HAVE CHOSEN THE RELATIONSHIP ');
DISPLAY('PROCESS_DERIVES_GROUP. ');
DISPLAY('ENTER THE PROCESS NAME ') REPLY(OWNER_NAME);
DISPLAY('ENTER THE GROUP NAME ') REPLY(MEMBER_NAME);
DISPLAY('DO YOU WISH TO (I) INSERT, ');
DISPLAY(' OR (D) DELETE THIS RELATIONSHIP?  ENTER YOUR CHOICE ');
DISPLAY(' OF I, D') REPLY(CHOICE);
IF CHOICE = 'I' THEN GO TO INSERT_PR_DER_GR;
IF CHOICE = 'D' THEN GO TO DELETE_PR_DER_GR;
DISPLAY('INVALID CHOICE, TRY AGAIN');
GO TO PR_DER_GR;

INSERT_PR_DER_GR :
PROCESS_OWNER_NO = '    1';

J = 1;

OBTAIN CALC RECORD (PROCESS_OWNER);
OBTAIN FIRST RECORD (PROCESS) SET (PRO_SUBPARTS_PR);
DO WHILE (J <= PRCNT);
    IF (PROCESS_NAME = OWNER_NAME) THEN DO;
        GROUP_OWNER_NO = '    1';
        I = 1;
        OBTAIN CALC RECORD (GROUP_OWNER);
        DO WHILE (I <= GRCNT);
            IF GROUP_OWNER_NAME = MEMBER_NAME THEN DO;

```

```

        GROUP_NAME = GROUP_OWNER_NAME;
        GROUP_NO = GROUP_OWNER_NO;
        GR_ATTRIBUTES = GRO_ATTRIBUTES;
        STORE RECORD(GROUP);
        CALL IDMS_STATUS;
        CONNECT RECORD(GROUP) SET(GRO_SUBPARTS_GR);
        CONNECT RECORD(GROUP) SET(PR_DERIVES_GR);
        CALL IDMS_STATUS;
        DISPLAY('RELATIONSHIP ' || OWNER_NAME || ' DERIVES ' ||
MEMBER_NAME || ' HAS BEEN INSERTED ');
        I = GRCNT + 1;
        J = PRCNT + 1;
        END; /* OF IF */
    ELSE DO;
        FIND CURRENT RECORD(GROUP_OWNER);
        PUT STRING(GROUP_OWNER_NO) EDIT(I+1) (F(4));
        OBTAIN CALC RECORD(GROUP_OWNER);
        END; /* ELSE */
        I = I + 1;
    END; /* OF WHILE */
END; /* OF THEN */
ELSE DO;
    OBTAIN NEXT RECORD(PROCESS) SET(PRO_SUBPARTS_PR);
    END; /* OF ELSE */
    J = J + 1;
END; /* OF WHILE */

```



```

GO TO ENTER_SETS;

DELETE_PR_DER_GR: PROCESS_OWNER_NO = ' 1';

J = 1;

OBTAIN CALC RECORD(PROCESS_OWNER);

OBTAIN FIRST RECORD(PROCESS) SET(PRO_SUBPARTS_PR);

DO WHILE (J<=PRCNT);

    IF (OWNER_NAME = PROCESS_NAME) THEN DO;

        OBTAIN FIRST RECORD(GROUP) SET(PR_DERIVES_GR);

        DO WHILE (ERROR_STATUS = OK);

            IF (GROUP_NAME = MEMBER_NAME) THEN DO;

                DISCONNECT RECORD(GROUP) SET(PR_DERIVES_GR);

                CALL IDMS_STATUS;

                DISPLAY('RELATIONSHIP ' || OWNER_NAME || ' DERIVES ' ||
MEMBER_NAME || ' HAS BEEN DELETED ');

                J = PRCNT + 1;

                ERROR_STATUS = END_OF_SET;

            END; /* OF THEN */

            ELSE DO;

                OBTAIN NEXT RECORD(GROUP) SET(PR_DERIVES_GR);

            END; /* OF ELSE */

        END; /* OF WHILE ERROR_STATUS */

    END; /* OF IF */

    ELSE DO;

        OBTAIN NEXT RECORD(PROCESS) SET(PRO_SUBPARTS_PR);

    END; /* OF ELSE */

    J = J + 1;

END; /* OF WHILE */

```

```

GO TO ENTER_SETS;

PR_UPD_GR :   DISPLAY('YOU HAVE CHOSEN THE RELATIONSHIP ');
DISPLAY('PROCESS_UPDATES_GROUP. ');
DISPLAY('ENTER THE PROCESS NAME ') REPLY(OWNER_NAME);
DISPLAY('ENTER THE GROUP NAME ') REPLY(MEMBER_NAME);
DISPLAY('DO YOU WISH TO (I) INSERT, ');
DISPLAY(' OR (D) DELETE THIS RELATIONSHIP?  ENTER YOUR CHOICE ');
DISPLAY(' OF I, D') REPLY(CHOICE);
IF CHOICE = 'I' THEN GO TO INSERT_PR_UPD_GR;
IF CHOICE = 'D' THEN GO TO DELETE_PR_UPD_GR;
DISPLAY('INVALID CHOICE, TRY AGAIN');
GO TO PR_UPD_GR;

INSERT_PR_UPD_GR :
PROCESS_OWNER_NO = '  1';

J = 1;

OBTAIN CALC RECORD(PROCESS_OWNER);
OBTAIN FIRST RECORD(PROCESS) SET(PRO_SUBPARTS_PR);
DO WHILE (J <= PRCNT);
  IF (PROCESS_NAME = OWNER_NAME) THEN DO;
    GROUP_OWNER_NO = '  1';
    I = 1;
    OBTAIN CALC RECORD(GROUP_OWNER);
    DO WHILE (I <= GRCNT);
      IF GROUP_OWNER_NAME = MEMBER_NAME THEN DO;
        GROUP_NAME = GROUP_OWNER_NAME;
        GROUP_NO = GROUP_OWNER_NO;

```

```

GR_ATTRIBUTES = GRO_ATTRIBUTES;
STORE RECORD(GROUP);
CALL IDMS_STATUS;
CONNECT RECORD(GROUP) SET(GRO_SUBPARTS_GR);
CONNECT RECORD(GROUP) SET(PR_UPDATES_GR);
CALL IDMS_STATUS;
DISPLAY('RELATIONSHIP ' || OWNER_NAME || ' UPDATES ' ||
MEMBER_NAME || ' HAS BEEN INSERTED ');
I = GRCNT + 1;
J = PRCNT + 1;
END; /* OF IF */
ELSE DO;
FIND CURRENT RECORD(GROUP_OWNER);
PUT STRING(GROUP_OWNER_NO) EDIT(I+1) (F(4));
OBTAIN CALC RECORD(GROUP_OWNER);
END; /* ELSE */
I = I + 1;
END; /* OF WHILE */
END; /* OF THEN */
ELSE DO;
OBTAIN NEXT RECORD(PROCESS) SET(PRO_SUBPARTS_PR);
END; /* OF ELSE */
J = J + 1;
END; /* OF WHILE */

GO TO ENTER_SETS;
DELETE_PR_UPD_GR: PROCESS_OWNER_NO = ' 1';

```

```

J = 1;
OBTAIN CALC RECORD (PROCESS_OWNER);
OBTAIN FIRST RECORD (PROCESS) SET (PRO_SUBPARTS_PR);
DO WHILE (J<=PRCNT);
    IF (OWNER_NAME = PROCESS_NAME) THEN DO;
        OBTAIN FIRST RECORD (GROUP) SET (PR_UPDATES_GR);
        DO WHILE (ERROR_STATUS = OK);
            IF (GROUP_NAME = MEMBER_NAME) THEN DO;
                DISCONNECT RECORD (GROUP) SET (PR_UPDATES_GR);
                CALL IDMS_STATUS;
                DISPLAY ('RELATIONSHIP ' || OWNER_NAME || ' UPDATES ' ||
MEMBER_NAME || ' HAS BEEN DELETED ');
                J = PRCNT + 1;
                ERROR_STATUS = END_OF_SET;
            END; /* OF THEN */
            ELSE DO;
                OBTAIN NEXT RECORD (GROUP) SET (PR_UPDATES_GR);
            END; /* OF ELSE */
        END; /* OF WHILE ERROR_STATUS */
    END; /* OF IF */
    ELSE DO;
        OBTAIN NEXT RECORD (PROCESS) SET (PRO_SUBPARTS_PR);
    END; /* OF ELSE */
    J = J + 1;
END; /* OF WHILE */

GO TO ENTER_SETS;

```

```

PR_USE_GR :   DISPLAY('YOU HAVE CHOSEN THE RELATIONSHIP ');
DISPLAY('PROCESS_USES_GROUP. ');
DISPLAY('ENTER THE PROCESS NAME ') REPLY(OWNER_NAME);
DISPLAY('ENTER THE GROUP NAME ') REPLY(MEMBER_NAME);
DISPLAY('DO YOU WISH TO (I) INSERT, ');
DISPLAY(' OR (D) DELETE THIS RELATIONSHIP?  ENTER YOUR CHOICE ');
DISPLAY(' OF I, D') REPLY(CHOICE);
IF CHOICE = 'I' THEN GO TO INSERT_PR_USE_GR;
IF CHOICE = 'D' THEN GO TO DELETE_PR_USE_GR;
DISPLAY('INVALID CHOICE, TRY AGAIN');
GO TO PR_USE_GR;
INSERT_PR_USE_GR :
PROCESS_OWNER_NO = '    1';
J = 1;
OBTAIN CALC RECORD(PROCESS_OWNER);
OBTAIN FIRST RECORD(PROCESS) SET(PRO_SUBPARTS_PR);
DO WHILE (J <= PRCNT);
    IF (PROCESS_NAME = OWNER_NAME) THEN DO;
        GROUP_OWNER_NO = '    1';
        I = 1;
        OBTAIN CALC RECORD(GROUP_OWNER);
        DO WHILE (I <= GRCNT);
            IF GROUP_OWNER_NAME = MEMBER_NAME THEN DO;
                GROUP_NAME = GROUP_OWNER_NAME;
                GROUP_NO = GROUP_OWNER_NO;
                GR_ATTRIBUTES = GRO_ATTRIBUTES;
                STORE RECORD(GROUP);
            
```

```

        CALL IDMS_STATUS;

        CONNECT RECORD (GROUP) SET (GRO_SUBPARTS_GR) ;

        CONNECT RECORD (GROUP) SET (PR_USES_GR) ;

        CALL IDMS_STATUS;

        DISPLAY ('RELATIONSHIP ' || OWNER_NAME || '   USES   ' ||
MEMBER_NAME || ' HAS BEEN INSERTED ');

        I = GRCNT + 1;

        J = PRCNT + 1;

        END; /* OF IF */

    ELSE DO;

        FIND CURRENT RECORD (GROUP_OWNER);

        PUT STRING (GROUP_OWNER_NO) EDIT (I+1) (F(4));

        OBTAIN CALC RECORD (GROUP_OWNER) ;

        END; /* ELSE */

        I = I + 1;

    END; /* OF WHILE */

END; /* OF THEN */

ELSE DO;

    OBTAIN NEXT RECORD (PROCESS) SET (PRO_SUBPARTS_PR);

    END; /* OF ELSE */

    J = J + 1;

END; /* OF WHILE */

GO TO ENTER_SETS;

DELETE_PR_USE_GR: PROCESS_OWNER_NO = '   1';

J = 1;

OBTAIN CALC RECORD (PROCESS_OWNER) ;

```

```

OBTAIN FIRST RECORD (PROCESS) SET (PRO_SUBPARTS_PR);
DO WHILE (J<=PRCNT);
    IF (OWNER_NAME = PROCESS_NAME) THEN DO;
        OBTAIN FIRST RECORD (GROUP) SET (PR_USES_GR);
        DO WHILE (ERROR_STATUS = OK);
            IF (GROUP_NAME = MEMBER_NAME) THEN DO;
                DISCONNECT RECORD (GROUP) SET (PR_USES_GR);
                CALL IDMS_STATUS;
                DISPLAY ('RELATIONSHIP ' || OWNER_NAME || '      USES      ' ||
MEMBER_NAME || ' HAS BEEN DELETED ');
                J = PRCNT + 1;
                ERROR_STATUS = END_OF_SET;
            END; /* OF THEN */
            ELSE DO;
                OBTAIN NEXT RECORD (GROUP) SET (PR_USES_GR);
            END; /* OF ELSE */
        END; /* OF WHILE ERROR_STATUS */
    END; /* OF IF */
    ELSE DO;
        OBTAIN NEXT RECORD (PROCESS) SET (PRO_SUBPARTS_PR);
    END; /* OF ELSE */
    J = J + 1;
END; /* OF WHILE */

GO TO ENTER_SETS;

PR_DER_EN :   DISPLAY ('YOU HAVE CHOSEN THE RELATIONSHIP ');
DISPLAY ('PROCESS_DERIVES_ENTITY. ');

```

```

DISPLAY('ENTER THE PROCESS NAME ')REPLY(OWNER_NAME);
DISPLAY('ENTER THE ENTITY NAME ')REPLY(MEMBER_NAME);
DISPLAY('DO YOU WISH TO (I) INSERT, ');
DISPLAY(' OR (D) DELETE THIS RELATIONSHIP?  ENTER YOUR CHOICE ');
DISPLAY(' OF I, D')REPLY(CHOICE);
IF CHOICE = 'I' THEN GO TO INSERT_PR_DER_EN;
IF CHOICE = 'D' THEN GO TO DELETE_PR_DER_EN;
DISPLAY('INVALID CHOICE, TRY AGAIN');
GO TO PR_DER_EN;
INSERT_PR_DER_EN :
PROCESS_OWNER_NO = '    1';
J = 1;
OBTAIN CALC RECORD(PROCESS_OWNER);
OBTAIN FIRST RECORD(PROCESS) SET(PRO_SUBPARTS_PR);
DO WHILE (J <= PRCNT);
    IF (PROCESS_NAME = OWNER_NAME) THEN DO;
        ENTITY_OWNER_NO = '    1';
        I = 1;
        OBTAIN CALC RECORD(ENTITY_OWNER);
        DO WHILE (I <= ENCNT);
            IF ENTITY_OWNER_NAME = MEMBER_NAME THEN DO;
                ENTITY_NAME = ENTITY_OWNER_NAME;
                ENTITY_NO = ENTITY_OWNER_NO;
                EN_ATTRIBUTES = ENO_ATTRIBUTES;
                STORE RECORD(ENTITY);
                CALL IDMS_STATUS;
                CONNECT RECORD(ENTITY) SET(ENO_SUBPARTS_EN);
            
```



```

        CONNECT RECORD(ENTITY) SET (PR_DERIVES_EN);

        CALL IDMS_STATUS;

        DISPLAY('RELATIONSHIP ' || OWNER_NAME || ' DERIVES ' ||
MEMBER_NAME || ' HAS BEEN INSERTED ');

        I = ENCNT +1;

        J = PRCNT + 1;

        END; /* OF IF */

    ELSE DO;

        FIND CURRENT RECORD(ENTITY_OWNER);

        PUT STRING(ENTITY_OWNER_NO) EDIT(I+1) (F(4));

        OBTAIN CALC RECORD(ENTITY_OWNER);

        END; /* ELSE */

        I = I + 1;

    END; /* OF WHILE */

END; /* OF THEN */

ELSE DO;

    OBTAIN NEXT RECORD(PROCESS) SET (PRO_SUBPARTS_PR);

    END; /* OF ELSE */

    J = J + 1;

END; /* OF WHILE */

GO TO ENTER_SETS;

DELETE_PR_DER_EN: PROCESS_OWNER_NO ='    1';

J = 1;

OBTAIN CALC RECORD(PROCESS_OWNER);

OBTAIN FIRST RECORD(PROCESS) SET (PRO_SUBPARTS_PR);

DO WHILE (J<=PRCNT);

```

```

IF (OWNER_NAME = PROCESS_NAME) THEN DO;
    OBTAIN FIRST RECORD(ENTITY) SET(PR_DERIVES_EN);
    DO WHILE (ERROR_STATUS = OK);
        IF (ENTITY_NAME = MEMBER_NAME) THEN DO;
            DISCONNECT RECORD(ENTITY) SET(PR_DERIVES_EN);
            CALL IDMS_STATUS;
            DISPLAY('RELATIONSHIP ' || OWNER_NAME || ' DERIVES ' ||
MEMBER_NAME || ' HAS BEEN DELETED ');
            J = PRCNT + 1;
            ERROR_STATUS = END_OF_SET;
        END; /* OF THEN */
        ELSE DO;
            OBTAIN NEXT RECORD(ENTITY) SET(PR_DERIVES_EN);
        END; /* OF ELSE */
    END; /* OF WHILE ERROR_STATUS */
END; /* OF IF */
ELSE DO;
    OBTAIN NEXT RECORD(PROCESS) SET(PRO_SUBPARTS_PR);
END; /* OF ELSE */
J = J + 1;
END; /* OF WHILE */

GO TO ENTER_SETS;

PR_UPD_EN :   DISPLAY('YOU HAVE CHOSEN THE RELATIONSHIP ');
DISPLAY('PROCESS_UPDATES_ENTITY. ');
DISPLAY('ENTER THE PROCESS NAME ') REPLY(OWNER_NAME);
DISPLAY('ENTER THE ENTITY NAME ') REPLY(MEMBER_NAME);

```

```

DISPLAY('DO YOU WISH TO (I) INSERT, ');
DISPLAY(' OR (D) DELETE THIS RELATIONSHIP? ENTER YOUR CHOICE ');
DISPLAY(' OF I, D')REPLY(CHOICE);
IF CHOICE = 'I' THEN GO TO INSERT_PR_UPD_EN;
IF CHOICE = 'D' THEN GO TO DELETE_PR_UPD_EN;
DISPLAY('INVALID CHOICE, TRY AGAIN');
GO TO PR_UPD_EN;
INSERT_PR_UPD_EN :
PROCESS_OWNER_NO = ' 1';
J = 1;
OBTAIN CALC RECORD(PROCESS_OWNER);
OBTAIN FIRST RECORD(PROCESS) SET(PRO_SUBPARTS_PR);
DO WHILE (J <= PRCNT);
    IF (PROCESS_NAME = OWNER_NAME) THEN DO;
        ENTITY_OWNER_NO = ' 1';
        I = 1;
        OBTAIN CALC RECORD(ENTITY_OWNER);
        DO WHILE (I <= ENCNT);
            IF ENTITY_OWNER_NAME = MEMBER_NAME THEN DO;
                ENTITY_NAME = ENTITY_OWNER_NAME;
                ENTITY_NO = ENTITY_OWNER_NO;
                EN_ATTRIBUTES = ENO_ATTRIBUTES;
                STORE RECORD(ENTITY);
                CALL IDMS_STATUS;
                CONNECT RECORD(ENTITY) SET(ENO_SUBPARTS_EN);
                CONNECT RECORD(ENTITY) SET(PR_UPDATES_EN);
                CALL IDMS_STATUS;
            
```

```

        DISPLAY('RELATIONSHIP ' || OWNER_NAME || '  UPDATES ' ||
MEMBER_NAME || ' HAS BEEN INSERTED ');

        I = ENCNT +1;

        J = PRCNT + 1;

        END; /* OF IF */

ELSE DO;

        FIND CURRENT RECORD(ENTITY_OWNER);

        PUT STRING(ENTITY_OWNER_NO) EDIT(I+1) (F(4));

        OBTAIN CALC RECORD(ENTITY_OWNER);

        END; /* ELSE */

        I = I + 1;

        END; /* OF WHILE */

END; /* OF THEN */

ELSE DO;

        OBTAIN NEXT RECORD(PROCESS) SET(PRO_SUBPARTS_PR);

        END; /* OF ELSE */

        J = J + 1;

        END; /* OF WHILE */

GO TO ENTER_SETS;

DELETE_PR_UPD_EN: PROCESS_OWNER_NO ='  1';

J = 1;

OBTAIN CALC RECORD(PROCESS_OWNER);

OBTAIN FIRST RECORD(PROCESS) SET(PRO_SUBPARTS_PR);

DO WHILE (J<=PRCNT);

        IF (OWNER_NAME = PROCESS_NAME) THEN DO;

                OBTAIN FIRST RECORD(ENTITY) SET(PR_UPDATES_EN);

```

```

DO WHILE (ERROR_STATUS = OK);
  IF (ENTITY_NAME = MEMBER_NAME) THEN DO;
    DISCONNECT RECORD(ENTITY) SET (PR_UPDATES_EN);
    CALL IDMS_STATUS;
    DISPLAY('RELATIONSHIP ' || OWNER_NAME || ' UPDATES ' ||
MEMBER_NAME || ' HAS BEEN DELETED ');
    J = PRCNT + 1;
    ERROR_STATUS = END_OF_SET;
  END; /* OF THEN */
  ELSE DO;
    OBTAIN NEXT RECORD(ENTITY) SET (PR_UPDATES_EN);
    END; /* OF ELSE */
  END; /* OF WHILE ERROR_STATUS */
END; /* OF IF */
ELSE DO;
  OBTAIN NEXT RECORD(PROCESS) SET (PRO_SUBPARTS_PR);
END; /* OF ELSE */
J = J + 1;
END; /* OF WHILE */

GO TO ENTER_SETS;

PR_USE_EN :   DISPLAY('YOU HAVE CHOSEN THE RELATIONSHIP ');
DISPLAY('PROCESS_USES_ENTITY. ');
DISPLAY('ENTER THE PROCESS NAME ') REPLY(OWNER_NAME);
DISPLAY('ENTER THE ENTITY NAME ') REPLY(MEMBER_NAME);
DISPLAY('DO YOU WISH TO (I) INSERT, ');
DISPLAY(' OR (D) DELETE THIS RELATIONSHIP?  ENTER YOUR CHOICE ');

```

```

DISPLAY(' OF I, D')REPLY(CHOICE) ;
IF CHOICE = 'I' THEN GO TO INSERT_PR_USE_EN;
IF CHOICE = 'D' THEN GO TO DELETE_PR_USE_EN;
DISPLAY('INVALID CHOICE, TRY AGAIN');
GO TO PR_USE_EN;
INSERT_PR_USE_EN :
PROCESS_OWNER_NO = '    1';
J = 1;
OBTAIN CALC RECORD (PROCESS_OWNER) ;
OBTAIN FIRST RECORD (PROCESS) SET (PRO_SUBPARTS_PR) ;
DO WHILE (J <= PRCNT) ;
    IF (PROCESS_NAME = OWNER_NAME) THEN DO ;
        ENTITY_OWNER_NO = '    1';
        I = 1;
        OBTAIN CALC RECORD (ENTITY_OWNER) ;
        DO WHILE (I <= ENCNT) ;
            IF ENTITY_OWNER_NAME = MEMBER_NAME THEN DO ;
                ENTITY_NAME = ENTITY_OWNER_NAME;
                ENTITY_NO = ENTITY_OWNER_NO;
                EN_ATTRIBUTES = ENO_ATTRIBUTES;
                STORE RECORD (ENTITY) ;
                CALL IDMS_STATUS;
                CONNECT RECORD (ENTITY) SET (ENO_SUBPARTS_EN) ;
                CONNECT RECORD (ENTITY) SET (PR_USES_EN) ;
                CALL IDMS_STATUS;
                DISPLAY('RELATIONSHIP ' || OWNER_NAME || '    USES    ' ||
MEMBER_NAME || ' HAS BEEN INSERTED ');

```

```

        I = ENCNT + 1;

        J = PRCNT + 1;

        END; /* OF IF */

    ELSE DO;

        FIND CURRENT RECORD(ENTITY_OWNER);

        PUT STRING(ENTITY_OWNER_NO) EDIT(I+1) (F(4));

        OBTAIN CALC RECORD(ENTITY_OWNER);

        END; /* ELSE */

        I = I + 1;

    END; /* OF WHILE */

END; /* OF THEN */

ELSE DO;

    OBTAIN NEXT RECORD(PROCESS) SET(PRO_SUBPARTS_PR);

    END; /* OF ELSE */

    J = J + 1;

END; /* OF WHILE */

GO TO ENTER_SETS;

DELETE_PR_USE_EN: PROCESS_OWNER_NO = '    1';

J = 1;

OBTAIN CALC RECORD(PROCESS_OWNER);

OBTAIN FIRST RECORD(PROCESS) SET(PRO_SUBPARTS_PR);

DO WHILE (J<=PRCNT);

    IF (OWNER_NAME = PROCESS_NAME) THEN DO;

        OBTAIN FIRST RECORD(ENTITY) SET(PR_USES_EN);

        DO WHILE (ERROR_STATUS = OK);

            IF (ENTITY_NAME = MEMBER_NAME) THEN DO;

```

```

        DISCONNECT RECORD(ENTITY) SET (PR_USES_EN);

        CALL IDMS_STATUS;

        DISPLAY('RELATIONSHIP ' || OWNER_NAME || '   USES   ' ||
MEMBER_NAME || ' HAS BEEN DELETED ');

        J = PRCNT + 1;

        ERROR_STATUS = END_OF_SET;

    END; /* OF THEN */

    ELSE DO;

        OBTAIN NEXT RECORD(ENTITY) SET (PR_USES_EN);

        END; /* OF ELSE */

    END; /* OF WHILE ERROR_STATUS */

    END; /* OF IF */

    ELSE DO;

        OBTAIN NEXT RECORD(PROCESS) SET (PRO_SUBPARTS_PR);

        END; /* OF ELSE */

        J = J + 1;

    END; /* OF WHILE */

GO TO ENTER_SETS;

IN_CON_EL: DISPLAY('YOU HAVE CHOSEN THE RELATIONSHIP ');

DISPLAY('INPUT CONSISTS OF ELEMENT. ');

DISPLAY('ENTER THE INPUT NAME ') REPLY(OWNER_NAME);

DISPLAY('ENTER THE ELEMENT NAME ') REPLY(MEMBER_NAME);

DISPLAY('DO YOU WISH TO (I) INSERT, ');

DISPLAY(' OR (D) DELETE THIS RELATIONSHIP?  ENTER YOUR CHOICE ');

DISPLAY(' OF I, D') REPLY(CHOICE);

IF CHOICE = 'I' THEN GO TO INSERT_IN_CON_EL;

```



```

IF CHOICE = 'D' THEN GO TO DELETE_IN_CON_EL;
DISPLAY('INVALID CHOICE, TRY AGAIN');
GO TO IN_CON_EL;
INSERT_IN_CON_EL:
INPUT_OWNER_NO = '    1';
J = 1;
OBTAIN CALC RECORD (INPUT_OWNER) ;
DO   WHILE (J <= INCNT) ;
IF (INPUT_OWNER_NAME = OWNER_NAME) THEN DO;
    ELEMENT_OWNER_NO = '    1';
    I = 1;
    OBTAIN CALC RECORD (ELEMENT_OWNER) ;
    DO WHILE (I <= ELCNT) ;
        IF (ELEMENT_OWNER_NAME = MEMBER_NAME) THEN DO;
            ELEMENT_NO =ELEMENT_OWNER_NO;
            ELEMENT_NAME = ELEMENT_OWNER_NAME;
            EL_ATTRIBUTES = ELO_ATTRIBUTES;
            STORE RECORD (ELEMENT) ;
            CALL IDMS_STATUS;
            CONNECT RECORD (ELEMENT) SET {ELO_SUBPARTS_EL} ;
            CALL IDMS_STATUS;
            IF SET (INO_SUBPARTS_IN) EMPTY THEN DO;
                INPUT_NAME = INPUT_OWNER_NAME;
                INPUT_NO = INPUT_OWNER_NO;
                IN_ATTRIBUTES = INO_ATTRIBUTES;
                STORE RECORD (INPUT) ;
                CALL IDMS_STATUS;

```

```

        CONNECT RECORD(INPUT) SET(INO_SUBPARTS_IN);

        CALL IDMS_STATUS;

    END; /* OF IF EMPTY */

ELSE DO;

    FIND FIRST RECORD(INPUT) SET(INO_SUBPARTS_IN);

END; /* OF ELSE DO */

    CONNECT RECORD(ELEMENT) SET(IN_CONSISTS_EL);

    CALL IDMS_STATUS;

    DISPLAY('RELATIONSHIP ' || OWNER_NAME || ' CONSISTS ' ||
MEMBER_NAME || ' HAS BEEN INSERTED ');

    I = ELCNT + 1;

    J = INCNT + 1;

    END; /* OF IF */

ELSE DO;

    FIND CURRENT RECORD(ELEMENT_OWNER);

    PUT STRING(ELEMENT_OWNER_NO) EDIT(I+1) (F(4));

    OBTAIN CALC RECORD(ELEMENT_OWNER);

    END; /* OF ELSE */

    I = I + 1;

    END; /* OF WHILE */

END; /* OF IF-THEN */

ELSE DO;

    PUT STRING(INPUT_OWNER_NO) EDIT(J+1) (F(4));

    OBTAIN CALC RECORD(INPUT_OWNER);

END; /* OF ELSE */

    J = J + 1;

END; /* OF WHILE */

```

```

GO TO ENTER_SETS;

DELETE_IN_CON_EL:

INPUT_OWNER_NO = ' 1';

I = 1;

OBTAIN CALC RECORD (INPUT_OWNER) ;

DO WHILE (I<=INCNT) ;

    IF (INPUT_OWNER_NAME = OWNER_NAME) THEN DO;

        OBTAIN FIRST RECORD (INPUT) SET (INO_SUBPARTS_IN) ;

        CALL IDMS_STATUS;

        DO WHILE (ERROR_STATUS = OK) ;

            IF NOT SET (IN_CONSISTS_EL) EMPTY THEN DO;

                OBTAIN FIRST RECORD (ELEMENT) SET (IN_CONSISTS_EL) ;

                DO WHILE (ERROR_STATUS = OK) ;

                    IF (ELEMENT_NAME = MEMBER_NAME) THEN DO;

                        DISCONNECT RECORD (ELEMENT) SET (IN_CONSISTS_EL) ;

                        CALL IDMS_STATUS;

                        ERASE RECORD (ELEMENT) ;

                        CALL IDMS_STATUS;

                        DISPLAY ('RELATIONSHIP ' || OWNER_NAME || ' CONSISTS ' ||
MEMBER_NAME || ' HAS BEEN DELETED ') ;

                        I = INCNT + 1;

                        ERROR_STATUS = END_OF_SET;

                        END; /* OF IF */

                    ELSE DO;

                        OBTAIN NEXT RECORD (ELEMENT) SET (IN_CONSISTS_EL) ;

                        END; /* OF ELSE */

                    END; /* OF WHILE */

                DO WHILE (ERROR_STATUS = OK) ;

                    IF NOT SET (IN_CONSISTS_EL) EMPTY THEN DO;

                        OBTAIN FIRST RECORD (ELEMENT) SET (IN_CONSISTS_EL) ;

                        DISCONNECT RECORD (ELEMENT) SET (IN_CONSISTS_EL) ;

                        CALL IDMS_STATUS;

                        ERASE RECORD (ELEMENT) ;

                        CALL IDMS_STATUS;

                        DISPLAY ('RELATIONSHIP ' || OWNER_NAME || ' CONSISTS ' ||
MEMBER_NAME || ' HAS BEEN DELETED ') ;

                        I = INCNT + 1;

                        ERROR_STATUS = END_OF_SET;

                        END; /* OF IF */

                    ELSE DO;

                        OBTAIN NEXT RECORD (ELEMENT) SET (IN_CONSISTS_EL) ;

                        END; /* OF ELSE */

                    END; /* OF WHILE */

                END; /* OF IF */

            END; /* OF WHILE */

        END; /* OF IF */

    END; /* OF WHILE */

```

```

END; /* OF IF */
ELSE DO;
    OBTAIN NEXT RECORD(INPUT) SET(INO_SUBPARTS_IN);
END; /* OF ELSE */
END; /* OF WHILE */
END; /* OF IF *NAME = *NAME */
ELSE DO;
    PUT STRING(INPUT_OWNER_NO) EDIT(I+1) (F(4));
    OBTAIN CALC RECORD(INPUT_OWNER);
END; /* OF ELSE */
I = I + 1;
END; /* OF WHILE */
GO TO ENTER_SETS;
IN_CON_GR: DISPLAY('YOU HAVE CHOSEN THE RELATIONSHIP ');
DISPLAY('INPUT_CONSISTS_OF_GROUP. ');
DISPLAY('ENTER THE INPUT NAME ') REPLY(OWNER_NAME);
DISPLAY('ENTER THE GROUP NAME ') REPLY(MEMBER_NAME);
DISPLAY('DO YOU WISH TO (I) INSERT, ');
DISPLAY(' OR (D) DELETE THIS RELATIONSHIP? ENTER YOUR CHOICE ');
DISPLAY(' OF I, D') REPLY(CHOICE);
IF CHOICE = 'I' THEN GO TO INSERT_IN_CON_GR;
IF CHOICE = 'D' THEN GO TO DELETE_IN_CON_GR;
DISPLAY('INVALID CHOICE, TRY AGAIN');
GO TO IN_CON_GR;
INSERT_IN_CON_GR:
INPUT_OWNER_NO = ' 1';
J = 1;

```

```

OBTAIN CALC RECORD (INPUT_OWNER) ;

DO WHILE (J <= INCNT) ;

IF (INPUT_OWNER_NAME = OWNER_NAME) THEN DO ;

    GROUP_OWNER_NO = '    1' ;

    I = 1 ;

    OBTAIN CALC RECORD (GROUP_OWNER) ;

    DO WHILE (I <= GRCNT) ;

        IF (GROUP_OWNER_NAME = MEMBER_NAME) THEN DO ;

            GROUP_NO = GROUP_OWNER_NO ;

            GROUP_NAME = GROUP_OWNER_NAME ;

            GR_ATTRIBUTES = GRO_ATTRIBUTES ;

            STORE RECORD (GROUP) ;

            CALL IDMS_STATUS ;

            CONNECT RECORD (GROUP) SET (GRO_SUBPARTS_GR) ;

            CALL IDMS_STATUS ;

            IF SET (INO_SUBPARTS_IN) EMPTY THEN DO ;

                INPUT_NAME = INPUT_OWNER_NAME ;

                INPUT_NO = INPUT_OWNER_NO ;

                IN_ATTRIBUTES = INO_ATTRIBUTES ;

                STORE RECORD (INPUT) ;

                CALL IDMS_STATUS ;

                CONNECT RECORD (INPUT) SET (INO_SUBPARTS_IN) ;

                CALL IDMS_STATUS ;

            END ; /* OF IF EMPTY */

        ELSE DO ;

            FIND FIRST RECORD (INPUT) SET (INO_SUBPARTS_IN) ;

        END ; /* OF ELSE DO */

```

```

CONNECT RECORD(GROUP) SET(IN_CONSISTS_GR);

CALL IDMS_STATUS;

DISPLAY('RELATIONSHIP ' || OWNER_NAME || ' CONSISTS ' ||
MEMBER_NAME || ' HAS BEEN INSERTED ');

I = GRCNT + 1;

J = INCNT + 1;

END; /* OF IF */

ELSE DO;

FIND CURRENT RECORD(GROUP_OWNER);

PUT STRING(GROUP_OWNER_NO) EDIT(I+1) (F(4));

OBTAIN CALC RECORD(GROUP_OWNER);

END; /* ELSE */

I = I + 1;

END; /* OF WHILE */

END; /* OF IF-THEN */

ELSE DO;

PUT STRING(INPUT_OWNER_NO) EDIT(J+1) (F(4));

OBTAIN CALC RECORD(INPUT_OWNER);

END; /* OF ELSE DO */

J = J + 1;

END; /* OF WHILE */

GO TO ENTER_SETS;

DELETE_IN_CON_GR:

INPUT_OWNER_NO = ' 1';

I = 1;

OBTAIN CALC RECORD(INPUT_OWNER);

DO WHILE (I<=INCNT);

```

```

IF (INPUT_OWNER_NAME = OWNER_NAME) THEN DO;
  OBTAIN FIRST RECORD (INPUT) SET (INO_SUBPARTS_IN);
  DO WHILE (ERROR_STATUS = OK);
    IF NOT SET (IN_CONSISTS_GR) EMPTY THEN DO;
      OBTAIN FIRST RECORD (GROUP) SET (IN_CONSISTS_GR);
      DO WHILE (ERROR_STATUS = OK);
        IF (GROUP_NAME = MEMBER_NAME) THEN DO;
          DISCONNECT RECORD (GROUP) SET (IN_CONSISTS_GR);
          CALL IDMS_STATUS;
          DISPLAY ('RELATIONSHIP ' || OWNER_NAME || ' CONSISTS ' ||
MEMBER_NAME || ' HAS BEEN DELETED ');
          I = INCNT + 1;
          ERROR_STATUS = END_OF_SET;
          END; /* OF IF */
        ELSE DO;
          OBTAIN NEXT RECORD (GROUP) SET (IN_CONSISTS_GR);
          END; /* OF ELSE */
        END; /* OF WHILE */
      END; /* OF IF */
    ELSE DO;
      OBTAIN NEXT RECORD (INPUT) SET (INO_SUBPARTS_IN);
      END; /* OF ELSE */
    END; /* OF WHILE */
  END; /* OF IF *NAME = *NAME */
ELSE DO;
  PUT STRING (INPUT_OWNER_NO) EDIT (I+1) (F(4));
  OBTAIN CALC RECORD (INPUT_OWNER);

```

```

END; /* OF ELSE */
I = I + 1;
END; /* OF WHILE */
GO TO ENTER_SETS;
GR_CON_EL: DISPLAY('YOU HAVE CHOSEN THE RELATIONSHIP ');
DISPLAY('GROUP_CONSISTS_OF_ELEMENT. ');
DISPLAY('ENTER THE GROUP NAME ') REPLY(OWNER_NAME);
DISPLAY('ENTER THE ELEMENT NAME ') REPLY(MEMBER_NAME);
DISPLAY('DO YOU WISH TO (I) INSERT, ');
DISPLAY(' OR (D) DELETE THIS RELATIONSHIP? ENTER YOUR CHOICE ');
DISPLAY(' OF I, D') REPLY(CHOICE);
IF CHOICE = 'I' THEN GO TO INSERT_GR_CON_EL;
IF CHOICE = 'D' THEN GO TO DELETE_GR_CON_EL;
DISPLAY('INVALID CHOICE, TRY AGAIN');
GO TO GR_CON_EL;
INSERT_GR_CON_EL:
GROUP_OWNER_NO = ' 1';
J = 1;
OBTAIN CALC RECORD(GROUP_OWNER);
DO WHILE (J <= GRCNT);
IF (GROUP_OWNER_NAME = OWNER_NAME) THEN DO;
ELEMENT_OWNER_NO = ' 1';
I = 1;
OBTAIN CALC RECORD(ELEMENT_OWNER);
DO WHILE (I <= ELCNT);
IF (ELEMENT_OWNER_NAME = MEMBER_NAME) THEN DO;
ELEMENT_NO = ELEMENT_OWNER_NO;

```



```

ELEMENT_NAME = ELEMENT_OWNER_NAME;
EL_ATTRIBUTES = ELO_ATTRIBUTES;
STORE RECORD (ELEMENT);
CALL IDMS_STATUS;
CONNECT RECORD(ELEMENT) SET (ELO_SUBPARTS_EL);
CALL IDMS_STATUS;

IF SET (GRO_SUBPARTS_GR) EMPTY THEN DO;
    GROUP_NAME = GROUP_OWNER_NAME;
    GROUP_NO = GROUP_OWNER_NO;
    GR_ATTRIBUTES = GRO_ATTRIBUTES;
    STORE RECORD (GROUP);
    CALL IDMS_STATUS;
    CONNECT RECORD (GROUP) SET (GRO_SUBPARTS_GR);
    CALL IDMS_STATUS;
END; /* OF IF EMPTY */
ELSE DO;
    FIND FIRST RECORD (GROUP) SET (GRO_SUBPARTS_GR);
END; /* OF ELSE DO */

CONNECT RECORD (ELEMENT) SET (GR_CONSISTS_EL);
CALL IDMS_STATUS;
DISPLAY ('RELATIONSHIP ' || OWNER_NAME || ' CONSISTS ' ||
MEMBER_NAME || ' HAS BEEN INSERTED ');

I = ELCNT + 1;
J = GRCNT + 1;
END; /* OF IF */
ELSE DO;
    FIND CURRENT RECORD (GROUP_OWNER);

```

```

        PUT STRING(ELEMENT_OWNER_NO) EDIT(I+1) (F(4));
        OBTAIN CALC RECORD(ELEMENT_OWNER);

        END; /* ELSE */

        I = I + 1;

        END; /* OF WHILE */

    END; /* OF IF-THEN */

ELSE DO;

    PUT STRING(GROUP_OWNER_NO) EDIT(J+1) (F(4));

    OBTAIN CALC RECORD(GROUP_OWNER);

    END; /* OF ELSE DO */

    J = J + 1;

END; /* OF WHILE */

GO TO ENTER_SETS;

DELETE_GR_CON_EL:

GROUP_OWNER_NO = '    1';

I = 1;

OBTAIN CALC RECORD(GROUP_OWNER);

DO WHILE (I<=GRCNT);

    IF (GROUP_OWNER_NAME = OWNER_NAME) THEN DO;

        OBTAIN FIRST RECORD(GROUP) SET(GRO_SUBPARTS_GR);

        DO WHILE (ERROR_STATUS = OK);

            IF NOT SET(GR_CONSISTS_EL) EMPTY THEN DO;

                OBTAIN FIRST RECORD(ELEMENT) SET(GR_CONSISTS_EL);

                DO WHILE (ERROR_STATUS = OK);

                    IF (ELEMENT_NAME = MEMBER_NAME) THEN DO;

                        DISCONNECT RECORD(ELEMENT) SET(GR_CONSISTS_EL);

                        CALL IDMS_STATUS;

```

```

        ERASE RECORD(ELEMENT);

        CALL IDMS_STATUS;

        DISPLAY('RELATIONSHIP ' || OWNER_NAME || ' CONSISTS ' ||
MEMBER_NAME || ' HAS BEEN DELETED ');

        I = GRCNT + 1;

        ERROR_STATUS = END_OF_SET;

        END; /* OF IF */

    ELSE DO;

        OBTAIN NEXT RECORD(ELEMENT) SET(GR_CONSISTS_EL);

        END; /* OF ELSE */

    END; /* OF WHILE */

END; /* OF IF */

ELSE DO;

    OBTAIN NEXT RECORD(GROUP) SET(GRO_SUBPARTS_GR);

    END; /* OF ELSE */

END; /* OF WHILE */

END; /* OF IF *NAME = *NAME */

ELSE DO;

    PUT STRING(GROUP_OWNER_NO) EDIT(I+1) (F(4));

    OBTAIN CALC RECORD(GROUP_OWNER);

    END; /* OF ELSE */

    I = I + 1;

END; /* OF WHILE */

GO TO ENTER_SETS;

EN_CON_EL: DISPLAY('YOU HAVE CHOSEN THE RELATIONSHIP ');

DISPLAY('ENTITY_CONSISTS_OF_ELEMENT. ');

DISPLAY('ENTER THE ENTITY NAME ') REPLY(OWNER_NAME);

```

```

DISPLAY('ENTER THE ELEMENT NAME ')REPLY(MEMBER_NAME);
DISPLAY('DO YOU WISH TO (I) INSERT, ');
DISPLAY(' OR (D) DELETE THIS RELATIONSHIP? ENTER YOUR CHOICE ');
DISPLAY(' OF I, D')REPLY(CHOICE);
IF CHOICE = 'I' THEN GO TO INSERT_EN_CON_EL;
IF CHOICE = 'D' THEN GO TO DELETE_EN_CON_EL;
DISPLAY('INVALID CHOICE, TRY AGAIN');
GO TO EN_CON_EL;
INSERT_EN_CON_EL:
ENTITY_OWNER_NO = ' 1';
J = 1;
OBTAIN CALC RECORD(ENTITY_OWNER);
DO WHILE (J <= ENCNT);
IF (ENTITY_OWNER_NAME = OWNER_NAME) THEN DO;
ELEMENT_OWNER_NO = ' 1';
I = 1;
OBTAIN CALC RECORD(ELEMENT_OWNER);
DO WHILE (I <= ELCNT);
IF (ELEMENT_OWNER_NAME = MEMBER_NAME) THEN DO;
ELEMENT_NO =ELEMENT_OWNER_NO;
ELEMENT_NAME = ELEMENT_OWNER_NAME;
EL_ATTRIBUTES = ELO_ATTRIBUTES;
STORE RECORD (ELEMENT);
CALL IDMS_STATUS;
CONNECT RECORD(ELEMENT) SET(ELO_SUBPARTS_EL);
CALL IDMS_STATUS;
IF SET(ENO_SUBPARTS_EN) EMPTY THEN DO;

```

```

    ENTITY_NAME = ENTITY_OWNER_NAME;

    ENTITY_NO = ENTITY_OWNER_NO;

    EN_ATTRIBUTES = ENO_ATTRIBUTES;

    STORE RECORD(ENTITY);

    CALL IDMS_STATUS;

    CONNECT RECORD(ENTITY) SET(ENO_SUBPARTS_EN);

    CALL IDMS_STATUS;

END; /* OF IF EMPTY */

ELSE DO;

    FIND FIRST RECORD(ENTITY) SET(ENO_SUBPARTS_EN);

END; /* OF ELSE DO */

    CONNECT RECORD(ELEMENT) SET(EN_CONSISTS_EL);

    CALL IDMS_STATUS;

    DISPLAY('RELATIONSHIP ' || OWNER_NAME || ' CONSISTS ' ||
MEMBER_NAME || ' HAS BEEN INSERTED ');

    I = ELCNT + 1;

    J = ENCNT + 1;

END; /* OF IF */

ELSE DO;

    FIND CURRENT RECORD(ENTITY_OWNER);

    PUT STRING(ELEMENT_OWNER_NO) EDIT(I+1) (F(4));

    OBTAIN CALC RECORD(ELEMENT_OWNER);

END; /* ELSE */

    I = I + 1;

END; /* OF WHILE */

END; /* OF IF-THEN */

ELSE DO;

```

```

        PUT STRING(ENTITY_OWNER_NO) EDIT(J+1) (F(4));

        OBTAIN CALC RECORD(ENTITY_OWNER);

    END; /* OF ELSE DO */

    J = J + 1;

END; /* OF WHILE */

GO TO ENTER_SETS;

DELETE_EN_CON_EL:

ENTITY_OWNER_NO = '    1';

I = 1;

OBTAIN CALC RECORD(ENTITY_OWNER);

DO WHILE (I<=ENCNT);

    IF (ENTITY_OWNER_NAME = OWNER_NAME) THEN DO;

        OBTAIN FIRST RECORD(ENTITY) SET(ENO_SUBPARTS_EN);

        DO WHILE (ERROR_STATUS = OK);

            IF NOT SET(EN_CONSISTS_EL) EMPTY THEN DO;

                OBTAIN FIRST RECORD(ELEMENT) SET(EN_CONSISTS_EL);

                DO WHILE (ERROR_STATUS = OK);

                    IF(ELEMENT_NAME = MEMBER_NAME) THEN DO;

                        DISCONNECT RECORD(ELEMENT) SET(EN_CONSISTS_EL);

                        CALL IDMS_STATUS;

                        ERASE RECORD(ELEMENT);

                        CALL IDMS_STATUS;

                        DISPLAY(' RELATIONSHIP ' || OWNER_NAME || ' CONSISTS ' ||
MEMBER_NAME || ' HAS BEEN DELETED ');

                        I = ENCNT + 1;

                        ERROR_STATUS = END_OF_SET;

                    END; /* OF IF */

```

```

        ELSE DO;

            OBTAIN NEXT RECORD(ELEMENT) SET(EN_CONSISTS_EL);

            END; /* OF ELSE */

        END; /* OF WHILE */

    END; /* OF IF */

    ELSE DO;

        OBTAIN NEXT RECORD(ENTITY) SET(ENO_SUBPARTS_EN);

        END; /* OF ELSE */

    END; /* OF WHILE */

END; /* OF IF *NAME = *NAME */

ELSE DO;

    PUT STRING(ENTITY_OWNER_NO) EDIT(I+1) (F(4));

    OBTAIN CALC RECORD(ENTITY_OWNER);

    END; /* OF ELSE */

    I = I + 1;

END; /* OF WHILE */

GO TO ENTER_SETS;

OU_CON_EL: DISPLAY('YOU HAVE CHOSEN THE RELATIONSHIP ');

DISPLAY('OUTPUT_CONSISTS_OF_ELEMENT. ');

DISPLAY('ENTER THE OUTPUT NAME ') REPLY(OWNER_NAME);

DISPLAY('ENTER THE ELEMENT NAME ') REPLY(MEMBER_NAME);

DISPLAY('DO YOU WISH TO (I) INSERT, ');

DISPLAY(' OR (D) DELETE THIS RELATIONSHIP? ENTER YOUR CHOICE ');

DISPLAY(' OF I, D') REPLY(CHOICE);

IF CHOICE = 'I' THEN GO TO INSERT_OU_CON_EL;

IF CHOICE = 'D' THEN GO TO DELETE_OU_CON_EL;

DISPLAY('INVALID CHOICE, TRY AGAIN');

```

```

GO TO OU_CON_EL;
INSERT_OU_CON_EL:
OUTPUT_OWNER_NO = '  1';
J = 1;
OBTAIN CALC RECORD(OUTPUT_OWNER);
DO WHILE (J <= OUCNT);
IF (OUTPUTOWNER_NAME = OWNER_NAME) THEN DO;
    ELEMENT_OWNER_NO = '  1';
    I = 1;
    OBTAIN CALC RECORD(ELEMENT_OWNER);
    DO WHILE (I <= ELCNT);
        IF (ELEMENT_OWNER_NAME = MEMBER_NAME) THEN DO;
            ELEMENT_NO = ELEMENT_OWNER_NO;
            ELEMENT_NAME = ELEMENT_OWNER_NAME;
            EL_ATTRIBUTES = ELO_ATTRIBUTES;
            STORE RECORD (ELEMENT);
            CALL IDMS_STATUS;
            CONNECT RECORD(ELEMENT) SET (ELO_SUBPARTS_EL);
            CALL IDMS_STATUS;
            IF SET(OUO_SUBPARTS_OU) EMPTY THEN DO;
                OUTPUT_NAME = OUTPUTOWNER_NAME;
                OUTPUT_NO = OUTPUT_OWNER_NO;
                OU_ATTRIBUTES = OUO_ATTRIBUTES;
                STORE RECORD(OUTPUT);
                CALL IDMS_STATUS;
                CONNECT RECORD(OUTPUT) SET(OUO_SUBPARTS_OU);
                CALL IDMS_STATUS;
            
```



```

        END; /* OF IF EMPTY */
    ELSE DO;
        FIND FIRST RECORD(OUTPUT) SET(OUO_SUBPARTS_OU);
    END; /* OF ELSE DO */
        CONNECT RECORD(ELEMENT) SET(OU_CONSISTS_EL);
        CALL IDMS_STATUS;
        DISPLAY('RELATIONSHIP ' || OWNER_NAME || ' CONSISTS ' ||
MEMBER_NAME || ' HAS BEEN INSERTED ');
        I = ELCNT + 1;
        J = OUCNT + 1;
        END; /* OF IF */
    ELSE DO;
        FIND CURRENT RECORD(OUTPUT_OWNER);
        PUT STRING(ELEMENT_OWNER_NO) EDIT(I+1) (F(4));
        OBTAIN CALC RECORD(ELEMENT_OWNER);
        END; /* ELSE */
        I = I + 1;
        END; /* OF WHILE */
    END; /* OF IF-THEN */
ELSE DO;
        PUT STRING(OUTPUT_OWNER_NO) EDIT(J+1) (F(4));
        OBTAIN CALC RECORD(OUTPUT_OWNER);
    END; /* OF ELSE DO */
    J = J + 1;
END; /* OF WHILE */
GO TO ENTER_SETS;
DELETE_OU_CON_EL:

```

```

OUTPUT_OWNER_NO = ' 1';

I = 1;

OBTAIN CALC RECORD(OUTPUT_OWNER);

DO WHILE (I<=OUCNT);

    IF (OUTPUTOWNER_NAME = OWNER_NAME) THEN DO;

        OBTAIN FIRST RECORD(OUTPUT) SET(OUO_SUBPARTS_OU);

        DO WHILE (ERROR_STATUS = OK);

            IF NOT SET(OU_CONSISTS_EL) EMPTY THEN DO;

                OBTAIN FIRST RECORD(ELEMENT) SET(OU_CONSISTS_EL);

                DO WHILE (ERROR_STATUS = OK);

                    IF (ELEMENT_NAME = MEMBER_NAME) THEN DO;

                        DISCONNECT RECORD(ELEMENT) SET(OU_CONSISTS_EL);

                        CALL IDMS_STATUS;

                        ERASE RECORD(ELEMENT);

                        CALL IDMS_STATUS;

                        DISPLAY('RELATIONSHIP ' || OWNER_NAME || ' CONSISTS ' ||
MEMBER_NAME || ' HAS BEEN DELETED ');

                        I = OUCNT + 1;

                        ERROR_STATUS = END_OF_SET;

                        END; /* OF IF */

                    ELSE DO;

                        OBTAIN NEXT RECORD(ELEMENT) SET(OU_CONSISTS_EL);

                        END; /* OF ELSE */

                    END; /* OF WHILE */

                END; /* OF IF */

            ELSE DO;

                OBTAIN NEXT RECORD(OUTPUT) SET(OUO_SUBPARTS_OU);

```

```

        END; /* OF ELSE */

    END; /* OF WHILE */

END; /* OF IF *NAME = *NAME */

ELSE DO;

    PUT STRING(OUTPUT_OWNER_NO)  EDIT(I+1)  (F(4));

    OBTAIN CALC RECORD(OUTPUT_OWNER);

END; /* OF ELSE */

I = I + 1;

END; /* OF WHILE */

GO TO ENTER_SETS;

EN_CON_GR: DISPLAY('YOU HAVE CHOSEN THE RELATIONSHIP ');

DISPLAY('ENTITY_CONSISTS_OF_GROUP. ');

DISPLAY('ENTER THE ENTITY NAME ')REPLY(OWNER_NAME);

DISPLAY('ENTER THE GROUP NAME ')REPLY(MEMBER_NAME);

DISPLAY('DO YOU WISH TO (I) INSERT, ');

DISPLAY(' OR (D) DELETE THIS RELATIONSHIP?  ENTER YOUR CHOICE ');

DISPLAY(' OF I, D')REPLY(CHOICE);

IF CHOICE = 'I' THEN GO TO INSERT_EN_CON_GR;

IF CHOICE = 'D' THEN GO TO DELETE_EN_CON_GR;

DISPLAY('INVALID CHOICE, TRY AGAIN');

GO TO EN_CON_GR;

INSERT_EN_CON_GR:

ENTITY_OWNER_NO = '    1';

J = 1;

OBTAIN CALC RECORD(ENTITY_OWNER);

DO  WHILE (J <= ENCNT);

IF (ENTITY_OWNER_NAME = OWNER_NAME) THEN DO;

```

```

GROUP_OWNER_NO = ' 1';

I = 1;

OBTAIN CALC RECORD(GROUP_OWNER);

DO WHILE (I <= GRCNT);

    IF (GROUP_OWNER_NAME = MEMBER_NAME) THEN DO;

        GROUP_NO = GROUP_OWNER_NO;

        GROUP_NAME = GROUP_OWNER_NAME;

        GR_ATTRIBUTES = GRO_ATTRIBUTES;

        STORE RECORD (GROUP);

        CALL IDMS_STATUS;

        CONNECT RECORD(GROUP) SET(GRO_SUBPARTS_GR);

        CALL IDMS_STATUS;

        IF SET(ENO_SUBPARTS_EN) EMPTY THEN DO;

            ENTITY_NAME = ENTITY_OWNER_NAME;

            ENTITY_NO = ENTITY_OWNER_NO;

            EN_ATTRIBUTES = ENO_ATTRIBUTES;

            STORE RECORD(ENTITY);

            CALL IDMS_STATUS;

            CONNECT RECORD(ENTITY) SET(ENO_SUBPARTS_EN);

            CALL IDMS_STATUS;

        END; /* OF IF EMPTY */

    ELSE DO;

        FIND FIRST RECORD(ENTITY) SET(ENO_SUBPARTS_EN);

    END; /* OF ELSE DO */

    CONNECT RECORD(GROUP) SET(EN_CONSISTS_GR);

    CALL IDMS_STATUS;

    DISPLAY('RELATIONSHIP ' || OWNER_NAME || ' CONSISTS ' ||

```

```

MEMBER_NAME || ' HAS BEEN INSERTED ');

    I = GRCNT + 1;

    J = ENCNT + 1;

    END; /* OF IF */

ELSE DO;

    FIND CURRENT RECORD(ENTITY_OWNER);

    PUT STRING(GROUP_OWNER_NO) EDIT(I+1) (F(4));

    OBTAIN CALC RECORD(GROUP_OWNER);

    END; /* ELSE */

    I = I + 1;

    END; /* OF WHILE */

END; /* OF IF-THEN */

ELSE DO;

    PUT STRING(ENTITY_OWNER_NO) EDIT(J+1) (F(4));

    OBTAIN CALC RECORD(ENTITY_OWNER);

    END; /* OF ELSE DO */

    J = J + 1;

END; /* OF WHILE */

GO TO ENTER_SETS;

DELETE_EN_CON_GR:

ENTITY_OWNER_NO = '    1';

I = 1;

OBTAIN CALC RECORD(ENTITY_OWNER);

DO WHILE (I<=ENCNT);

    IF (ENTITY_OWNER_NAME = OWNER_NAME) THEN DO;

        OBTAIN FIRST RECORD(ENTITY) SET(ENO_SUBPARTS_EN);

        DO WHILE (ERROR_STATUS = OK);

```

```

        IF NOT SET(EN_CONSISTS_GR) EMPTY THEN DO;
        OBTAIN FIRST RECORD(GROUP) SET(EN_CONSISTS_GR);
        DO WHILE (ERROR_STATUS = OK);
            IF (GROUP_NAME = MEMBER_NAME) THEN DO;
                DISCONNECT RECORD(GROUP) SET(EN_CONSISTS_GR);
                CALL IDMS_STATUS;
                DISPLAY('RELATIONSHIP ' || OWNER_NAME || ' CONSISTS ' ||
MEMBER_NAME || ' HAS BEEN DELETED ');
                I = ENCNT + 1;
                ERROR_STATUS = END_OF_SET;
                END; /* OF IF */
            ELSE DO;
                OBTAIN NEXT RECORD(GROUP) SET(EN_CONSISTS_GR);
                END; /* OF ELSE */
            END; /* OF WHILE */
        END; /* OF IF */
    ELSE DO;
        OBTAIN NEXT RECORD(ENTITY) SET(ENO_SUBPARTS_EN);
        END; /* OF ELSE */
    END; /* OF WHILE */
END; /* OF IF *NAME = *NAME */
ELSE DO;
    PUT STRING(ENTITY_OWNER_NO) EDIT(I+1) (F(4));
    OBTAIN CALC RECORD(ENTITY_OWNER);
    END; /* OF ELSE */
    I = I + 1;
END; /* OF WHILE */

```

```

GO TO ENTER_SETS;

OU_CON_GR: DISPLAY('YOU HAVE CHOSEN THE RELATIONSHIP ');
DISPLAY('OUTPUT_CONSISTS_OF_GROUP. ');
DISPLAY('ENTER THE OUTPUT NAME ')REPLY(OWNER_NAME);
DISPLAY('ENTER THE GROUP NAME ')REPLY(MEMBER_NAME);
DISPLAY('DO YOU WISH TO (I) INSERT, ');
DISPLAY(' OR (D) DELETE THIS RELATIONSHIP? ENTER YOUR CHOICE ');
DISPLAY(' OF I, D')REPLY(CHOICE);
IF CHOICE = 'I' THEN GO TO INSERT_OU_CON_GR;
IF CHOICE = 'D' THEN GO TO DELETE_OU_CON_GR;
DISPLAY('INVALID CHOICE, TRY AGAIN');
GO TO OU_CON_GR;
INSERT_OU_CON_GR:
OUTPUT_OWNER_NO = ' 1';
J = 1;
OBTAIN CALC RECORD(OUTPUT_OWNER);
DO WHILE (J <= OUCNT);
IF (OUTPUTOWNER_NAME = OWNER_NAME) THEN DO;
    GROUP_OWNER_NO = ' 1';
    I = 1;
    OBTAIN CALC RECORD(GROUP_OWNER);
    DO WHILE (I <= GRCNT);
        IF (GROUP_OWNER_NAME = MEMBER_NAME) THEN DO;
            GROUP_NO = GROUP_OWNER_NO;
            GROUP_NAME = GROUP_OWNER_NAME;
            GR_ATTRIBUTES = GRO_ATTRIBUTES;
            STORE RECORD (GROUP);

```

```

CALL IDMS_STATUS;
CONNECT RECORD(GROUP) SET(GRO_SUBPARTS_GR);
CALL IDMS_STATUS;

IF SET(OUO_SUBPARTS_OU) EMPTY THEN DO;

    OUTPUT_NAME = OUTPUTOWNER_NAME;
    OUTPUT_NO = OUTPUT_OWNER_NO;
    OU_ATTRIBUTES = OUO_ATTRIBUTES;
    STORE RECORD(OUTPUT);
    CALL IDMS_STATUS;
    CONNECT RECORD(OUTPUT) SET(OUO_SUBPARTS_OU);
    CALL IDMS_STATUS;
END; /* OF IF EMPTY */

ELSE DO;

    FIND FIRST RECORD(OUTPUT) SET(OUO_SUBPARTS_OU);
END; /* OF ELSE DO */

    CONNECT RECORD(GROUP) SET(OU_CONSISTS_GR);
    CALL IDMS_STATUS;
    DISPLAY('RELATIONSHIP ' || OWNER_NAME || ' CONSISTS ' ||
MEMBER_NAME || ' HAS BEEN INSERTED ');
    I = GRCNT + 1;
    J = OUCNT + 1;
END; /* OF IF */

ELSE DO;

    FIND CURRENT RECORD(OUTPUT_OWNER);
    PUT STRING(GROUP_OWNER_NO) EDIT(I+1) (F(4));
    OBTAIN CALC RECORD(GROUP_OWNER);
END; /* ELSE */

```



```

        I = I + 1;

    END; /* OF WHILE */

END; /* OF IF-THEN */

ELSE DO;

    PUT STRING(OUTPUT_OWNER_NO) EDIT(J+1) (F(4));

    OBTAIN CALC RECORD(OUTPUT_OWNER);

END; /* OF ELSE DO */

J = J + 1;

END; /* OF WHILE */

GO TO ENTER_SETS;

DELETE_OU_CON_GR:

OUTPUT_OWNER_NO = '    1';

I = 1;

OBTAIN CALC RECORD(OUTPUT_OWNER);

DO WHILE (I<=OUCNT);

    IF (OUTPUTOWNER_NAME = OWNER_NAME) THEN DO;

        OBTAIN FIRST RECORD(OUTPUT) SET(OUO_SUBPARTS_OU);

        DO WHILE (ERROR_STATUS = OK);

            IF NOT SET(OU_CONSISTS_GR) EMPTY THEN DO;

                OBTAIN FIRST RECORD(GROUP) SET(OU_CONSISTS_GR);

                DO WHILE (ERROR_STATUS = OK);

                    IF (GROUP_NAME = MEMBER_NAME) THEN DO;

                        DISCONNECT RECORD(GROUP) SET(OU_CONSISTS_GR);

                        CALL IDMS_STATUS;

                        DISPLAY('RELATIONSHIP ' || OWNER_NAME || ' CONSISTS ' ||
MEMBER_NAME || ' HAS BEEN DELETED ');

                        I = OUCNT + 1;

```

```

        ERROR_STATUS = END_OF_SET;

        END; /* OF IF */

    ELSE DO;

        OBTAIN NEXT RECORD(GROUP) SET(OU_CONSISTS_GR);

        END; /* OF ELSE */

    END; /* OF WHILE */

END; /* OF IF */

ELSE DO;

    OBTAIN NEXT RECORD(OUTPUT) SET(OUO_SUBPARTS_OU);

    END; /* OF ELSE */

END; /* OF WHILE */

END; /* OF IF *NAME = *NAME */

ELSE DO;

    PUT STRING(OUTPUT_OWNER_NO) EDIT(I+1) (F(4));

    OBTAIN CALC RECORD(OUTPUT_OWNER);

    END; /* OF ELSE */

    I = I + 1;

END; /* OF WHILE */

GO TO ENTER_SETS;

PR_INC_EV:  DISPLAY('YOU HAVE CHOSEN THE RELATIONSHIP ');

DISPLAY('PROCESS_INCEPTION_CAUSES_EVENT. ');

DISPLAY('ENTER THE PROCESS NAME ')REPLY(OWNER_NAME);

DISPLAY('ENTER THE EVENT NAME ')REPLY(MEMBER_NAME);

DISPLAY('DO YOU WISH TO (I) INSERT, ');

DISPLAY(' OR (D) DELETE THIS RELATIONSHIP? ENTER YOUR CHOICE ');

DISPLAY(' OF I, D')REPLY(CHOICE);

IF CHOICE = 'I' THEN GO TO INSERT_PR_INC_EV;

```

```

IF CHOICE = 'D' THEN GO TO DELETE_PR_INC_EV;
DISPLAY('INVALID CHOICE, TRY AGAIN');
GO TO PR_INC_EV;
INSERT_PR_INC_EV :
PROCESS_OWNER_NO = '    1';
J = 1;
OBTAIN CALC RECORD(PROCESS_OWNER);
OBTAIN FIRST RECORD(PROCESS) SET(PRO_SUBPARTS_PR);
DO WHILE (J <= PRCNT);
    IF (PROCESS_NAME = OWNER_NAME) THEN DO;
        EVENT_NO = '    1';
        I = 1;
        OBTAIN CALC RECORD(EVENT);
        DO WHILE (I <= EVCNT);
            IF EVENT_NAME = MEMBER_NAME THEN DO;
                CONNECT RECORD(EVENT) SET(PR_INCEPTION_EV);
                CALL IDMS_STATUS;
                DISPLAY('RELATIONSHIP ' || OWNER_NAME || ' INCEPTION '
|| MEMBER_NAME || ' HAS BEEN INSERTED ');
                I = EVCNT + 1;
                J = PRCNT + 1;
                END; /* OF IF */
            ELSE DO;
                FIND CURRENT RECORD(EVENT);
                PUT STRING(EVENT_NO) EDIT(I+1) (F(4));
                OBTAIN CALC RECORD(EVENT);
                END; /* ELSE */
            END;
        END;
    END;

```

```

        I = I + 1;
    END; /* OF WHILE */
END; /* OF THEN */
ELSE DO;
    OBTAIN NEXT RECORD(PROCESS) SET (PRO_SUBPARTS_PR);
END; /* OF ELSE */
J = J + 1;
END; /* OF WHILE */

GO TO ENTER_SETS;
DELETE_PR_INC_EV: PROCESS_OWNER_NO = '    1';
J = 1;
OBTAIN CALC RECORD (PROCESS_OWNER);
OBTAIN FIRST RECORD (PROCESS) SET (PRO_SUBPARTS_PR);
DO WHILE (J<=PRCNT);
    IF (OWNER_NAME = PROCESS_NAME) THEN DO;
        OBTAIN FIRST RECORD (EVENT) SET (PR_INCEPTION_EV);
        DO WHILE (ERROR_STATUS = OK);
            IF (EVENT_NAME = MEMBER_NAME) THEN DO;
                DISCONNECT RECORD (EVENT) SET (PR_INCEPTION_EV);
                CALL IDMS_STATUS;
                DISPLAY(' RELATIONSHIP ' || OWNER_NAME || ' INCEPTION '
|| MEMBER_NAME || ' HAS BEEN DELETED ');
                J = PRCNT + 1;
                ERROR_STATUS = END_OF_SET;
            END; /* OF THEN */
        ELSE DO;

```

```

        OBTAIN NEXT RECORD(EVENT) SET (PR_INCEPTION_EV);

    END; /* OF ELSE */

    END; /* OF WHILE ERROR_STATUS */

    END; /* OF THEN */

    ELSE DO;

        OBTAIN NEXT RECORD(PROCESS) SET (PRO_SUBPARTS_PR);

    END; /* OF ELSE */

    J = J + 1;

    END; /* OF WHILE */

GO TO ENTER_SETS;

PR_TER_EV:   DISPLAY('YOU HAVE CHOSEN THE RELATIONSHIP ');

DISPLAY('PROCESS_TERMINATION_CAUSES_EVENT. ');

DISPLAY('ENTER THE PROCESS NAME ') REPLY(OWNER_NAME);

DISPLAY('ENTER THE EVENT NAME ') REPLY(MEMBER_NAME);

DISPLAY('DO YOU WISH TO (I) INSERT, ');

DISPLAY(' OR (D) DELETE THIS RELATIONSHIP? ENTER YOUR CHOICE ');

DISPLAY(' OF I, D') REPLY(CHOICE);

IF CHOICE = 'I' THEN GO TO INSERT_PR_TER_EV;

IF CHOICE = 'D' THEN GO TO DELETE_PR_TER_EV;

DISPLAY('INVALID CHOICE, TRY AGAIN');

GO TO PR_TER_EV;

INSERT_PR_TER_EV:

PROCESS_OWNER_NO = ' 1';

J = 1;

OBTAIN CALC RECORD(PROCESS_OWNER);

OBTAIN FIRST RECORD(PROCESS) SET (PRO_SUBPARTS_PR);

```

```

DO WHILE (J <= PRCNT) ;
  IF (PROCESS_NAME = OWNER_NAME) THEN DO;
    EVENT_NO = '  1';
    I = 1;
    OBTAIN CALC RECORD(EVENT);
    DO WHILE (I <= EVCNT);
      IF EVENT_NAME = MEMBER_NAME THEN DO;
        CONNECT RECORD(EVENT) SET(PR_TERMINATE_EV);
        DISPLAY(' RELATIONSHIP ' || OWNER_NAME || ' TERMINATE ' ||
MEMBER_NAME || ' HAS BEEN INSERTED ');
        CALL IDMS_STATUS;
        I = EVCNT + 1;
        J = PRCNT + 1;
        END; /* OF IF */
      ELSE DO;
        FIND CURRENT RECORD(EVENT);
        PUT STRING(EVENT_NO) EDIT(I+1) (F(4));
        OBTAIN CALC RECORD(EVENT);
        END; /* ELSE */
        I = I + 1;
      END; /* OF WHILE */
    END; /* OF THEN */
  ELSE DO;
    OBTAIN NEXT RECORD(PROCESS) SET(PRO_SUBPARTS_PR);
    END; /* OF ELSE */
    J = J + 1;
  END; /* OF WHILE */

```

```

GO TO ENTER_SETS;

DELETE_PR_TER_EV: PROCESS_OWNER_NO =' 1';

J = 1;

OBTAIN CALC RECORD (PROCESS_OWNER);

OBTAIN FIRST RECORD (PROCESS) SET (PRO_SUBPARTS_PR);

DO WHILE (J<=PRCNT);

    IF (OWNER_NAME = PROCESS_NAME) THEN DO;

        OBTAIN FIRST RECORD (EVENT) SET (PR_TERMINATE_EV);

        DO WHILE (ERROR_STATUS = OK);

            IF (EVENT_NAME = MEMBER_NAME) THEN DO;

                DISCONNECT RECORD (EVENT) SET (PR_TERMINATE_EV);

                CALL IDMS_STATUS;

                DISPLAY(' RELATIONSHIP ' || OWNER_NAME || ' TERMINATE ' ||
MEMBER_NAME || ' HAS BEEN DELETED ');

                J = PRCNT + 1;

                ERROR_STATUS = END_OF_SET;

            END; /* OF THEN */

            ELSE DO;

                OBTAIN NEXT RECORD (EVENT) SET (PR_TERMINATE_EV);

            END; /* OF ELSE */

        END; /* OF WHILE ERROR_STATUS */

    END; /* OF THEN */

    ELSE DO;

        OBTAIN NEXT RECORD (PROCESS) SET (PRO_SUBPARTS_PR);

    END; /* OF ELSE */

    J = J + 1;

```

```

END; /* OF WHILE */

GO TO ENTER_SETS;

PR_TRI_EV:  DISPLAY('YOU HAVE CHOSEN THE RELATIONSHIP ');
DISPLAY('PROCESS_TRIGGERED_BY_EVENT. ');
DISPLAY('ENTER THE PROCESS NAME ') REPLY(OWNER_NAME);
DISPLAY('ENTER THE EVENT NAME ') REPLY(MEMBER_NAME);
DISPLAY('DO YOU WISH TO (I) INSERT, ');
DISPLAY(' OR (D) DELETE THIS RELATIONSHIP? ENTER YOUR CHOICE ');
DISPLAY(' OF I, D') REPLY(CHOICE);
IF CHOICE = 'I' THEN GO TO INSERT_PR_TRI_EV;
IF CHOICE = 'D' THEN GO TO DELETE_PR_TRI_EV;
DISPLAY('INVALID CHOICE, TRY AGAIN');
GO TO PR_TRI_EV;
INSERT_PR_TRI_EV:
PROCESS_OWNER_NO = ' 1';
J = 1;
OBTAIN CALC RECORD(PROCESS_OWNER);
OBTAIN FIRST RECORD(PROCESS) SET(PRO_SUBPARTS_PR);
DO WHILE (J <= PRCNT);
  IF (PROCESS_NAME = OWNER_NAME) THEN DO;
    EVENT_NO = ' 1';
    I = 1;
    OBTAIN CALC RECORD(EVENT);
    DO WHILE (I <= EVCNT);
      IF EVENT_NAME = MEMBER_NAME THEN DO;
        CONNECT RECORD(EVENT) SET(PR_TRIGGERED_EV);

```



```

        CALL IDMS_STATUS;

        DISPLAY(' RELATIONSHIP ' || OWNER_NAME || ' TRIGGERED ' ||
MEMBER_NAME || ' HAS BEEN INSERTED ');

        I = EVCNT +1;
        J = PRCNT + 1;

        END; /* OF IF */

ELSE DO;

        FIND CURRENT RECORD(EVENT);

        PUT STRING(EVENT_NO) EDIT(I+1) (F(4));

        OBTAIN CALC RECORD(EVENT);

        END; /* ELSE */

        I = I + 1;

        END; /* OF WHILE */

END; /* OF THEN */

ELSE DO;

        OBTAIN NEXT RECORD(PROCESS) SET (PRO_SUBPARTS_PR);

        END; /* OF ELSE */

        J = J + 1;

        END; /* OF WHILE */

GO TO ENTER_SETS;

DELETE_PR_TRI_EV: PROCESS_OWNER_NO = ' 1';

J = 1;

OBTAIN CALC RECORD (PROCESS_OWNER);

OBTAIN FIRST RECORD (PROCESS) SET (PRO_SUBPARTS_PR);

DO WHILE (J<=PRCNT);

        IF (OWNER_NAME = PROCESS_NAME) THEN DO;

```

```

OBTAIN FIRST RECORD(EVENT) SET(PR_TRIGGERED_EV);

DO WHILE (ERROR_STATUS = OK);

IF (EVENT_NAME = MEMBER_NAME) THEN DO;

DISCONNECT RECORD(EVENT) SET(PR_TRIGGERED_EV);

CALL IDMS_STATUS;

DISPLAY(' RELATIONSHIP ' || OWNER_NAME || ' TRIGGERED ' ||
MEMBER_NAME || ' HAS BEEN DELETED ');

J = PRCNT + 1;

ERROR_STATUS = END_OF_SET;

END; /* OF THEN */

ELSE DO;

OBTAIN NEXT RECORD(EVENT) SET(PR_TRIGGERED_EV);

END; /* OF ELSE */

END; /* OF WHILE ERROR_STATUS */

END; /* OF THEN */

ELSE DO;

OBTAIN NEXT RECORD(PROCESS) SET(PRO_SUBPARTS_PR);

END; /* OF ELSE */

J = J + 1;

END; /* OF WHILE */

GO TO ENTER_SETS;

EV_WHE_CO: DISPLAY('YOU HAVE CHOSEN THE RELATIONSHIP ');

DISPLAY('EVENT_WHEN_CONDITION. ');

DISPLAY('ENTER THE EVENT NAME ') REPLY(OWNER_NAME);

DISPLAY('ENTER THE CONDITION NAME ') REPLY(MEMBER_NAME);

DISPLAY('DO YOU WISH TO (I) INSERT, ');

```

```

DISPLAY(' OR (D) DELETE THIS RELATIONSHIP? ENTER YOUR CHOICE ');
DISPLAY(' OF I, D')REPLY(CHOICE);
IF CHOICE = 'I' THEN GO TO INSERT_EV_WHE_CO;
IF CHOICE = 'D' THEN GO TO DELETE_EV_WHE_CO;
DISPLAY('INVALID CHOICE, TRY AGAIN');
GO TO EV_WHE_CO;
INSERT_EV_WHE_CO:
EVENT_NO = ' 1';
J = 1;
OBTAIN CALC RECORD(EVENT);
DO WHILE (J <= EVCNT);
IF (EVENT_NAME = OWNER_NAME) THEN DO;
    CONDITION_NO = ' 1';
    I = 1;
    OBTAIN CALC RECORD(CONDITION);
    DO WHILE (I <= COCNT);
        IF (CONDITION_NAME = MEMBER_NAME) THEN DO;
            CONNECT RECORD(CONDITION) SET(EV_WHEN_CO);
            CALL IDMS_STATUS;
            DISPLAY(' RELATIONSHIP ' || OWNER_NAME || ' WHEN ' ||
MEMBER_NAME || ' HAS BEEN INSERTED ');
            I = COCNT + 1;
            J = EVCNT + 1;
            END; /* OF IF */
        ELSE DO;
            FIND CURRENT RECORD(CONDITION);
            PUT STRING(CONDITION_NO) EDIT(I+1) (F(4));

```

```

        OBTAIN CALC RECORD(CONDITION) ;

        END; /* ELSE */

        I = I + 1;

    END; /* OF WHILE */

END; /* OF IF-THEN */

ELSE DO;

    PUT STRING(EVENT_NO) EDIT(J+1) (F(4));

    OBTAIN CALC RECORD(EVENT);

END; /* OF ELSE DO */

J = J + 1;

END; /* OF WHILE */

GO TO ENTER_SETS;

DELETE_EV_WHE_CO: CONDITION_NO = '    1';

I = 1;

OBTAIN CALC RECORD(CONDITION) ;

DO WHILE (I<=COCNT) ;

    IF (CONDITION_NAME = MEMBER_NAME) THEN DO;

        DO WHILE (ERROR_STATUS = OK) ;

            IF NOT SET(EV_WHEN_CO) MEMBER THEN DO;

                IF (ERROR_STATUS ^= '1601') THEN CALL IDMS_STATUS;

                OBTAIN NEXT RECORD(CONDITION) SET(EV_WHEN_CO);

            END; /* OF THEN */

            ELSE DO; /* FOUND MEMBER RECORD */

                DISCONNECT RECORD(CONDITION) SET(EV_WHEN_CO) ;

                CALL IDMS_STATUS;

                ERASE RECORD(CONDITION) ;

                CALL IDMS_STATUS;

```

```

        DISPLAY(' RELATIONSHIP ' || OWNER_NAME || ' WHEN ' ||
MEMBER_NAME || ' HAS BEEN DELETED ');

        I = COCNT + 1;

        ERROR_STATUS = END_OF_SET;

        END; /* OF ELSE */

    END; /* OF WHILE */

END; /* OF IF *NAME = *NAME */

ELSE DO;

    PUT STRING(CONDITION_NO) EDIT(I+1) (F(4));

    OBTAIN CALC RECORD(CONDITION);

    END; /* OF ELSE */

    I = I + 1;

END; /* OF WHILE */

GO TO ENTER_SETS;

FIND_RECORDS:

DISPLAY(' PSL/PSA OBJECTS AND RELATIONSHIPS REPORT');

DISPLAY(' ');

PROCESS_OWNER_NO = ' 1';

I = 0;

OBTAIN CALC RECORD(PROCESS_OWNER);

DISPLAY(' ');

DISPLAY(' OBJECTS OF TYPE PROCESS ');

CALL IDMS_STATUS;

OBTAIN FIRST RECORD(PROCESS) SET(PRO_SUBPARTS_PR);

DO WHILE (ERROR_STATUS = OK);

    I = I + 1;

```

```

PUT STRING (REC_NUM) EDIT (I) (F(4));
DISPLAY (REC_NUM || ' ' || PROCESS_NAME);
DISPLAY (PR_ATTRIBUTES);
OBTAIN NEXT RECORD (PROCESS) SET (PRO_SUBPARTS_PR);
END; /* OF WHILE */
IF (ERROR_STATUS ^= END_OF_SET) THEN CALL IDMS_STATUS;
PRCNT = I;
INPUT_OWNER_NO = ' 1';
I = 1;

PUT STRING (INPUT_OWNER_NO) EDIT (I) (F(4));
OBTAIN CALC RECORD (INPUT_OWNER);
DISPLAY (' ');
DISPLAY (' OBJECTS OF TYPE INPUT ');
DO WHILE (ERROR_STATUS = OK);

    DISPLAY (INPUT_OWNER_NO || ' ' || INPUT_OWNER_NAME);
    DISPLAY (INO_ATTRIBUTES);
    I = I + 1;
    PUT STRING (INPUT_OWNER_NO) EDIT (I) (F(4));
    OBTAIN CALC RECORD (INPUT_OWNER);
END; /* OF WHILE */
IF (ERROR_STATUS ^= '0326') THEN CALL IDMS_STATUS;
INCNT = I-1;
OUTPUT_OWNER_NO = ' 1';
I = 1;

PUT STRING (OUTPUT_OWNER_NO) EDIT (I) (F(4));
OBTAIN CALC RECORD (OUTPUT_OWNER);
DISPLAY (' ');

```

```

DISPLAY('  OBJECTS OF TYPE OUTPUT  ');
DO WHILE (ERROR_STATUS = OK);
    DISPLAY(OUTPUT_OWNER_NO || '      ' || OUTPUTOWNER_NAME);
    DISPLAY(OUO_ATTRIBUTES);
    I = I + 1;
    PUT STRING(OUTPUT_OWNER_NO) EDIT(I) (F(4));
    OBTAIN CALC RECORD(OUTPUT_OWNER);
END; /* OF WHILE */
IF(ERROR_STATUS = '0326') THEN CALL IDMS_STATUS;
OUCNT = I-1;
ENTITY_OWNER_NO = '    1';
I = 1;
    PUT STRING(ENTITY_OWNER_NO) EDIT(I) (F(4));
OBTAIN CALC RECORD(ENTITY_OWNER);
DISPLAY('  ');
DISPLAY('  OBJECTS OF TYPE ENTITY  ');
DO WHILE (ERROR_STATUS = OK);
    DISPLAY(ENTITY_OWNER_NO || '      ' || ENTITY_OWNER_NAME);
    DISPLAY(ENO_ATTRIBUTES);
    I = I + 1;
    PUT STRING(ENTITY_OWNER_NO) EDIT(I) (F(4));
    OBTAIN CALC RECORD(ENTITY_OWNER);
END; /* OF WHILE */
IF(ERROR_STATUS = '0326') THEN CALL IDMS_STATUS;
ENCNT = I-1;
GROUP_OWNER_NO = '    1';
I = 1;

```

```

    PUT STRING(GROUP_OWNER_NO) EDIT(I) (F(4));
OBTAIN CALC RECORD(GROUP_OWNER);
DISPLAY(' ');
DISPLAY(' OBJECTS OF TYPE GROUP ');
DO WHILE (ERROR_STATUS = OK);
    DISPLAY(GROUP_OWNER_NO || ' ' || GROUP_OWNER_NAME);
    DISPLAY(GRO_ATTRIBUTES);
    I = I + 1;
    PUT STRING(GROUP_OWNER_NO) EDIT(I) (F(4));
    OBTAIN CALC RECORD(GROUP_OWNER);
END; /* OF WHILE */
IF(ERROR_STATUS ^= '0326') THEN CALL IDMS_STATUS;
GRCNT = I-1;
ELEMENT_OWNER_NO = ' 1';
I = 1;
    PUT STRING(ELEMENT_OWNER_NO) EDIT(I) (F(4));
OBTAIN CALC RECORD(ELEMENT_OWNER);
DISPLAY(' ');
DISPLAY(' OBJECTS OF TYPE ELEMENT ');
DO WHILE (ERROR_STATUS = OK);
    DISPLAY(ELEMENT_OWNER_NO || ' ' || ELEMENT_OWNER_NAME);
    DISPLAY(ELO_ATTRIBUTES);
    I = I + 1;
    PUT STRING(ELEMENT_OWNER_NO) EDIT(I) (F(4));
    OBTAIN CALC RECORD(ELEMENT_OWNER);
END; /* OF WHILE */
IF(ERROR_STATUS ^= '0326') THEN CALL IDMS_STATUS;

```



```

ELCNT = I-1;
EVENT_NO = ' 1';
I = 1;

  PUT STRING(EVENT_NO) EDIT(I) (F(4));
OBTAIN CALC RECORD(EVENT);
DISPLAY(' ');
DISPLAY(' OBJECTS OF TYPE EVENT ');
DO WHILE (ERROR_STATUS = OK);

  DISPLAY(EVENT_NO || ' ' || EVENT_NAME);
  DISPLAY(EV_ATTRIBUTES);
  I = I + 1;
  PUT STRING(EVENT_NO) EDIT(I) (F(4));
  OBTAIN CALC RECORD(EVENT);

END; /* OF WHILE */
IF(ERROR_STATUS ^= '0326') THEN CALL IDMS_STATUS;
EVCNT = I-1;
CONDITION_NO = ' 1';
I = 1;

  PUT STRING(CONDITION_NO) EDIT(I) (F(4));
OBTAIN CALC RECORD(CONDITION);
DISPLAY(' ');
DISPLAY(' OBJECTS OF TYPE CONDITION ');
DO WHILE (ERROR_STATUS = OK);

  DISPLAY(CONDITION_NO || ' ' || CONDITION_NAME);
  DISPLAY(CO_ATTRIBUTES);
  I = I + 1;
  PUT STRING(CONDITION_NO) EDIT(I) (F(4));

```

```

    OBTAIN CALC RECORD(CONDITION);
END; /* OF WHILE */
IF(ERROR_STATUS = '0326') THEN CALL IDMS_STATUS;
COCNT = I-1;
FIND_SETS: /* FIRST FIND SETS WHOSE OWNERS ARE PROCESS */
PROCESS_OWNER_NO = ' 1';
OBTAIN CALC RECORD(PROCESS_OWNER);
OBTAIN FIRST RECORD(PROCESS) SET(PRO_SUBPARTS_PR);
DO WHILE (ERROR_STATUS = OK);
    IF NOT SET(PR_RECEIVES_IN) EMPTY THEN DO;
        DISPLAY('RELATIONSHIP(S) PROCESS RECEIVES INPUT');
        OBTAIN FIRST RECORD(INPUT) SET(PR_RECEIVES_IN);
        DO WHILE (ERROR_STATUS = OK);
            DISPLAY(PROCESS_NAME || ' RECEIVES ' || INPUT_NAME);
            OBTAIN NEXT RECORD(INPUT) SET(PR_RECEIVES_IN);
        END; /* OF WHILE */
    END; /* OF IF THEN */
    OBTAIN NEXT RECORD(PROCESS) SET(PRO_SUBPARTS_PR);
END; /* OF WHILE */

PROCESS_OWNER_NO = ' 1';
OBTAIN CALC RECORD(PROCESS_OWNER);
OBTAIN FIRST RECORD(PROCESS) SET(PRO_SUBPARTS_PR);
DO WHILE (ERROR_STATUS = OK);
    IF NOT SET(PR_USES_IN) EMPTY THEN DO;
        DISPLAY('RELATIONSHIP(S) PROCESS USES INPUT');
        OBTAIN FIRST RECORD(INPUT) SET(PR_USES_IN);

```

```

DO WHILE (ERROR_STATUS = OK) ;
    DISPLAY (PROCESS_NAME || ' USES ' || INPUT_NAME) ;
    OBTAIN NEXT RECORD (INPUT) SET (PR_USES_IN) ;
END; /* OF WHILE */

END; /* OF IF THEN */

OBTAIN NEXT RECORD (PROCESS) SET (PRO_SUBPARTS_PR) ;
END; /* OF WHILE */

PROCESS_OWNER_NO = ' 1';
OBTAIN CALC RECORD (PROCESS_OWNER) ;
OBTAIN FIRST RECORD (PROCESS) SET (PRO_SUBPARTS_PR) ;
DO WHILE (ERROR_STATUS = OK) ;
    IF NOT SET (PR_GENERATES_OU) EMPTY THEN DO ;
        DISPLAY ('RELATIONSHIP(S) PROCESS GENERATES OUTPUT') ;
        OBTAIN FIRST RECORD (OUTPUT) SET (PR_GENERATES_OU) ;
        DO WHILE (ERROR_STATUS = OK) ;
            DISPLAY (PROCESS_NAME || ' GENERATES ' || OUTPUT_NAME) ;
            OBTAIN NEXT RECORD (OUTPUT) SET (PR_GENERATES_OU) ;
        END; /* OF WHILE */
    END; /* OF IF THEN */

    OBTAIN NEXT RECORD (PROCESS) SET (PRO_SUBPARTS_PR) ;
END; /* OF WHILE */

PROCESS_OWNER_NO = ' 1';
OBTAIN CALC RECORD (PROCESS_OWNER) ;
OBTAIN FIRST RECORD (PROCESS) SET (PRO_SUBPARTS_PR) ;
DO WHILE (ERROR_STATUS = OK) ;

```

```

IF NOT SET (PR_DERIVES_OU) EMPTY THEN DO;
    DISPLAY ('RELATIONSHIP(S) PROCESS DERIVES OUTPUT');
    OBTAIN FIRST RECORD (OUTPUT) SET (PR_DERIVES_OU);
    DO WHILE (ERROR_STATUS = OK);
        DISPLAY (PROCESS_NAME || ' DERIVES ' || OUTPUT_NAME);
        OBTAIN NEXT RECORD (OUTPUT) SET (PR_DERIVES_OU);
    END; /* OF WHILE */

END; /* OF IF THEN */

    OBTAIN NEXT RECORD (PROCESS) SET (PRO_SUBPARTS_PR);
END; /* OF WHILE */

PROCESS_OWNER_NO = ' 1';

OBTAIN CALC RECORD (PROCESS_OWNER);

OBTAIN FIRST RECORD (PROCESS) SET (PRO_SUBPARTS_PR);

DO WHILE (ERROR_STATUS = OK);

    IF NOT SET (PR_USES_EL) EMPTY THEN DO;

        DISPLAY ('RELATIONSHIP(S) PROCESS USES ELEMENT');

        OBTAIN FIRST RECORD (ELEMENT) SET (PR_USES_EL);

        DO WHILE (ERROR_STATUS = OK);

            DISPLAY (PROCESS_NAME || ' USES ' || ELEMENT_NAME);

            OBTAIN NEXT RECORD (ELEMENT) SET (PR_USES_EL);

        END; /* OF WHILE */

    END; /* OF IF THEN */

    OBTAIN NEXT RECORD (PROCESS) SET (PRO_SUBPARTS_PR);
END; /* OF WHILE */

PROCESS_OWNER_NO = ' 1';

```

```

OBTAIN CALC RECORD(PROCESS_OWNER);
OBTAIN FIRST RECORD(PROCESS) SET(PRO_SUBPARTS_PR);
DO WHILE (ERROR_STATUS = OK);
    IF NOT SET(PR_UPDATES_EL) EMPTY THEN DO;
        DISPLAY('RELATIONSHIP(S) PROCESS UPDATES ELEMENT');
        OBTAIN FIRST RECORD(ELEMENT) SET(PR_UPDATES_EL);
        DO WHILE (ERROR_STATUS = OK);
            DISPLAY{PROCESS_NAME || ' UPDATES ' || ELEMENT_NAME};
            OBTAIN NEXT RECORD(ELEMENT) SET(PR_UPDATES_EL);
        END; /* OF WHILE */
    END; /* OF IF THEN */
    OBTAIN NEXT RECORD(PROCESS) SET(PRO_SUBPARTS_PR);
END; /* OF WHILE */

PROCESS_OWNER_NO = ' 1';
OBTAIN CALC RECORD(PROCESS_OWNER);
OBTAIN FIRST RECORD(PROCESS) SET(PRO_SUBPARTS_PR);
DO WHILE (ERROR_STATUS = OK);
    IF NOT SET(PR_DERIVES_EL) EMPTY THEN DO;
        DISPLAY('RELATIONSHIP(S) PROCESS DERIVES ELEMENT');
        OBTAIN FIRST RECORD(ELEMENT) SET(PR_DERIVES_EL);
        DO WHILE (ERROR_STATUS = OK);
            DISPLAY{PROCESS_NAME || ' DERIVES ' || ELEMENT_NAME};
            OBTAIN NEXT RECORD(ELEMENT) SET(PR_DERIVES_EL);
        END; /* OF WHILE */
    END; /* OF IF THEN */

```

```

        OBTAIN NEXT RECORD(PROCESS) SET(PRO_SUBPARTS_PR);
END;  /* OF WHILE */

PROCESS_OWNER_NO = '  1';
OBTAIN CALC RECORD(PROCESS_OWNER);
OBTAIN FIRST RECORD(PROCESS) SET(PRO_SUBPARTS_PR);
DO WHILE (ERROR_STATUS = OK);
    IF NOT SET(PR_DERIVES_GR) EMPTY THEN DO;
        DISPLAY('RELATIONSHIP(S) PROCESS DERIVES GROUP');
        OBTAIN FIRST RECORD(GROUP) SET(PR_DERIVES_GR);
        DO WHILE (ERROR_STATUS = OK);
            DISPLAY(PROCESS_NAME || ' DERIVES ' || GROUP_NAME);
            OBTAIN NEXT RECORD(GROUP) SET(PR_DERIVES_GR);
        END; /* OF WHILE */
    END; /* OF IF THEN */
    OBTAIN NEXT RECORD(PROCESS) SET(PRO_SUBPARTS_PR);
END;  /* OF WHILE */

PROCESS_OWNER_NO = '  1';
OBTAIN CALC RECORD(PROCESS_OWNER);
OBTAIN FIRST RECORD(PROCESS) SET(PRO_SUBPARTS_PR);
DO WHILE (ERROR_STATUS = OK);
    IF NOT SET(PR_USES_GR) EMPTY THEN DO;
        DISPLAY('RELATIONSHIP(S) PROCESS USES GROUP');
        OBTAIN FIRST RECORD(GROUP) SET(PR_USES_GR);
        DO WHILE (ERROR_STATUS = OK);
            DISPLAY(PROCESS_NAME || ' USES ' || GROUP_NAME);

```

```

        OBTAIN NEXT RECORD(GROUP) SET(PR_USES_GR);

    END; /* OF WHILE */

END; /* OF IF THEN */

    OBTAIN NEXT RECORD(PROCESS) SET(PRO_SUBPARTS_PR);
END; /* OF WHILE */

PROCESS_OWNER_NO = ' 1';

OBTAIN CALC RECORD(PROCESS_OWNER);

OBTAIN FIRST RECORD(PROCESS) SET(PRO_SUBPARTS_PR);

DO WHILE (ERROR_STATUS = OK);

    IF NOT SET(PR_UPDATES_GR) EMPTY THEN DO;

        DISPLAY('RELATIONSHIP(S) PROCESS UPDATES GROUP');

        OBTAIN FIRST RECORD(GROUP) SET(PR_UPDATES_GR);

        DO WHILE (ERROR_STATUS = OK);

            DISPLAY(PROCESS_NAME || ' UPDATES ' || GROUP_NAME);

            OBTAIN NEXT RECORD(GROUP) SET(PR_UPDATES_GR);

        END; /* OF WHILE */

    END; /* OF IF THEN */

    OBTAIN NEXT RECORD(PROCESS) SET(PRO_SUBPARTS_PR);
END; /* OF WHILE */

PROCESS_OWNER_NO = ' 1';

OBTAIN CALC RECORD(PROCESS_OWNER);

OBTAIN FIRST RECORD(PROCESS) SET(PRO_SUBPARTS_PR);

DO WHILE (ERROR_STATUS = OK);

    IF NOT SET(PR_DERIVES_EN) EMPTY THEN DO;

        DISPLAY('RELATIONSHIP(S) PROCESS DERIVES ENTITY');

```

```

OBTAIN FIRST RECORD(ENTITY) SET(PR_DERIVES_EN);
DO WHILE (ERROR_STATUS = OK);
    DISPLAY(PROCESS_NAME || ' DERIVES ' || ENTITY_NAME);
    OBTAIN NEXT RECORD(ENTITY) SET(PR_DERIVES_EN);
END; /* OF WHILE */

END; /* OF IF THEN */

OBTAIN NEXT RECORD(PROCESS) SET(PRO_SUBPARTS_PR);
END; /* OF WHILE */

PROCESS_OWNER_NO = ' 1';
OBTAIN CALC RECORD(PROCESS_OWNER);
OBTAIN FIRST RECORD(PROCESS) SET(PRO_SUBPARTS_PR);
DO WHILE (ERROR_STATUS = OK);
    IF NOT SET(PR_UPDATES_EN) EMPTY THEN DO;
        DISPLAY('RELATIONSHIP(S) PROCESS UPDATES ENTITY');
        OBTAIN FIRST RECORD(ENTITY) SET(PR_UPDATES_EN);
        DO WHILE (ERROR_STATUS = OK);
            DISPLAY(PROCESS_NAME || ' UPDATES ' || ENTITY_NAME);
            OBTAIN NEXT RECORD(ENTITY) SET(PR_UPDATES_EN);
        END; /* OF WHILE */
    END; /* OF IF THEN */

    OBTAIN NEXT RECORD(PROCESS) SET(PRO_SUBPARTS_PR);
END; /* OF WHILE */

PROCESS_OWNER_NO = ' 1';
OBTAIN CALC RECORD(PROCESS_OWNER);
OBTAIN FIRST RECORD(PROCESS) SET(PRO_SUBPARTS_PR);

```



```

DO WHILE (ERROR_STATUS = OK);
  IF NOT SET (PR_USES_EN) EMPTY THEN DO;
    DISPLAY ('RELATIONSHIP(S) PROCESS USES ENTITY');
    OBTAIN FIRST RECORD (ENTITY) SET (PR_USES_EN);
    DO WHILE (ERROR_STATUS = OK);
      DISPLAY (PROCESS_NAME || ' USES ' || ENTITY_NAME);
      OBTAIN NEXT RECORD (ENTITY) SET (PR_USES_EN);
    END; /* OF WHILE */
  END; /* OF IF THEN */
  OBTAIN NEXT RECORD (PROCESS) SET (PRO_SUBPARTS_PR);
END; /* OF WHILE */

PROCESS_OWNER_NO = ' 1';
OBTAIN CALC RECORD (PROCESS_OWNER);
OBTAIN FIRST RECORD (PROCESS) SET (PRO_SUBPARTS_PR);
DO WHILE (ERROR_STATUS = OK);
  IF NOT SET (PR_INCEPTION_EV) EMPTY THEN DO;
    DISPLAY ('RELATIONSHIP(S) PROCESS INCEPTION CAUSES EVENT');
    OBTAIN FIRST RECORD (EVENT) SET (PR_INCEPTION_EV);
    DO WHILE (ERROR_STATUS = OK);
      DISPLAY (PROCESS_NAME || ' INCEPTION CAUSES ' || EVENT_NAME);
      OBTAIN NEXT RECORD (EVENT) SET (PR_INCEPTION_EV);
    END; /* OF WHILE */
  END; /* OF IF THEN */
  OBTAIN NEXT RECORD (PROCESS) SET (PRO_SUBPARTS_PR);
END; /* OF WHILE */

```

```

PROCESS_OWNER_NO = ' 1';
OBTAIN CALC RECORD (PROCESS_OWNER);
OBTAIN FIRST RECORD (PROCESS) SET (PRO_SUBPARTS_PR);
DO WHILE (ERROR_STATUS = OK);
    IF NOT SET (PR_TERMINATE_EV) EMPTY THEN DO;
        DISPLAY ('RELATIONSHIP(S) PROCESS TERMINATION CAUSES EVENT');
        OBTAIN FIRST RECORD (EVENT) SET (PR_TERMINATE_EV);
        DO WHILE (ERROR_STATUS = OK);
            DISPLAY (PROCESS_NAME || 'TERMINATION CAUSES' || EVENT_NAME);
            OBTAIN NEXT RECORD (EVENT) SET (PR_TERMINATE_EV);
        END; /* OF WHILE */
    END; /* OF IF THEN */
    OBTAIN NEXT RECORD (PROCESS) SET (PRO_SUBPARTS_PR);
END; /* OF WHILE */

PROCESS_OWNER_NO = ' 1';
OBTAIN CALC RECORD (PROCESS_OWNER);
OBTAIN FIRST RECORD (PROCESS) SET (PRO_SUBPARTS_PR);
DO WHILE (ERROR_STATUS = OK);
    IF NOT SET (PR_TRIGGERED_EV) EMPTY THEN DO;
        DISPLAY ('RELATIONSHIP(S) PROCESS TRIGGERED BY EVENT');
        OBTAIN FIRST RECORD (EVENT) SET (PR_TRIGGERED_EV);
        DO WHILE (ERROR_STATUS = OK);
            DISPLAY (PROCESS_NAME || ' TRIGGERED BY ' || EVENT_NAME);
            OBTAIN NEXT RECORD (EVENT) SET (PR_TRIGGERED_EV);
        END; /* OF WHILE */
    END; /* OF IF THEN */
    OBTAIN NEXT RECORD (PROCESS) SET (PRO_SUBPARTS_PR);
END; /* OF WHILE */

```

```

    END; /* OF IF THEN */

        OBTAIN NEXT RECORD(PROCESS) SET(PRO_SUBPARTS_PR);
END; /* OF WHILE */

I = 1;

PUT STRING(INPUT_OWNER_NO) EDIT(I) (F(4));

OBTAIN CALC RECORD(INPUT_OWNER);

DO WHILE (ERROR_STATUS = OK);

OBTAIN FIRST RECORD(INPUT) SET(INO_SUBPARTS_IN);

DO WHILE (ERROR_STATUS = OK);

    IF NOT SET(IN_CONSISTS_EL) EMPTY THEN DO;

        DISPLAY('RELATIONSHIP(S) INPUT CONSISTS OF ELEMENT');

        OBTAIN FIRST RECORD(ELEMENT) SET(IN_CONSISTS_EL);

        DO WHILE (ERROR_STATUS = OK);

            DISPLAY(INPUT_NAME || 'CONSISTS OF ' || ELEMENT_NAME);

            OBTAIN NEXT RECORD(ELEMENT) SET(IN_CONSISTS_EL);

        END; /* OF WHILE */

    END; /* OF IF THEN */

        OBTAIN NEXT RECORD(INPUT) SET(INO_SUBPARTS_IN);
END; /* OF WHILE */

I = I + 1;

PUT STRING(INPUT_OWNER_NO) EDIT(I) (F(4));

OBTAIN CALC RECORD(INPUT_OWNER);

END; /* OF OUTER WHILE */

IF (ERROR_STATUS = '0326') THEN CALL IDMS_STATUS;

I = 1;

```

```

PUT STRING (INPUT_OWNER_NO) EDIT (I) (F(4));
OBTAIN CALC RECORD (INPUT_OWNER);
DO WHILE (ERROR_STATUS = OK);
OBTAIN FIRST RECORD (INPUT) SET (INO_SUBPARTS_IN);
DO WHILE (ERROR_STATUS = OK);
    IF NOT SET (IN_CONSISTS_GR) EMPTY THEN DO;
        DISPLAY ('RELATIONSHIP(S) INPUT CONSISTS OF GROUP');
        OBTAIN FIRST RECORD (GROUP) SET (IN_CONSISTS_GR);
        DO WHILE (ERROR_STATUS = OK);
            DISPLAY (INPUT_NAME || 'CONSISTS OF ' || GROUP_NAME);
            OBTAIN NEXT RECORD (GROUP) SET (IN_CONSISTS_GR);
        END; /* OF WHILE */
    END; /* OF IF THEN */
    OBTAIN NEXT RECORD (INPUT) SET (INO_SUBPARTS_IN);
END; /* OF WHILE */

I = I + 1;
PUT STRING (INPUT_OWNER_NO) EDIT (I) (F(4));
OBTAIN CALC RECORD (INPUT_OWNER);
END; /* OF OUTER WHILE */

I = 1;
PUT STRING (OUTPUT_OWNER_NO) EDIT (I) (F(4));
OBTAIN CALC RECORD (OUTPUT_OWNER);
DO WHILE (ERROR_STATUS = OK);
OBTAIN FIRST RECORD (OUTPUT) SET (OUO_SUBPARTS_OU);
DO WHILE (ERROR_STATUS = OK);
    IF NOT SET (OU_CONSISTS_EL) EMPTY THEN DO;

```

```

    DISPLAY('RELATIONSHIP(S) OUTPUT CONSISTS OF ELEMENT');
    OBTAIN FIRST RECORD(ELEMENT) SET(OU_CONSISTS_EL);
    DO WHILE (ERROR_STATUS = OK);
        DISPLAY(OUTPUT_NAME || 'CONSISTS OF ' || ELEMENT_NAME);
        OBTAIN NEXT RECORD(ELEMENT) SET(OU_CONSISTS_EL);
    END; /* OF WHILE */

END; /* OF IF THEN */

    OBTAIN NEXT RECORD(OUTPUT) SET(OUO_SUBPARTS_OU);
END; /* OF WHILE */

I = I + 1;
PUT STRING(OUTPUT_OWNER_NO) EDIT(I) (F(4));
OBTAIN CALC RECORD(OUTPUT_OWNER);
END; /* OF OUTER WHILE */

I = 1;
PUT STRING(GROUP_OWNER_NO) EDIT(I) (F(4));
OBTAIN CALC RECORD(GROUP_OWNER);
DO WHILE (ERROR_STATUS = OK);
    OBTAIN FIRST RECORD(GROUP) SET(GRO_SUBPARTS_GR);
    DO WHILE (ERROR_STATUS = OK);
        IF NOT SET(GR_CONSISTS_EL) EMPTY THEN DO;
            DISPLAY('RELATIONSHIP(S) GROUP CONSISTS OF ELEMENT');
            OBTAIN FIRST RECORD(ELEMENT) SET(GR_CONSISTS_EL);
            DO WHILE (ERROR_STATUS = OK);
                DISPLAY(GROUP_NAME || 'CONSISTS OF ' || ELEMENT_NAME);
                OBTAIN NEXT RECORD(ELEMENT) SET(GR_CONSISTS_EL);
            END; /* OF WHILE */
        END IF;
    END DO;
END DO;

```

```

    END; /* OF IF THEN */

    OBTAIN NEXT RECORD(GROUP) SET(GRO_SUBPARTS_GR);

END; /* OF WHILE */

I = I + 1;

PUT STRING(GROUP_OWNER_NO) EDIT(I) (F(4));

OBTAIN CALC RECORD(GROUP_OWNER);

END; /* OF OUTER WHILE */

I = 1;

PUT STRING(ENTITY_OWNER_NO) EDIT(I) (F(4));

OBTAIN CALC RECORD(ENTITY_OWNER);

DO WHILE (ERROR_STATUS = OK);

OBTAIN FIRST RECORD(ENTITY) SET(ENO_SUBPARTS_EN);

DO WHILE (ERROR_STATUS = OK);

    IF NOT SET(EN_CONSISTS_EL) EMPTY THEN DO;

        DISPLAY('RELATIONSHIP(S) ENTITY CONSISTS OF ELEMENT');

        OBTAIN FIRST RECORD(ELEMENT) SET(EN_CONSISTS_EL);

        DO WHILE (ERROR_STATUS = OK);

            DISPLAY(ENTITY_NAME || 'CONSISTS OF ' || ELEMENT_NAME);

            OBTAIN NEXT RECORD(ELEMENT) SET(EN_CONSISTS_EL);

        END; /* OF WHILE */

    END; /* OF IF THEN */

    OBTAIN NEXT RECORD(ENTITY) SET(ENO_SUBPARTS_EN);

END; /* OF WHILE */

I = I + 1;

```

```

PUT STRING(ENTITY_OWNER_NO) EDIT(I) (F(4));
OBTAIN CALC RECORD(ENTITY_OWNER);
END; /* OF OUTER WHILE */
I = 1;
PUT STRING(ENTITY_OWNER_NO) EDIT(I) (F(4));
OBTAIN CALC RECORD(ENTITY_OWNER);
DO WHILE (ERROR_STATUS = OK);
OBTAIN FIRST RECORD(ENTITY) SET(ENO_SUBPARTS_EN);
DO WHILE (ERROR_STATUS = OK);
    IF NOT SET(EN_CONSISTS_GR) EMPTY THEN DO;
        DISPLAY('RELATIONSHIP(S) ENTITY CONSISTS OF GROUP');
        OBTAIN FIRST RECORD(GROUP) SET(EN_CONSISTS_GR);
        DO WHILE (ERROR_STATUS = OK);
            DISPLAY(ENTITY_NAME || 'CONSISTS OF ' || GROUP_NAME);
            OBTAIN NEXT RECORD(GROUP) SET(EN_CONSISTS_GR);
        END; /* OF WHILE */
    END; /* OF IF THEN */
    OBTAIN NEXT RECORD(ENTITY) SET(ENO_SUBPARTS_EN);
END; /* OF WHILE */

I = I + 1;
PUT STRING(ENTITY_OWNER_NO) EDIT(I) (F(4));
OBTAIN CALC RECORD(ENTITY_OWNER);
END; /* OF OUTER WHILE */
IF (ERROR_STATUS ^= '0326') THEN CALL IDMS_STATUS;
I = 1;
PUT STRING(OUTPUT_OWNER_NO) EDIT(I) (F(4));

```

```

OBTAIN CALC RECORD(OUTPUT_OWNER);
DO WHILE (ERROR_STATUS = OK);
OBTAIN FIRST RECORD(OUTPUT) SET(OUO_SUBPARTS_OU);
DO WHILE (ERROR_STATUS = OK);
    IF NOT SET(OU_CONSISTS_GR) EMPTY THEN DO;
        DISPLAY('RELATIONSHIP(S) OUTPUT CONSISTS OF GROUP');
        OBTAIN FIRST RECORD(GROUP) SET(OU_CONSISTS_GR);
        DO WHILE (ERROR_STATUS = OK);
            DISPLAY(OUTPUT_NAME || 'CONSISTS OF ' || GROUP_NAME);
            OBTAIN NEXT RECORD(GROUP) SET(OU_CONSISTS_GR);
        END; /* OF WHILE */
    END; /* OF IF THEN */
    OBTAIN NEXT RECORD(OUTPUT) SET(OUO_SUBPARTS_OU);
END; /* OF WHILE */

I = I + 1;
PUT STRING(OUTPUT_OWNER_NO) EDIT(I) (F(4));
OBTAIN CALC RECORD(OUTPUT_OWNER);
END; /* OF OUTER WHILE */
IF (ERROR_STATUS ->= '0326') THEN CALL IDMS_STATUS;
I = 1;
PUT STRING(EVENT_NO) EDIT(I) (F(4));
OBTAIN CALC RECORD(EVENT);
DO WHILE (ERROR_STATUS = OK);
    IF NOT SET(EV_WHEN_CO) EMPTY THEN DO;
        DISPLAY('RELATIONSHIP(S) EVENT WHEN CONDITION');
        OBTAIN FIRST RECORD(CONDITION) SET(EV_WHEN_CO);
    
```



```

DO WHILE (ERROR_STATUS = OK) ;
    DISPLAY(EVENT_NAME || ' WHEN ' || CONDITION_NAME) ;
    OBTAIN NEXT RECORD(CONDITION) SET(EV_WHEN_CO) ;
END; /* OF WHILE */

END; /* OF IF THEN */
ELSE DO;
    I = I + 1;
    PUT STRING(EVENT_NO) EDIT(I) (F(4));
    OBTAIN CALC RECORD(EVENT);
END; /* OF ELSE */
END; /* OF WHILE */
IF (ERROR_STATUS ^= '0326' & ERROR_STATUS ^= END_OF_AREA)
THEN CALL IDMS_STATUS;
DISPLAY(' PRESS ''ENTER'' TO CONTINUE') REPLY(CHOICE);
DISPLAY(' ');
DISPLAY(' ');
GO TO MENU;
EXIT_PROGRAM: FINISH;
CALL IDMS_STATUS;
DISPLAY('END OF PROGRAM');
DEC:PROCEDURE(REC_COUNT);
    DCL REC_COUNT CHAR(4) VAR,
        Y FIXED DECIMAL ;
    GET STRING(REC_COUNT) EDIT(Y) (F(4));
    Y = Y - 1;
    PUT STRING(REC_COUNT) EDIT(Y) (F(4));

```

END DEC;

END PSLPROG;

AN IMPLEMENTATION OF A SUBSET
OF PSL/PSA

by

FRANCIS B. HAJEK

B.A., Peru State College, 1961
M.S., Oklahoma State University 1966
Ed.D., Oklahoma State University, 1970

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree of

MASTER OF SCIENCE

Department of Computer Science

Kansas State University
Manhattan, Kansas

1984

ABSTRACT

This report describes an implementation of some of the capabilities of PSL/PSA using the IDMS data base management system. The project was implemented on the National Advanced Systems AS-5 computer at Kansas State University.

Problem Statement Language - Problem Statement Analyzer, PSL/PSA, is a software tool for automated systems analysis and documentation. PSL is a computer-processable language designed primarily to describe a software system in the requirements specification phase of the software life cycle. PSA is a software package that processes PSL statements and provides several reports constructed from the PSL information in the PSA data base.

The purpose of this implementation project was to provide a subset of PSL/PSA for use by the Software Engineering classes at Kansas State University.. Students using the system should learn something about the concepts of PSL/PSA and about writing requirements specifications while using an automated system.

The implementation was accomplished by mapping PSL objects to IDMS record types and PSL relationships to IDMS sets. This seems to be a natural mapping since PSL consists of objects and relationships between these objects and IDMS sets are named relationships between record types.

The project required an IDMS schema to be developed to incorporate the PSL information and an application program to interactively store, manipulate, and retrieve the information from the IDMS data base. The application program is menu driven and provides the above capabilities without the user having to deal with the complexities of IDMS.

The project could be expanded to include more of the objects and relationships of PSL and more of the report generating capabilities of PSA. However, there are some limitations of IDMS, particularly that of not being able to make a record type be related to itself in a set, which make a full implementation of PSL/PSA questionable. For the subset of PSL/PSA implemented in this project the program works well and provides a usable automated system for requirements specification.