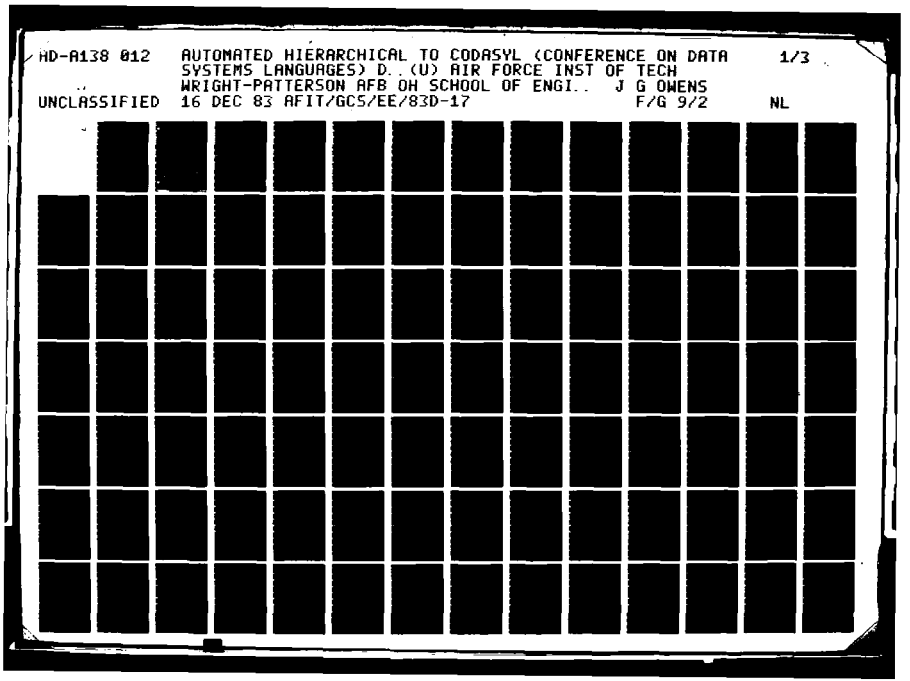
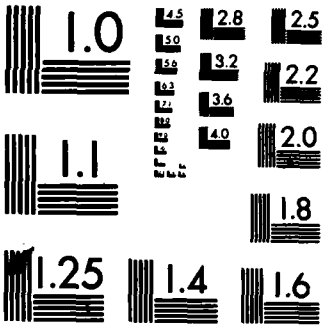

AD-A138 012 AUTOMATED HIERARCHICAL TO CODASYL (CONFERENCE ON DATA 1/3
SYSTEMS LANGUAGES) D..(U) AIR FORCE INST OF TECH
WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. J G OWENS
UNCLASSIFIED 16 DEC 83 AFIT/GCS/EE/83D-17 F/G 9/2 NL



AUTOMATED HIERARCHICAL TO CODASYL DATABASE INTERFACE SCHEMA TRANSLATOR

THESIS

AFIT/GCS/EE/83D-17

James G. Owens
Capt USAF

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY (ATC)
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GCS/EE/83D-17

AUTOMATED HIERARCHICAL TO CODASYL DATABASE INTERFACE SCHEMA TRANSLATOR

THESIS

AFIT/GCS/EE/83D-17 James G. Owens
Capt USAF

AFIT/GCS/EE/83D-17

AUTOMATED HIERARCHICAL TO CODASYL DATABASE INTERFACE SCHEMA TRANSLATOR

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University (ATC)
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science

by

James G. Owens
Capt USAF

Graduate Information Systems

16 December 1983

Preface

This thesis effort presents a user friendly system to translate a System 2000 schema into an equivalent IDMS schema, its associated DMCL, and subschema. I feel with the progress made on this thesis, the complete System 2000 to IDMS translation can be accomplished with other thesis efforts or special study projects.

I would like to express my sincere appreciation to Major Charles Lillie, who as my thesis advisor gave me guidance and encouragement. Thanks is also extended to Dr. Henry Potoczny and Dr. Thomas Hartrum, who as my thesis readers provided constructive comments to improve the quality of this thesis; to Captain Ed Oliver for his technical support in the area of IDMS; to Mr. James S. Pinckney for his technical support in the areas of System 2000 and IDMS; to Mr. Fred Pitts and the entire ASD/ADOM systems shop for their endless computer support.

Finally, I wish to thank my wife Dawn for enduring hours of complaining and separation which enabled me to complete this graduate program.

Contents

List of Figures

Abstract

An interactive hierarchical to CODASYL schema translator was designed and implemented with the goal of taking a System 2000 (hierarchical) schema as input and producing an equivalent IDMS (CODASYL) schema, storage schema (DMCL), and subschema. In addition, this system is to serve as part of an entire System 2000 to IDMS database translator system.

This thesis effort includes an analysis of the hierarchical and CODASYL models, an analysis of the System 2000 implementation of the hierarchical model, an analysis of the IDMS implementation of the CODASYL model, Data Flow Diagrams for the general hierarchical to CODASYL schema translator, structure charts for the System 2000 to IDMS schema translator, translator source code, and data dictionary.

I. Introduction

Background

The purpose of a database management system (DBMS) is to record and maintain information used by an organization in the organization's decision-making process. Some advantages of a DBMS are reduced data redundancy, improved data consistency, sharing data between users, enforcing data standards, maintaining data integrity and data independence.

Database Management Systems are classified into three major models; relational, network, and hierarchical. Each model uses a software design which will support a certain physical data representation in an efficient manner. Figure I - 1 illustrates the hierarchical model, figure I - 2 illustrates the network model, and figure I - 3 illustrates the relational model.

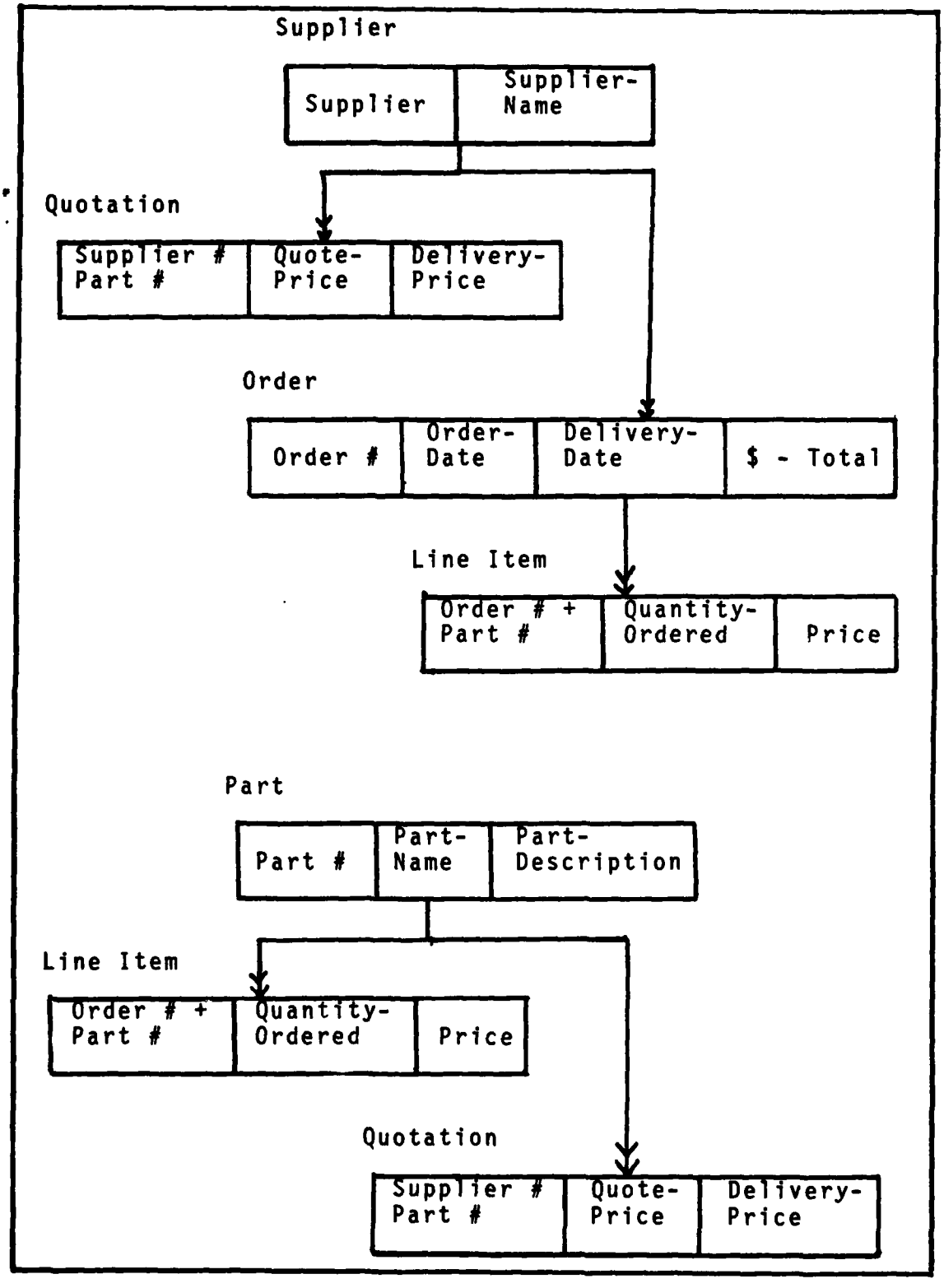


Figure I - 1. A Hierarchical View of a Database

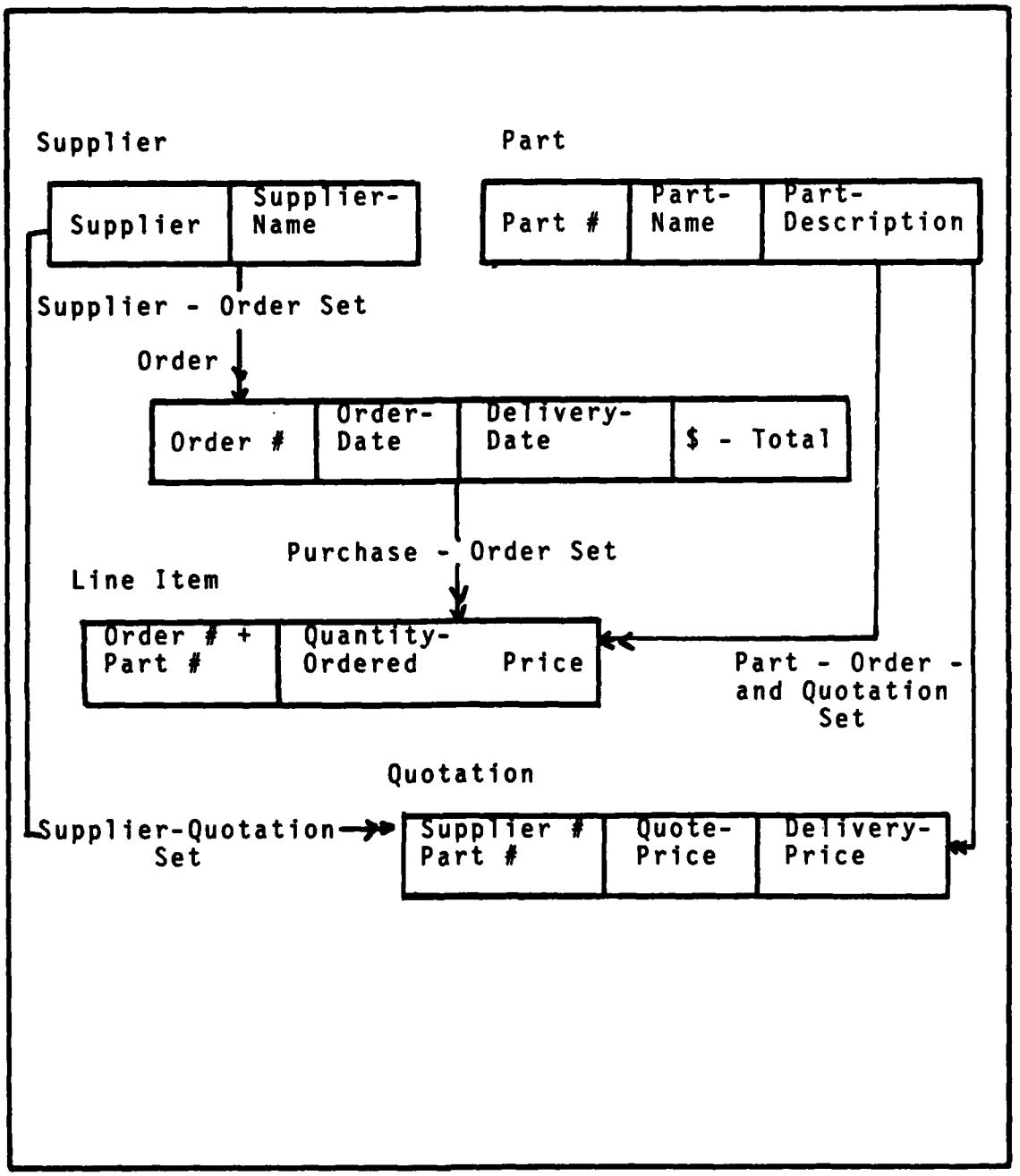


Figure I - 2. A Network View of a Database

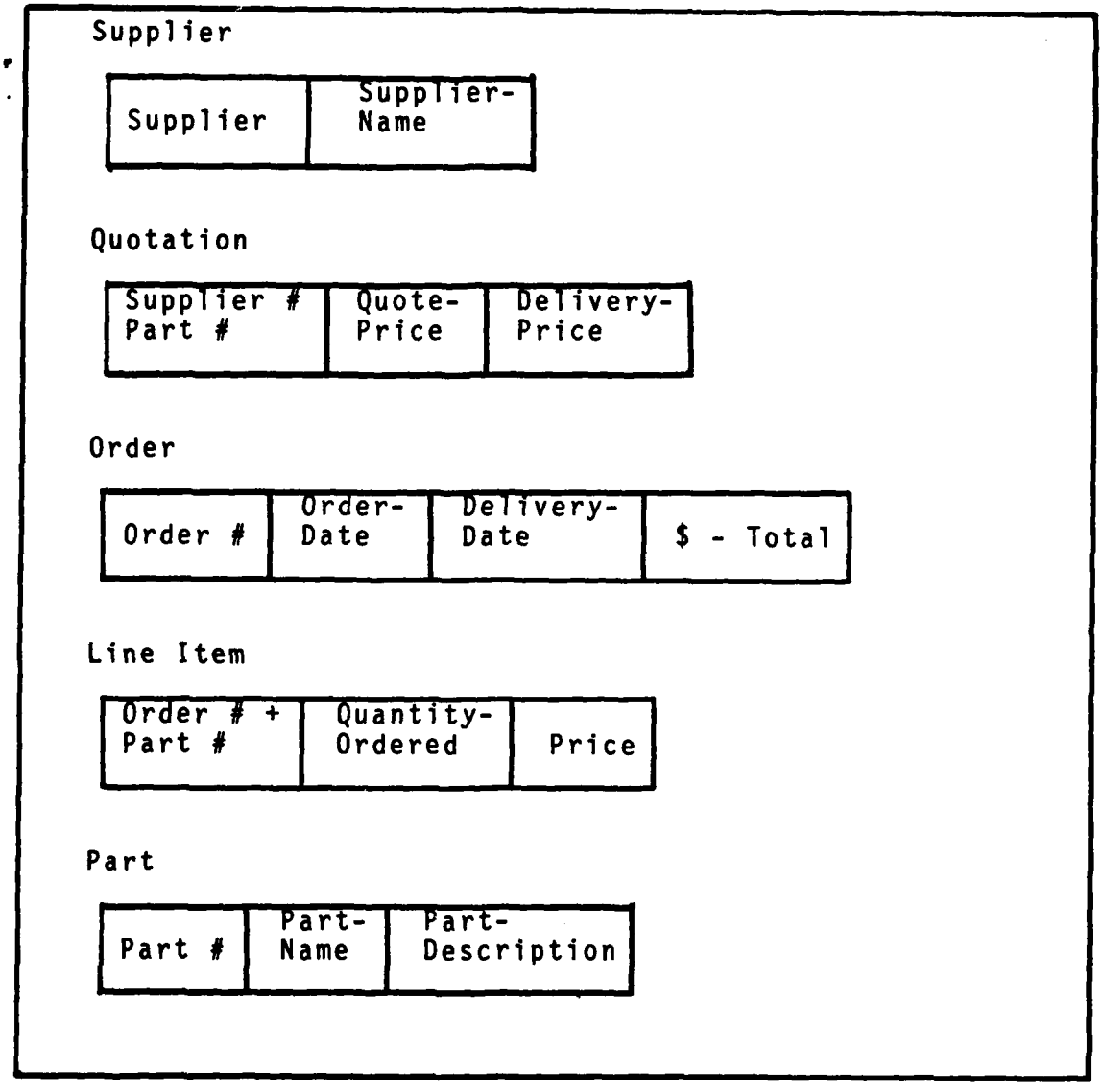


Figure I - 3. A Relational View of a Database

Since this thesis will deal with the inter-relationship between the hierarchical and the network approaches, they will now be discussed in further detail.

Hierarchical Approach

Some characteristics of the hierarchical model are record types, relationships connecting record types, only one relationship between record types, relationships (arcs) pointing only to the leaves of the tree, and 1:N relationships only. (Ref 1:121)

The hierarchical approach does not provide a means for providing many-to-many relationships between record types, but most hierarchical DBMS do provide the capability to support many hierarchical data trees. The user may thus represent many-to-many relationships by duplicating record types. (Ref 1:122)

Some specific advantages of the hierarchical approach are:

1. It is a simple data model which provides the user with few, easy-to-master, commands.

2. The types of relationships are much easier to implement than the other DBMS approaches. (Ref 1:122)

Some specific disadvantages of the hierarchical approach:

1. The restrictions imposed sometimes force an unnatural organization of the data.

2. Operations such as insertion and deletion become quite complex.

3. A delete operation can lead to the loss of information contained in a descendant record if null records are not allowed.

4. Symmetrical queries can not always be answered easily in a hierarchical system. (Ref 1:123)

Network Approach

The most comprehensive specification of a network model is in the April 1971 report published by the Data Base Task Group (DBTG) of the Conference on Data Systems Languages (CODASYL). Proposed changes to this document were submitted in 1973 and 1978. Interested readers are referred to [CODASYL 1971, 1973, 1977] for further information. The objective of the report was to specify the facilities of a DBMS and to propose the facility for embedding it into COBOL. The hierarchical and network approaches are very similar in most respects, therefore only the significant differences will be mentioned in this thesis.

The advantages of the network model are:

1. The network approach is more general, allowing more than one owner record type.

2. The network approach allows the user to model many-to-many relationships with greater ease.

3. The network approach is more symmetric.

4. Insertion and deletion operations are much easier to perform on the data base.

The prime disadvantage of the network approach is the complexity contained in both the data structure and in the Data Manipulation Language (DML). Since the network approach does allow more types of record constructs than the hierarchical approach, it will obviously need more operators to handle them, which will cause more complication in the DML. (Ref 2:70)

Description of the Problem

Nearly everyone would agree that any automated system developed should be "user friendly" to the point that it is at least usable by the end user for whom it was designed. Although there have been many meanings associated with the term "user friendly", for the purpose of this thesis, a software package is "user friendly" if it has the qualities of being usable by the user, supplies informative messages to

the user, and the user can exit the software package at any time. This is particularly true of database management systems (DBMS) where the end user typically has little or no data processing background.

The ASD Computer Center currently has Intel Corporations' System 2000 (S2k) and Cullinet Database Systems' Integrated Database Management System (IDMS). S2k is a hierarchical DBMS, where IDMS is a network structured implementation of the CODASYL Data Base Task Group Language specification. IDMS was designed for the end user. Current users of S2k feel that S2k was not designed for the end user. This feeling impedes the overall effectiveness of the Acquisition Management Information System (AMIS), which currently uses S2k. The size of the AMIS organization and the importance of their mission, would make a complete conversion to IDMS very costly.

Objective

The objective of this thesis is to develop a series of application programs which will accept a S2k schema as input and will return an equivalent IDMS schema, its associated DMCL, and also a subschema which can then be used to update the IDMS data dictionary.

Since the main objective of this thesis is a product that will aid the DBA of the ASD Computer Center in the translation of a S2k schema to an IDMS schema, the following

user requirements were established:

1. The systems must be interactive with the user.

2. The system will take a S2K schema as input and will produce an IDMS schema, an appropriate DMCL, and an appropriate subschema.

3. The produced software must allow the user to make changes to the newly generated schema easily.

4. The produced IDMS schema, DMCL, and subschema must be in the format that can be readily used as input by the supplied IDMS utilities.

Each of these requirements is necessary for a completely viable system.

The following assumptions are established in this thesis effort:

1. The S2k schema used as input will be created by using the supplied S2k UNLOAD utility.

2. All information needed by IDMS for a record description and an item description will be found in 2 input cards or less.

3. The S2k schema is expected to be in the form of 80 character records.

Approach

The thesis will be accomplished by performing the following steps:

1. Conduct a literature search for previously conducted work in the area of translating a hierarchical database to a network database.

2. Develop an algorithm to accept user input from the terminal. This module will be used to store schema description information, file description information, and area description information into a file which will be used in step 4.

3. Develop an algorithm which will accept a S2k schema as input and produce an output file which will be used to create the IDMS schema, DMCL, and subschema. This file will be available to the user for editing if desired.

4. Develop an algorithm which will take the files created in steps 1 and 3 as input and produce a schema file, a DMCL file, and a subschema file. These files will be used as input to the supplied IDMS programs to update the IDMS data dictionary. A user's manual (Appendix A) will be supplied upon completion of the software.

II. Analysis

This chapter will present a brief overview of SYSTEM 2000 and IDMS including a discussion of their respective physical data formats, schema characteristics, and host programming languages.

SYSTEM 2000

SYSTEM 2000 is a database management system developed and maintained by Intel Systems Corporation. This DBMS is operational with CDC, IBM, and Univac computer systems.

This thesis will involve only the IBM version of SYSTEM 2000. In this version the IBM Basic Direct Access Method (BDAM) is used for physical I/O between SYSTEM 2000 and the direct access storage device.

Physical Data Format

The physical storage structure of SYSTEM 2000 consists of seven fixed-length-record direct (BDAM) files. The first file (Master Record) contains information about control of the database structure (database name, cycle, passwords, etc...). An update log flag for the database is also included in the master record. File 2 (Definition Table) contains a description of the database structure. It describes all elements, giving characteristics such as

length, type, key or non-key, and other structure information. File 3 (Distinct Values Table) contains a pointer for every item that has been designated a key by the user. A key in SYSTEM 2000 can be any item that the user wishes to improve access to. If the item has more than one value assigned to it then the pointer points to the Multiple Occurrence Table, otherwise the pointer points to the Hierarchical Table. File 4 (Multiple Occurrence Table) contains all of the distinct values that an item can have that has been specified in the Distinct Values Table. File 5 (Hierarchical Table) contains a pointer for every item whether it is a key item or a non-key item. This pointer contains the location of the item in the Data Table. File 6 (Data Table) contains the actual fixed-length data records. File 7 (Extended Field Table) contains values exceeding the picture width allotted for textual or character items. (Ref 3:271)

Schema

The database schema in SYSTEM 2000 is organized as a hierarchical tree with a maximum of 32 levels and a maximum of 1000 schema items (components). Any number of records can be specified at a given level, and any number of items can be defined in a record. The user must decide which items will be keys, the type of each item, and the picture of each item. The DEFINE processor in the Data Base Definition

Facility is used to define the schema. DEFINE, the data definition language (DDL), is used to declare individual components of the database, to map a definition, and to modify a definition either before or after a database has been loaded. (Ref 4:3)

Defining a schema in SYSTEM 2000 consists of assigning component numbers, component labels, specifying record memberships for items, specifying record relationships for records, and designating the item type, picture, key status, and padding for schema items. Padding is used when the user wishes to reserve physically contiguous index space. This can make the use of the index area more efficient and easier to maintain. Any number of schema items may be declared a key. (Ref 4:104)

String and Function statements are also allowed in the schema definition. Strings allow the user to store entire SCF commands, parts of SCF commands, or a series of SCF commands in their schema. Functions allow the user to store arithmetic expressions and to specify a numeric item type for the display of output. Strings and Functions also require a component number and a component name. Strings and Functions will not be handled by this thesis effort. (Ref 5:6-1)

Host Programming Languages

SYSTEM 2000 supports Assembler, COBOL, FORTRAN, and PL/1 as its host programming languages. The Programming Language Extension (PLEX) is a Data Manipulation Language. PLEX allows users of the previously mentioned languages to use SYSTEM 2000 within their application programs. The SYSTEM 2000 PLEX commands are embedded in the program to locate the desired information in the database. SYSTEM 2000 supplies a preprocessor for each of the programming languages which will then replace the PLEX instructions with the appropriate CALL statements. The output from the preprocessor is then compiled and linked before execution. (Ref 6:1)

IDMS

The Integrated Database Management System (IDMS) is an implementation of the CODASYL Data Base Task Group Language specification. It operates in any IBM 360/370 OS/DOS/VS environment including 30XX, 4300 series with FBA devices, and all plug-compatible mainframes. The IBM Basic Direct Access Method (BDAM) or Virtual Sequential Access Method (VSAM) is used for physical I/O between IDMS and the direct access storage device. IDMS provides separate facilities for the description of data and the manipulation of data. This separation removes the data description function from

the scope of the application programs and allows integration of all data and data relations into a database which is common to all application programs which use it.

Physical Data Format

The physical structure of a database in IDMS is determined by the contents of the database keys, by the Database Administrator's definition of pages, areas, files, and location modes. (Ref 7:2-26)

IDMS assigns a database key to each record occurrence when it is entered into the database. This key remains unchanged as long as the record occurrence remains in the database. Each record occurrence appears as two components (Prefix and Data) in the database. The Prefix component contains a description of the set relationships for the record occurrence, in the form of pointers (next, prior, and owner records) in all sets in which the record is a member. The Data component contains the actual value of the record occurrence in the form specified by the user. (Ref 7:2-26)

The database is subdivided into a number of blocks (pages). A page can contain as many record occurrences as will fit. Variable-length records can be fragmented across several pages. Each page has a unique page number and each record within a page is assigned a unique line number. The record's database key is a concatenation of the page and line numbers. (Ref 7:2-29)

Pages are grouped into larger subdivisions called areas. The user can specify one or more areas, but the following rules must be satisfied:

1. The pages that make up an area must be a range of sequentially numbered pages.

2. Gaps in the page numbers can occur between pages.

3. A page number can only belong to one area.

4. All record occurrences of one type must exist in one area.

5. All pages in one area must be the same size.

The database is stored as one or more files or data sets on a direct access storage device. These devices are formatted into BDAM blocks or VSAM control intervals. Each database page corresponds to a direct access block, which means that data transfers are always performed a page at a time. (Ref 7:2-30)

The assignment of a record to a particular page in its area is determined by the user's specification of location modes. CALC, VIA, DIRECT, and Physical Sequential are the four different location modes which can be specified by the user. (Ref 7:2-31)

Schema

The schema is defined by means of the schema DDL, which is comprised of a schema description, a file description, an area description, a record description, and a set description.

The schema description identifies the schema and includes any number of comment and documentation entries. The file description identifies the database files and assigns a ddname or filename. The area description identifies the area by name, specifies the page range of the area, and maps the area into a file or files. Each area in the database must be included in the area description. Each record in the database is included in the record description, which contains the record name, record id, synonym, location mode, within, minimum root, minimum fragment, call, element description, and copy. For each set in the data base, the set description specifies the name, order, mode, owner, member, and copy. Further information on the IDMS schema can be found in (Ref 7:4-7).

Host Programming Languages

IDMS supports COBOL, PL/1, FORTRAN, Assembler, and any other programming languages supporting a CALL statement or equivalent. Data Manipulation statements may be included anywhere within the procedure coding of an application program. Before compilation, a preprocessor validates and converts all DML statements into CALL statements and updates the data dictionary accordingly.

System Description & Specification

This section will discuss in further detail the user requirements and assumptions that were established in this thesis.

The user requirements, as stated in chapter 1, established for this thesis were:

1. The system must be interactive with the user.
2. The system will take a S2k schema as input and will produce an IDMS schema, an appropriate DMCL, and an appropriate subschema.
3. The produced software must allow the user to make changes to the newly generated schema easily.
4. The produced IDMS schema, DMCL, and subschema must be in the format that can be readily used as input by the supplied IDMS utilities.

User requirement 1 establishes that the translation software must be very easy to use and should prompt the user with a clear distinct question if any information is required from him. All communication between the software and the user should be conducted via a CRT.

User requirement 2 establishes that the translation software must do the translation with as little user intervention as possible. The software should require a S2k schema definition as input and should produce an IDMS schema, DMCL, and subschema for that particular S2k schema.

User requirement 3 establishes that the translation software must create an intermediate file which the user can modify, if desired via the editor, before the final schema, DMCL, and subschema have been produced. These changes include changing record names, changing item/element names, changing record id numbers, changing picture definitions, changing owner/member relations, changing set names, and changing location modes for a particular set.

User requirement 4 establishes that the produced IDMS schema, DMCL, subschema should be ready for use as input to the supplied IDMS utilities for updating of the IDMS data dictionary upon completion of the translation software. No syntax errors should be encountered due to the translation software. Checking for IDMS reserved words does NOT fall into this category.

The assumptions, as stated in chapter 1, established for this thesis were:

1. The S2k schema used as input will be created by using the supplied S2k UNLOAD utility.

2. All information needed by IDMS for a record description and an item description will be found in 2 input cards or less.

3. The S2k schema is expected to be in the form of 80 character records.

Assumption 1 establishes that the S2k schema used as input to the translation software must be formed by the S2k UNLOAD utility. The UNLOAD utility formats the schema in a certain fashion which will be expected by the translation software. If the schema is NOT created by the UNLOAD utility, errors may arise.

Assumption 2 establishes that the necessary information needed for an IDMS schema will be found in 2 cards or less. The S2k UNLOAD utility uses indentation to make the schema easy to read by the user. This indentation could cause numerous blank characters to be placed at the beginning of a record. The translation software will allow 2 input cards per description.

Assumption 3 establishes that the S2k schema must be in the form of 80 character records. The S2k UNLOAD utility allows the user to specify the desired record size, therefore the user should request 80 as the record size.

Requirement Analysis

The S2k schema contains record names, record element names and descriptions, along with information showing the hierarchical tree representation of the records in the database. No information concerning virtual records or the linking of two or more hierarchical trees is available in the schema. For this reason the resulting IDMS schema will also show the database in a hierarchical tree representation. However, flexibility should be incorporated in the design to allow for the different implementations of the hierarchical model.

The IDMS schema requires, in addition to the information contained in the S2k schema, the IDMS schema name, author's name, database file name, the type of storage device where the database is stored, the IDMS area name and its associated pages, and finally the desired record id number of the first record in the schema along with the id number increment for the following records. The associated DMCL and subschema will require the schema information along with the desired DMCL and subschema names. The additional information for the schema, DMCL, and subschema will need to be provided by the DBA.

Figure II - 1 is the top level DFD for the schema translator system. Appendix B contains the detailed DFD's of the system and Appendix D contains the data dictionary.

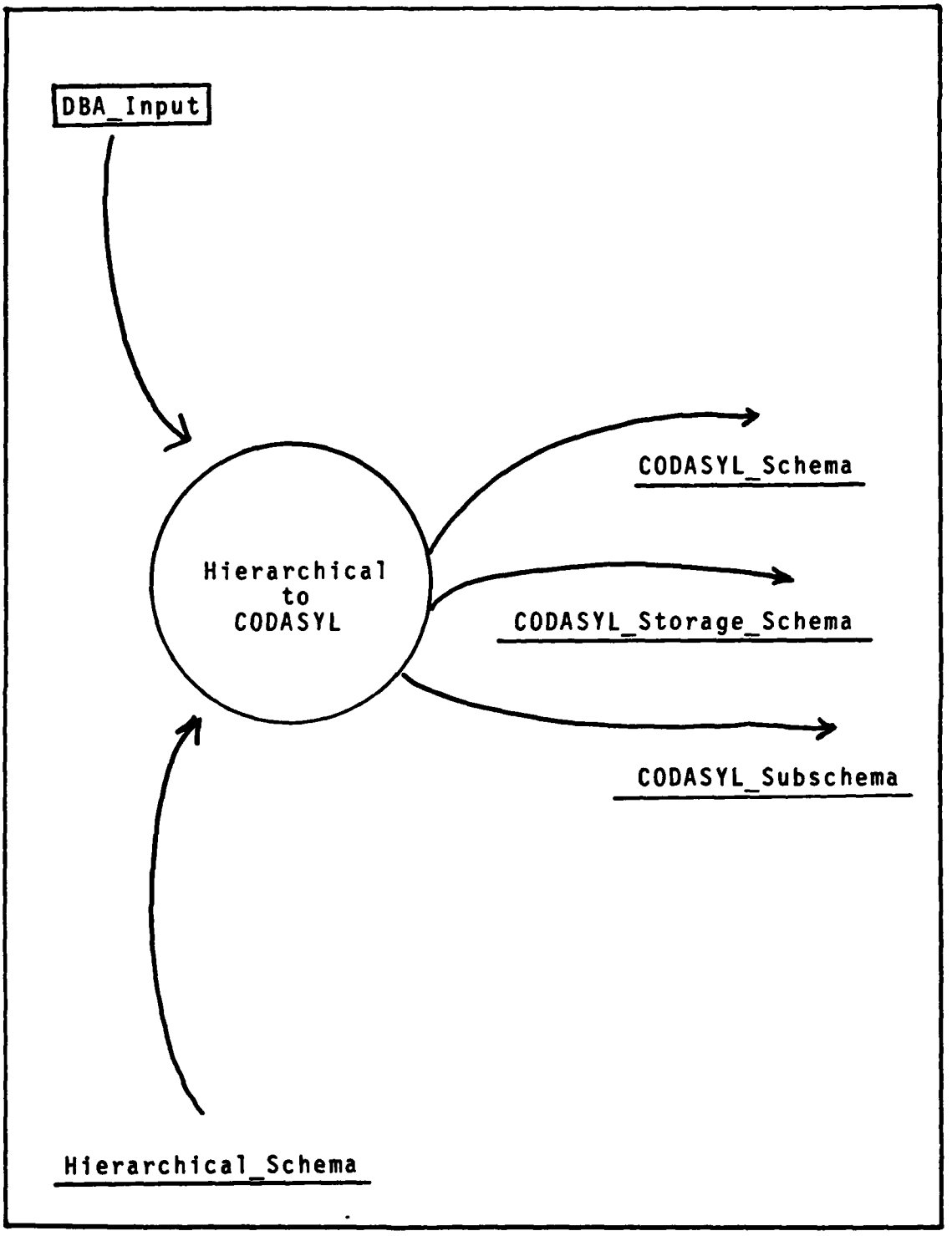


Figure II - 1. Context Diagram for this Thesis

III. Design and Implementation

Introduction

This chapter will present the software engineering tools used and why, a discussion of the design considerations, a brief description of each of the modules, and a discussion of the implementation.

Software Engineering Tools

The software engineering tool used in this thesis was structure charts. Structure charts were used as opposed to the Structured Analysis and Design Technique (SADT) method or the HIPO (Hierarchy, plus Input, Process, Output) because structure charts are easier to use, show a hierarchical representation, and show data and control flow.

Figure III - 1 is the top level structure chart for the schema translator system. Appendix C contains the detail structure charts and Appendix D contains the data dictionary.

Design Considerations

Most hierarchical DBMS supply an UNLOAD facility which will place the schema into a sequential data file in a standard format. A common format will aid in the gathering of information from the hierarchical schema.

The mapping of the record names and element names will require research concerning the restraints placed upon them by the two database management systems. For example, S2k allows the record and element names to be a maximum of 256 characters, where IDMS restricts record names to be 16 characters or less and element names to be 32 characters or less.

Element descriptions can also pose a problem. Some DBMS have special element descriptions specifically for their particular system. For example, S2k has a DATE description. This type of description is used specifically for storing date information in the database. IDMS does not support this description, therefore the DATE description must be mapped into an acceptable IDMS description.

The CODASYL model also requires "set" information to be included in the CODASYL schema. If the hierarchical schema does not contain virtual records or redundant data, then the mapping will be a hierarchical tree representaion in the CODASYL DBMS. This type of arrangement requires that the top level record be accessed by the "CALC" mode, and all other records of the tree to be accessed by the "VIA" mode. The set names will need to be generated or received from the DBA by some means. If virtual records or redundant records do exist, then the appropriate type of set must be generated to replace the virtual record or to remove the need of having redundant data in the database.

Additional information not found in the hierarchical schema may be required for the CODASYL schema. This information must be received from the DBA.

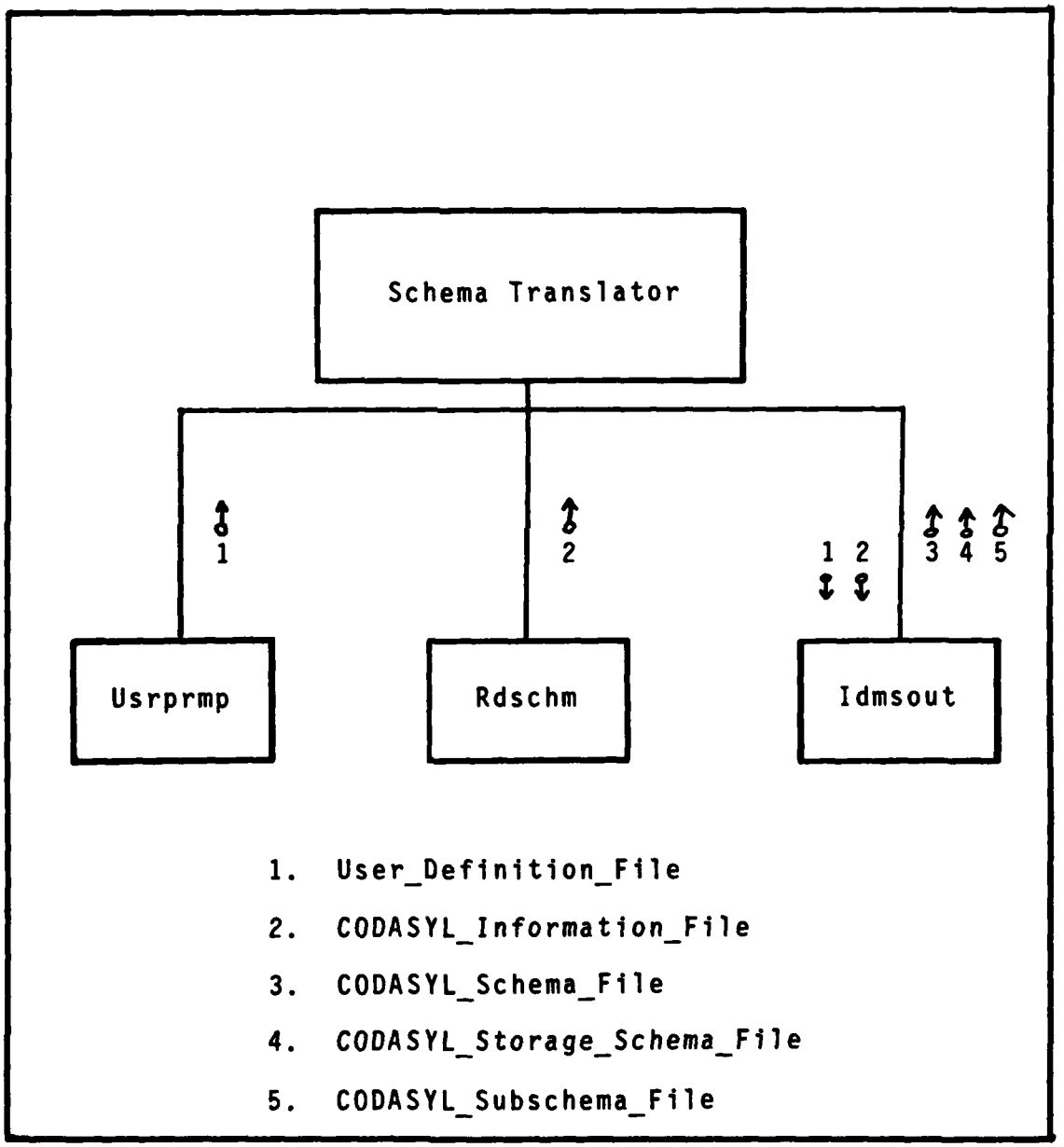


Figure III - 1. Top level Structure Chart for this Thesis

Module Descriptions

The three main modules will now be discussed including their sub-modules.

Usrprmp is designed to prompt the DBA for the IDMS schema name, schema version number, author name, author phone number, IDMS file name, the device for which the file resides, IDMS area name, starting and ending page numbers of the area, IDMS DMCL name, and the IDMS subschema name. This information will be stored in a file (USERDEFS) which will be used as input by IDMSOUT for the building of the IDMS schema, DMCL, and subschema. Usrprmp will put the appropriate prompt request and will then call Rdwrit. This loop will continue until all of the necessary information has been received.

Rdwrit will be the module responsible for reading the input buffer from the terminal in response to a prompt request. Rdwrit will then take the information and write it to the appropriate external file which will be used at a later time.

Rdschm is designed to accept a S2k schema as input (S2KSCHEM) and to produce as output the intermediate file (IDMSMODL) which can be edited by the DBA to make any necessary changes. The unchanged file will be in a valid format to generate an IDMS schema, DMCL, and subschema if desired by the DBA. Rdschm will call Rdbuff, Frstrec,

Readout, Getnum, Getname, Gettype, and Schmbl to complete its assigned task and will terminate when the end-of-file marker of the S2k schema file is encountered.

Rdbuff is designed to read in all of the information pertaining to a record or item definition. The information will be found in two cards or less. Rdbuff will call Chkpar, Findpat, Nonblnk to accomplish its task.

Chkpar is designed to check for matching left and right parenthesis. If all of the parenthesis have matches, then Chkpar will inform Rdbuff not to read another card, otherwise Rdbuff will be informed to read another input card.

Findpat is designed to find a string of characters in a particular character string at a given starting position in the character string. Findpat will return the position of the desired pattern in the character string if it is found, or Findpat will return a zero value if the string is not found. This procedure is used by various other modules throughout Rdschm.

Nonblnk is designed to find the next nonblnk character position in a given character string starting with a given position in the string. Nonblnk will return the position if a nonblnk character is found, otherwise Nonblnk will return a zero value to the calling module. Nonblnk is used by many modules throughout Rdschm.

Frstrec is designed to find the name of record zero in the given S2k schema. Frstrec uses Findpat and Nonblnk to accomplish its task.

Readout is designed to write the IDMS record information to the external file (S2KTEMP) which will be used later, and also to write a log report for the user showing the current S2k schema information record along with the created IDMS schema information record.

Getnum is designed to find the S2k component number of the current S2k schema record being processed. Getnum will use Nonblnk and Findpat to accomplish its task.

Getname is designed to find the name of the current S2k schema record being processed. Getname will use Nonblnk and Findpat to accomplish its task.

Gettype is designed to determine whether the current S2k schema record being processed is a S2k record description, or a S2k item description with an integer picture definition, or a S2k item description with a decimal picture definition, or a S2k item description with a money picture definition, or a S2k item description with a date picture definition, or a S2k item description with a text picture definition, or, finally, if it is a S2k item description with a character picture definition. Gettype will call Findpat, Recfnd, Intfnd, Chrfnd, Datefnd, Decfnd, and Money depending upon which is the appropriate action to take.

Recfnd is designed to build an output record containing the necessary information for an IDMS record description. Recfnd will insure that the record name is 16 characters or less and that all invalid characters in the name have been replaced by a hyphen. Recfnd will use Findpat to accomplish this task.

Intfnd is designed to build an output record containing a correctly formed IDMS item description of a S2k item description with a S2k integer picture definition. Intfnd will insure that the item name is 32 characters or less and that all invalid characters in the name have been replaced by a hyphen. Intfnd will use Findpat to accomplish this task.

Chrfnd is designed to build an output record containing a correctly formed IDMS item description of a S2k item description with a S2k character picture definition. Chrfnd will insure that the item name is 32 characters or less and that all of the invalid characters have been replaced by a hyphen. Chrfnd will use Findpat to accomplish this task.

Datfnd is designed to build an output record containing a correctly formed IDMS item description of a S2k item description with a S2k date picture definition. Datfnd will insure that the item name is 32 characters or less and that all invalid characters have been replaced by a hyphen. Datfnd will use Findpat to accomplish this task.

Decfnd is designed to build an output record containing a correctly formed IDMS item description of a S2k item description with a S2k decimal picture definition. Decfnd will insure that the item name is 32 characters or less and that all invalid characters in the name have been replaced by a hyphen. Decfnd will use Findpat to accomplish this task.

Money is designed to build an output record containing a correctly formed IDMS item description of a S2k item description with a S2k money picture definition. Money will insure that the item name is 32 characters or less and that all invalid characters have been replaced by a hyphen. Money will use Findpat to accomplish this task.

Schmbl is designed to be the driver module responsible for using the file (S2KTEMP) created by Readout as input to produce the intermediate file (IDMSMODL) which will be edited by the DBA. This file will contain the record description information along with its respective location mode information, as well as all of the items which make up each record. Schmbl will use Getinfo and Bildlog to accomplish this task.

Getinfo is designed to place all of the information contained in the file S2KTEMP into memory for future reference by the other modules which make up Schmbl.

Bildlog is designed to actually construct the IDMSMODL file. Bildlog will use Recdum and Getmode to complete this task.

Recdum is designed to build a "dummy item" if the current S2k record has no items which belong to it. S2k supports dummy records where IDMS does not.

Getmode is designed to determine whether a given record's location mode is CALC or VIA. If the location mode is determined to be VIA then Calcrec is called, otherwise Getmode will build the correct output information for a location mode of CALC.

Calcrec is designed to build the necessary output record for a location mode of VIA for a given record. A unique set name will be created as well as the owner and member information.

Idmsout is designed to use the file IDMSMODL as input to produce the IDMS schema file, DMCL file, and the subschema file. Idmsout will call Schmdes, Filedes, Areades, Bildrec, Bildset, DMCL, and Subschm to complete its assigned task.

Schmdes is designed to call the module Getdate and to use the file USERDEFS as input to produce the schema description section of the IDMS schema. The produced portion of the schema will be written to the file SCHEMA.

Getdate is designed to return the current date to the Schmdes module.

Filedes is designed to use the USERDEFS file as input to produce the file description section of the IDMS schema. The produced portion of the schema will be written to the file SCHEMA.

Areades is designed to use the file USERDEFS as input to produce the area description section of the IDMS schema. The produced portion of the code will be written to the file SCHEMA.

Bildrec is designed to build the record description section of the IDMS schema. Bildrec will call Recinfo and Itminfo to accomplish this task.

Recinfo is designed to use the file IDMSMODL as input to build the record information portion for each record in the IDMS schema. Information included is the record name, record id number, record location mode, and its associated set name if the location mode is VIA. Recinfo will call Loadrec and Loadset to accomplish this task. The produced information will be stored in the file SCHEMA.

Loadrec is designed to store each record name, next pointer number, and prior pointer number into memory for future use.

Loadset is designed to store each set name, its owner, and its member into memory for future use.

Itminfo is designed to build the item information portion of each record in the record description section of the IDMS schema. Itminfo will accept the file IDMSMODL as input and will store the produced information in the file SCHEMA.

Bildset is designed to build the set description section of the IDMS schema. Bildset will call Dbkey to accomplish its assigned task.

Dbkey is designed to produce the owner, member, next pointer, and prior pointer for each set in the schema. The produced output will be stored in the file SCHEMA.

DMCL is designed to produce the IDMS DMCL source code to be used as input by the DBA for the associated schema. The produced output will be stored in the file DMCL.

Subschm is designed to produce the IDMS subschema source code to be used by the DBA for the associated schema. The produced output will be stored in the file SUBS.

Implementation

USRPRMP, RDSCHM, and IDMSOUT were implemented on a National Semiconductor model 7000 computer at the ASD Computer Center. This computer is one of the many IBM 370 compatible computers that is available in today's market. I was restricted to the IBM architecture because the versions of S2k and IDMS were also IBM.

The high level language chosen to code the software was PL/1. My familiarity with the language and the string manipulation features provided by PL/1 were the deciding factors in the selection of the language.

Date, Index, Length, Substring, and Translate were the PL/1 built-in functions used in the implementation of this thesis.

Date returns a character string of length six composed of digits. The first pair of digits are the current year, the second pair the current month, and the final pair the current day. (Ref 8:437)

Index is used to find a certain pattern of characters in a parent string of characters. Index will return the integer position of the character pattern if it is found, otherwise it will return a zero. (Ref 8:434)

Length will return the number of bits or characters in a given parent string. (Ref 8:434)

Substring extracts a substring of length j from the parent string starting in the ith position. If no j is given, the remainder of the string from i on is assumed. (Ref 8:434)

Translate returns a character string r of the same length as the source string s. However, any character in s which is included in the string b is translated to the character which appears in the corresponding position of the

string a. Thus a and b should be of the same length, and the pairs of characters in strings a and b represent the before and after states of the translation. (Ref 8:434)

Implementation Changes

Implementation of the software went according to plan with no major difficulties. The software worked as required by the DBA with no failures experienced. Execution time was very good since the software was running in the foreground to meet the user requirement of being interactive. However it was noted that once the files were given to the translator software, there was no need to continue running in the foreground during the actual translation of the S2k schema. No further input would be required from the user, therefore the translation should be executing in the background, allowing better utilization of the CPU. Satisfying this request prompted the development of TSO command procedures, which would prompt the user for the necessary information, build the appropriate JCL for the particular job, and then submit the job in the background for execution. This method of operation gave the user more efficient utilization of the CPU and the interactive capability.

IV. Testing

This chapter will discuss the test plan and test results of this thesis effort.

Test Plan

This thesis will be designed in a top-down structure by means of structure charts. The top-down design will allow me the advantage of discovering problems in the design before total implementation has been completed.

Usrprmp has several prompts for information from the user. These prompts will be given a null response in test1, a response of 80 or less characters in test2, and a response of more than 80 characters in test3. The USERDEFS file will then be edited to verify that the file contains the actual input from the user in test4. No checking on the validity of the data is necessary at this point, since the IDMS supplied utilities will perform the checks automatically.

Rdschm will be tested by performing the following group of tests:

Test5 will check the Findpat module for accuracy. This module will be needed for syntax checking in the S2k input schema.

Test6 will check the module Nonblnk for accuracy. This module will be needed for throwing away blank characters in the input stream. This module will be very important in the

case of the input information exceeding more than one card.

Test7 will check Rdschm for the ability of finding the name of the S2k record zero. This information is located in the user information section of the S2k schema as opposed to the component information of the schema.

Test8 will check Rdschm for the ability of finding all S2k schema statements which contain the system separator (*). The separator is the key character used in determining whether or not a statement contains needed schema information. All statements which do not meet this requirement should be discarded. This test does not allow for information spanning across two cards.

Test9 will test Rdschm for the ability of retrieving component information if it spans across two cards. The key here is balanced parenthesis. Each complete S2k component statement will have balanced parenthesis.

Test10 will test Rdschm for the ability of merging the information, if it spans across two cards, into one continuous card. As stated previously, the S2k UNLOAD utility will indent information to make it more readable for the user. However, this will put blanks in the middle of the component information when it is merged. The blanks will then have to be removed to get the information correctly.

Test11 will test Rdschm for the ability of finding the correct S2k component number.

Test12 will test Rdschm for the ability of finding the correct S2k component name.

Test13 will test Rdschm for the ability of determining whether a component is a record or an item.

Test14 will test Rdschm for the ability of truncating the current component name based on whether the current component is a record or an item. If the component is a record, then the current component name should be truncated to 16 characters or less, otherwise the current component is an item and the name should be truncated to 32 characters or less. In both cases all illegal characters found in the name should be replaced with a hyphen with the exception that the name cannot end with a hyphen.

Test15 will test Rdschm for the ability of determining the owner of a given component. If the current component is a record, then the owner will be the record of the previous level in the hierarchical tree. Otherwise the owner will be the record of which the current item component is a member.

Test16 will test Rdschm for the ability of taking an S2k item component and determining whether it is an integer item, a decimal item, a date item, a money item, a text item, or a character item.

Test17 will test Rdschm for the ability of translating the current S2k item picture description to a valid IDMS picture description based on the S2k component type.

Test18 will test Rdschm's log and S2KTEMP file. The log should contain each S2k component and its associated IDMS translation. Tests 5 - 17 were verified by reviewing the log after each test. The S2KTEMP file contains only the IDMS

translation of the S2k components and will be used as input by the remainder Rdschm modules. This file will be verified by editing the file.

Test19 will test Rdschm for the ability of reading the 'S2KTEMP file and storing the information into memory.

Test20 will test Rdschm for the ability of determining whether a record has no members. If the record has no members then a dummy item must be generated for the record.

Test21 will test Rdschm for the ability of determining the location mode for each the records in the schema. The first record (S2k record zero) should be assigned the location mode CALC with the first item of the record being used as the hashing value. All other records will be assigned the location mode VIA.

Test22 will Rdschm for the ability of generating unique set names for each of the records assigned the location mode of VIA.

Test23 will test Rdschm for the ability of finding the owner and member record names for each of the sets being built for the location mode VIA.

Test24 will review the IDMSMODL file created by Rdschm. The first record in the IDMSMODL should contain the number of records and items in the file. The remaining IDMSMODL file should be comprised of four types of 80 character records TYPE1, TYPE2, TYPE3, and TYPE4. TYPE1 represents the record description, TYPE2 represents record location mode of CALC, TYPE3 represents location mode of VIA, and TYPE4 represents

the items of each of the records. The format of these records should be:

TYPE1 -	col 1	'R'	length 1
	col 3	IDMS record name	length 16
	col 20	IDMS record number	length 4
	col 25	S2k record number	length 4
	col 30	S2k owner record number	length 4
	col 35	record item pointer	length 4
TYPE2 -	col 1	'C'	length 1
	col 5	Calc item id	length 32
TYPE3 -	col 1	'V'	length 1
	col 5	set name	length 16
	col 25	owner name	length 16
	col 45	member name	length 16
TYPE4 -	col 1	'I'	length 1
	col 3	S2k component number	length 4
	col 8	IDMS item name	length 32
	col 41	IDMS item description	length 28
	col 70	S2k record number	length 4

The file should be checked by editing the file for verification.

Idmsout will be tested by performing the following group of test:

Test25 will test Idmsout for the ability of generating the schema description section of the IDMS schema.

Test26 will test Idmsout for the ability of generating the area description section of the IDMS schema.

Test27 will test Idmsout for the ability of generating the record description section of the IDMS schema.

Test28 will test Idmsout for the ability of generating the DMCL file of the current IDMS schema.

Test29 will test Idmsout for the ability of generating the subschema file for the current IDMS schema.

Test30 will be an execution of the IDMS supplied schema compile program using the Idmsout produced SCHEMA file as input.

Test31 will be an execution of the IDMS supplied DMCL compile program using the Idmsout produced DMCL file as input.

Test32 will be an execution of the IDMS supplied subschema program using the Idmsout produced SUBS file as input.

Test33 will consist of logging onto the IDMS DBMS and verifying that the newly generated schema is in fact contained in the data dictionary.

Test Results

The individual modules of the thesis were tested in the previously defined fashion and were then compiled together producing the final product of the thesis. Two S2k schemas were used in the final series of tests.

Test schema one contained all of the different types of S2k records and items, but did not contain record or item descriptions that exceeded one card of the S2k schema input

file. Test schema one produced no errors in the translation, but two errors were encountered when the IDMS schema compiler found two items having IDMS reserved words as item names. The DBA then changed the names in the IDMSMODL file and executed Idmsout once again. The produced IDMS schema, DMCL, and subschema were readily accepted by the IDMS database system.

Test schema two contained all of the different types of S2k records and items plus record and item descriptions that exceeded one card of the S2k schema input file. Test schema two produced no errors in the translation and the produced IDMS schema, DMCL, and subschema were readily accepted by the IDMS database system. The DBA then edited the IDMSMODL file and changed a record location mode from VIA to CALC. Idmsout was then executed producing no errors and the produced IDMS schema, DMCL, and subschema were readily accepted by the IDMS database system.

Upon completion of executing the supplied IDMS batch jobs using the files produced by the thesis as input, the DBA then logged onto the IDMS system to verify that the newly generated schema and subschema were contained in the IDMS data dictionary. A sample of the USERDEFS file, IDMSMODL file, Log file, SCHEMA file, DMCL file, and SUBS file from test schema one can be found in Appendix G.

V. Conclusions and Recommendations

This chapter will present the conclusions and recommendations of this thesis.

Conclusions

The objective of this thesis was to develop a series of application programs which would accept a S2k schema as input and return an equivalent IDMS schema, its associated DMCL, and subschema as output. The developed software was to be as interactive with the user as possible, but also leave the DBA with flexibility.

The developed programs USRPRMP, RDSCHM, and IDMSOUT were coded in the high level language PL/1 on an IBM compatible computer at the ASD Computer Center. The IBM architecture was chosen, because the versions of S2k and IDMS being used were also under the IBM architecture. These programs satisfied all of the previously mentioned requirements except for being interactive with the user.

Time Sharing Option (TSO) command programs (Appendix F) were created to execute the developed batch programs from the terminal, and thusly satisfying the interactive requirement. The TSO command programs after communicating with the user submit the appropriate application program in the batch mode.

This type of arrangement saves a tremendous amount of computer resources as compared to executing the entire process in the foreground.

Recommendations

This thesis effort opens up the areas of Data translation, Query translation, and Program translation from S2k to IDMS as future thesis efforts. The DBA has the capability of translating the schemas as a result of this thesis, but does not have any way to use the existing S2k data, queries, or programs under IDMS.

The translation of the existing S2k data will require the unloading of the S2k data file into a sequential file, the stripping of the dollar sign from the S2k money items, the stripping of the slashes from the S2k date items, and then finally the loading of the data into the IDMS system. The programmer should review chapter 2 on how the data is stored in S2k and should review the IDMSMODL file. This file will show the relationship between the S2k record component number and its IDMS counterpart. This information will be needed since the S2k component number is contained in the S2k unloaded data file.

The translation of the S2k application programs will require the familiarity of the S2k DML and the IDMS DML. The thesis student will also have to be familiar with the IDMSMODL file. The IDMSMODL file will show the relationship

between the two schemas which will be necessary for the translation.

The translation of the S2k online queries will require the user to be familiar with the S2k and IDMS online query languages. The thesis student will again have to be familiar with the IDMSMODL file for the reasons stated previously.

Bibliography

1. Tsichritzis, D.C. and F. H. Lochovsky. "Hierarchical Data-Base Management: A Survey," ACM Computing Surveys, 8 (1): 105-123 (March 1976).

2. Date, C.J. An Introduction to Database Systems (Third Edition). Reading, Massachusetts: Addison-Wesley Publishing Company, 1981.

3. Kroenke, David. Database Processing: Fundamentals, Modeling, Applications. Chicago: Science Research Associates, Inc., 1976.

4. LSM-DEF-10,4210/R10/PD11M. SYSTEM 2000: Language Specification Manual for the DEFINE Language for Release 10. Austin, Texas: Intel Systems Corporation, 1980.

5. S2K-EU-ALL,#221144. SYSTEM 2000: Professional User's Guide. Austin, Texas: Intel Corporation, 1983.

6. LSM-COBOL-10,4220/R10/PD11M. SYSTEM 2000: Language Specification Manual for The COBOL Programming Language Extension (PLEX) for Release 10. Austin, Texas: Intel Systems Corporation, 1980.

7. TDDB-0180-5501. IDMS: System Overview, Revision 0.1. USA: Cullinane Database Systems Inc, 1982.

8. Kennedy, Michael and Martin B. Solomon. Eight Statement PL/C (PL/ZERO). Englewood Cliffs, N. J.: Prentice-Hall, Inc., 1972

9. TDDB-0300--5700. IDMS: Database Design and Definition Guide, Revision 0.0. USA: Cullinane Database Systems Inc, 1982.

Appendix A

Automated Hierarchical - CODASYL Database Interface

Schema Translator User's Guide

Introduction

The schema translation software as described in this thesis is intended to run on an IBM compatible mainframe under the control of MVS 3.8 . The software for the actual schema translation is written in PL/1 and the command files used to execute the software are written in TSO command language.

This system is designed to be used primarily with a IBM 3270 CRT as the terminal device, but will support other terminals with a minor modification to the TSO user's profile.

The function of the Schema Translator software is to provide the DBA with an IDMS schema, DMCL, and subschema for an already existing S2k schema. Three programs USRPRMP, RDSCHM, and IDMSOUT make up the Schema Translator software.

Before using this software, the user should be familiar with S2k, IDMS, TSO editor, TSO CLISTS, and IBM partitioned data sets.

User Preparation

Before executing the Schema Translator software, the user must do the following:

1. Unload the desired S2k schema into a file using the S2k supplied UNLOAD utility.

2. Userid.IDMS.Data should be allocated. For example, if your TSO id is AAA, then you should allocate a file with the name of AAA.IDMS.DATA. One track should be enough.

3. If you are NOT using a 3270 type terminal, then modify your TSO terminal configuration so that 80 characters make up one line.

4. Allocate a PDS to be used by the translator. One cylinder should be enough.

Using the Translator

The translator is executed via a TSO clist (MENU). Menu will prompt you for every piece of information necessary to create the IDMS schema, DMCL and subschema. The translator software is currently stored in a file called JGO.DEMO.CNTL and the clist software is currently stored in JGO.CLIST. These files will therefore be used in the examples throughout this user's guide. To execute the translator type in

EXECUTE JGO.CLIST(MENU)

the system should respond with the following menu on the screen

```
************ S2k to IDMS  translator Menu ***********
*                                                     *
*    1) RUN USRPRMP                 3) RUN IDMSOUT    *
*    2) RUN RDSCHM                 >3) EXIT MENU      *
*                                                     *
*******************************************************
```

you now have the option of entering

1 - to execute the USRPRMP program.

2 - to execute the RDSCHM program.

3 - to execute the IDSMOUT program.

or any number greater than 3 to return back to TSO.

Executing Usrprmp

Usrprmp will prompt you for information necessary to create the IDMS schema, DMCL, and subschema which can not be obtained from the S2k schema. Initially, USRPRMP will prompt you for a file (USERDEFS) in which to store the information received by USRPRMP. A member of the previously allocated PDS is the recommended choice. A sample terminal session follows with the USRPRMP system prompts in capital letters.

ENTER SCHEMA NAME (1 - 8 CHARACTERS)
reply idms schema name
(required entry)

ENTER SCHEMA VERSION NUMBER (1 - 3 DIGITS)
reply idms version number of this schema
(required entry)

ENTER AUTHOR NAME (1 30 CHARACTERS)
reply authors name
(optional entry)

ENTER AUTHOR PHONE NUMBER (AREA CODE - NUMBER)
reply author's phone number
(optional entry)

ENTER DATABASE FILE NAME (1 - 8 CHARACTERS)
reply name of your IDMS database file
(required entry)

ENTER DEVICE TYPE (3330/3350)
reply type of storage device that contains database
(required entry)

ENTER AREA NAME (1 - 16 CHARACTERS)
reply the desired IDMS area name for this schema
(required entry)

ENTER AREA STARTING RANGE # (1-8 DIGITS)
reply starting page number
(required entry)

ENTER AREA ENDING RANGE # (1-8 DIGITS)
reply ending page number
(required entry)

ENTER DMCL NAME (1 - 8 CHARACTERS)
reply name of the dmcl for this schema
(required entry)

ENTER SUBSCHEMA NAME (1 - 8 CHARACTERS)
reply name of the subschema for this schema
(required entry)

USRPRMP will then return control back to MENU. If the user makes a mistake while entering data in USRPRMP, corrections can be made easily by exiting the menu, editing the USERDEFS file, and returning to the MENU. USRPRMP does not have to be reexecuted.

Executing Rdschm

Rdschm will prompt you for the following information:

1. S2KSCHM - Name of file containing the S2k schema to be translated.

2. S2KTEMP - Name of file to be used for temporary storage.

3. IDMSMODL - Name of file to store the final output of RDSCHM.

4. STARTNUM - Value to be used for the first IDMS record id number.

5. INCNUM - Value to be used as the increment for each successive IDMS record id number.

6. MSGCLASS - which output device to use
   A - printer
   X - terminal

After answering the questions, a batch job submit message will appear on the screen. For example, with a TSO user id of AAA, the message will be AAATRNS(JOBXXXX) SUBMITTED, with XXXX representing the job number assigned. Upon completion of the job, the user will receive a listing of the RDSCHM log file and the IDMSMODL file. The user is encouraged to review these files before executing IDMSOUT.

Changing the IDMSMODL File

The IDMSMODL file is a model of what the created IDMS schema will look like. Changes to this file can be made very easily by using the TSO editor. The first record in the IDMSMODL contains the number of records and items in the file. The remaining IDMSMODL file is comprised of four types of 80 character records TYPE1, TYPE2, TYPE3,and TYPE4. TYPE1 represents the record description, TYPE2 represents record location mode of CALC, TYPE3 represents location mode of VIA, and TYPE4 represents the items of each of the records. The format of these records are:

TYPE1 -	col 1	'R'	length 1
	col 3	IDMS record name	length 16
	col 20	IDMS record number	length 4
	col 25	S2k record number	length 4
	col 30	S2k owner record number	length 4
	col 35	record item pointer	length 4
TYPE2 -	col 1	'C'	length 1
	col 5	Calc item id	length 32
TYPE3 -	col 1	'V'	length 1
	col 5	set name	length 16
	col 25	owner name	length 16
	col 45	member name	length 16
TYPE4 -	col 1	'I'	length 1
	col 3	S2k component number	length 4
	col 8	IDMS item name	length 32
	col 41	IDMS item description	length 28
	col 70	S2k record number	length 4

The user should note that TYPE2 and TYPE3 records will not appear in the same IDMS record description. Changes can be made to the IDMSMODL file and will be reflected in the produced IDMS schema, DMCL, and subschema provided that the previously mentioned record format guidelines are followed. For example, if the user wishes to change a record's location mode from VIA to CALC, the user simply changes the record's TYPE3 record to a TYPE2 record.

Executing Idmsout

IDMSOUT will prompt the user for the following information:

1. USERDEFS - Name of the file used by USRPRMP.

2. SCHEMA - Name of the file which will contain the IDMS schema.

3. IDMSMODL - Name of the file containing the IDMS model.

4. DMCL - Name of the file which will contain the DMCL.

5. SUBSCHEM - Name of the file which will contain the subschema.

6. MSGCLASS - which output device to use
   A - printer
   X - terminal.

After answering the questions, a batch job submit message will appear on the screen. For example, with a TSO user id of AAA, the message will be AAAIDMS(JOBXXXX) SUBMITTED, with

XXXX representing the job number assigned. Upon completion of the job, the user will receive a listing of the IDMS schema, DMCL, and subschema files.

Further Information

More information can be found on the schema translator in the following locations in this thesis:

1. Appendix B - Data Flow Diagrams.
2. Appendix C - Structure Charts.
3. Appendix D - Data Dictionary.
4. Appendix E - PL/1 source code.
5. Appendix F - TSO CLIST source code.
6. Appendix G - Sample output from the translator.

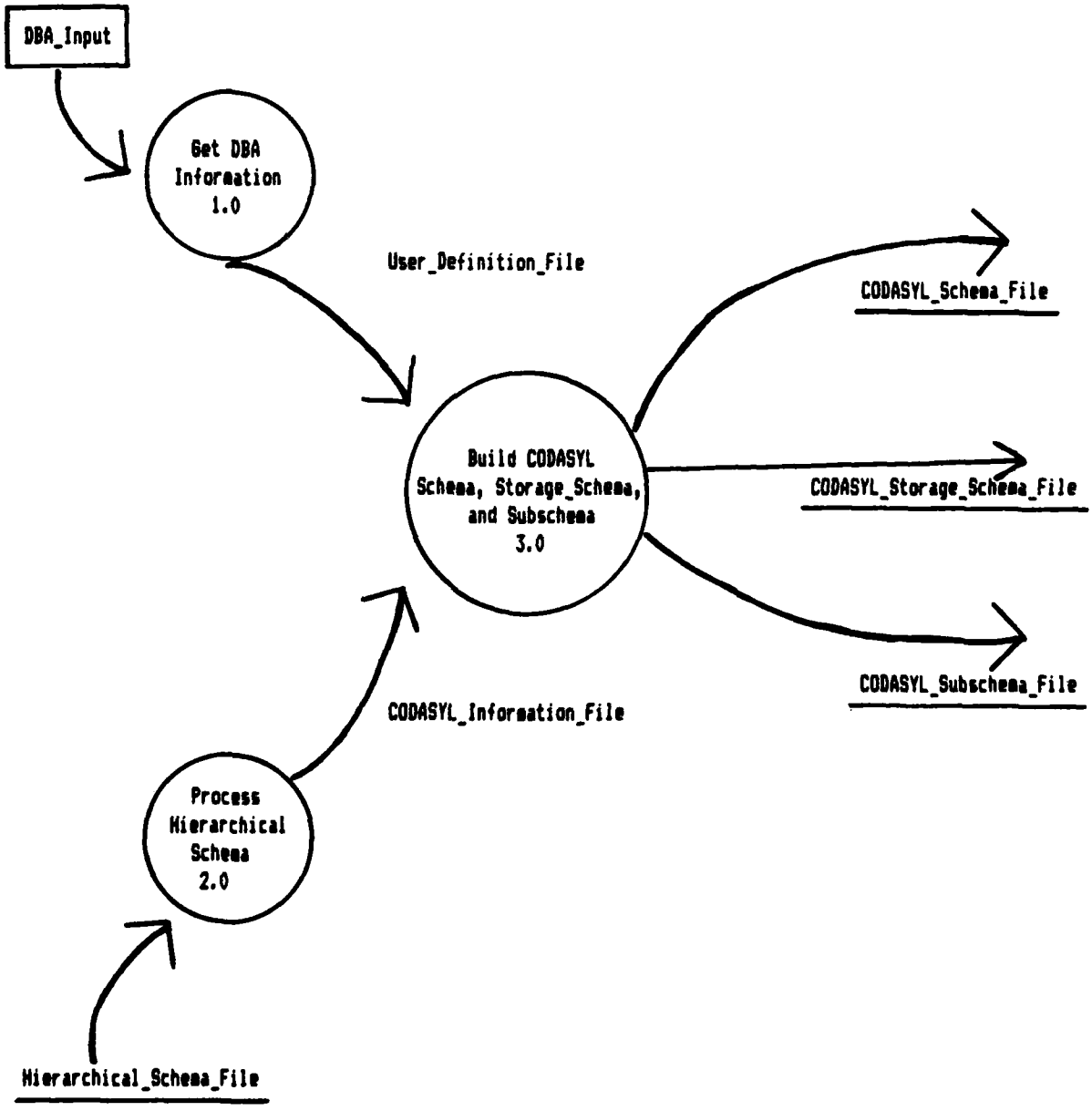
Appendix B

Automated Hierarchical - CODASYL Database Interface

Schema Translator Data Flow Diagrams

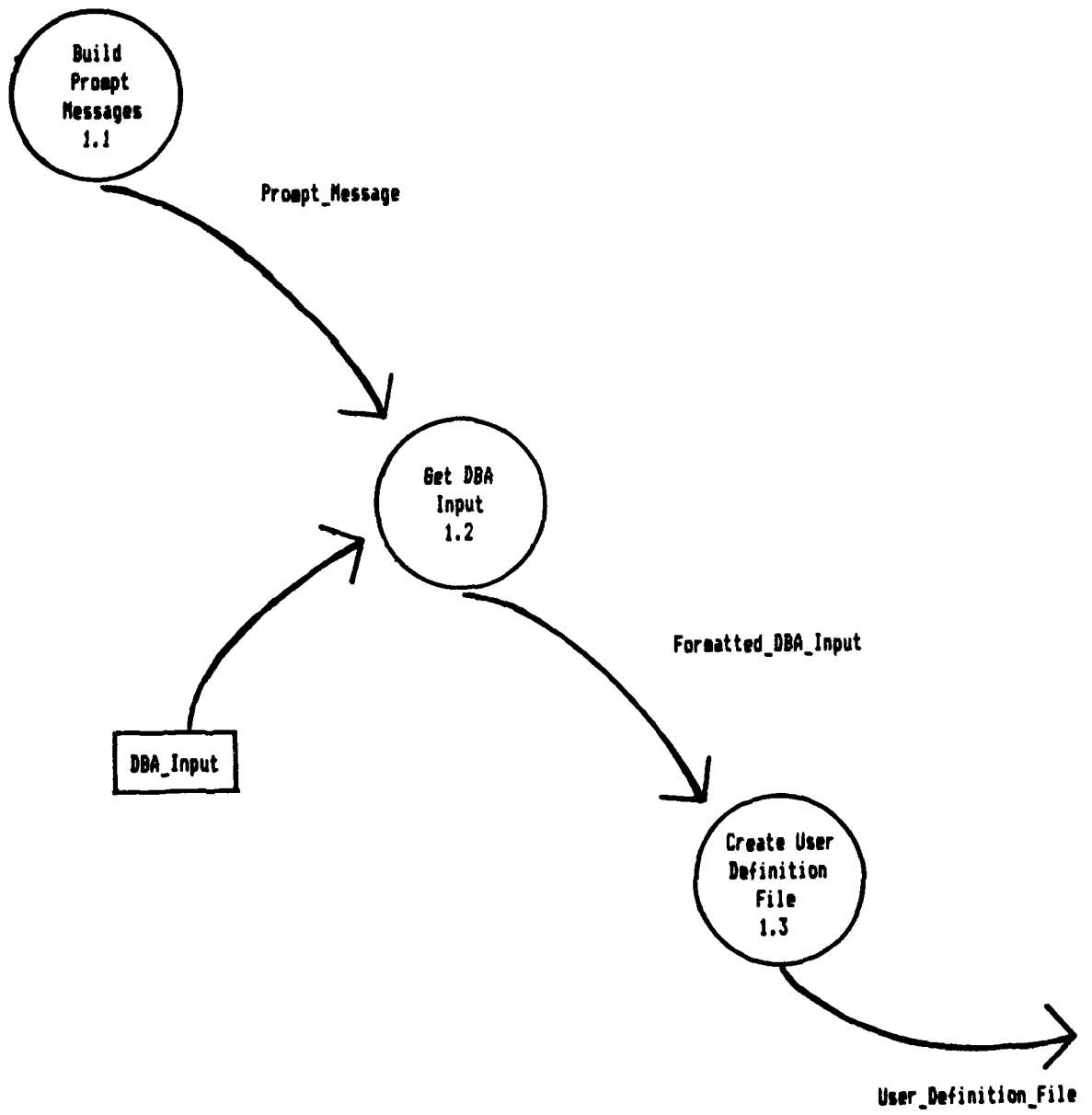
Process Number: 0.0 Author: Capt Jerry Owens

Process Name: Hierarchical to CODASYL



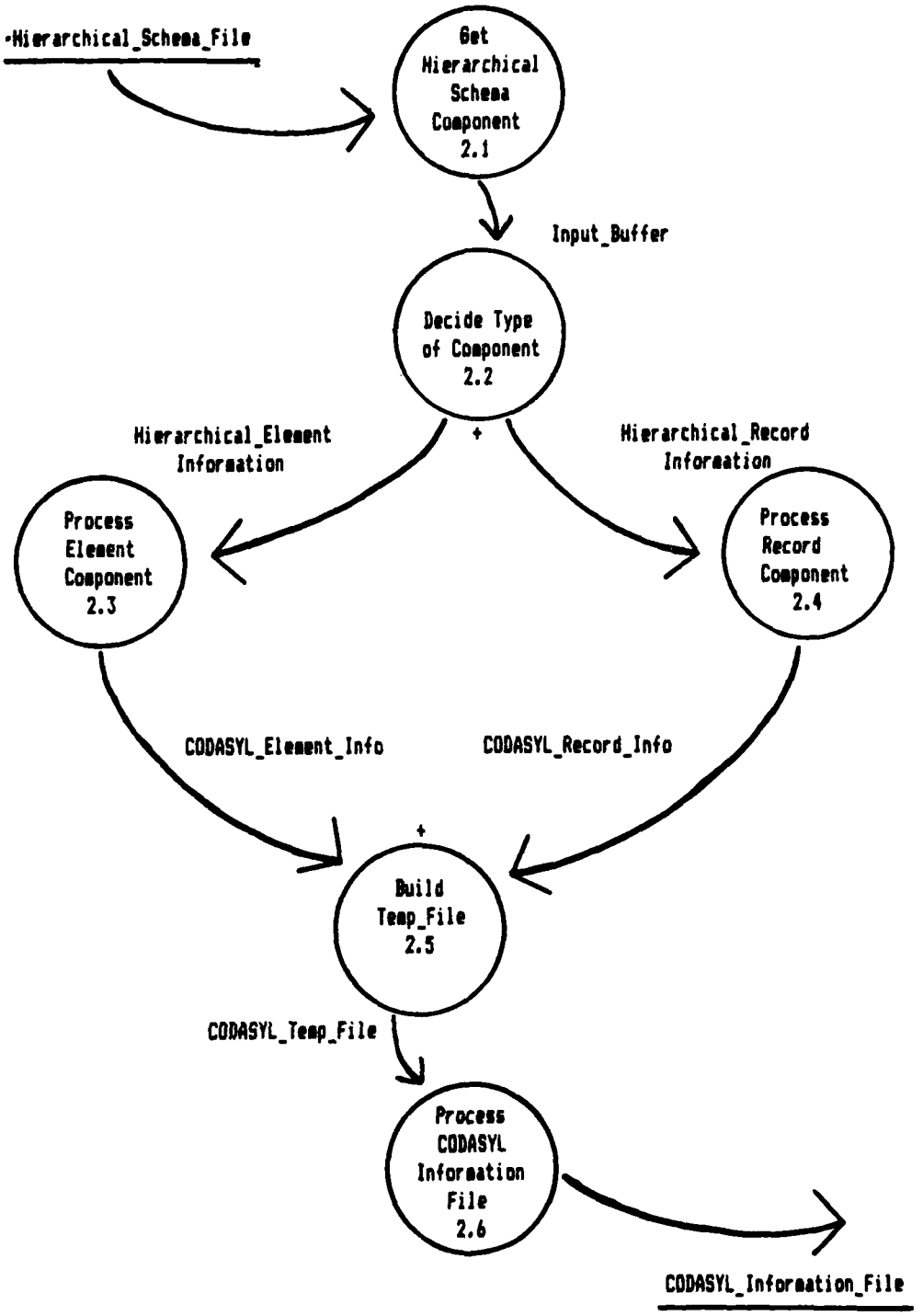
Process Number: 1.0 Author: Capt Jerry Owens

Process Name: Get DBA Information



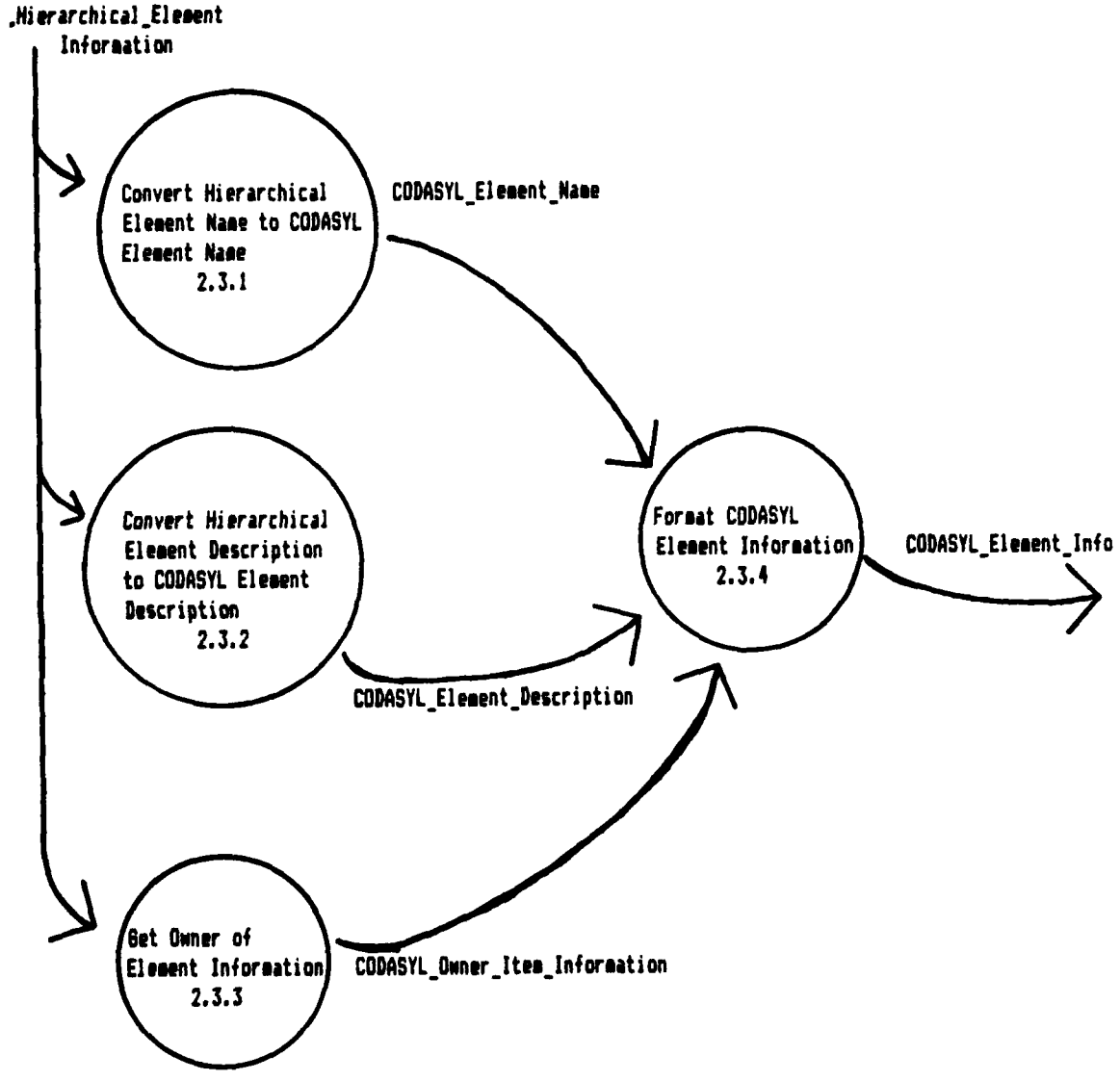
Process Number: 2.0 Author: Capt Jerry Owens

Process Name: Process Hierarchical Schema



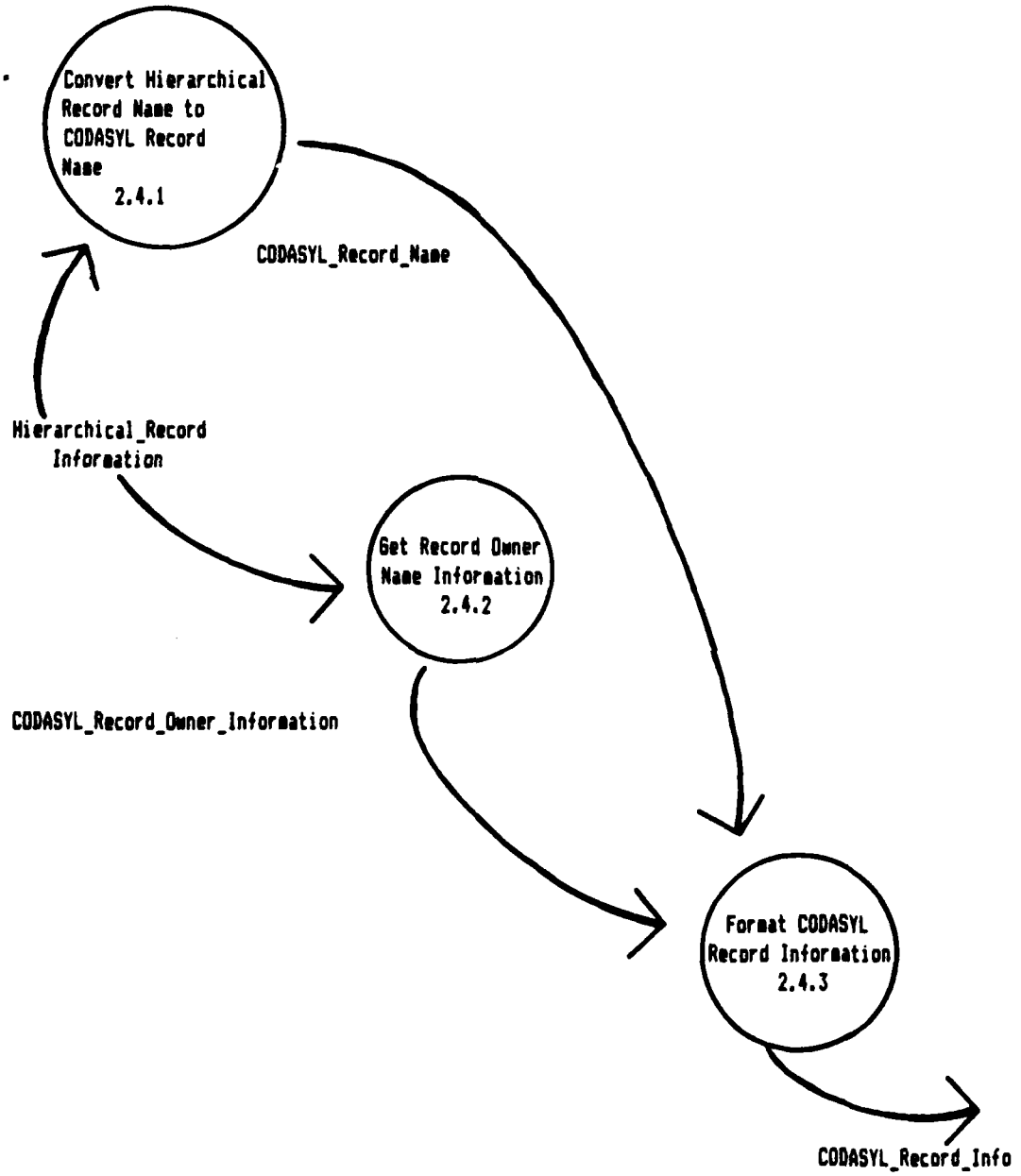
Process Number: 2.3 **Author: Capt Jerry Owens**

Process Name: Process Element Component



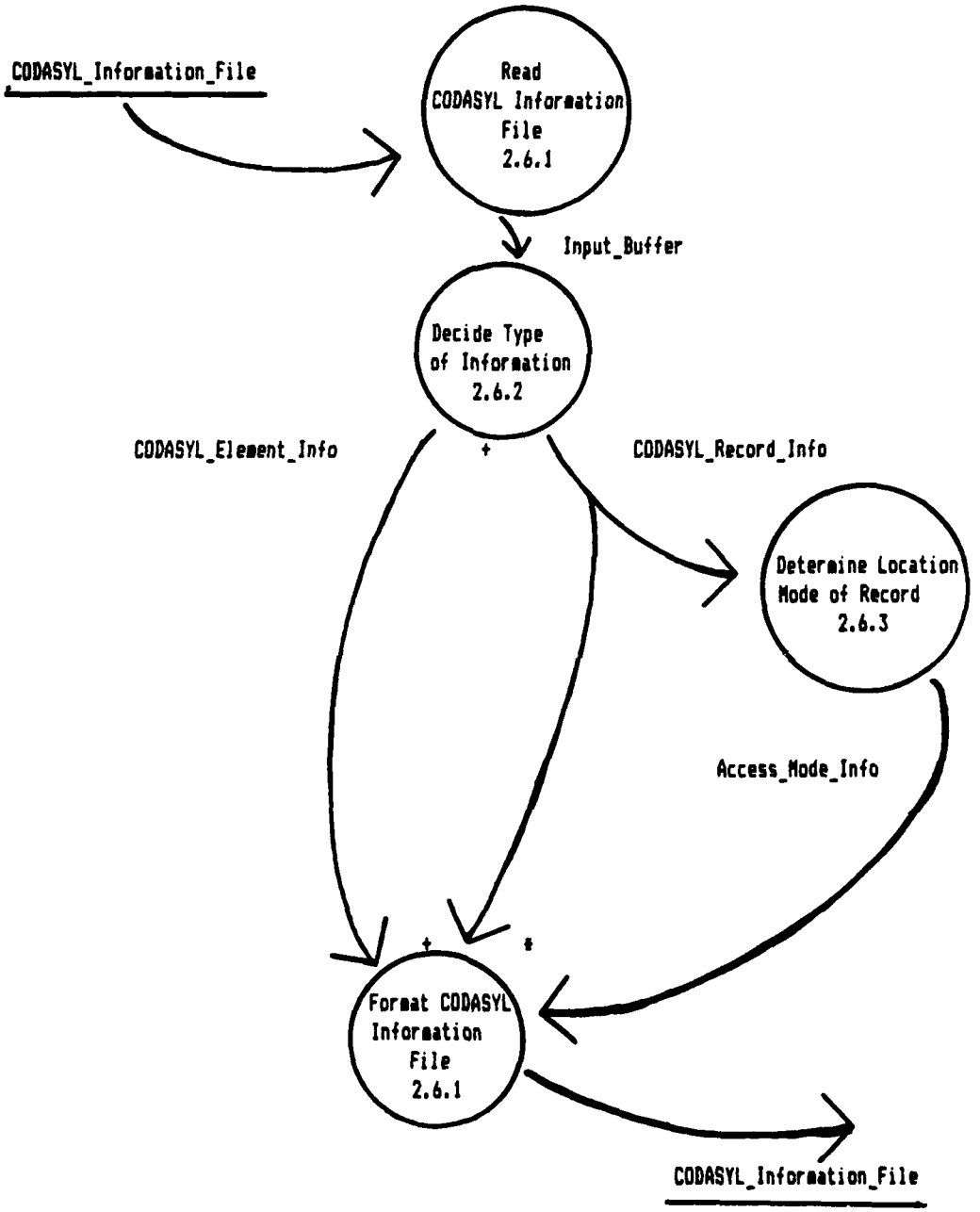
Process Number: 2.4 **Author: Capt Jerry Owens**

Process Name: Process Record Component



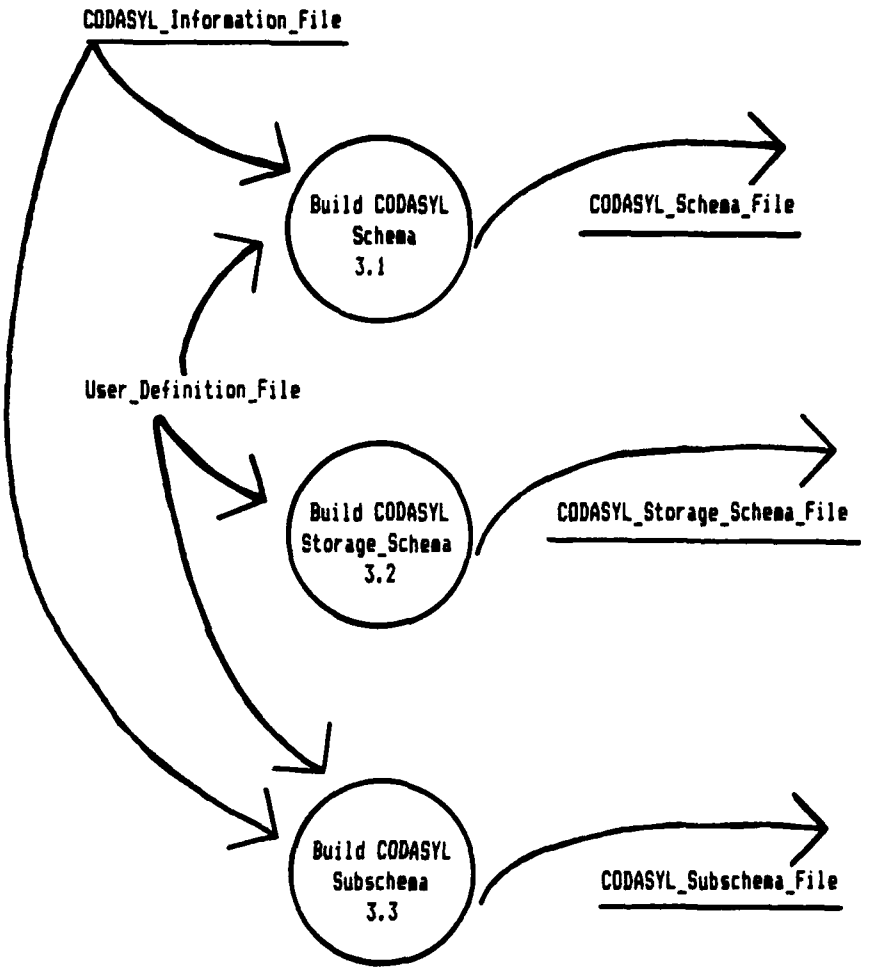
Process Number: 2.6 **Author: Capt Jerry Owens**

Process Name: Process CODASYL Information File



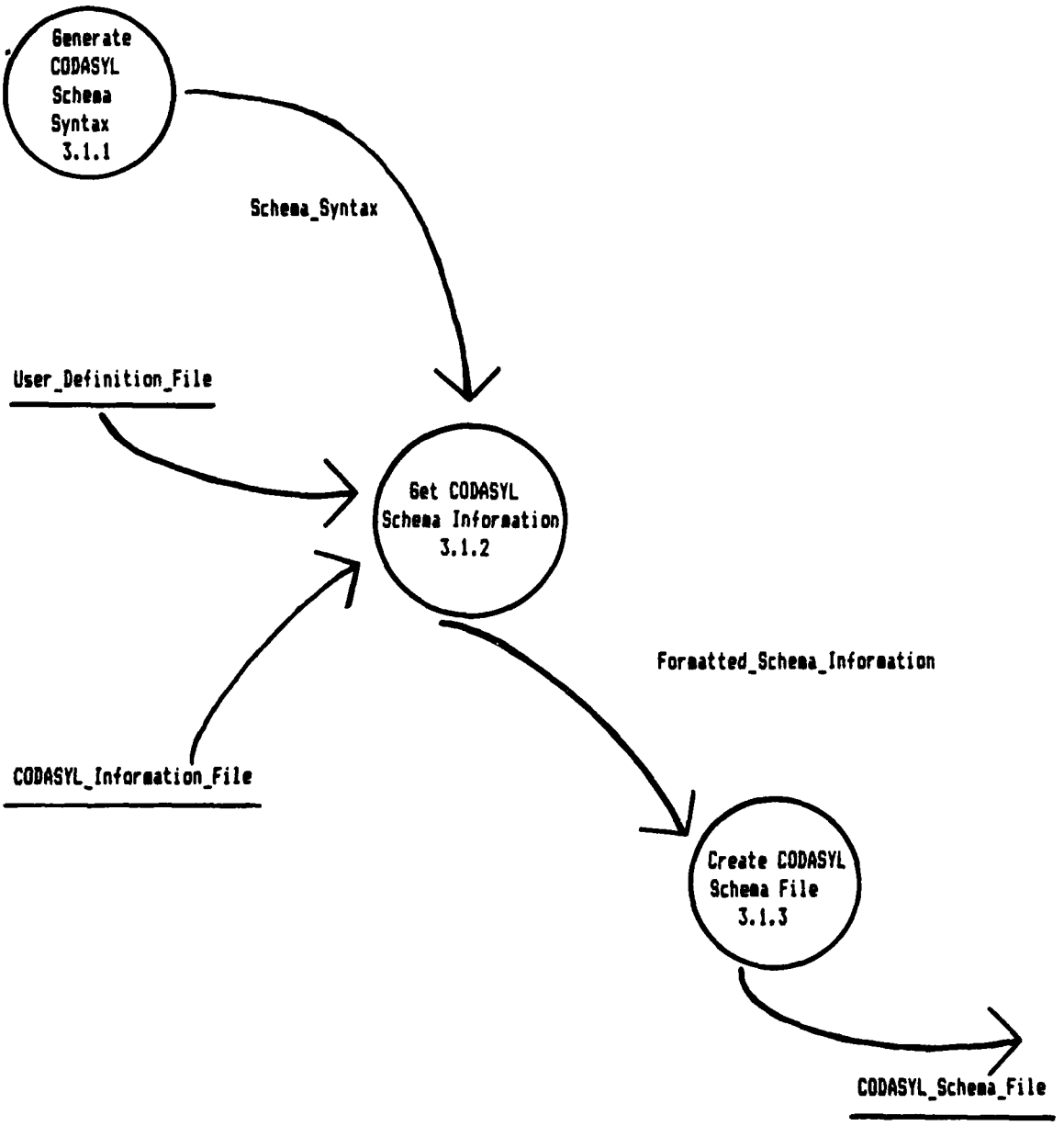
Process Number: 3.0 Author: Capt Jerry Owens

Process Name: Build CODASYL Schema, Storage_Schema, and Subschema



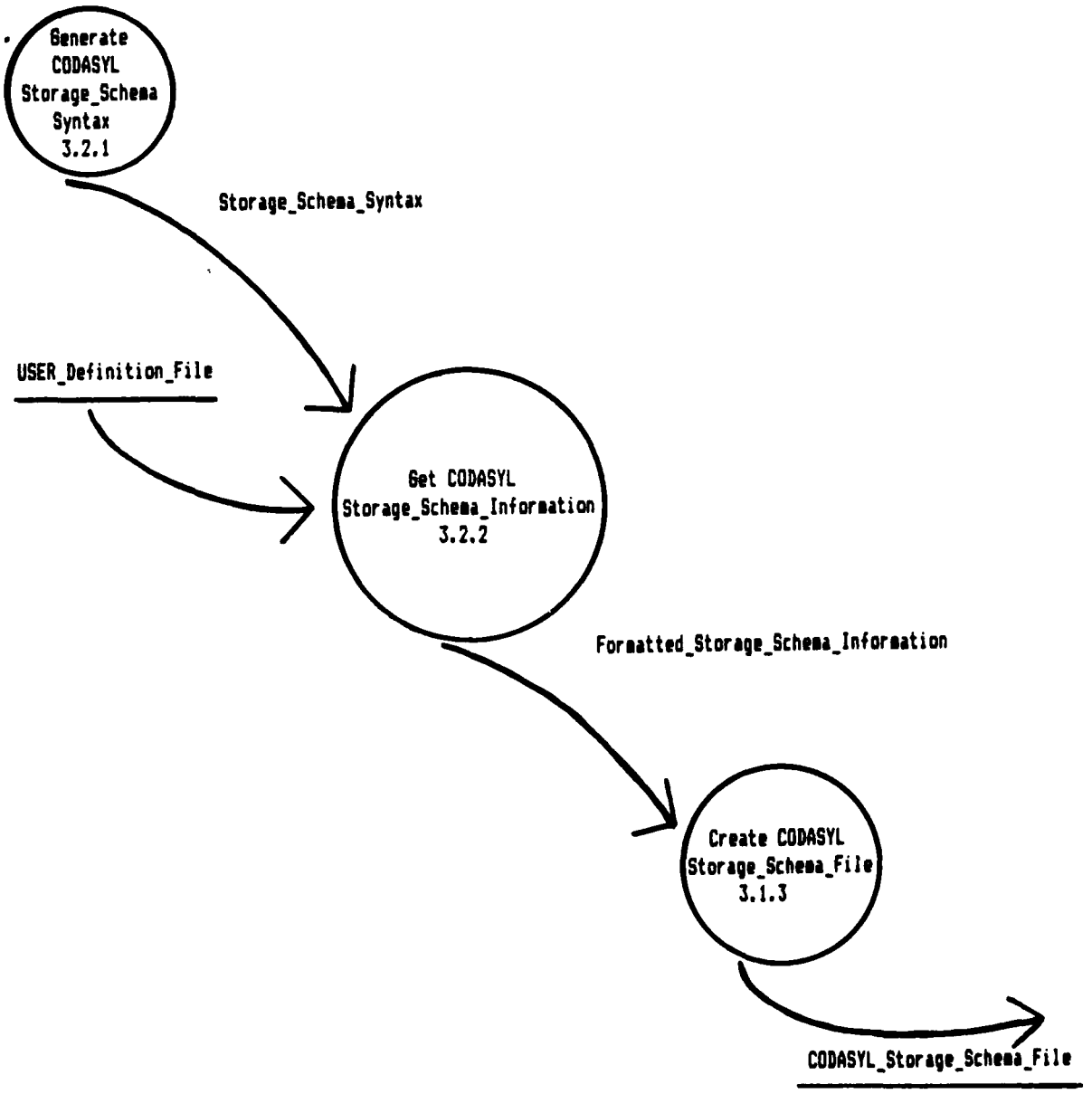
Process Number: 3.1 Author: Capt Jerry Owens

Process Name: Build CODASYL Schema



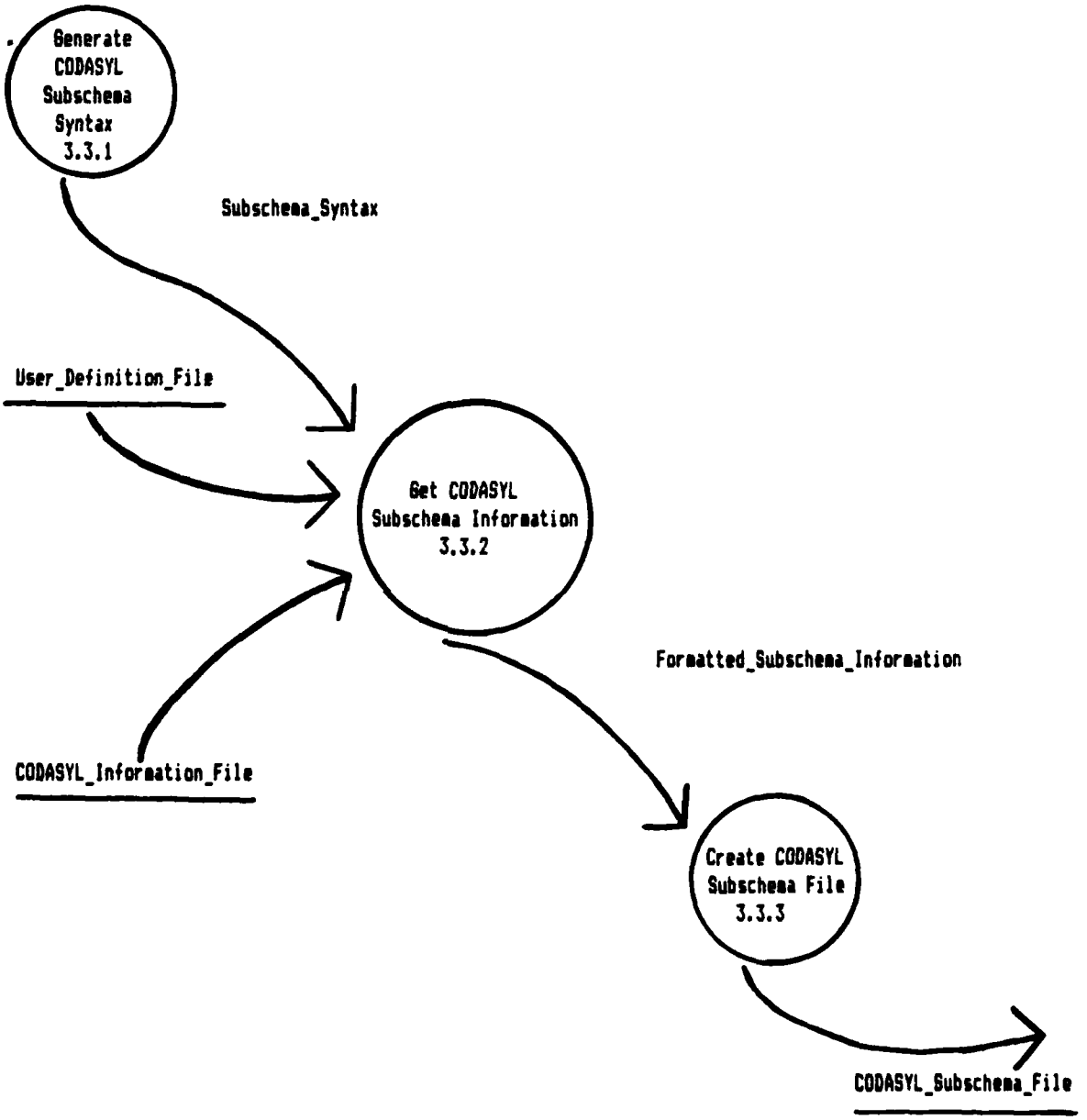
Process Number: 3.2 Author: Capt Jerry Owens

Process Name: Build CODASYL Storage_Schema



Process Number: 3.3 Author: Capt Jerry Owens

Process Name: Build CODASYL Subschema



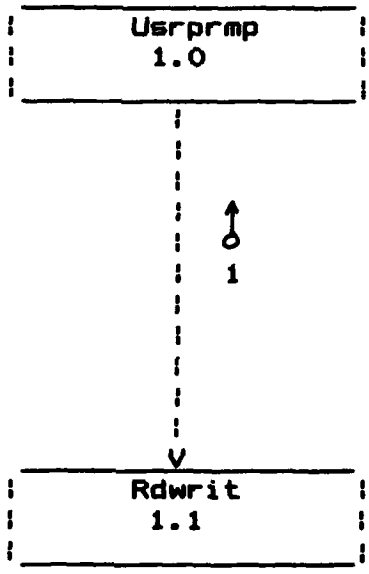
Appendix C

Automatic Hierarchical - CODASYL Database Interface

Schema Translator Structure Charts

Module Name: Usrprmp Module Number: 1.0

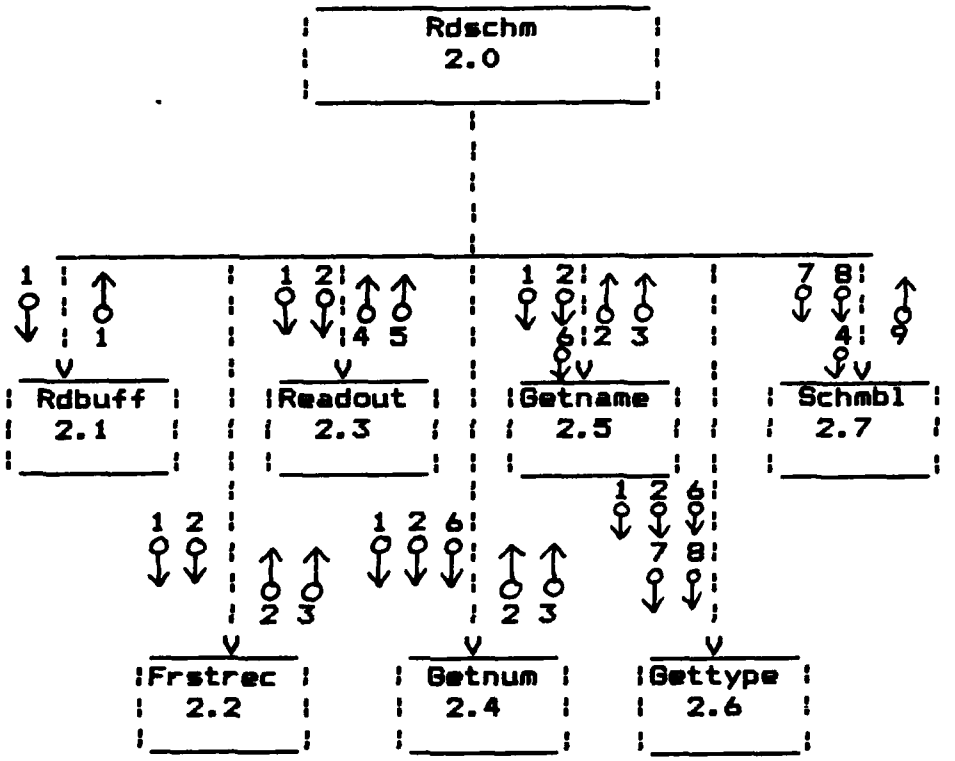
Program Name: Usrprmp Author: Capt Jerry Owens



1. User_Definition_File

Module Name: Rdschm Module Number: 2.0

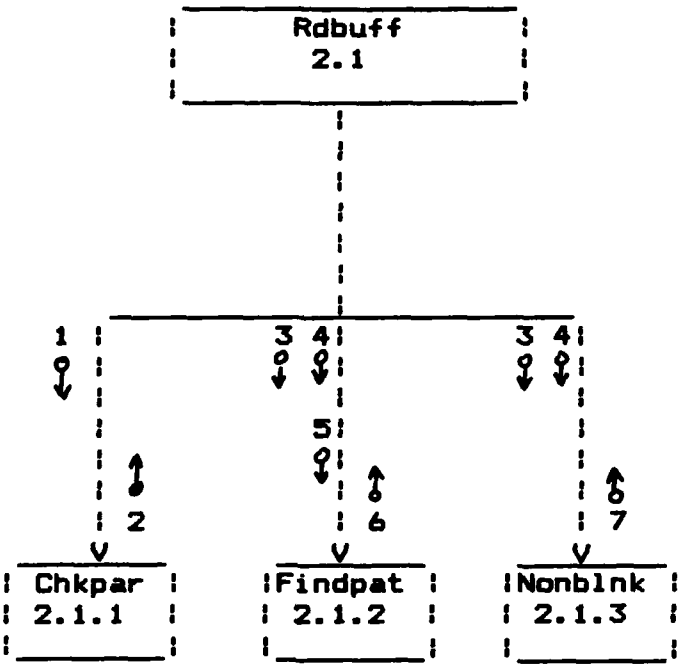
Program Name: Rdschm Author: Capt Jerry Owens



1. Recin_Buffer
2. Recout_Buffer
3. Found_Position
4. S2k_Temp_File
5. S2k_Log_File
6. Recin_Buffer_Start_Pos
7. Number_of_Records
8. Number_of_Elements
9. IDMS_Model_File

Module Name: Rdbuff Module Number: 2.1

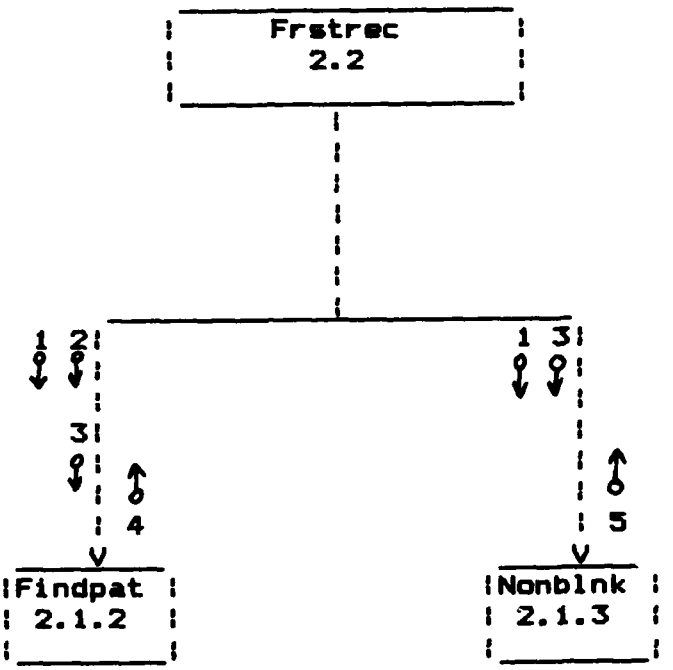
Program Name: Rdschm Author: Capt Jerry Owens



1. Buffer_80
2. Matching_Paren_Found
3. Recin_Buffer
4. Recin_Buffer_Start_Pos
5. Pattern_to_Be_Found
6. Starting_Pos_of_Pattern
7. Found_Position

Module Name: Frstrec Module Number: 2.2

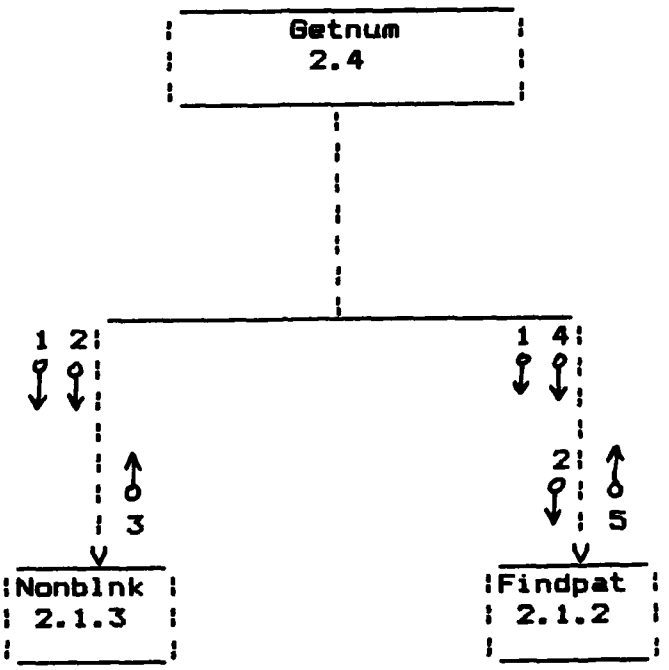
Program Name: Rdschm Author: Capt Jerry Owens



1. Recin_Buffer
2. Pattern_to_Be_Found
3. Recin_Buffer_Start_Pos
4. Starting_Pos_of_Pattern
5. Found_Position

Module Name: Getnum Module Number: 2.4

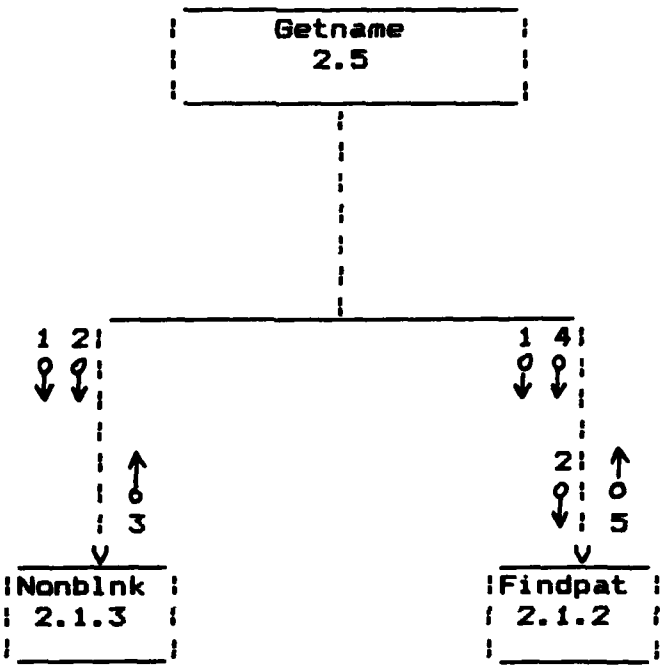
Program Name: Rdschm Author: Capt Jerry Owens



1. Recin_Buffer
2. Recin_Buffer_Start_Pos
3. Found_Pos
4. Pattern_to_Be_Found
5. Starting_Pos_of_Pattern

Module Name: Getname Module Number: 2.5

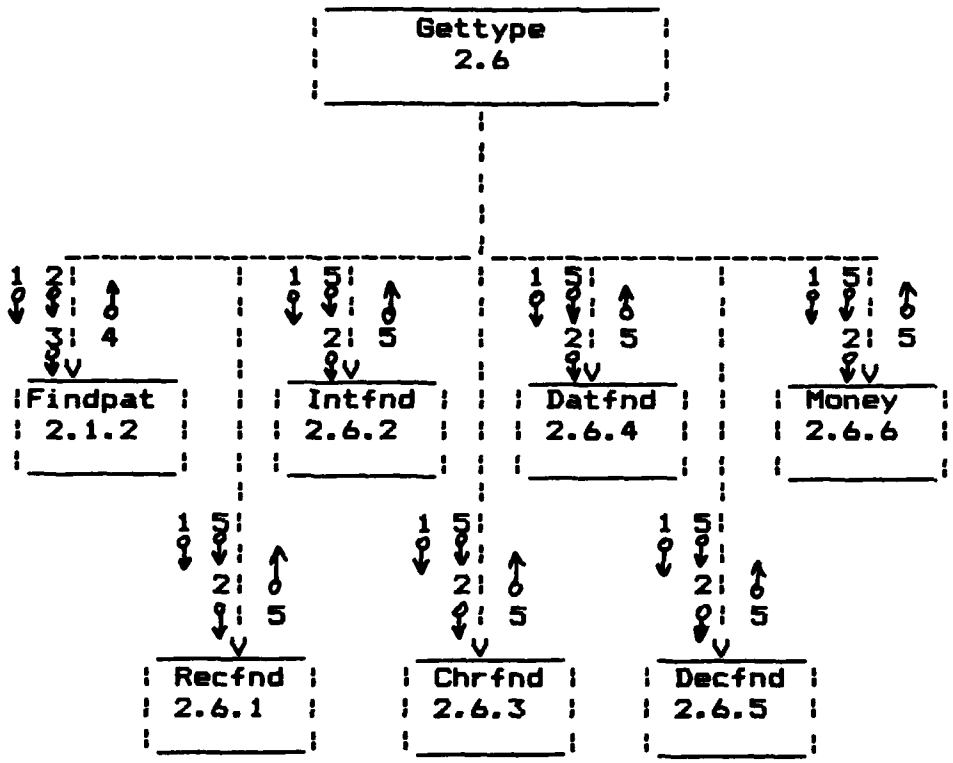
Program Name: Rdschm Author: Capt Jerry Owens



1. Recin_Buffer
2. Recin_Buffer_Start_Pos
3. Found_Pos
4. Pattern_to_Be_Found
5. Starting_Pos_of_Pattern

Module Name: Gettype Module Number: 2.6

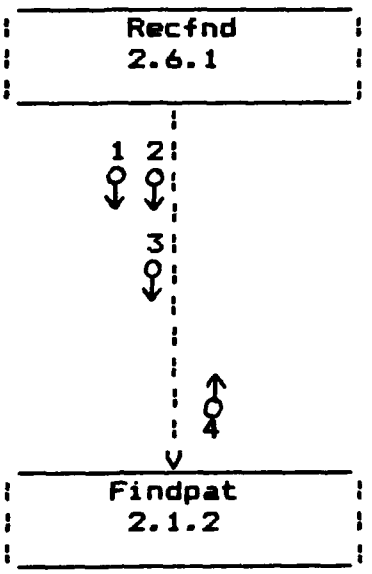
Program Name: Rdschm Author: Capt Jerry Owens



1. Recin_Buffer
2. Recin_Buffer_Start_Pos
3. Pattern_to_Be_Found
4. Starting_Pos_of_Pattern
5. Recout_Buffer

Module Name: Recfnd Module Number: 2.6.1

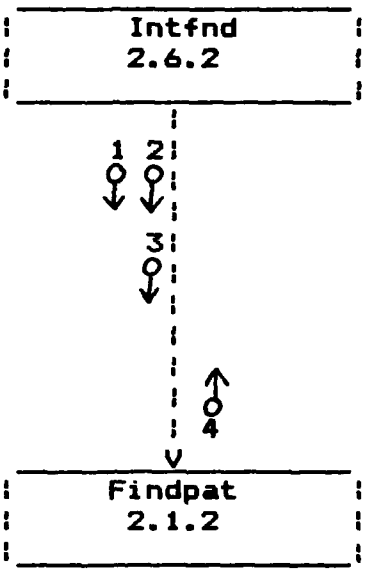
Program Name: Rdschm Author: Capt Jerry Owens



1. Recin_Buffer
2. Recin_Buffer_Start_Pos
3. Pattern_to_Be_Found
4. Starting_Pos_of_Pattern

Module Name: Intfnd Module Number: 2.6.2

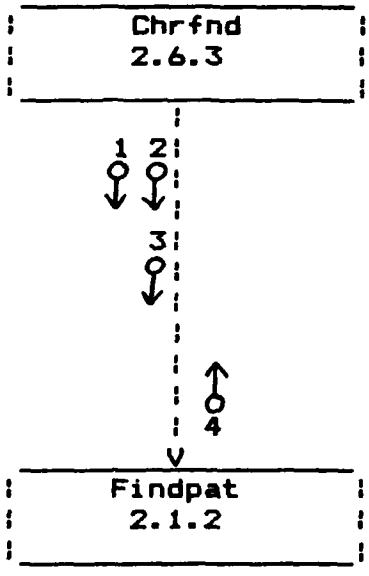
Program Name: Rdschm Author: Capt Jerry Owens



1. Recin_Buffer
2. Recin_Buffer_Start_Pos
3. Pattern_to_Be_Found
4. Starting_Pos_of_Pattern

Module Name: Chrfnd Module Number: 2.6.3

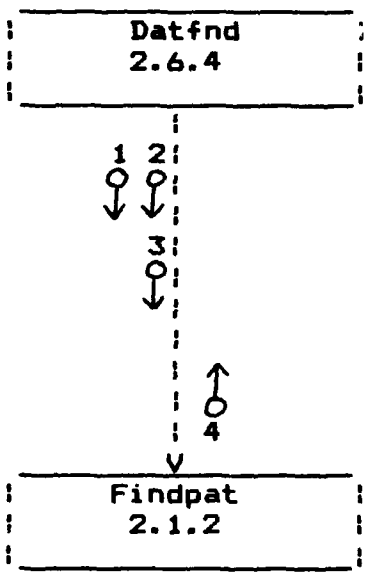
Program Name: Rdschm Author: Capt Jerry Owens



1. Recin_Buffer
2. Recin_Buffer_Start_Pos
3. Pattern_to_Be_Found
4. Starting_Pos_of_Pattern

Module Name: Datfnd
Program Name: Rdschm

Module Number: 2.6.4
Author: Capt Jerry Owens



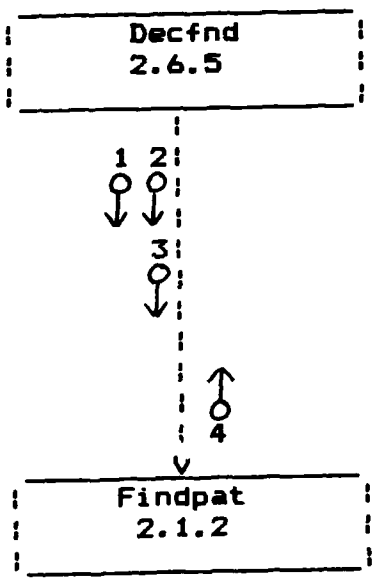
1. Recin_Buffer
2. Recin_Buffer_Start_Pos
3. Pattern_to_Be_Found
4. Starting_Pos_of_Pattern

Module Name: Decfnd

Module Number: 2.6.5

Program Name: Rdschm

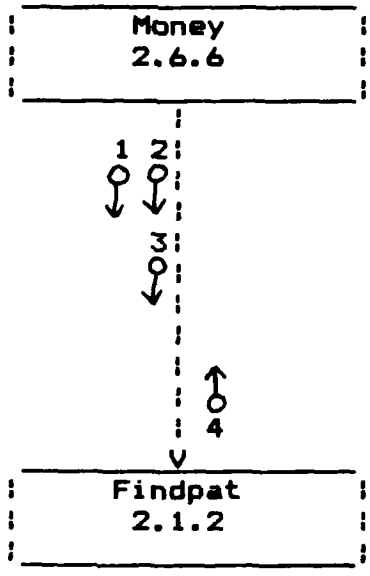
Author: Capt Jerry Owens



1. Recin_Buffer
2. Recin_Buffer_Start_Pos
3. Pattern_to_Be_Found
4. Starting_Pos_of_Pattern

Module Name: Money Module Number: 2.6.6

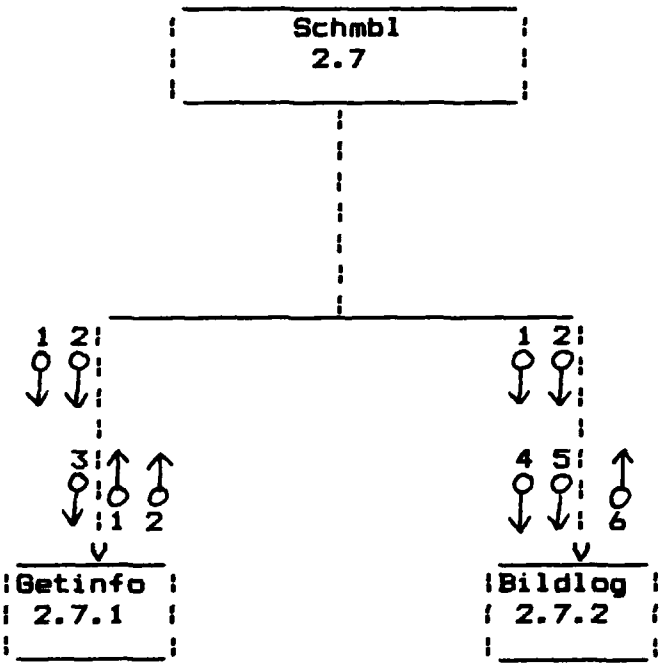
Program Name: Rdschm Author: Capt Jerry Owens



1. Recin_Buffer
2. Recin_Buffer_Start_Pos
3. Pattern_to_Be_Found
4. Starting_Pos_of_Pattern

Module Name: Schmbl Module Number: 2.7

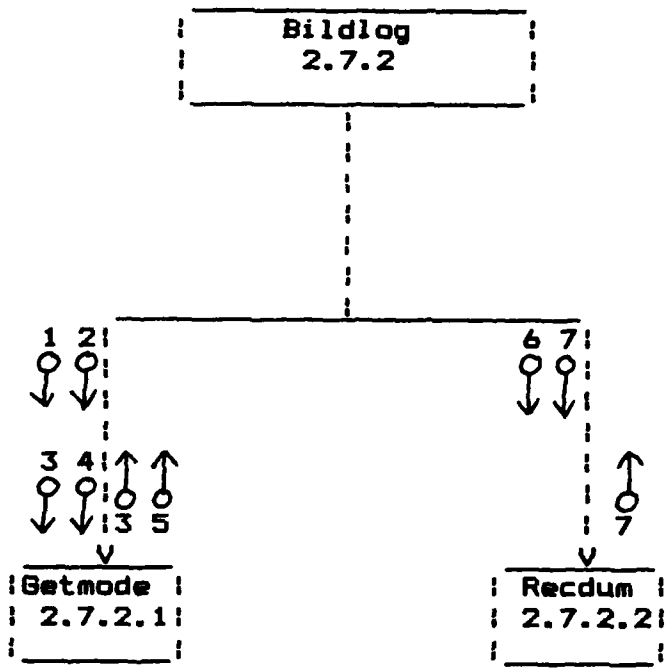
Program Name: Rdschm Author: Capt Jerry Owens



1. Record_Desc_Info
2. Element_Desc_Info
3. Total_Records_and_Elements
4. Number_of_Records
5. Number_of_Elements
6. IDMS_Model_File

Module Name: Bildlog Module Number: 2.7.2

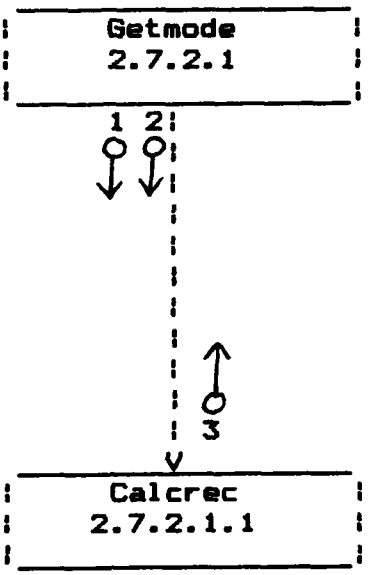
Program Name: Rdschm Author: Capt Jerry Owens



1. Record_Desc_Info
2. Element_Desc_Info
3. Set_Count
4. Number_of_Records
5. IDMS_Model_File
6. Record_Name
7. Dummy_Record_Name

Module Name: Getmode Module Number: 2.7.2.1

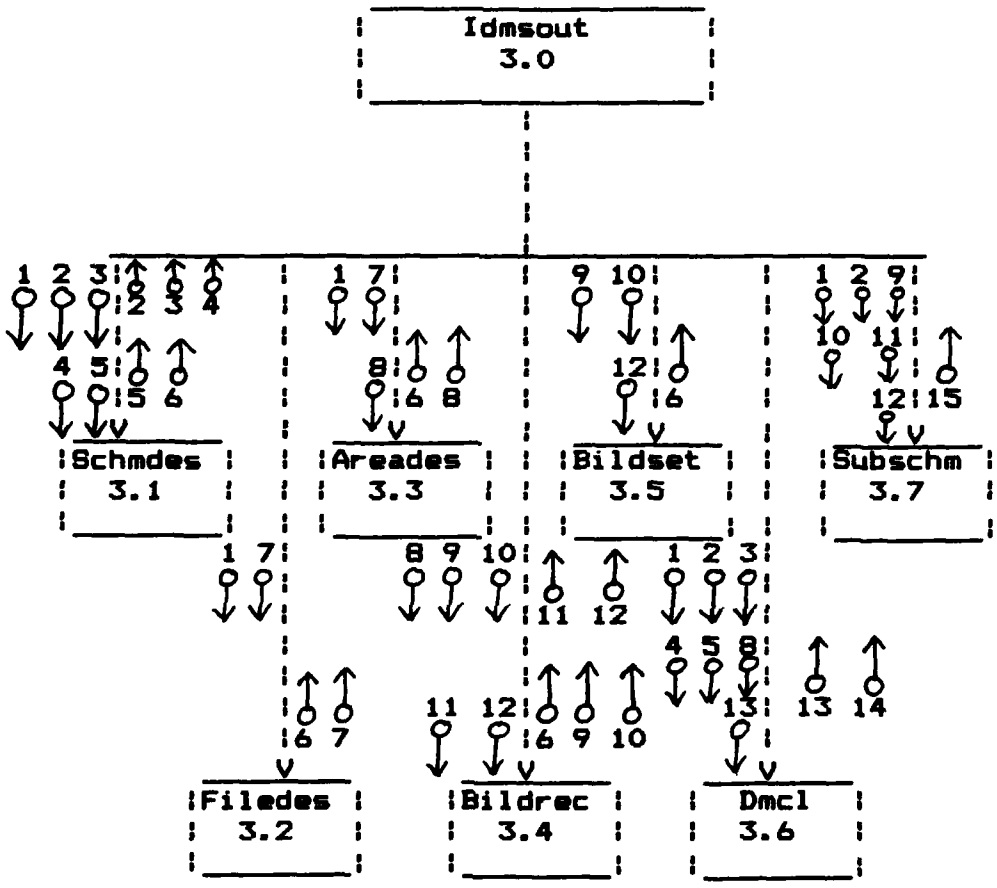
Program Name: Rdschm Author: Capt Jerry Owens



1. Record_Desc_Info
2. Record_Desc_Info_Pos
3. IDMS_Model_File

Module Name: Idmsout Module Number: 3.0

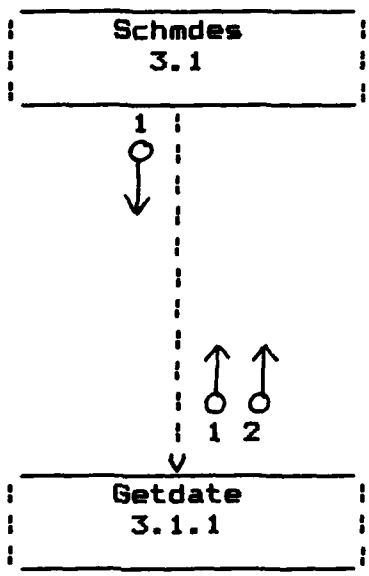
Program Name: Idmsout Author: Capt Jerry Owens



1. User_Definition_File
2. IDMS_Schema_Name
3. IDMS_Schema_Version
4. Author_of_Schema
5. Current_System_Date
6. CODASYL_Schema_File
7. IDMS_File_Name
8. IDMS_Area_Name
8. Record_Next_Prior_Info
10. Set_Owner_Member_Info
11. Record_Next_Prior_Pos
12. Set_Owner_Member_Pos
13. IDMS_DMCL_Name
14. IDMS_DMCL_File
15. CODASYL_Subschema_File

Module Name: Schmdes Module Number: 3.1

Program Name: Idmsout Author: Capt Jerry Owens



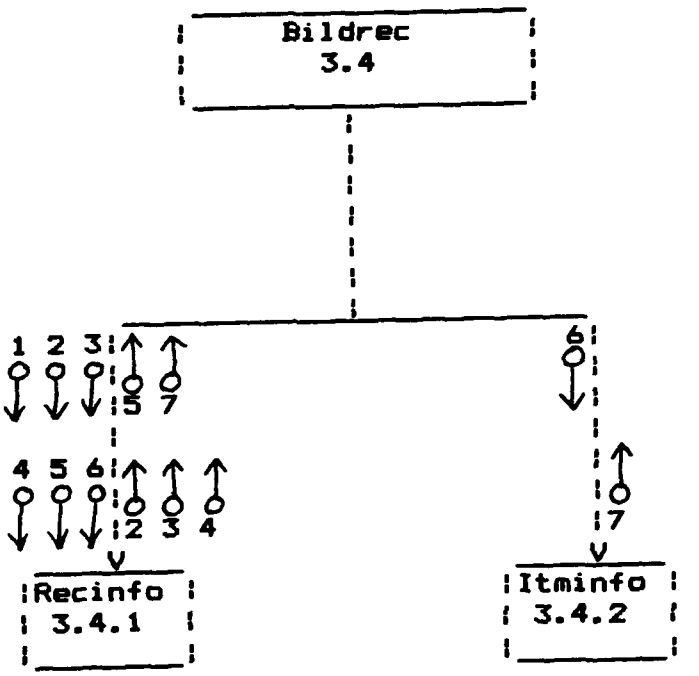
1. Current_System_Date 2. CODASYL_Schema_File

Module Name: Bildrec

Module Number: 3.4

Program Name: Idmsout

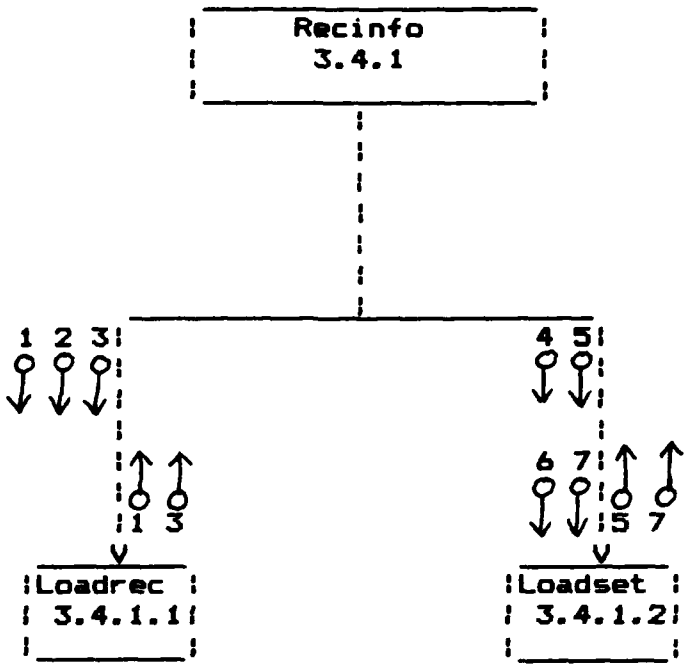
Author: Capt Jerry Owens



1. IDMS_Area_Name
2. Record_Next_Prior_Info
3. Set_Owner_Member_Info
4. Set_Owner_Member_Pos
5. Record_Next_Prior_Pos
6. IDMS_Model_File
7. CODASYL_Schema_File

Module Name: Recinfo Module Number: 3.4.1

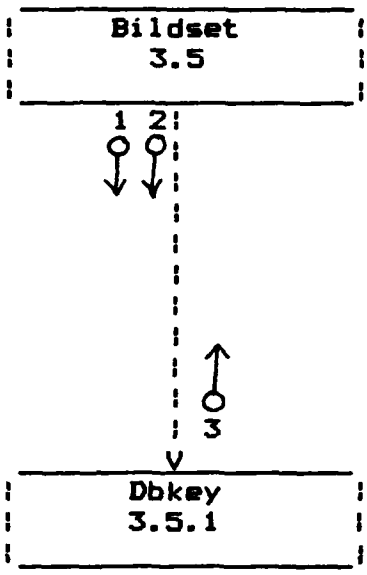
Program Name: Idmsout Author: Capt Jerry Owens



1. Record_Next_Pointer_Info
2. Record_Name
3. Record_Next_Prior_Pos
4. IDMS_Model_File
5. Set_Owner_Member_Info
6. Set_Name
7. Set_Owner_Member_Pos

Module Name: Bildset Module Number: 3.5

Program Name: Idmsout Author: Capt Jerry Owens



1. Record_Next_Prior_Info
2. Set_Owner_Member_Info
3. CODASYL_Schema_File

Appendix D

Automated Hiearchical - CODASYL Database Interface

Schema Translator Data Dictionary

NAME: Access_Mode_Info
TYPE: VARIABLE
DATE: 24 Nov 1983
DESCRIPTION: Information showing the CODASYL access mode for the current CODASYL record. If the mode is to be CALC, then the element to perform the hashing on, will be included. If the mode is VIA, then the owner and member record names will be included.
ALIASES:
PART OF: CODASYL_Information_File
DATA CHARACTERISTICS: character string
VALUES:

NAME: Author_of_Schema
TYPE: VARIABLE
DATE: 27 Nov 1983
DESCRIPTION: Variable containing the name of the author of the schema.
ALIASES:
PART OF:
DATA CHARACTERISTICS: character string
VALUES:

NAME: Buffer_80
TYPE: VARIABLE
DATE: 26 Nov 1983
DESCRIPTION: This buffer will hold the first 80 characters of an S2k component.
ALIASES:
PART OF: Recin_Buffer
DATA CHARACTERISTICS: character string
VALUES:

NAME: CODASYL_Element_Description
TYPE: VARIABLE
DATE: 24 Nov 1983
DESCRIPTION: An equivalent CODASYL element description for a hierarchical element description.
ALIASES:
PART OF: CODASYL_Element_Info
DATA CHARACTERISTICS: character string
VALUES:

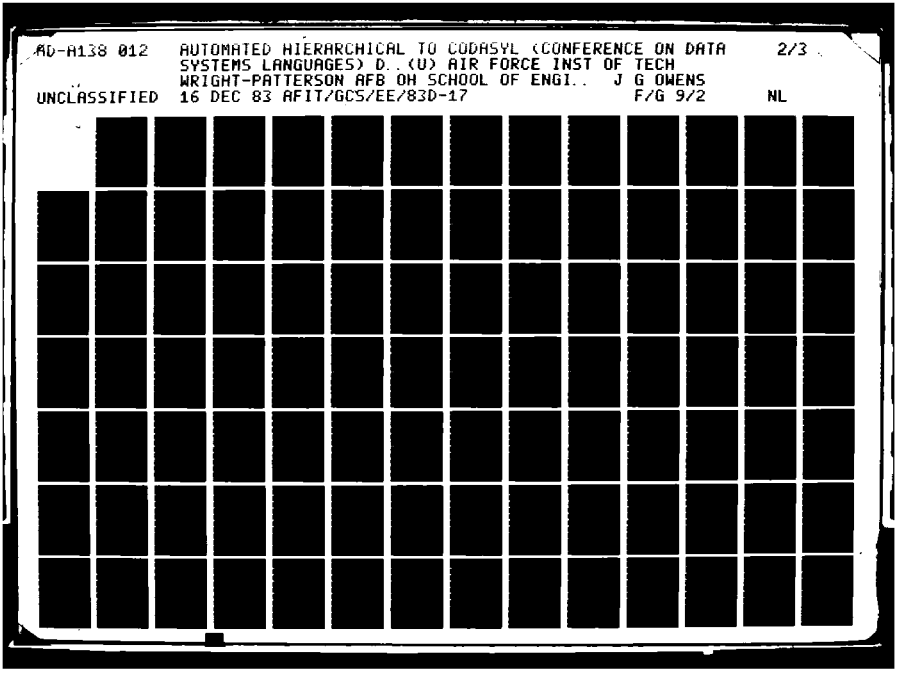
NAME: CODASYL_Element_Info
TYPE: VARIABLE
DATE: 24 Nov 1983
DESCRIPTION: Element information needed for the CODASYL schema found in the hierarchical schema.
ALIASES:
PART OF: CODASYL_Information_File
DATA CHARACTERISTICS: character string
VALUES:

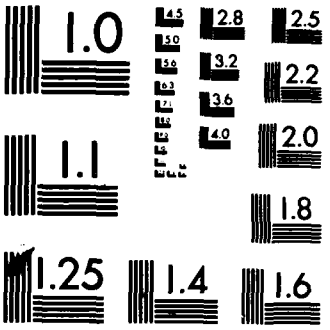
NAME: CODASYL_Element_Name
TYPE: VARIABLE
DATE: 24 Nov 1983
DESCRIPTION: An equivalent CODASYL schema element name for a hierarchical schema element name.
ALIASES:
PART OF: CODASYL_Element_Info
DATA CHARACTERISTICS: character_string
VALUES:

NAME: CODASYL_Information_File
TYPE: FILE
DATE: 24 Nov 1983
DESCRIPTION: File conatining necessary information to build the CODASYL schema, storage_schema, and subschema that was found in the Hierarchical_Schema_File.
ALIASES:
PART OF:
DATA CHARACTERISTICS: character string
VALUES:

NAME: CODASYL_Owner_Item_Information
TYPE: VARIABLE
DATE: 24 Nov 1983
DESCRIPTION: Information showing the record of which this element is a member of.
ALIASES:
PART OF: CODASYL_Element_Info
DATA CHARACTERISTICS: character string
VALUES:

NAME: CODASYL_Record_Info
TYPE: VARIABLE
DATE: 24 Nov 1983
DESCRIPTION: Record information found in the hierarchical schema needed for the CODASYL schema.
ALIASES:
PART OF: CODASYL_Information_File
DATA CHARACTERISTICS: character string
VALUES:


AD-A138 012
AUTOMATED HIERARCHICAL TO CODASYL (CONFERENCE ON DATA SYSTEMS LANGUAGES) D..(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. J G OWENS
2/3
UNCLASSIFIED
16 DEC 83 AFIT/GCS/EE/83D-17
F/G 9/2
NL



NAME: CODASYL_Record_Name
TYPE: VARIABLE
DATE: 24 Nov 1983
DESCRIPTION: An equivalent CODASYL record name for a hierarchical record name.
ALIASES:
PART OF: CODASYL_Record_Info
DATA CHARACTERISTICS: character string
VALUES:

NAME: CODASYL_Record_Owner_Information
TYPE: VARIABLE
DATE: 24 Nov 1983
DESCRIPTION: Information showing the parent record of the current record in the hierarchical schema.
ALIASES:
PART OF: CODASYL_Record_Info
DATA CHARACTERISTICS: character string
VALUES:

NAME: CODASYL_Schema_File
TYPE: FILE
DATE: 24 Nov 1983
DESCRIPTION: This file will contain the definitions of the various types of records in the database, the elements they contain, and the sets into which they are grouped.
ALIASES:
PART OF:
DATA CHARACTERISTICS: character string
VALUES:

NAME: CODASYL_Storage_Schema_File
TYPE: FILE
DATE: 24 Nov 1983
DESCRIPTION: This file will describe the storage structure (internal view) ofthe database.
ALIASES:
PART OF:
DATA CHARACTERISTICS: character string
VALUES:

NAME: CODASYL_Subschema_File
TYPE: FILE
DATE: 24 Nov 1983
DESCRIPTION: This file will consist essentially of a specification of all schema record types,all elements in those records, and all schema relationships (sets) linking those records.
ALIASES:
PART OF:
DATA CHARACTERISTICS: character string
VALUES:

NAME: CODASYL_Temp_File
TYPE: FILE
DATE: 24 Nov 1983
DESCRIPTION: Temporary file containing only information found in the hierarchical schema. This file will be used as input for further processing.
ALIASES:
PART OF:
DATA CHARACTERISTICS: character string
VALUES:

NAME: Current_System_Date
TYPE: VARIABLE
DATE: 27 Nov 1983
DESCRIPTION: Variable contains the current date found in the computer system. Used for documentation purposes.
ALIASES:
PART OF:
DATA CHARACTERISTICS: character string
VALUES:

NAME: DBA_Input
TYPE: VARIABLE
DATE: 24 Nov 1983
DESCRIPTION: Keyboard input from the DBA containing information necessary to build a CODASYL schema, storage_schema, and subschema that is not contained in the hierarchical schema.
ALIASES:
PART OF:
DATA CHARACTERISTICS: character string
VALUES:

NAME: Dummy_Record_Name
TYPE: VARIABLE
DATE: 26 Nov 1983
DESCRIPTION: Variable used to store a generated element name for a S2k record which has no elements. Idms does not support records with no elements therefore a dummy element must be generated.
ALIASES: CODASYL_Element_Name
PART OF: CODASYL_Element_Info
DATA CHARACTERISTICS: character string
VALUES:

NAME: Element_Desc_Info
TYPE: VARIABLE
DATE: 26 Nov 1983
DESCRIPTION: This variable is an array of records which contain element information needed for the translation. Each record of the array will contain the element's S2k component number, IDMS element name, IDMS element description, and the record of which this item is a member of.
ALIASES: CODASYL_Element_Info
PART OF:
DATA CHARACTERISTICS: array of records
VALUES:

NAME: Formatted_DBA_Input
TYPE: VARIABLE
DATE: 24 Nov 1983
DESCRIPTION: Input from the DBA placed into a specified form
ALIASES: User_Definition_File
PART OF:
DATA CHARACTERISTICS: character string
VALUES:

NAME: Formatted_Schema_Information
TYPE: VARIABLE
DATE: 24 Nov 1983
DESCRIPTION: A combination of the information found in the User_Definition_File, CODASYL_Information_File, and the Schema_Syntax needed for the CODASYL schema.
ALIASES: CODASYL_Schema_File
PART OF:
DATA CHARACTERISTICS: character_string
VALUES:

NAME: Formatted_Storage_Schema_Information
TYPE: VARIABLE
DATE: 24 Nov 1983
DESCRIPTION: A combination of Storage_Schema_Syntax and the User_Definition_File to produce the correct form of the CODASYL_Storage_Schema_File.
ALIASES: CODASYL_Storage_Schema_File
PART OF:
DATA CHARACTERISTICS: character_string
VALUES:

NAME: Formatted_Subschema_Information
TYPE: VARIABLE
DATE: 24 Nov 1983
DESCRIPTION: A combination of the Subschema_Syntax, the User_Definition_File, and the CODASYL_Information_File needed for the CODASYL subschema.
ALIASES: CODASYL_Subschema_File
PART OF:
DATA CHARACTERISTICS: character string
VALUES:

NAME: Found_Position
TYPE: VARIABLE
DATE: 26 Nov 1983
DESCRIPTION: This variable will inform the calling module of the next position in the Recin_Buffer to continue processing from. All positions less than this value have already been processed by a Rdschm module.
ALIASES:
PART OF:
DATA CHARACTERISTICS: integer
VALUES:

NAME: Hierarchical_Element Information
TYPE: VARIABLE
DATE: 24 Nov 1983
DESCRIPTION: Element information conatined in the hierarchical schema.
ALIASES:
PART OF:
DATA CHARACTERISTICS: character string
VALUES:

NAME: Hierarchical_Record_Information
TYPE: VARIABLE
DATE: 24 Nov 1983
DESCRIPTION: Record information found in the hierarchical schema.
ALIASES:
PART OF:
DATA CHARACTERISTICS: character string
VALUES:

NAME: Hierarchical_Schema_File
TYPE: FILE
DATE: 24 Nov 1983
DESCRIPTION: File containing the hierarchical schema that is to be translated to a CODASYL schema.
ALIASES:
PART OF:
DATA CHARACTERISTICS: character string
VALUES:

NAME: IDMS_Area_Name
TYPE: VARIABLE
DATE: 27 Nov 1983
DESCRIPTION: Variable containing the name of the area region in the IDMS schema.
ALIASES:
PART OF:
DATA CHARACTERISTICS:
VALUES:

NAME: IDMS_DMCL_File
TYPE: FILE
DATE: 27 Nov 1983
DESCRIPTION: File containing definitions of areas and files of the schema as well as information on the buffers and journals that are used.
ALIASES:
PART OF:
DATA CHARACTERISTICS: character string
VALUES:

NAME: IDMS_DMCL_Name
TYPE: VARIABLE
DATE: 27 Nov 1983
DESCRIPTION: Variable storing the name of the IDMS_DMCL.
ALIASES:
PART OF:
DATA CHARACTERISTICS: character string
VALUES:

NAME: IDMS_File_Name
TYPE: VARIABLE
DATE: 27 Nov 1983
DESCRIPTION: Variable containing the name of the file which contains the database.
ALIASES:
PART OF:
DATA CHARACTERISTICS: character string
VALUES:

NAME: IDMS_Model_File
TYPE: FILE
DATE: 26 Nov 1983
DESCRIPTION: This file contains all of the processed S2k information that was contained in the S2k_Temp_File plus it now contains the record location mode information. The information is in an easy to read form so the DBA can make changes if necessary.
ALIASES: CODASYL_Information_File
PART OF:
DATA CHARACTERISTICS:
VALUES:

NAME: IDMS_Schema_Name
TYPE: VARIABLE
DATE: 27 Nov 1983
DESCRIPTION: Variable containing the name of the IDMS schema that is being created by the thesis software.
ALIASES:
PART OF:
DATA CHARACTERISTICS: character string
VALUES:

NAME: IDMS_Schema_Version
TYPE: VARIABLE
DATE: 27 Nov 1983
DESCRIPTION: Variable containing the version number of this particular schema.
ALIASES:
PART OF:
DATA CHARACTERISTICS: character string
VALUES:

NAME: Input_Buffer
TYPE: VARIABLE
DATE: 24 Nov 1983
DESCRIPTION: A buffer used to store information read from a file.
ALIASES:
PART OF:
DATA CHARACTERISTICS: character string
VALUES:

NAME: Matching_Paren_Found
TYPE: VARIABLE
DATE: 26 Nov 1983
DESCRIPTION: This variable will indicate to the calling procedure whether or not matching parenthesis has been found in the string being processed. The value will be a 1 if true and a 0 if false.
ALIASES:
PART OF:
DATA CHARACTERISTICS: integer
VALUES: 0, 1

NAME: Number_of_Elements
TYPE: VARIABLE
DATE: 26 Nov 1983
DESCRIPTION: This variable will contain the number of element entries that have been processed. This value will be used for the dynamic allocation of an array of element information.
ALIASES:
PART OF:
DATA CHARACTERISTICS: integer
VALUES:

NAME: Number_of_Records
TYPE: VARIABLE
DATE: 26 Nov 1983
DESCRIPTION: This variable will contain the current count of the number of record entries that have been processed. This value will be used for the dynamic allocation of an array of record information.
ALIASES:
PART OF:
DATA CHARACTERISTICS: integer
VALUES:

NAME: Pattern_to_Be_Found
TYPE: VARIABLE
DATE: 26 Nov 1983
DESCRIPTION: This variable ia a character string of 1 - 20 characters that is to be searched for in a parent string.
ALIASES:
PART OF:
DATA CHARACTERISTICS: character string
VALUES:

NAME: Prompt_Message
TYPE: VARIABLE
DATE: 24 Nov 1983
DESCRIPTION: A message sent to the DBA for the required information needed for the CODASYL schema, storage_schema, or subschema not found in the hieracrhical schema.
ALIASES:
PART OF:
DATA CHARACTERISTICS: character string
VALUES:

NAME: Recin_Buffer
TYPE: VARIABLE
DATE: 26 Nov 1983
DESCRIPTION: An input buffer with the capacity of 160 characters used to store the hierarchical schema file information. This size was chosen to support information being across two cards of input. One card having the size of 80 charcaters.
ALIASES: Input_Buffer, Hierarchical_Element_Information, Hierarchical_Record_Information
PART OF:
DATA CHARACTERISTICS: character string
VALUES:

NAME: Recin_Buffer_Start_Position
TYPE: VARIABLE
DATE: 26 Nov 1983
DESCRIPTION: This variable will contain the next position in the Recin_Buffer to be processed.
ALIASES:
PART OF:
DATA CHARACTERISTICS: integer
VALUES:

NAME: Record_Desc_Info
TYPE: VARIABLE
DATE: 26 Nov 1983
DESCRIPTION: This variable is an array of records which contain the record information needed to perform the translation. The information contained is the IDMS record name, IDMS record number, S2k component number, S2k owner component number, and the position in the Element_Desc_Info array of the record's first element.
ALIASES: CODASYL_Record_Info
PART OF:
DATA CHARACTERISTICS: array of records
VALUES:

NAME: Record_Desc_Info_Pos
TYPE: VARIABLE
DATE: 26 Nov 1983
DESCRIPTION: A variable containing the current position of the current record of information in the Record_Desc_Info array. This variable is very important in the task of finding the owner record for the current record.
ALIASES:
PART OF:
DATA CHARACTERISTICS: integer
VALUES:

NAME: Record_Name
TYPE: VARIABLE
DATE: 26 Nov 1983
DESCRIPTION: A buffer used to contain the IDMS record name.
ALIASES: CODASYL_Record_Name
PART OF:
DATA CHARACTERISTICS: character string
VALUES:

NAME: Record_Next_Prior_Info
TYPE: VARIABLE
DATE: 27 Nov 1983
DESCRIPTION: An array of records containing the IDMS record name, its next pointer value, and its prior pointer value.
ALIASES:
PART OF:
DATA CHARACTERISTICS: array of records
VALUES:

NAME: Record_Next_Prior_Pos
TYPE: VARIABLE
DATE: 27 Nov 1983
DESCRIPTION: Variable used as a subscript into the Record_Next_Pos_Info array.
ALIASES:
PART OF:
DATA CHARACTERISTICS: integer
VALUES:

NAME: Recout_Buffer
TYPE: VARIABLE
DATE: 26 Nov 1983
DESCRIPTION: An 80 character buffer used for storing the information that was contained in the Recin_Buffer and is needed for the IDMS translation. After the entire Recin_Buffer has been processed, the Recout_Buffer will contain all of the necessary record information or all of the necessary element information, depending on the type of information currently being processed, needed for the translation. Certain parts of this buffer will be updated by various modules.
ALIASES: CODASYL_Element_Info, CODASYL_Record_Info
PART OF:
DATA CHARACTERISTICS: character string
VALUES:

NAME: S2k_Log_File
TYPE: FILE
DATE: 26 Nov 1983
DESCRIPTION: This file is an information file for the DBA. Every S2k component record will be shown with its mapped IDMS translation.
ALIASES:
PART OF:
DATA CHARACTERISTICS: character string
VALUES:

NAME: S2k_Temp_File
TYPE: FILE
DATE: 26 Nov 1983
DESCRIPTION: This file will contain all of the information needed for the IDMS translation that was found in the S2k input schema. The record and element names will be truncated to the correct size as restricted by IDMS. Also the S2k element descriptions will have been mapped to an equivalent IDMS element description.
ALIASES: CODASYL_Temp_File
PART OF:
DATA CHARACTERISTICS: character string
VALUES:

NAME: Schema_Syntax
TYPE: VARIABLE
DATE: 24 Nov 1983
DESCRIPTION: Syntax necessary for the correct format of the CODASYL schema.
ALIASES:
PART OF: Formatted_Schema_Information
DATA CHARACTERISTICS: character_string
VALUES:

NAME: Set_Count
TYPE: VARIABLE
DATE: 26 Nov 1983
DESCRIPTION: Variable which contains the current number of set relations using the VIA location mode. This value is used in generating the unique IDMS set names.
ALIASES:
PART OF:
DATA CHARACTERISTICS: integer
VALUES:

NAME: Set_Name
TYPE: VARIABLE
DATE: 27 Nov 1983
DESCRIPTION: variable used to store the name of an IDMS set.
ALIASES:
PART OF:
DATA CHARACTERISTICS: character set
VALUES:

NAME: Set_Owner_Member_Info
TYPE: VARIABLE
DATE: 27 Nov 1983
DESCRIPTION: An array of records containing the IDMS set name, the name of the record owning the set, and the name of the record which is the member of the set.
ALIASES:
PART OF:
DATA CHARACTERISTICS: array of records
VALUES:

NAME: Set_Owner_Member_Pos
TYPE: VARIABLE
DATE: 27 Nov 1983
DESCRIPTION: Variable used as a subscript into the Set_Owner_Member_Info array.
ALIASES:
PART OF:
DATA CHARACTERISTICS: integer
VALUES:

NAME: Starting_Pos_of_Pattern
TYPE: VARIABLE
DATE: 26 Nov 1983
DESCRIPTION: This variable will contain the starting position in the parent character string of the pattern of characters being searched for. If the desired pattern is not found then the value is 0.
ALIASES:
PART OF:
DATA CHARACTERISTICS: integer
VALUES:

NAME: Storage_Schema_Syntax
TYPE: VARIABLE
DATE: 24 Nov 1983
DESCRIPTION: Synatx information needed for the correct generation of the CODASYL_Storage_Schema_File.
ALIASES:
PART OF: Formatted_Storage_Schema_Information
DATA CHARACTERISTICS: character_string
VALUES:

NAME: Subschema_Syntax
TYPE: VARIABLE
DATE: 24 Nov 1983
DESCRIPTION: Synatx for the CODASYL Subschema generation.
ALIASES:
PART OF: Formatted_Subschema_Information
DATA CHARACTERISTICS: character_string
VALUES:

NAME: Total_Records_and_Elements
TYPE: VARIABLE
DATE: 26 Nov 1983
DESCRIPTION: An integer containing the total number of records and elements in the schema.
ALIASES:
PART OF:
DATA CHARACTERISTICS: integer
VALUES:

NAME: User_Definition_File
TYPE: FILE
DATE: 24 Nov 1983
DESCRIPTION: File which contains all of the DBA_Input.
ALIASES:
PART OF:
DATA CHARACTERISTICS: character string
VALUES:

Appendix E

Automated Hierarchical - CODASYL Database Interface

Schema Translator Pl/1 Source Code

```
/* USRPRMP 1.0 */
/*********************************************************************
 *  DATE:  15 AUG 1983                                               *
 *  VERSION:  1.1                                                    *
 *  NAME:  USRPRMP                                                   *
 *  MODULE NUMBER:  1.0                                              *
 *  FUNCTION:  MAIN DRIVER. PROMPTS USER FOR THE NECESSARY INFO      *
 *             TO BUILD THE IDMS SCHEMA, SUBSCHEMA, AND DMCL.        *
 *  INPUTS:                                                          *
 *  OUTPUTS:                                                         *
 *  GLOBAL VARIABLES USED:                                           *
 *  GLOBAL VARIABLES CHANGED:                                        *
 *  GLOBAL TABLES USED:                                              *
 *  FILES READ:                                                      *
 *  FILES WRITTEN:                                                   *
 *  MODULES CALLED:  RDWRIT                                          *
 *  CALLING MODULES:                                                 *
 *  AUTHOR:  CAPT. JERRY OWENS                                       *
 *  HISTORY:                                                         *
 *********************************************************************/
/*

*/
USRPRMP: PROC OPTIONS (MAIN);
PUT SKIP;
PUT SKIP EDIT('ENTER SCHEMA NAME  (1 - 8 CHARACTERS)  ')
            (COL(1),A);
PUT SKIP;
CALL RDWRIT;
PUT SKIP;
PUT SKIP EDIT('ENTER SCHEMA VERSION NUMBER ( 1 - 3 DIGITS )  ')
            (COL(1),A);
PUT SKIP;
CALL RDWRIT;
PUT SKIP;
PUT SKIP EDIT('ENTER AUTHOR NAME  (1 - 30 CHARACTERS)    ')
            (COL(1),A);
PUT SKIP;
CALL RDWRIT;
PUT SKIP;
PUT SKIP EDIT('ENTER AUTHOR PHONE NUMBER  (AREA CODE - NUMBER)  ')
            (COL(1),A);
PUT SKIP;
CALL RDWRIT;
```

```
PUT SKIP;
PUT SKIP EDIT('ENTER DATABASE FILE NAME  (1 - 8 CHARACTERS)    ')
            (COL(1),A);
PUT SKIP;
CALL RDWRIT;
PUT SKIP;
PUT SKIP EDIT('ENTER DEVICE TYPE (3330/3350)   ')(COL(1),A);
PUT SKIP;
CALL RDWRIT;
PUT SKIP;
PUT SKIP EDIT('ENTER AREA NAME (1 - 16 CHARACTERS)    ')(COL(1),A);
PUT SKIP;
CALL RDWRIT;
PUT SKIP;
PUT SKIP EDIT('ENTER AREA STARTING RANGE # (1 - 8 DIGITS) ')
            (COL(1),A);
PUT SKIP;
CALL RDWRIT;
PUT SKIP;
PUT SKIP EDIT('ENTER AREA ENDING RANGE # (1 - 8 DIGITS)  ')
            (COL(1),A);
PUT SKIP;
CALL RDWRIT;
PUT SKIP;
PUT SKIP EDIT('ENTER DMCL NAME (1 - 8 CHARACTERS)  ')
            (COL(1),A);
PUT SKIP;
CALL RDWRIT;
PUT SKIP;
PUT SKIP EDIT('ENTER SUBSCHEMA NAME (1 - 8 CHARACTERS)   ')
            (COL(1),A);
PUT SKIP;
CALL RDWRIT;
```

```
/* RDWRIT 1.1 */
/*****************************************************************
 *  DATE:  15 AUG 1983                                           *
 *  VERSION:  1.0                                                *
 *  NAME:  RDWRIT                                                *
 *  MODULE NUMBER:  1.1                                          *
 *  FUNCTION:  READ TERMINAL INPUT AND WRITE IT FILE USERDEFS    *
 *  INPUTS:                                                      *
 *  OUTPUTS:                                                     *
 *  GLOBAL VARIABLES USED:                                       *
 *  GLOBAL VARIABLES CHANGED:                                    *
 *  GLOBAL TABLES USED:                                          *
 *  FILES READ:                                                  *
 *  FILES WRITTEN: USERDEFS                                      *
 *  MODULES CALLED:                                              *
 *  CALLING MODULES: USRPRMP                                     *
 *  AUTHOR:  CAPT. JERRY OWENS                                   *
 *  HISTORY:                                                     *
 *****************************************************************/
/*

*/
     RDWRIT: PROCEDURE;
        DCL USERIN FILE RECORD SEQUENTIAL INPUT;
        DCL USEROUT FILE STREAM OUTPUT;
        DCL USER_INPUT CHAR(80);
     READ FILE (USERIN) INTO (USER_INPUT);
     PUT FILE(USEROUT) EDIT (USER_INPUT) (COL(1),A);
END RDWRIT;
END USRPRMP;
```

```
/* RDSCHM 2.0 */
/*****************************************************************
 *  DATE:   7 SEP 1983                                           *
 *  VERSION:  1.1                                                *
 *  NAME:  RDSCHM                                                *
 *  MODULE NUMBER:  2.0                                          *
 *  FUNCTION:  MAIN DRIVER. CONTINUES TO CALL THE APPROPIATE     *
 *             MODULES UNTIL AND EOF OF THE INPUT FILE IS        *
 *             ENCOUNTERED.                                      *
 *  INPUTS:                                                      *
 *  OUTPUTS:                                                     *
 *  GLOBAL VARIABLES USED:                                       *
 *  GLOBAL VARIABLES CHANGED:                                    *
 *  GLOBAL TABLES USED:                                          *
 *  FILES READ:                                                  *
 *  FILES WRITTEN:                                               *
 *  MODULES CALLED:  RDBUFF, FRSTREC, READOUT, GETNUM, GETNAME,  *
 *                   GETTYPE, SCHMBL                             *
 *  CALLING MODULES:                                             *
 *  AUTHOR:  CAPT. JERRY OWENS                                   *
 *  HISTORY:                                                     *
 *****************************************************************/
/*

*/
RDSCHM: PROC OPTIONS (MAIN);
        DCL GETNUM ENTRY (CHAR(160),CHAR(80),FIXED BIN(15))
            RETURNS (FIXED BIN(15));
        DCL RECFND ENTRY (CHAR(160),CHAR(80),FIXED BIN(15));
        DCL INTFND ENTRY (CHAR(160),CHAR(80),FIXED BIN(15));
        DCL CHRFND ENTRY (CHAR(160),CHAR(80),FIXED BIN(15));
        DCL DECFND ENTRY (CHAR(160),CHAR(80),FIXED BIN(15));
        DCL MONEY ENTRY (CHAR(160),CHAR(80),FIXED BIN(15));
        DCL DATFND ENTRY (CHAR(160),CHAR(80),FIXED BIN(15));
        DCL READOUT ENTRY (CHAR(160),CHAR(80));
        DCL RDBUFF ENTRY (CHAR(160));
        DCL FRSTREC ENTRY (CHAR(160),CHAR(80))
            RETURNS (FIXED BIN(15));
        DCL GETNAME ENTRY (CHAR(160),CHAR(80),FIXED BIN(15))
            RETURNS (FIXED BIN(15));
        DCL GETTYPE ENTRY (CHAR(160),CHAR(80),FIXED BIN(15),
                           FIXED BIN(15),FIXED BIN(15));
        DCL NONBLNK ENTRY (CHAR(160),FIXED BIN(15))
            RETURNS (FIXED BIN(15));
        DCL CHKPAR ENTRY (CHAR(80))
            RETURNS (FIXED BIN(15));
        DCL FINDPAT ENTRY (CHAR(160),FIXED BIN(15),CHAR(20) VAR)
            RETURNS (FIXED BIN(15));
        DCL SCHMBL ENTRY;
        DCL S2KSCHM FILE STREAM INPUT;
        DCL S2KTEMP FILE STREAM;
        DCL IDMSKDL FILE STREAM OUTPUT;
        DCL RECIN_BUFFER CHAR(160);
```

```
        DCL RECOUT_BUFFER CHAR(80);
        DCL BUF_POS FIXED BIN(15);
        DCL REC_CNT FIXED BIN(15) INIT(0);
        DCL ELEM_CNT FIXED BIN(15) INIT(0);
        ON ENDFILE (S2KSCHM) GO TO SETBL;
OPEN FILE (S2KTEMP) OUTPUT;
PUT PAGE;
BUF_POS = 0;
/*
 ************************************************************
 *    FIND THE NAME OF S2K RECORD ZERO.                     *
 ************************************************************
*/
DO WHILE (BUF_POS = 0);
  CALL RDBUFF(RECIN_BUFFER);
  BUF_POS = FRSTREC(RECIN_BUFFER,RECOUT_BUFFER);
END;
CALL READOUT (RECIN_BUFFER,RECOUT_BUFFER);
REC_CNT = REC_CNT + 1;
CALL RDBUFF(RECIN_BUFFER);
/*
  ************************************************************
  *   CONTINUE PROCESSING UNTIL EOF HAS BEEN REACHED         *
  ************************************************************
*/
   DO WHILE ('1'B);
     BUF_POS = 1;
     BUF_POS = GETNUM(RECIN_BUFFER,RECOUT_BUFFER,BUF_POS);
     IF BUF_POS > 0 THEN DO;
       BUF_POS = GETNAME(RECIN_BUFFER,RECOUT_BUFFER,BUF_POS);
       CALL GETTYPE(RECIN_BUFFER,RECOUT_BUFFER,BUF_POS,
                    REC_CNT,ELEM_CNT);
       CALL READOUT (RECIN_BUFFER,RECOUT_BUFFER);
     END;
     CALL RDBUFF(RECIN_BUFFER);
     RECOUT_BUFFER = '';
   END;
/*
  ************************************************************
  *   THIS PORTION WILL BE EXECUTED WHEN THE EOF HAS BEEN    *
  *   REACHED                                                *
  ************************************************************
*/
 SETBL: CALL SCHMBL(REC_CNT,ELEM_CNT);
```

```
/*  RDBUFF 2.1  */
/*********************************************************************
 *  DATE:  30 AUG 83                                                 *
 *  VERSION:  1.0                                                    *
 *  NAME:  RDBUFF                                                    *
 *  MODULE NUMBER:  2.1                                              *
 *  FUNCTION:  READ IN THE BUFFER FOR PROCESSING                     *
 *  INPUTS:                                                          *
 *  OUTPUTS:  RECIN_BUFFER                                           *
 *  GLOBAL VARIABLES USED:                                           *
 *  GLOBAL VARIABLES CHANGED:                                        *
 *  GLOBAL TABLES USED:                                              *
 *  FILES READ:  S2KSCHM                                             *
 *  FILES WRITTEN:                                                   *
 *  MODULES CALLED:  CHKPAR, FINDPAT, NONBLNK, SUBSTR                *
 *  CALLING MODULES:  RDSCHM                                         *
 *  AUTHOR:  CAPT JERRY OWENS                                        *
 *  HISTORY:                                                         *
 *********************************************************************/
/*

*/
RDBUFF: PROCEDURE (RECIN_BUFFER);
        DCL RECIN_BUFFER CHAR(160);
        DCL IN_BUF1 CHAR(80);
        DCL IN_BUF2 CHAR(80);
        DCL BLNK_REC CHAR(80) INIT ((80)' ');
        DCL RDBUF_LOC FIXED BIN(15);
        DCL PAREN_CNT FIXED BIN(15);
   IN_BUF2 = BLNK_REC;
/*
  ***********************************************************
  *  READ IN 80 COLUMNS OF THE S2K SCHEMA FILE.....         *
  ***********************************************************
*/
   GET FILE (S2KSCHM) EDIT (IN_BUF1)(COL(1),A(80));
   PAREN_CNT = CHKPAR(IN_BUF1);
/*
  ***********************************************************
  * ALL INFO IN FIRST 80 COLUMNS ?????                      *
  ***********************************************************
*/
   IF PAREN_CNT = 0 THEN DO;
/*
  ***********************************************************
  *  NOPE, MUST READ IN SECOND CARD .....                   *
  ***********************************************************
*/
     GET FILE (S2KSCHM) EDIT (IN_BUF2)(COL(1),A(80));
     RECIN_BUFFER = IN_BUF1 :: IN_BUF2;
```

```
/*
  ************************************************************
  * HOW FAR IS THIS CARD INDENTED ???                        *
  ************************************************************
*/
     RDBUF_LOC = FINDPAT(RECIN_BUFFER,1,'* ');
     RDBUF_LOC = NONBLNK(RECIN_BUFFER,RDBUF_LOC + 1);
/*
  ************************************************************
  *  PLACE THE INDENTION BLANKS AT THE END OF THE CARD       *
  ************************************************************
*/
     SUBSTR(IN_BUF2,1,((80-RDBUF_LOC)+1)) =
       SUBSTR(IN_BUF2,RDBUF_LOC,((80-RDBUF_LOC)+1));
     SUBSTR(IN_BUF2,((80-RDBUF_LOC)+2),(RDBUF_LOC - 1)) =
       SUBSTR(BLNK_REC,1,(RDBUF_LOC - 1));
/*
  ************************************************************
  * MERGE THE TWO CARDS TO LOOK LIKE ONE LONG CARD           *
  ************************************************************
*/
     SUBSTR(RECIN_BUFFER,1,80) = IN_BUF1;
     SUBSTR(RECIN_BUFFER,73,80) = IN_BUF2;
     SUBSTR(RECIN_BUFFER,153,8) =
       SUBSTR(BLNK_REC,1,8);
   END;
   ELSE
     RECIN_BUFFER = IN_BUF1 :: IN_BUF2;
END RDBUFF;
```

```
/*  CHKPAR 2.1.1  */
/*********************************************************************
 *  DATE:  24 AUG 83                                                 *
 *  VERSION:  1.0                                                    *
 *  NAME:  CHKPAR                                                    *
 *  MODULE NUMBER:  2.1.1                                            *
 *  FUNCTION:  CHECKS FOR MATCHING PARENTHESIS IN A GIVEN BUFFER     *
 *  INPUTS:  IN_BUF1  -  80 CHARACTER INPUT BUFFER                   *
 *  OUTPUTS:  1  - IF MATCHING PARENS ARE FOUND                      *
 *            0  - IF NO MATCHING PARENS ARE FOUND                   *
 *  GLOBAL VARIABLES USED:                                           *
 *  GLOBAL VARIABLES CHANGED:                                        *
 *  GLOBAL TABLES USED:                                              *
 *  FILES READ:                                                      *
 *  FILES WRITTEN:                                                   *
 *  MODULES CALLED:  SUBSTR                                          *
 *  CALLING MODULES:  RDBUFF                                         *
 *  AUTHOR:  CAPT. JERRY OWENS                                       *
 *  HISTORY:                                                         *
 *********************************************************************/
/*

*/
CHKPAR: PROCEDURE (IN_BUF1) RETURNS (FIXED BIN(15));
        DCL IN_BUF1 CHAR(80);
        DCL PARCNT FIXED BIN(15);
        DCL I FIXED BIN(15);
        DCL LEFT_PAR FIXED BIN(15) INIT(0);
        DCL RIGHT_PAR FIXED BIN(15) INIT(0);
     DO I = 1 TO 80;
       IF (SUBSTR(IN_BUF1,I,1)) = '(' THEN
         LEFT_PAR = LEFT_PAR + 1;
       IF (SUBSTR(IN_BUF1,I,1)) = ')' THEN
         RIGHT_PAR = RIGHT_PAR + 1;
     END;
     IF (LEFT_PAR = RIGHT_PAR) THEN
       I = 1;
         ELSE
           I = 0;
     RETURN (I);
   END CHKPAR;
```

```
/* FINDPAT 2.1.2 */
/*********************************************************************
 *  DATE:  24 AUG 83                                                 *
 *  VERSION:  1.0                                                    *
 *  NAME:  FINDPAT                                                   *
 *  MODULE NUMBER: 2.1.2                                             *
 *  FUNCTION:  FIND A PASSED CHARACTER PATTERN IN A PASSED           *
 *             CHARACTER STRING BEGINNING WITH A PASSED STARTING     *
 *             POSISTION IN THE STRING.                              *
 *  INPUTS:  RECIN_BUFFER   -  160 CHARACTER INPUT BUFFER            *
 *           FINDPAT_POS    -  STARTING POS IN THE INPUT BUFFER      *
 *                             FIXED BIN(15)                        *
 *           FINDCHR        -  PATTERN TO BE SEARCHED FOR            *
 *                             CHAR(20) VAR                         *
 *  OUTPUTS:  FINDPAT_POS   -  STARTING POS OF THE PATTERN           *
 *                             FIXED BIN(15)                        *
 *  GLOBAL VARIABLES USED:                                           *
 *  GLOBAL VARIABLES CHANGED:                                        *
 *  GLOBAL TABLES USED:                                              *
 *  FILES READ:                                                      *
 *  FILES WRITTEN:                                                   *
 *  MODULES CALLED:  LENGTH, SUBSTR, INDEX                           *
 *  CALLING MODULES:  RECFND, INTFND, CHRFND, DATFND, DECFND,        *
 *                    MONEY, RDBUFF, FRSTREC, GETNUM, GETNAME,       *
 *                    GETTYPE                                        *
 *  AUTHOR:  CAPT JERRY OWENS                                        *
 *  HISTORY:                                                         *
 *********************************************************************/
/*

*/
FINDPAT: PROCEDURE (RECIN_BUFFER,FINDPAT_POS,FINDCHR) RETURNS
                   (FIXED BIN(15));
       DCL RECIN_BUFFER CHAR(160);
       DCL FINDPAT_POS FIXED BIN(15);
       DCL TEMP_BUFFER CHAR(160) VAR;
       DCL FINDCHR CHAR(20) VAR;
       DCL TEMP_LNGTH FIXED BIN(15);
       DCL LOCATE_POS FIXED BIN(15);
/*
  ***************************************************************
  *   GET THE LENGTH OF THE STRING STARTING WITH THE GIVEN      *
  *   STARTING POSITION.                                        *
  ***************************************************************
*/
   TEMP_LNGTH = (LENGTH(RECIN_BUFFER) - FINDPAT_POS) + 1;
   TEMP_BUFFER = SUBSTR(RECIN_BUFFER,FINDPAT_POS,TEMP_LNGTH);
```

```
/*
  ****************************************************************
  *  SEARCH IN THE NEWLY CREATED SUBSTRING FOR THE GIVEN PATTERN*
  *  IF THE GIVEN PATTERN IS FOUND, THEN UPDATE THE POSITION    *
  *  WITH THE STARTING POSITION ADDED. THIS WILL GIVE THE RIGHT *
  *  POSITION FOR THE ENTIRE STRING.                            *
  ****************************************************************
*/
   LOCATE_POS = INDEX(TEMP_BUFFER,FINDCHR);
   IF LOCATE_POS := 0 THEN
     LOCATE_POS = LOCATE_POS + (FINDPAT_POS - 1);
   RETURN (LOCATE_POS);
END FINDPAT;
```

```
/*  NONBLNK 2.1.3  */
/*********************************************************************
 *  DATE:  24 AUG 83                                                 *
 *  VERSION:  1.0                                                    *
 *  NAME:  NONBLNK                                                   *
 *  MODULE NUMBER:  2.1.3                                            *
 *  FUNCTION:  FIND THE FIRST NONBLNK CHARACTER IN A GIVEN STRING    *
 *             BEGINNING WITH A GIVEN STARTING POINT.                *
 *  INPUTS:  RECIN_BUFFER   - 160 CHARACTER INPUT BUFFER             *
 *           NONBLNK_POS    - STARTING POSISTION IN BUFFER           *
 *                            FIXED BIN(15)                          *
 *  OUTPUTS:  NONBLNK_POS   - POINTS TO THE FIRST NON BLANK          *
 *                            CHARACTER FROM THE GIVEN STARTING      *
 *                            POSISTION                              *
 *  GLOBAL VARIABLES USED:                                           *
 *  GLOBAL VARIABLES CHANGED:                                        *
 *  GLOBAL TABLES USED:                                              *
 *  FILES READ:                                                      *
 *  FILES WRITTEN:                                                   *
 *  MODULES CALLED:  SUBSTR                                          *
 *  CALLING MODULES:  RDSCHM                                         *
 *  AUTHOR:  CAPT. JERRY OWENS                                       *
 *  HISTORY:                                                         *
 *********************************************************************/
/*

*/
NONBLNK: PROCEDURE (RECIN_BUFFER,NONBLNK_POS) RETURNS
                   (FIXED BIN(15));
        DCL RECIN_BUFFER CHAR(160);
        DCL NONBLNK_POS FIXED BIN(15);
   DO WHILE ((SUBSTR(RECIN_BUFFER,NONBLNK_POS,1) = ' ') &
            (NONBLNK_POS <= 160));
     NONBLNK_POS = NONBLNK_POS + 1;
   END;
   IF NONBLNK_POS > 160 THEN NONBLNK_POS = 0;
   RETURN (NONBLNK_POS);
   END NONBLNK;
```

```
/* FRSTREC 2.2 */
/*********************************************************************
 * DATE:  28 AUG 83                                                  *
 * VERSION:  1.0                                                     *
 * NAME:  FRSTREC                                                    *
 * MODULE NUMBER:  2.2                                               *
 * FUNCTION:  FIND THE RECORD NAME OF RECORD ZERO (0).               *
 * INPUTS:  RECIN_BUFFER  - 160 CHARACTER INPUT BUFFER               *
 *          RECOUT_BUFFER -  80 CHARACTER OUTPUT BUFFER              *
 * OUTPUTS: FRSTREC_POS  -  POSISTION IN THE BUFFER OF THE NAME.     *
 *                          FIXED BIN(15)                            *
 * GLOBAL VARIABLES USED:                                            *
 * GLOBAL VARIABLES CHANGED:                                         *
 * GLOBAL TABLES USED:                                               *
 * FILES READ:                                                       *
 * FILES WRITTEN:                                                    *
 * MODULES CALLED:  FINDPAT, NONBLNK, SUBSTR                         *
 * CALLING MODULES:  RDSCHM                                          *
 * AUTHOR:  CAPT JERRY OWENS                                         *
 * HISTORY:                                                          *
 *********************************************************************/
/*

*/
FRSTREC: PROCEDURE (RECIN_BUFFER,RECOUT_BUFFER) RETURNS
                  (FIXED BIN(15));
       DCL RECIN_BUFFER CHAR(160);
       DCL RECOUT_BUFFER CHAR(80);
       DCL FRSTREC_POS FIXED BIN(15);
FRSTREC_POS = FINDPAT(RECIN_BUFFER,1,'NAME IS');
IF FRSTREC_POS := 0 THEN DO;
  FRSTREC_POS = FRSTREC_POS + 7;
  FRSTREC_POS = NONBLNK(RECIN_BUFFER,FRSTREC_POS);
  SUBSTR(RECOUT_BUFFER,1,4) = '0   ';
  SUBSTR(RECOUT_BUFFER,5,5) = '     ';
  SUBSTR(RECOUT_BUFFER,10,16) = SUBSTR(RECIN_BUFFER,FRSTREC_POS,16);
  SUBSTR(RECOUT_BUFFER,26,22) = '                      ';
  SUBSTR(RECOUT_BUFFER,48,4) = 'REC ';
  SUBSTR(RECOUT_BUFFER,52,29) = '';
END;
RETURN(FRSTREC_POS);
END FRSTREC;
```

```
/*  READOUT 2.3  */
/*************************************************************************
 *  DATE:  1 SEP 83                                                      *
 *  VERSION:  1.0                                                        *
 *  NAME:  READOUT                                                       *
 *  MODULE NUMBER:  2.3                                                  *
 *  FUNCTION:  CREATES OUTPUT LOG FOR THE USER, AND CREATES THE          *
 *             INPUT DATA FILE FOR SCHMBILD.                             *
 *  INPUTS:  RECIN_BUFFER  -  160 CHARACTER INPUT BUFFER                 *
 *           RECOUT_BUFFER -  80 CHARACTER OUTPUT BUFFER                 *
 *  OUTPUTS:                                                             *
 *  GLOBAL VARIABLES USED:                                               *
 *  GLOBAL VARIABLES CHANGED:                                            *
 *  GLOBAL TABLES USED:                                                  *
 *  FILES READ:                                                          *
 *  FILES WRITTEN:  SYSPRINT FILE                                        *
 *                  S2KOUT                                               *
 *  MODULES CALLED:                                                      *
 *  CALLING MODULES: RDSCHM                                              *
 *  AUTHOR:  CAPT JERRY OWENS                                            *
 *  HISTORY:                                                             *
 *************************************************************************/
/*

*/
READOUT: PROCEDURE (RECIN_BUFFER,RECOUT_BUFFER);
        DCL RECIN_BUFFER CHAR(160);
        DCL RECOUT_BUFFER CHAR(80);
        DCL INFOSEP CHAR (100) INIT ((100)'*');
   PUT SKIP;
   PUT SKIP;
   PUT SKIP;
   PUT EDIT (INFOSEP) (COL(1),A);
   PUT EDIT ('*',SUBSTR(RECIN_BUFFER,1,80),'*')
            (COL(1),A,COL(11),A,COL(100),A);
   PUT EDIT ('*',SUBSTR(RECIN_BUFFER,81,80),'*')
            (COL(1),A,COL(11),A,COL(100),A);
   PUT EDIT ('*',RECOUT_BUFFER,'*') (COL(1),A,COL(11),A,COL(100),A);
   PUT EDIT (INFOSEP) (COL(1),A);
   PUT SKIP;
   PUT SKIP;
   PUT FILE(S2KTEMP) EDIT (RECOUT_BUFFER)(COL(1),A(80));
 END READOUT;
```

```
/* GETNUM 2.4 */
/*********************************************************************
 *  DATE:  24 AUG 83                                                 *
 *  VERSION:  1.0                                                    *
 *  NAME:  GETNUM                                                    *
 *  MODULE NUMBER:  2.4                                              *
 *  FUNCTION:  FIND THE S2K COMPONENT ID FOR EACH INPUT RECORD IF    *
 *             ONE EXISTS.  IT WILL THEN PLACE THE NUMBER INTO THE   *
 *             OUTPUT RECORD BEING CREATED FOR USE IN THE IDMS       *
 *             SCHEMA TRANSLATION.                                   *
 *  INPUTS:  RECIN_BUFFER  - 160 CHARACTER INPUT BUFFER              *
 *           RECOUT_BUFFER - 80 CHARACTER OUTPUT BUFFER              *
 *           GETNUM_POS      STARTING POSISTION IN RECIN_BUFFER      *
 *                           FIXED BIN(15)                           *
 *  OUTPUTS: GETNUM_POS -  RETURNS THE POSISTION IN THE INPUT        *
 *                         BUFFER, ONE POSISTION PAST THE SYSTEM     *
 *                         SEPERATOR.                                *
 *           RECOUT_BUFFER - IS UPDATED TO CONTAIN THE NUMBER FOUND  *
 *  GLOBAL VARIABLES USED:                                           *
 *  GLOBAL VARIABLES CHANGED:                                        *
 *  GLOBAL TABLES USED:                                              *
 *  FILES READ:                                                      *
 *  FILES WRITTEN:                                                   *
 *  MODULES CALLED:  NONBLNK, FINDPAT, SUBSTR                        *
 *  CALLING MODULES:  RDSCHM                                         *
 *  AUTHOR:  CAPT. JERRY OWENS                                       *
 *  HISTORY:                                                         *
 *********************************************************************/
/*

*/
GETNUM: PROCEDURE (RECIN_BUFFER,RECOUT_BUFFER,GETNUM_POS) RETURNS
                  (FIXED BIN(15));
        DCL RECIN_BUFFER CHAR(160);
        DCL RECOUT_BUFFER CHAR(80);
        DCL GETNUM_POS FIXED BIN(15);
        DCL GETNUM_FRST FIXED BIN(15);
        DCL GETNUM_END FIXED BIN(15);
        DCL GETNUM_LNGTH FIXED BIN(15);
        DCL SYSTEM_SEP CHAR(20) VAR INIT ('* ');
        DCL ID_NUMBER CHAR(4);
/*
  ***********************************************************
  *   CHECK FOR BLANK LINE.  IF GETNUM_POS = 0 THEN THE     *
  *   CURRENT LINE IS BLANK.                                *
  ***********************************************************
*/
     GETNUM_POS = NONBLNK(RECIN_BUFFER,GETNUM_POS);
     IF GETNUM_POS > 0 THEN
       DO;
         GETNUM_FRST = GETNUM_POS;
```

```
/*
 ************************************************************
 *    CHECK FOR THE S2K SYSTEM SEPERATOR.  IF GETNUM_END = 0*
 *    THEN THERE IS NOT A SYSTEM SEPERATOR IN THE CURRENT   *
 *    LINE, THEREFORE IT IS INVALID.                        *
 ************************************************************
*/
        GETNUM_END = FINDPAT(RECIN_BUFFER,GETNUM_POS,SYSTEM_SEP);
        GETNUM_POS = GETNUM_END;
/*
  ***********************************************************
  *    GET THE CURRENT S2K COMPONENT NUMBER AND ITS LENGTH  *
  ***********************************************************
*/
        IF GETNUM_END > 0 THEN
          DO;
           GETNUM_POS = GETNUM_POS + 1;
           GETNUM_LNGTH = GETNUM_END - GETNUM_FRST;
           ID_NUMBER = SUBSTR(RECIN_BUFFER,GETNUM_FRST,GETNUM_LNGTH);
           SUBSTR(RECOUT_BUFFER,1,4) = ID_NUMBER;
          END;
      END;
    RETURN (GETNUM_POS);
END GETNUM;
```

```
/* GETNAME 2.5 */
/*********************************************************************
 *  DATE:  24 AUG 83                                                 *
 *  VERSION:  1.0                                                    *
 *  NAME:  GETNAME                                                   *
 *  MODULE NUMBER:  2.5                                              *
 *  FUNCTION:  FIND THE COMPONENT NAME OF EACH S2K COMPONENT STMT     *
 *             PASSED.  TRUNCATE THE NAME TO 50 CHARACTERS.          *
 *             REPLACE ALL IDMS ILLEGAL CHARACTERS WITH '-'.         *
 *  INPUTS:  RECIN_BUFFER  -  160 CHARACTER INPUT BUFFER             *
 *           RECOUT_BUFFER -  80 CHARACTER OUTPUT BUFFER             *
 *           GETNAME_POS   -  STARTING POSISTION IN THE INPUT        *
 *                            BUFFER. FIXED BIN(15)                  *
 *  OUTPUTS: GETNAME_POS   -  POSISTION IN THE INPUT BUFFER, ONE     *
 *                            PAST THE NAME.                         *
 *           RECOUT_BUFFER -  IS UPDATED TO CONTAIN THE NAME         *
 *  GLOBAL VARIABLES USED:                                           *
 *  GLOBAL VARIABLES CHANGED:                                        *
 *  GLOBAL TABLES USED:                                              *
 *  FILES READ:                                                      *
 *  FILES WRITTEN:                                                   *
 *  MODULES CALLED:  FINDPAT, NONBLNK, SUBSTR,TRANSLATE              *
 *  CALLING MODULES:  RDSCHM                                         *
 *  AUTHOR:  CAPT. JERRY OWENS                                       *
 *  HISTORY:                                                         *
 *********************************************************************/
/*

*/
GETNAME: PROCEDURE (RECIN_BUFFER,RECOUT_BUFFER,GETNAME_POS) RETURNS
                   (FIXED BIN(15));
       DCL RECIN_BUFFER CHAR(160);
       DCL RECOUT_BUFFER CHAR(80);
       DCL GETNAME_POS FIXED BIN(15);
       DCL GETNAME_START FIXED BIN(15);
       DCL GETNAME_END FIXED BIN(15);
       DCL GETNAME_LNGTH FIXED BIN(15);
       DCL S2K_NAME CHAR(50);
       DCL TEMP_NAME CHAR(50) VAR;
       DCL INVALID_CHR CHAR(21) INIT(' =<;:%>_+&:@!$#?/:.:,');
       DCL REPLACE_CHR CHAR(21) INIT('----------------------');
/*
  ******************************************************
  *  GET THE POSITION OF THE NEXT NON-BLANK CHARACTER  *
  ******************************************************
*/
       GETNAME_POS = NONBLNK(RECIN_BUFFER,GETNAME_POS);
       GETNAME_START = GETNAME_POS;
```

```
/*
  ******************************************************
  *   FIND THE S2K COMPONENT NAME                      *
  ******************************************************
*/
        GETNAME_END = FINDPAT(RECIN_BUFFER,GETNAME_POS,' (');
        GETNAME_POS = GETNAME_END;
        GETNAME_LNGTH = GETNAME_END - GETNAME_START;
        TEMP_NAME = SUBSTR(RECIN_BUFFER,GETNAME_START,GETNAME_LNGTH);
/*
  ******************************************************
  *    REPLACE ALL INVALID IDMS CHARACTERS WITH        *
  *    HYPHENS.                                        *
  ******************************************************
*/
        TEMP_NAME = TRANSLATE(TEMP_NAME,REPLACE_CHR,INVALID_CHR);
        S2K_NAME = TEMP_NAME;
        SUBSTR(RECOUT_BUFFER,10,50) = S2K_NAME;
        RETURN(GETNAME_POS);
END GETNAME;
```

```
/*  GETTYPE 2.6  */
/*********************************************************************
 *  DATE:  24 AUG 83                                                 *
 *  VERSION:  1.0                                                    *
 *  NAME:  GETTYPE                                                   *
 *  MODULE NUMBER:  2.6                                              *
 *  FUNCTION:  DETERMINE WHETHER THE PASSED S2K SCHEMA INPUT STMT    *
 *             IS A RECORD, INTGER, DECIMAL, CHARACTER, TEXT, MONEY,*
 *             DATE INPUT STATEMENT.                                 *
 *  INPUTS:  RECIN_BUFFER   - 160 CHARACTER INPUT BUFFER             *
 *           RECOUT_BUFFER  - 80 CHARACTER OUTPUT BUFFER             *
 *           GETTYPE_POS    - STARTING POSISTION IN THE INPUT        *
 *                            BUFFER. FIXED BIN(15)                 *
 *           REC_CNT        - NUMBER OF RECORDS PROCESSED            *
 *           ELEM_CNT       - NUMBER OF ELEMENTS PROCESSED           *
 *  OUTPUTS: RECOUT_BUFFER  - IS UPDATED TO CONTAIN THE TYPE INFO    *
 *           REC_CNT, ELEM_CNT                                       *
 *  GLOBAL VARIABLES USED:                                           *
 *  GLOBAL VARIABLES CHANGED:                                        *
 *  GLOBAL TABLES USED:                                              *
 *  FILES READ:                                                      *
 *  FILES WRITTEN:                                                   *
 *  MODULES CALLED:  FINDPAT, RECFND, INTFND, CHRFND, DATFND,        *
 *                   DECFND, MONEY                                   *
 *  CALLING MODULES:  RDSCHM                                         *
 *  AUTHOR:  CAPT. JERRY OWENS                                       *
 *  HISTORY:                                                         *
 *********************************************************************/
/*

*/
GETTYPE: PROCEDURE (RECIN_BUFFER,RECOUT_BUFFER,GETTYPE_POS,
                    REC_CNT,ELEM_CNT);
        DCL RECIN_BUFFER CHAR(160);
        DCL RECOUT_BUFFER CHAR(80);
        DCL GETTYPE_POS FIXED BIN(15);
        DCL REC_CNT FIXED BIN(15);
        DCL ELEM_CNT FIXED BIN(15);
        DCL REC_TYPE FIXED BIN(15);
/*
  ********************************************************
  *  IS THE CURRENT S2K COMPONENT A RECORD ?             *
  ********************************************************
*/
        REC_TYPE = FINDPAT(RECIN_BUFFER,GETTYPE_POS,'RECORD');
        IF REC_TYPE > 0 THEN DO;
          REC_CNT = REC_CNT + 1;
          CALL RECFND(RECIN_BUFFER,RECOUT_BUFFER,REC_TYPE);
          END;
            ELSE DO;
              ELEM_CNT = ELEM_CNT + 1;
/*
```

```
  ************************************************************
  *  IS THE CURRENT ITEM COMPONENT AN INTEGER ??       _    *
  ************************************************************
*/
            REC_TYPE = FINDPAT(RECIN_BUFFER,GETTYPE_POS,
                               'INTEGER NUMBER ');
             IF REC_TYPE > 0 THEN
               CALL INTFND(RECIN_BUFFER,RECOUT_BUFFER,REC_TYPE);
                  ELSE DO;
/*
  ************************************************************
  *  IS THE CURRENT COMPONENT A CHARACTER FORMAT ??          *
  ************************************************************
*/
                   REC_TYPE = FINDPAT(RECIN_BUFFER,GETTYPE_POS,
                                      'CHAR X');
                    IF REC_TYPE = 0 THEN
                      REC_TYPE = FINDPAT(RECIN_BUFFER,GETTYPE_POS,
                                         'TEXT X');
          IF REC_TYPE > 0 THEN
            CALL CHRFND(RECIN_BUFFER,RECOUT_BUFFER,REC_TYPE);
              ELSE DO;
/*
  ************************************************************
  *  IS THE CURRENT ITEM COMPONENT A DATE ???                *
  ************************************************************
*/
                REC_TYPE = FINDPAT(RECIN_BUFFER,GETTYPE_POS,
                                   'DATE');
                 IF REC_TYPE > 0 THEN
                   CALL DATFND(RECIN_BUFFER,RECOUT_BUFFER,REC_TYPE);
                ELSE DO;
/*
  ************************************************************
  *  IS THE CURRENT COMPONENT A DECIMAL NUMBER    ??         *
  ************************************************************
*/
                REC_TYPE = FINDPAT(RECIN_BUFFER,GETTYPE_POS,
                                   'DECIMAL NUMBER ');
                 IF REC_TYPE > 0 THEN
                   CALL DECFND(RECIN_BUFFER,RECOUT_BUFFER,REC_TYPE);
                ELSE DO;
/*
  ************************************************************
  *  IT MUST A MONEY COMPONENT.                              *
  ************************************************************
*/
                REC_TYPE = FINDPAT(RECIN_BUFFER,GETTYPE_POS,
                                   'MONEY $');
                  CALL MONEY(RECIN_BUFFER,RECOUT_BUFFER,REC_TYPE);
                END;
                 END;  /* DECIMAL NUMBER */
                     END;  /*  DATE  */
```

```
                END; /* CHAR OR TEXT */
            END; /* INTEGER NUMBER */
END GETTYPE;
```

```
/* RECFND 2.6.1 */
/*********************************************************************
 *  DATE:  30 AUG 83                                                 *
 *  VERSION:  1.0                                                    *
 *  NAME:  RECFND                                                    *
 *  MODULE NUMBER:  2.6.1                                            *
 *  FUNCTION:  GET THE NECESSARY INFO FOR AN IDMS RECORD             *
 *  INPUTS:  RECIN_BUFFER  -  160 CHARACTER INPUT BUFFER             *
 *           RECOUT_BUFFER -  80 CHARACTER OUTPUT BUFFER             *
 *           REC_POS       -  POSISTION TO START SEARCH              *
 *  OUTPUTS: RECOUT_BUFFER IS MODIFIED TO CONTAIN THE INFO           *
 *  GLOBAL VARIABLES USED:                                           *
 *  GLOBAL VARIABLES CHANGED:                                        *
 *  GLOBAL TABLES USED:                                              *
 *  FILES READ:                                                      *
 *  FILES WRITTEN:                                                   *
 *  MODULES CALLED:  SUBSTR, TRANSLATE, LENGTH, FINDPAT              *
 *  CALLING MODULES:  GETTYPE                                        *
 *  AUTHOR:  CAPT JERRY OWENS                                        *
 *  HISTORY:                                                         *
 *********************************************************************/
/*

*/
RECFND: PROCEDURE (RECIN_BUFFER,RECOUT_BUFFER,REC_POS);
        DCL RECIN_BUFFER CHAR(160);
        DCL RECOUT_BUFFER CHAR(80);
        DCL REC_TEMP FIXED BIN(15);
        DCL RECOWNER_LNGTH FIXED BIN(15);
        DCL RECOWNER_START FIXED BIN(15);
        DCL RECOWNER_END FIXED BIN(15);
        DCL REC_POS FIXED BIN(15);
        DCL RECFND_TYPE CHAR(4);
        DCL REC_OWNER CHAR(4);
        DCL REC_NAME CHAR(16);
          REC_NAME = SUBSTR(RECOUT_BUFFER,10,16);
/*
  ************************************************************
  *  INSURE THAT THE NAME DOES NOT END WITH A HYPHEN         *
  ************************************************************
*/
          SUBSTR(REC_NAME,16,1) =
            TRANSLATE(SUBSTR(REC_NAME,16,1),' ','-');
          RECFND_TYPE = 'REC ';
          REC_TEMP = REC_POS + LENGTH(RECFND_TYPE);
```

```
/*
  ***************************************************************
  *   IF REC_POS = 0 THEN THIS RECORD IS OWNED BY RECORD ZERO,  *
  *   OTHERWISE MUST SCAN THE REST OF THE INPUT STRING TO FIND  *
  *   WHO THE OWNER IS.                                         *
  ***************************************************************
*/
         REC_POS = FINDPAT(RECIN_BUFFER,REC_TEMP,'IN ');
         IF REC_POS = 0 THEN REC_OWNER = '0   ';
           ELSE DO;
             REC_TEMP = REC_POS + 3;
             RECOWNER_START = REC_TEMP;
             RECOWNER_END = FINDPAT(RECIN_BUFFER,REC_TEMP,') ');
             RECOWNER_LNGTH = RECOWNER_END - RECOWNER_START;
             REC_OWNER = SUBSTR(RECIN_BUFFER,RECOWNER_START,
                                RECOWNER_LNGTH);
           END;
         SUBSTR(RECOUT_BUFFER,10,16) = REC_NAME;
         SUBSTR(RECOUT_BUFFER,26,17) = '                 ';
         SUBSTR(RECOUT_BUFFER,43,4) = REC_OWNER;
         SUBSTR(RECOUT_BUFFER,47,1) = ' ';
         SUBSTR(RECOUT_BUFFER,48,4) = RECFND_TYPE;
         SUBSTR(RECOUT_BUFFER,52,29) = '';
END RECFND;
```

```
/*  INTFND 2.6.2  */
/*********************************************************************
 *  DATE:  30 AUG 83                                                 *
 *  VERSION: 1.0                                                     *
 *  NAME:  INTFND                                                    *
 *  MODULE NUMBER:  2.6.2                                            *
 *  FUNCTION:  FIND THE NECESSARY INFO FOR AN IDMS INTEGER ITEM.     *
 *  INPUTS:  RECIN_BUFFER  - 160 CHARACTER INPUT BUFFER              *
 *           RECOUT_BUFFER  - 80 CHARACTER OUTPUT BUFFER             *
 *           INTFND_POS     - STARTING POSISTION IN BUFFER           *
 *  OUTPUTS: RECOUT_BUFFER IS MODIFIED TO CONTAIN THE INFO           *
 *  GLOBAL VARIABLES USED:                                           *
 *  GLOBAL VARIABLES CHANGED:                                        *
 *  GLOBAL TABLES USED:                                              *
 *  FILES READ:                                                      *
 *  FILES WRITTEN:                                                   *
 *  MODULES CALLED:  SUBSTR, TRANSLATE, FINDPAT                      *
 *  CALLING MODULES:  GETTYPE                                        *
 *  AUTHOR:  CAPT JERRY OWENS                                        *
 *  HISTORY:                                                         *
 *********************************************************************/
/*

*/
INTFND: PROCEDURE (RECIN_BUFFER,RECOUT_BUFFER,INTFND_POS);
        DCL RECIN_BUFFER CHAR(160);
        DCL RECOUT_BUFFER CHAR(80);
        DCL INTFND_TEMP FIXED BIN(15);
        DCL INTFND_LAST FIXED BIN(15);
        DCL INTFND_FRST FIXED BIN(15);
        DCL INTFND_LNGTH FIXED BIN(15);
        DCL INTOWNER_LNGTH FIXED BIN(15);
        DCL INTOWNER_START FIXED BIN(15);
        DCL INTOWNER_END FIXED BIN(15);
        DCL INTFND_POS FIXED BIN(15);
        DCL INTFND_TYPE CHAR(4);
        DCL INTFND_OWNER CHAR(4);
        DCL INTFND_NAME CHAR(32);
        DCL INTFND_DESC CHAR(16);
          INTFND_OWNER = '0   ';
          INTFND_NAME = SUBSTR(RECOUT_BUFFER,10,32);
/*
  ********************************************************
  * MAKE SURE NAME DOES NOT END WITH A HYPHEN            *
  ********************************************************
*/
          SUBSTR(INTFND_NAME,32,1) =
            TRANSLATE(SUBSTR(INTFND_NAME,32,1),' ','-');
          INTFND_TYPE = 'PIC ';
          INTFND_TEMP = INTFND_POS + 15;
```

```
/*
  ********************************************
  *  IS DEFINITION IN THE FORM PIC 9(9...) ? *
  ********************************************
*/
INTFND_FRST = FINDPAT(RECIN_BUFFER,INTFND_TEMP,'9(');
IF INTFND_FRST > 0 THEN DO;
  INTFND_LAST = FINDPAT(RECIN_BUFFER,INTFND_FRST,')');
  INTFND_LNGTH = (INTFND_LAST - INTFND_FRST) + 1;
END;
  ELSE DO;
    /*
      ********************************
      *   MUST BE IN THE FORM OF 9999 *
      ********************************
    */
    INTFND_FRST = FINDPAT(RECIN_BUFFER,INTFND_TEMP,'9');
    INTFND_LAST = FINDPAT(RECIN_BUFFER,INTFND_FRST,'9)');
    IF INTFND_LAST > 0 THEN
      INTFND_LNGTH = (INTFND_LAST - INTFND_FRST) + 1;
       ELSE DO;
         INTFND_LAST = FINDPAT(RECIN_BUFFER,INTFND_FRST,
                               '9 ');
         INTFND_LNGTH = (INTFND_LAST - INTFND_FRST) + 1;
       END;
  END;
/*
  ********************************************
  *  DOES THIS ITEM BELON TO S2K RECORD ZERO ? *
  *  IF INTFND_TEMP > 0, THEN 'IN' WAS FOUND   *
  *  WHICH MEANS A RECORD OWNER WILL FOLLOW.   *
  ********************************************
*/
INTFND_TEMP = FINDPAT(RECIN_BUFFER,INTFND_LAST,'IN ');
IF INTFND_TEMP > 0 THEN DO;
  INTOWNER_START = INTFND_TEMP + 3;
  INTOWNER_END = FINDPAT(RECIN_BUFFER,INTFND_TEMP,
                         ' WITH ');
  IF INTOWNER_END = 0 THEN
    INTOWNER_END = FINDPAT(RECIN_BUFFER,INTFND_TEMP,') ');
    INTOWNER_LNGTH = INTOWNER_END - INTOWNER_START;
    INTFND_OWNER = SUBSTR(RECIN_BUFFER,INTOWNER_START,
                          INTOWNER_LNGTH);
END;
```

```
        INTFND_DESC = SUBSTR(RECIN_BUFFER,INTFND_FRST,INTFND_LNGTH);
        SUBSTR(RECOUT_BUFFER,10,32) = INTFND_NAME;
        SUBSTR(RECOUT_BUFFER,42,1) = ' ';
        SUBSTR(RECOUT_BUFFER,43,4) = INTFND_OWNER;
        SUBSTR(RECOUT_BUFFER,47,1) = ' ';
        SUBSTR(RECOUT_BUFFER,48,4) = INTFND_TYPE;
        SUBSTR(RECOUT_BUFFER,52,1) = ' ';
        SUBSTR(RECOUT_BUFFER,53,16) = INTFND_DESC;
        SUBSTR(RECOUT_BUFFER,69,1) = ' ';
        SUBSTR(RECOUT_BUFFER,70,6) = 'COMP-3';
        SUBSTR(RECOUT_BUFFER,76,5) = '     ';
END INTFND;
```

```
/* CHRFND 2.6.3 */
/*****************************************************************
 *  DATE:  30 AUG 83                                             *
 *  VERSION:  1.0                                                *
 *  NAME:  CHRFND                                                *
 *  MODULE NUMBER:  2.6.3                                        *
 *  FUNCTION:  GET THE NECESSARY INFO FOR AN IDMS CHARACTER ITEM. *
 *  INPUTS:  RECIN_BUFFER  - 160 CHARACTER INPUT BUFFER          *
 *           RECOUT_BUFFER -  80 CHARACTER OUTPUT BUFFER         *
 *           CHRFND_POS    -  STARTING POSISTION IN INPUT BUFFER *
 *  OUTPUTS:  RECOUT_BUFFER IS MODIFIED TO CONTAIN THE INFO      *
 *  GLOBAL VARIABLES USED:                                       *
 *  GLOBAL VARIABLES CHANGED:                                    *
 *  GLOBAL TABLES USED:                                          *
 *  FILES READ:                                                  *
 *  FILES WRITTEN:                                               *
 *  MODULES CALLED:  SUBSTR, TRANSLATE, FINDPAT                  *
 *  CALLING MODULES:  GETTYPE                                    *
 *  AUTHOR:  CAPT JERRY OWENS                                    *
 *  HISTORY:                                                     *
 *****************************************************************/
/*

*/
CHRFND: PROCEDURE (RECIN_BUFFER,RECOUT_BUFFER,CHRFND_POS);
        DCL RECIN_BUFFER CHAR(160);
        DCL RECOUT_BUFFER CHAR(80);
        DCL CHRFND_TEMP FIXED BIN(15);
        DCL CHRFND_LAST FIXED BIN(15);
        DCL CHRFND_FRST FIXED BIN(15);
        DCL CHRFND_LNGTH FIXED BIN(15);
        DCL CHROWNER_LNGTH FIXED BIN(15);
        DCL CHROWNER_START FIXED BIN(15);
        DCL CHROWNER_END FIXED BIN(15);
        DCL CHRFND_POS FIXED BIN(15);
        DCL CHRFND_TYPE CHAR(4);
        DCL CHRFND_OWNER CHAR(4);
        DCL CHRFND_NAME CHAR(32);
        DCL CHRFND_DESC CHAR(16);
          CHRFND_OWNER = '0   ';
          CHRFND_NAME = SUBSTR(RECOUT_BUFFER,10,32);
/*
  ********************************************************
  *  MAKE SURE NAME DOES NOT END WITH A HYPHEN           *
  ********************************************************
*/
          SUBSTR(CHRFND_NAME,32,1) =
            TRANSLATE(SUBSTR(CHRFND_NAME,32,1),' ','-');
          CHRFND_TYPE = 'PIC ';
          CHRFND_TEMP = CHRFND_POS + 5;
```

```
/*
  ************************************************************
  *  IS ITEM IN THE FORM X(X.........) ?                     *
  ************************************************************
*/
          CHRFND_FRST = FINDPAT(RECIN_BUFFER,CHRFND_TEMP,'X(');
          IF CHRFND_FRST > 0 THEN DO;
            CHRFND_LAST = FINDPAT(RECIN_BUFFER,CHRFND_FRST,')');
            CHRFND_LNGTH = (CHRFND_LAST - CHRFND_FRST) + 1;
          END;
            ELSE DO;
              /*
                ***********************************
                * ITEM IS IN THE FORM OF XXXXX *
                ***********************************
              */
              CHRFND_FRST = FINDPAT(RECIN_BUFFER,CHRFND_TEMP,'X');
              CHRFND_LAST = FINDPAT(RECIN_BUFFER,CHRFND_FRST,'X)');
              IF CHRFND_LAST > 0 THEN
                CHRFND_LNGTH = (CHRFND_LAST - CHRFND_FRST) + 1;
                  ELSE DO;
                    CHRFND_LAST = FINDPAT(RECIN_BUFFER,CHRFND_FRST,
                                          'X ');
                    CHRFND_LNGTH = (CHRFND_LAST - CHRFND_FRST) + 1;
                  END;
            END;
/*
  ************************************************************
  *  IS S2K RECORD ZERO THE OWNER OF THE ITEM ?              *
  ************************************************************
*/
          CHRFND_TEMP = FINDPAT(RECIN_BUFFER,CHRFND_LAST,'IN ');
          IF CHRFND_TEMP > 0 THEN DO;
            CHROWNER_START = CHRFND_TEMP + 3;
            CHROWNER_END = FINDPAT(RECIN_BUFFER,CHRFND_TEMP,
                                   ' WITH ');
            IF CHROWNER_END = 0 THEN
              CHROWNER_END = FINDPAT(RECIN_BUFFER,CHRFND_TEMP,') ');
              CHROWNER_LNGTH = CHROWNER_END - CHROWNER_START;
              CHRFND_OWNER = SUBSTR(RECIN_BUFFER,CHROWNER_START,
                                    CHROWNER_LNGTH);
          END;
          CHRFND_DESC = SUBSTR(RECIN_BUFFER,CHRFND_FRST,CHRFND_LNGTH);
          SUBSTR(RECOUT_BUFFER,10,32) = CHRFND_NAME;
          SUBSTR(RECOUT_BUFFER,42,1) = ' ';
          SUBSTR(RECOUT_BUFFER,43,4) = CHRFND_OWNER;
          SUBSTR(RECOUT_BUFFER,47,1) = ' ';
          SUBSTR(RECOUT_BUFFER,48,4) = CHRFND_TYPE;
          SUBSTR(RECOUT_BUFFER,52,1) = ' ';
          SUBSTR(RECOUT_BUFFER,53,16) = CHRFND_DESC;
          SUBSTR(RECOUT_BUFFER,69,12) = '           ';
END CHRFND;
```

```
/*  DATFND 2.6.4  */
/*********************************************************************
 *  DATE:  30 AUG 83                                                 *
 *  VERSION:  1.0                                                    *
 *  NAME:  DATFND                                                    *
 *  MODULE NUMBER:  2.6.4                                            *
 *  FUNCTION:  GET THE NECESSARY INFO FOR AN IDMS DATE ITEM          *
 *  INPUTS:  RECIN_BUFFER  -  160 CHARACTER INPUT BUFFER             *
 *           RECOUT_BUFFER  -  80 CHARACTER OUTPUT BUFFER            *
 *           DATFND_POS     -  POSISTION TO START IN BUFFER          *
 *  OUTPUTS: RECOUT_BUFFER IS MODIFIED TO CONTAIN THE INFO           *
 *  GLOBAL VARIABLES USED:                                           *
 *  GLOBAL VARIABLES CHANGED:                                        *
 *  GLOBAL TABLES USED:                                              *
 *  FILES READ:                                                      *
 *  FILES WRITTEN:                                                   *
 *  MODULES CALLED:  SUBSTR, TRANSLATE, FINDPAT                      *
 *  CALLING MODULES:  GETTYPE                                        *
 *  AUTHOR:  CAPT JERRY OWENS                                        *
 *  HISTORY:                                                         *
 *********************************************************************/
/*

*/
DATFND: PROCEDURE (RECIN_BUFFER,RECOUT_BUFFER,DATFND_POS);
        DCL RECIN_BUFFER CHAR(160);
        DCL RECOUT_BUFFER CHAR(80);
        DCL DATFND_TEMP FIXED BIN(15);
        DCL DATOWNER_LNGTH FIXED BIN(15);
        DCL DATOWNER_START FIXED BIN(15);
        DCL DATOWNER_END FIXED BIN(15);
        DCL DATFND_POS FIXED BIN(15);
        DCL DATFND_TYPE CHAR(4);
        DCL DATFND_OWNER CHAR(4);
        DCL DATFND_NAME CHAR(32);
        DCL DATFND_DESC CHAR(16);
          DATFND_OWNER = '0   ';
          DATFND_NAME = SUBSTR(RECOUT_BUFFER,10,32);
/*
  ********************************************************
  *  MAKE SURE DOES NOT END WITH A HYPHEN                *
  ********************************************************
*/
          SUBSTR(DATFND_NAME,32,1) =
            TRANSLATE(SUBSTR(DATFND_NAME,32,1),' ','-');
          DATFND_TYPE = 'PIC ';
          DATFND_TEMP = DATFND_POS + 4;
```

```
/*
 ************************************************************
 *  IS THIS ITEM OWNED BY S2K RECORD ZERO ??                *
 ************************************************************
*/
        DATFND_TEMP = FINDPAT(RECIN_BUFFER,DATFND_TEMP,'IN ');
        IF DATFND_TEMP > 0 THEN DO;
          DATOWNER_START = DATFND_TEMP + 3;
          DATOWNER_END = FINDPAT(RECIN_BUFFER,DATFND_TEMP,
                                 ' WITH ');
          IF DATOWNER_END = 0 THEN
            DATOWNER_END = FINDPAT(RECIN_BUFFER,DATFND_TEMP,') ');
            DATOWNER_LNGTH = DATOWNER_END - DATOWNER_START;
            DATFND_OWNER = SUBSTR(RECIN_BUFFER,DATOWNER_START,
                                  DATOWNER_LNGTH);
        END;
        SUBSTR(RECOUT_BUFFER,10,32) = DATFND_NAME;
        SUBSTR(RECOUT_BUFFER,42,1) = ' ';
        SUBSTR(RECOUT_BUFFER,43,4) = DATFND_OWNER;
        SUBSTR(RECOUT_BUFFER,47,1) = ' ';
        SUBSTR(RECOUT_BUFFER,48,4) = DATFND_TYPE;
        SUBSTR(RECOUT_BUFFER,52,1) = ' ';
        SUBSTR(RECOUT_BUFFER,53,16) = '9(7)           ';
        SUBSTR(RECOUT_BUFFER,69,1) = ' ';
        SUBSTR(RECOUT_BUFFER,70,6) = 'COMP-3';
        SUBSTR(RECOUT_BUFFER,76,5) = '     ';
END DATFND;
```

```
/*  DECFND 2.6.5  */
/*********************************************************************
 *  DATE:  30 AUG 83                                                 *
 *  VERSION:  1.0                                                    *
 *  NAME:  DECFND                                                    *
 *  MODULE NUMBER:  2.6.5                                            *
 *  FUNCTION:  GET THE NECESSARY INFO FOR AN IDMS DECIMAL ITEM       *
 *  INPUTS:  RECIN_BUFFER  -  160 CHARACTER INPUT BUFFER             *
 *           RECOUT_BUFFER  -  80 CHARACTER OUTPUT BUFFER            *
 *           DECFND_POS     -  STARTING POSISTION IN INPUT BUFFER    *
 *  OUTPUTS: RECOUT_BUFFER IS MODIFIED TO CONTAIN THE INFO           *
 *  GLOBAL VARIABLES USED:                                           *
 *  GLOBAL VARIABLES CHANGED:                                        *
 *  GLOBAL TABLES USED:                                              *
 *  FILES READ:                                                      *
 *  FILES WRITTEN:                                                   *
 *  MODULES CALLED:  SUBSTR, TRANSLATE, FINDPAT                      *
 *  CALLING MODULES:  GETTYPE                                        *
 *  AUTHOR:  CAPT JERRY OWENS                                        *
 *  HISTORY:                                                         *
 *********************************************************************/
/*       .

*/
DECFND: PROCEDURE (RECIN_BUFFER,RECOUT_BUFFER,DECFND_POS);
        DCL RECIN_BUFFER CHAR(160);
        DCL RECOUT_BUFFER CHAR(80);
        DCL DECFND_TEMP FIXED BIN(15);
        DCL DECFND_LAST FIXED BIN(15);
        DCL DECFND_FRST FIXED BIN(15);
        DCL DECFND_LNGTH FIXED BIN(15);
        DCL DECOWNER_LNGTH FIXED BIN(15);
        DCL DECOWNER_START FIXED BIN(15);
        DCL DECOWNER_END FIXED BIN(15);
        DCL DECFND_POS FIXED BIN(15);
        DCL DECFND_TYPE CHAR(4);
        DCL DECFND_OWNER CHAR(4);
        DCL DECFND_NAME CHAR(32);
        DCL DECFND_DESC CHAR(16);
          DECFND_OWNER = '0   ';
          DECFND_NAME = SUBSTR(RECOUT_BUFFER,10,32);
/*
  ********************************************************
  *  INSURE NAME DOES NOT END WITH A HYPHEN              *
  ********************************************************
*/
          SUBSTR(DECFND_NAME,32,1) =
            TRANSLATE(SUBSTR(DECFND_NAME,32,1),' ','-');
          DECFND_TYPE = 'PIC ';
          DECFND_TEMP = DECFND_POS + 15;
          DECFND_FRST = FINDPAT(RECIN_BUFFER,DECFND_TEMP,'9');
          DECFND_LAST = FINDPAT(RECIN_BUFFER,DECFND_FRST,'9 ');
```

```
/*
  ************************************************************
  * IS THE DESCRIPTION IN THE FORM 999.99  ???                *
  ************************************************************
*/
        IF DECFND_LAST > 0 THEN
          DECFND_LNGTH = (DECFND_LAST - DECFND_FRST) + 1;
            ELSE DO;
/*
  ************************************************************
  * NOPE IS IT THE FORM 999.99)    ?????                      *
  ************************************************************
*/
              DECFND_LAST = FINDPAT(RECIN_BUFFER,DECFND_FRST,'9)');
                IF DECFND_LAST > 0 THEN
                  DECFND_LNGTH = (DECFND_LAST - DECFND_FRST) + 1;
                    ELSE DO;
/*
  ************************************************************
  * NOPE IS IT THE FORM 9(X).9(X)) OR 99.9(X)) ?????          *
  ************************************************************
*/
                      DECFND_LAST = FINDPAT(RECIN_BUFFER,DECFND_FRST,
                                            '))');
                        IF DECFND_LAST > 0 THEN
                          DECFND_LNGTH = (DECFND_LAST - DECFND_FRST)
                                         + 1;
                            ELSE DO;
/*
  ************************************************************
  *  IT MUST BE 9(X).9(X) OR 99.9(X)    ....                  *
  ************************************************************
*/
                              DECFND_LAST = FINDPAT(RECIN_BUFFER,
                                                    DECFND_FRST,') ');
                              DECFND_LNGTH = (DECFND_LAST -
                                              DECFND_FRST) + 1;
                            END;
                      END;
            END;
        DECFND_TEMP = FINDPAT(RECIN_BUFFER,DECFND_LAST,'IN ');
/*
  ************************************************************
  *  IS THIS ITEM OWNED BY RECORD ZERO ?????                  *
  ************************************************************
*/
        IF DECFND_TEMP > 0 THEN DO;
          DECOWNER_START = DECFND_TEMP + 3;
          DECOWNER_END = FINDPAT(RECIN_BUFFER,DECFND_TEMP,
                                 ' WITH ');
          IF DECOWNER_END = 0 THEN
            DECOWNER_END = FINDPAT(RECIN_BUFFER,DECFND_TEMP,') ');
            DECOWNER_LNGTH = DECOWNER_END - DECOWNER_START;
```

```
            DECFND_OWNER = SUBSTR(RECIN_BUFFER,DECOWNER_START,
                                  DECOWNER_LNGTH);
        END;
        DECFND_DESC = SUBSTR(RECIN_BUFFER,DECFND_FRST,DECFND_LNGTH);
        DECFND_DESC = TRANSLATE(DECFND_DESC,'V','.');
        SUBSTR(RECOUT_BUFFER,10,32) = DECFND_NAME;
        SUBSTR(RECOUT_BUFFER,42,1) = ' ';
        SUBSTR(RECOUT_BUFFER,43,4) = DECFND_OWNER;
        SUBSTR(RECOUT_BUFFER,47,1) = ' ';
        SUBSTR(RECOUT_BUFFER,48,4) = DECFND_TYPE;
        SUBSTR(RECOUT_BUFFER,52,1) = ' ';
        SUBSTR(RECOUT_BUFFER,53,16) = DECFND_DESC;
        SUBSTR(RECOUT_BUFFER,69,1) = ' ';
        SUBSTR(RECOUT_BUFFER,70,6) = 'COMP-3';
        SUBSTR(RECOUT_BUFFER,76,5) = '     ';
END DECFND;
```

```
/*  MONEY 2.6.6  */
/*********************************************************************
 *  DATE:  30 AUG 83                                                 *
 *  VERSION:  1.0                                                    *
 *  NAME:  MONEY                                                     *
 *  MODULE NUMBER:  2.6.6                                            *
 *  FUNCTION:  GET THE NECESSARY INFO FROM AN S2K MONEY ITEM.        *
 *  INPUTS:  RECIN_BUFFER  -  160 CHARACTER INPUT BUFFER             *
 *           RECOUT_BUFFER  -  80 CHARACTER OUTPUT BUFFER            *
 *           MONEY_POS      -  STARTING POSISTION IN THE BUFFER      *
 *  OUTPUTS:  RECOUT_BUFFER IS MODIFIED TO CONTAIN THE INFO          *
 *  GLOBAL VARIABLES USED:                                           *
 *  GLOBAL VARIABLES CHANGED:                                        *
 *  GLOBAL TABLES USED:                                              *
 *  FILES READ:                                                      *
 *  FILES WRITTEN:                                                   *
 *  MODULES CALLED:  SUBSTR, TRANSLATE, FINDPAT                      *
 *  CALLING MODULES:  GETTYPE                                        *
 *  AUTHOR:  CAPT JERRY OWENS                                        *
 *  HISTORY:                                                         *
 *********************************************************************/
/*      .

*/
MONEY: PROCEDURE (RECIN_BUFFER,RECOUT_BUFFER,MONEY_POS);
       DCL RECIN_BUFFER CHAR(160);
       DCL RECOUT_BUFFER CHAR(80);
       DCL MONEY_TEMP FIXED BIN(15);
       DCL MONEY_LAST FIXED BIN(15);
       DCL MONEY_FRST FIXED BIN(15);
       DCL MONEY_LNGTH FIXED BIN(15);
       DCL MONOWNER_LNGTH FIXED BIN(15);
       DCL MONOWNER_START FIXED BIN(15);
       DCL MONOWNER_END FIXED BIN(15);
       DCL MONEY_POS FIXED BIN(15);
       DCL MONEY_TYPE CHAR(4);
       DCL MONEY_OWNER CHAR(4);
       DCL MONEY_NAME CHAR(32);
       DCL MONEY_DESC CHAR(16);
         MONEY_OWNER = '0   ';
         MONEY_NAME = SUBSTR(RECOUT_BUFFER,10,32);
/*
  ********************************************************
  *  INSURE NAME DOES NOT END WITH A HYPHEN .....        *
  ********************************************************
*/
         SUBSTR(MONEY_NAME,32,1) =
           TRANSLATE(SUBSTR(MONEY_NAME,32,1),' ','-');
         MONEY_TYPE = 'PIC ';
         MONEY_TEMP = MONEY_POS + 7;
         MONEY_FRST = FINDPAT(RECIN_BUFFER,MONEY_TEMP,'9');
```

```
/*
  ********************************************************
  * IN THE FORM OF $99999.99    ??                       *
  ********************************************************
*/
        MONEY_LAST = FINDPAT(RECIN_BUFFER,MONEY_FRST,'9 ');
        IF MONEY_LAST > 0 THEN
          MONEY_LNGTH = (MONEY_LAST - MONEY_FRST) + 1;
            ELSE DO;
/*
  ********************************************************
  * IN THE FORM $99999.999) ????                         *
  ********************************************************
*/
              MONEY_LAST = FINDPAT(RECIN_BUFFER,MONEY_FRST,'9)');
                IF MONEY_LAST > 0 THEN
                  MONEY_LNGTH = (MONEY_LAST - MONEY_FRST) + 1;
                    ELSE DO;
/*
  ********************************************************
  * IN THE FORM $9(2).9(2)) OR $99.9(2))  ???            *
  ********************************************************
*/
                      MONEY_LAST = FINDPAT(RECIN_BUFFER,MONEY_FRST,
                                           '))');
                        IF MONEY_LAST > 0 THEN
                          MONEY_LNGTH = (MONEY_LAST - MONEY_FRST)
                                        + 1;
                            ELSE DO;
/*
  ********************************************************
  * MUST BE $9(5).9(2)   OR $999.9(5)                    *
  ********************************************************
*/
                              MONEY_LAST = FINDPAT(RECIN_BUFFER,
                                                   MONEY_FRST,') ');
                            MONEY_LNGTH = (MONEY_LAST - MONEY_FRST)
                                          + 1;
                            END;
                      END;
            END;
        MONEY_TEMP = FINDPAT(RECIN_BUFFER,MONEY_LAST,'IN ');
/*
  ********************************************************
  * DOES S2K RECORD ZERO OWN THIS ITEM  ????             *
  ********************************************************
*/
        IF MONEY_TEMP > 0 THEN DO;
          MONOWNER_START = MONEY_TEMP + 3;
          MONOWNER_END = FINDPAT(RECIN_BUFFER,MONEY_TEMP,
                                 ' WITH ');
          IF MONOWNER_END = 0 THEN
            MONOWNER_END = FINDPAT(RECIN_BUFFER,MONEY_TEMP,') ');
```

```
            MONOWNER_LNGTH = MONOWNER_END - MONOWNER_START;
            MONEY_OWNER = SUBSTR(RECIN_BUFFER,MONOWNER_START,
                                 MONOWNER_LNGTH);
        END;
        MONEY_DESC = SUBSTR(RECIN_BUFFER,MONEY_FRST,MONEY_LNGTH);
        MONEY_DESC = TRANSLATE(MONEY_DESC,'V','.');
        SUBSTR(RECOUT_BUFFER,10,32) = MONEY_NAME;
        SUBSTR(RECOUT_BUFFER,42,1) = ' ';
        SUBSTR(RECOUT_BUFFER,43,4) = MONEY_OWNER;
        SUBSTR(RECOUT_BUFFER,47,1) = ' ';
        SUBSTR(RECOUT_BUFFER,48,4) = MONEY_TYPE;
        SUBSTR(RECOUT_BUFFER,52,1) = ' ';
        SUBSTR(RECOUT_BUFFER,53,16) = MONEY_DESC;
        SUBSTR(RECOUT_BUFFER,69,1) = ' ';
        SUBSTR(RECOUT_BUFFER,70,6) = 'COMP-3';
        SUBSTR(RECOUT_BUFFER,76,5) = '     ';
END MONEY;
```

```
/*  SCHMBL 2.7  */
/*****************************************************************
 *  DATE: 3 SEP 83                                               *
 *  VERSION: 1.3                                                 *
 *  NAME:  SCHMBL                                                *
 *  MODULE NUMBER: 2.7                                           *
 *  FUNCTION: CALL THE APPROPIATE MODULES TO BUILD THE IDMSMDL    *
 *           FILE.                                               *
 *  INPUTS: REC_CNT, ELEM_CNT                                    *
 *  OUTPUTS:                                                     *
 *  GLOBAL VARIABLES USED:                                       *
 *  GLOBAL VARIABLES CHANGED:                                    *
 *  GLOBAL TABLES USED:                                          *
 *  FILES READ:                                                  *
 *  FILES WRITTEN:                                               *
 *  MODULES CALLED: GETINFO, BILDLOG                             *
 *  CALLING MODULES: RDSCHM                                      *
 *  AUTHOR: CAPT. JERRY OWENS                                    *
 *  HISTORY:                                                     *
 *****************************************************************/
/*

*/
SCHMBL: PROCEDURE (REC_CNT,ELEM_CNT);
   DCL REC_CNT FIXED BIN(15);
   DCL ELEM_CNT FIXED BIN(15);
   DCL CNTR FIXED BIN(15);
   DCL 1 REC_DESC (REC_CNT) CONTROLLED,
         2  IDMS_NAME CHAR(16) ,
         2  IDMS_NUM FIXED BIN(15) ,
         2  S2K_NUM CHAR(4),
         2  OWNER_S2K CHAR(4),
         2  FRST_PTR FIXED BIN(15);
   DCL 1 ELEM_DESC (ELEM_CNT) CONTROLLED,
         2  S2KREF_NUM CHAR(4),
         2  ELEM_NAME CHAR(32),
         2  ELEM_PIC CHAR(28),
         2  ELEM_MEMBER CHAR(4);
   ALLOCATE REC_DESC;
   ALLOCATE ELEM_DESC;
   CNTR = REC_CNT + ELEM_CNT;
   CALL GETINFO (REC_DESC,ELEM_DESC,CNTR);
   CALL BILDLOG (REC_DESC,ELEM_DESC,REC_CNT,ELEM_CNT);
END SCHMBL;
```

```
/*  GETINFO 2.7.1 */
/*********************************************************************
 *  DATE:  3 SEP 83                                                  *
 *  VERSION: 1.1                                                     *
 *  NAME:  GETINFO                                                   *
 *  MODULE NUMBER: 2.7.1                                             *
 *  FUNCTION: READ THE S2KTEMP FILE INTO MEMORY FPR FUTURE           *
 *           PROCESS'NG.                                             *
 *  INPUTS: REC_DESC, ELEM_DESC, CNTR                                *
 *  OUTPUTS: REC_DESC, ELEM_DESC                                     *
 *  GLOBAL VARIABLES USED:                                           *
 *  GLOBAL VARIABLES CHANGED:                                        *
 *  GLOBAL TABLES USED:                                              *
 *  FILES READ: S2KTEMP                                              *
 *  FILES WRITTEN:                                                   *
 *  MODULES CALLED:                                                  *
 *  CALLING MODULES: SCHMBL                                          *
 *  AUTHOR: CAPT. JERRY OWENS                                        *
 *  HISTORY:                                                         *
 *********************************************************************/
/*

*/
GETINFO: PROCEDURE(REC_DESC,ELEM_DESC,CNTR);
     DCL 1 REC_DESC (*),
             2  IDMS_NAME CHAR(16) ,
             2  IDMS_NUM FIXED BIN(15) ,
             2  S2K_NUM CHAR(4),
             2  OWNER_S2K CHAR(4),
             2  FRST_PTR FIXED BIN(15);
     DCL 1 ELEM_DESC (*),
             2  S2KREF_NUM CHAR(4),
             2  ELEM_NAME CHAR(32),
             2  ELEM_PIC CHAR(28),
             2  ELEM_MEMBER CHAR(4);
     DCL ECNT FIXED BIN(15) INIT(1);
     DCL RCNT FIXED BIN(15) INIT(1);
     DCL CNTR FIXED BIN(15);
     DCL I,J FIXED BIN(15);
     DCL STRT_NUM FIXED BIN(15);
     DCL INC_NUM FIXED BIN(15);
     DCL TEMP_REC CHAR(80);
/*
  ********************************************************
  * THIS INFO CAME FROM CLIST .....                      *
  ********************************************************
*/
   GET LIST (STRT_NUM,INC_NUM);
   CLOSE FILE(S2KTEMP);
   OPEN FILE(S2KTEMP) INPUT;
   DO I = 1 TO CNTR;
     GET FILE(S2KTEMP) EDIT (TEMP_REC)(COL(1),A(80));
```

```
/*
  ********************************************************
  * IS THIS INFORMATION FOR A RECORD OR AN ITEM ????     *
  ********************************************************
*/
    IF SUBSTR(TEMP_REC,48,4) = 'REC ' THEN DO;
      REC_DESC(RCNT).IDMS_NAME = SUBSTR(TEMP_REC,10,16);
      REC_DESC(RCNT).S2K_NUM = SUBSTR(TEMP_REC,1,4);
      REC_DESC(RCNT).OWNER_S2K = SUBSTR(TEMP_REC,43,4);
      REC_DESC(RCNT).IDMS_NUM = STRT_NUM;
      REC_DESC(RCNT).FRST_PTR = 0;
      RCNT = RCNT + 1;
      STRT_NUM = STRT_NUM + INC_NUM;
    END;
        ELSE DO;
          ELEM_DESC(ECNT).S2KREF_NUM = SUBSTR(TEMP_REC,1,4);
          ELEM_DESC(ECNT).ELEM_NAME = SUBSTR(TEMP_REC,10,32);
          ELEM_DESC(ECNT).ELEM_MEMBER = SUBSTR(TEMP_REC,43,4);
          ELEM_DESC(ECNT).ELEM_PIC = SUBSTR(TEMP_REC,48,28);
          DO J = 1 TO RCNT;
            IF (REC_DESC(J).S2K_NUM =
                 ELEM_DESC(ECNT).ELEM_MEMBER) THEN
              DO;
                IF REC_DESC(J).FRST_PTR = 0 THEN
                  REC_DESC(J).FRST_PTR = ECNT;
              END;
          END;
          ECNT = ECNT + 1;
        END;
  END;
END GETINFO;
```

```
/* BILDLOG 2.7.2 */
/*****************************************************************
 *  DATE: 3 SEP 83                                               *
 *  VERSION: 1.3                                                 *
 *  NAME:  BILDLOG                                               *
 *  MODULE NUMBER: 2.7.2                                         *
 *  FUNCTION: WRITE RECORD DESCRIPTION SECTION TO IDMSMDL FILE.  *
 *  INPUTS: REC_DESC, ELEM_DESC, REC_CNT, ELEM_CNT               *
 *  OUTPUTS:                                                     *
 *  GLOBAL VARIABLES USED:                                       *
 *  GLOBAL VARIABLES CHANGED:                                    *
 *  GLOBAL TABLES USED:                                          *
 *  FILES READ:                                                  *
 *  FILES WRITTEN: IDMSMDL                                       *
 *  MODULES CALLED: GETMODE, RECDUM                              *
 *  CALLING MODULES:                                             *
 *  AUTHOR:  CAPT. JERRY OWENS                                   *
 *  HISTORY:                                                     *
 *****************************************************************/
/*


*/
BILDLOG: PROCEDURE(REC_DESC,ELEM_DESC,REC_CNT,ELEM_CNT);
     DCL 1 REC_DESC (*),
             2  IDMS_NAME CHAR(16) ,
             2  IDMS_NUM FIXED BIN(15) ,
             2  S2K_NUM CHAR(4),
             2  OWNER_S2K CHAR(4),
             2  FRST_PTR FIXED BIN(15);
     DCL 1 ELEM_DESC (*),
             2  S2KREF_NUM CHAR(4),
             2  ELEM_NAME CHAR(32),
             2  ELEM_PIC CHAR(28),
             2  ELEM_MEMBER CHAR(4);
     DCL REC_CNT FIXED BIN(15);
     DCL ELEM_CNT FIXED BIN(15);
     DCL I,J FIXED BIN(15);
     DCL DUMNAME CHAR(22);
     DCL SET_CNT FIXED BIN(15) INIT(1);
    PUT FILE (IDMSMDL) EDIT (REC_CNT,ELEM_CNT)
                            (COL(1),F(4),COL(5),F(4));
    DO I = 1 TO REC_CNT;
      PUT FILE (IDMSMDL) EDIT
            ('R',REC_DESC(I).IDMS_NAME,REC_DESC(I).IDMS_NUM,
             REC_DESC(I).S2K_NUM,REC_DESC(I).OWNER_S2K,
             REC_DESC(I).FRST_PTR)
            (COL(1),A,COL(3),A,COL(20),F(4),COL(25),A,COL(30),A,
             COL(35),F(4));
      CALL GETMODE(REC_DESC,ELEM_DESC,SET_CNT,I);
```

```
/*
  ***********************************************************
  *  DOES THIS RECORD HAVE ANY ITEMS IN IT ???              *
  ***********************************************************
*/
      IF REC_DESC(I).FRST_PTR = 0 THEN DO;
        CALL RECDUM(REC_DESC(I).IDMS_NAME,DUMNAME);
        PUT FILE (IDMSMDL) EDIT
              ('I',DUMNAME,'PIC X(4)',REC_DESC(I).S2K_NUM)
              (COL(1),A,COL(8),A,COL(41),A,COL(70),A);
      END;
        ELSE DO;
/*
  ***********************************************************
  * GET ALL OF THE ITEMS FOR THE CURRENT RECORD ....        *
  ***********************************************************
*/
          DO J = 1 TO ELEM_CNT;
            IF REC_DESC(I).S2K_NUM = ELEM_DESC(J).ELEM_MEMBER THEN
              PUT FILE (IDMSMDL) EDIT
                     ('I',ELEM_DESC(J).S2KREF_NUM,
                      ELEM_DESC(J).ELEM_NAME,ELEM_DESC(J).ELEM_PIC,
                      ELEM_DESC(J).ELEM_MEMBER)(COL(1),A,COL(3),A,
                      COL(8),A,COL(41),A,COL(70),A);
          END;
        END;
    END;
 END BILDLOG;
```

```
/*  GETMODE 2.7.2.1  */
/*********************************************************************
 *  DATE: 4 SEP 83                                                   *
 *  VERSION: 1.1                                                     *
 *  NAME:  GETMODE                                                   *
 *  MODULE NUMBER: 2.7.2.1                                           *
 *  FUNCTION: DECIDE THE LOCATION MODE FOR A RECORD AND WRITES IT    *
 *           TO THE IDMSMDL FILE.                                    *
 *  INPUTS: REC_DESC, ELEM_DESC, SET_CNT, REC_NUM                    *
 *  OUTPUTS: SET_CNT                                                 *
 *  GLOBAL VARIABLES USED:                                           *
 *  GLOBAL VARIABLES CHANGED:                                        *
 *  GLOBAL TABLES USED:                                              *
 *  FILES READ:                                                      *
 *  FILES WRITTEN: IDMSMDL                                           *
 *  MODULES CALLED: CALCREC                                          *
 *  CALLING MODULES: BILDLOG                                         *
 *  AUTHOR: CAPT. JERRY OWENS                                        *
 *  HISTORY:                                                         *
 *********************************************************************/
/*

*/
GETMODE: PROCEDURE(REC_DESC,ELEM_DESC,SET_CNT,REC_NUM);
   DCL 1 REC_DESC (*),
         2  IDMS_NAME CHAR(16) ,
         2  IDMS_NUM FIXED BIN(15) ,
         2  S2K_NUM CHAR(4),
         2  OWNER_S2K CHAR(4),
         2  FRST_PTR FIXED BIN(15);
   DCL 1 ELEM_DESC (*),
         2  S2KREF_NUM CHAR(4),
         2  ELEM_NAME CHAR(32),
         2  ELEM_PIC CHAR(28),
         2  ELEM_MEMBER CHAR(4);
   DCL SET_CNT FIXED BIN(15);
   DCL REC_NUM FIXED BIN(15);
/*
  ***********************************************************
  * THIS IS RECORD S2K RECORD ZERO, MAKE IT CALC ..         *
  ***********************************************************
*/
IF REC_DESC(REC_NUM).OWNER_S2K = '    ' THEN
  PUT FILE (IDMSMDL) EDIT
          ('C',ELEM_DESC(REC_DESC(REC_NUM).FRST_PTR).ELEM_NAME)
          (COL(1),A,COL(5),A);
    ELSE DO;
/*
  ***********************************************************
  * MAKE IT VIA WITH A UNIQUE SET NAME .....                *
  ***********************************************************
*/
      IF SET_CNT < 10 THEN
```

```
            PUT FILE (IDMSMDL) EDIT ('V','RELATION-',SET_CNT)
                    (COL(1),A,COL(5),A,COL(14),F(1));
              ELSE IF SET_CNT < 100 THEN
                PUT FILE (IDMSMDL) EDIT ('V','RELATION-',SET_CNT)
                        (COL(1),A,COL(5),A,COL(14),F(2));
                  ELSE IF SET_CNT < 1000 THEN
                    PUT FILE (IDMSMDL) EDIT ('V','RELATION-',SET_CNT)
                            (COL(1),A,COL(5),A,COL(14),F(3));
          SET_CNT = SET_CNT + 1;
          CALL CALCREC(REC_DESC,REC_NUM);
        END;
    END GETMODE;
```

```
/*  CALCREC 2.7.2.1.1  */
/*********************************************************************
 *  DATE: 4 SEP 83                                                   *
 *  VERSION: 1.0                                                     *
 *  NAME:  CALCREC                                                   *
 *  MODULE NUMBER: 2.7.2.1.1                                         *
 *  FUNCTION: FINDS THE OWNER AND MEMBER OF A GIVEN SET RELATION.    *
 *            THEN WRITES THEM TO THE IDMSMDL FILE.                  *
 *  INPUTS: REC_DESC, REC_NUM                                        *
 *  OUTPUTS:                                                         *
 *  GLOBAL VARIABLES USED:                                           *
 *  GLOBAL VARIABLES CHANGED:                                        *
 *  GLOBAL TABLES USED:                                              *
 *  FILES READ:                                                      *
 *  FILES WRITTEN: IDMSMDL                                           *
 *  MODULES CALLED:                                                  *
 *  CALLING MODULES: GETMODE                                         *
 *  AUTHOR: CAPT. JERRY OWENS                                        *
 *  HISTORY:                                                         *
 *********************************************************************/
/*

*/
CALCREC: PROCEDURE (REC_DESC,REC_NUM);
   DCL 1 REC_DESC (*),
         2  IDMS_NAME CHAR(16) ,
         2  IDMS_NUM FIXED BIN(15) ,
         2  S2K_NUM CHAR(4),
         2  OWNER_S2K CHAR(4),
         2  FRST_PTR FIXED BIN(15);
   DCL REC_NUM FIXED BIN(15);
   DCL I FIXED BIN(15)INIT (1);
   DCL REC_FLAG FIXED BIN(15) INIT (0);
   DCL OWNER_NAME CHAR(16);
   DCL MEMBER_NAME CHAR(16);
/*
  ***********************************************************
  * THE CURRENT RECORD IS THE MEMBER OF THE CURRENT SET     *
  ***********************************************************
*/
MEMBER_NAME = REC_DESC(REC_NUM).IDMS_NAME;
/*
  ***********************************************************
  *  FIND THE OWNER OF THE CURRENT SET  OR MEMBER ...       *
  ***********************************************************
*/
DO WHILE (REC_FLAG = 0);
  IF REC_DESC(REC_NUM).OWNER_S2K = REC_DESC(I).S2K_NUM THEN
    DO;
      OWNER_NAME = REC_DESC(I).IDMS_NAME;
      REC_FLAG = 1;
    END;
      ELSE I = I + 1;
```

```
  END;
  PUT FILE (IDMSMDL) EDIT (OWNER_NAME,MEMBER_NAME)
                          (COL(25),A,COL(45),A);
END CALCREC;
```

```
/*  RECDUM 2.7.2.2  */
/*********************************************************************
 *  DATE: 4 SEP 83                                                   *
 *  VERSION: 1.0                                                     *
 *  NAME:  RECDUM                                                    *
 *  MODULE NUMBER: 2.7.2.2                                           *
 *  FUNCTION: BUILD AN ITEM NAME FOR A DUMMY RECORD.                 *
 *  INPUTS: RECNAME, DUMNAME                                         *
 *  OUTPUTS: DUMNAME                                                 *
 *  GLOBAL VARIABLES USED:                                           *
 *  GLOBAL VARIABLES CHANGED:                                        *
 *  GLOBAL TABLES USED:                                              *
 *  FILES READ:                                                      *
 *  FILES WRITTEN:                                                   *
 *  MODULES CALLED:                                                  *
 *  CALLING MODULES: BILDLOG                                         *
 *  AUTHOR: CAPT. JERRY OWENS                                        *
 *  HISTORY:                                                         *
 *********************************************************************/
/*

*/
RECDUM: PROCEDURE(RECNAME,DUMNAME);
   DCL RECNAME CHAR(16);
   DCL DUM_POS FIXED BIN(15);
   DCL DUM_REC CHAR(160) INIT ((160)' ');
   DCL DUMNAME CHAR(22);
SUBSTR(DUM_REC,1,16) = RECNAME;
DUM_POS = FINDPAT(DUM_REC,1,'  ');
SUBSTR(DUM_REC,DUM_POS,6) = '-DUMMY';
DUMNAME = SUBSTR(DUM_REC,1,22);
END RECDUM;
END RDSCHM;
```

```
/*  IDMSOUT 3.0  */
/*********************************************************************
 *  DATE: 10 SEP 83                                                  *
 *  VERSION: 1.0                                                     *
 *  NAME:  IDMSOUT                                                   *
 *  MODULE NUMBER: 3.0                                               *
 *  FUNCTION: MAIN DRIVER WHICH CALLS THE APPROPIATE MODULES         *
 *           TO BUILD THE IDMS SCHEMA, DMCL, AND SUBSCHEMA.          *
 *  INPUTS:                                                          *
 *  OUTPUTS:                                                         *
 *  GLOBAL VARIABLES USED:                                           *
 *  GLOBAL VARIABLES CHANGED:                                        *
 *  GLOBAL TABLES USED:                                              *
 *  FILES READ: IDMSMDL                                              *
 *  FILES WRITTEN:                                                   *
 *  MODULES CALLED: SCHMDES, FILEDES, AREADES, BILDREC, BILDSET,     *
 *                  DMCL, SUBSCHM                                    *
 *  CALLING MODULES:                                                 *
 *  AUTHOR:  CAPT. JERRY OWENS                                       *
 *  HISTORY:                                                         *
 *********************************************************************/
/*       .

*/
IDMSOUT: PROCEDURE OPTIONS(MAIN);
   DCL USRDEFS FILE STREAM INPUT;
   DCL SCHEMA FILE STREAM OUTPUT;
   DCL IDMSMDL FILE STREAM INPUT;
   DCL SUBS FILE STREAM OUTPUT;
   DCL DMCLFLE FILE STREAM OUTPUT;
   DCL AREA_NAME CHAR(16) INIT ((16)' ');
   DCL FILE_NAME CHAR(8) INIT ((8)' ');
   DCL REC_CNT FIXED BIN(15);
   DCL ELEM_CNT FIXED BIN(15);
   DCL SETNUM FIXED BIN(15) INIT (1);
   DCL RECNUM FIXED BIN(15) INIT (1);
   DCL SCHEMA_NAME CHAR(8);
   DCL DMCL_NAME CHAR(8);
   DCL SCHEMA_VERSION CHAR(3);
   DCL AUTHOR CHAR(30);
   DCL TODATE CHAR(9);
   DCL 1 REC_DESC (REC_CNT) CONTROLLED,
         2 NAME_REC CHAR(16),
         2 NEXT_PTR FIXED BIN(15),
         2 PRIOR_PTR FIXED BIN(15);
   DCL 1 SET_DESC (REC_CNT - 1) CONTROLLED,
         2 SET_NAME CHAR(16),
         2 SET_OWNER CHAR(16),
         2 SET_MEMBER CHAR(16);
   ON ENDFILE(IDMSMDL) GOTO SETBILD;
GET FILE(IDMSMDL) LIST(REC_CNT,ELEM_CNT);
ALLOCATE REC_DESC;
ALLOCATE SET_DESC;
```

```
CALL SCHMDES(SCHEMA_NAME,SCHEMA_VERSION,AUTHOR,TODATE);
CALL FILEDES(FILE_NAME);
CALL AREADES(AREA_NAME,FILE_NAME);
CALL BILDREC(AREA_NAME,REC_DESC,SET_DESC,RECNUM,SETNUM);
SETBILD: CALL BILDSET(REC_DESC,SET_DESC,SETNUM);
        CLOSE FILE (SCHEMA);
        CALL DMCL(SCHEMA_NAME,SCHEMA_VERSION,AUTHOR,TODATE,
                AREA_NAME,DMCL_NAME);
        CALL SUBSCHM(SCHEMA_NAME,REC_DESC,SET_DESC,RECNUM,SETNUM);
```

```
/*  SCHMDES 3.1  */
/*********************************************************************
 *  DATE: 10 SEP 83                                                  *
 *  VERSION: 1.1                                                     *
 *  NAME:  SCHMDES                                                   *
 *  MODULE NUMBER: 3.1                                               *
 *  FUNCTION: BUILD THE SCHEMA DESCRIPTION SECTION OF THE IDMS       *
 *            SCHEMA.                                                *
 *  INPUTS: SCHEMA_NAME, SCHEMA_VERSION, AUTHOR, TODATE              *
 *  OUTPUTS: SCHEMA_NAME, SCHEMA_VERSION, AUTHOR, TODATE             *
 *  GLOBAL VARIABLES USED:                                           *
 *  GLOBAL VARIABLES CHANGED:                                        *
 *  GLOBAL TABLES USED:                                              *
 *  FILES READ: USRDEFS                                              *
 *  FILES WRITTEN: SCHEMA                                            *
 *  MODULES CALLED: GETDATE                                          *
 *  CALLING MODULES: IDMSOUT                                         *
 *  AUTHOR: CAPT. JERRY OWENS                                        *
 *  HISTORY:                                                         *
 *********************************************************************/
/*

*/
SCHMDES: PROCEDURE(SCHEMA_NAME,SCHEMA_VERSION,AUTHOR,TODATE);
   DCL SCHEMA_NAME CHAR(8);
   DCL SCHEMA_VERSION CHAR(3);
   DCL AUTHOR CHAR(30);
   DCL AUTHOR_PHONE CHAR(12);
   DCL TODATE CHAR(9);
GET FILE(USRDEFS) EDIT (SCHEMA_NAME) (COL(1),A(8));
GET FILE(USRDEFS) EDIT (SCHEMA_VERSION) (COL(1),A(30));
GET FILE(USRDEFS) EDIT (AUTHOR) (COL(1),A(30));
GET FILE(USRDEFS) EDIT (AUTHOR_PHONE) (COL(1),A(12));
PUT FILE (SCHEMA) EDIT ('*')(COL(7),A);
PUT FILE (SCHEMA) EDIT
        ('*    ************************************************')
        (COL(7),A);
PUT FILE(SCHEMA) EDIT ('********')(COL(62),A);
PUT FILE(SCHEMA) EDIT
        ('*    *            SCHEMA DESCRIPTION STATEMENTS')
        (COL(7),A);
PUT FILE(SCHEMA) EDIT ('            *')(COL(55),A);
PUT FILE(SCHEMA) EDIT
        ('*    ************************************************')
        (COL(7),A);
PUT FILE(SCHEMA) EDIT ('********')(COL(62),A);
PUT FILE(SCHEMA) EDIT ('* ')(COL(7),A);
PUT FILE(SCHEMA) EDIT ('SCHEMA DESCRIPTION.')(COL(8),A);
PUT FILE(SCHEMA) SKIP(2) EDIT ('SCHEMA NAME IS ',SCHEMA_NAME,
                               'VERSION ',SCHEMA_VERSION,'.')
        (COL(8),A,COL(23),A,COL(40),A,COL(48),A,COL(51),A);
PUT FILE(SCHEMA) SKIP(2) EDIT ('DATE. ')(COL(8),A);
CALL GETDATE(TODATE);
```

```
PUT FILE(SCHEMA) SKIP(2) EDIT
       ('INSTALLATION.','NAS 7000 - 2')
       (COL(8),A,COL(40),A);
PUT FILE(SCHEMA) EDIT
       ('WRIGHT-PATTERSON AFB, OHIO.') (COL(40),A);
PUT FILE(SCHEMA) SKIP(2) EDIT ('REMARKS.') (COL(8),A);
PUT FILE(SCHEMA) EDIT
       ('THIS IS A SAMPLE IDMS SCHEMA') (COL(40),A);
PUT FILE(SCHEMA) EDIT
       ('DERIVED FROM THE ',SCHEMA_NAME) (COL(40),A,COL(57),A);
PUT FILE(SCHEMA) EDIT
       ('DATABASE SCHEMA CURRENTLY') (COL(40),A);
PUT FILE(SCHEMA) EDIT
       ('INSTALLED UNDER S2000.') (COL(40),A);
PUT FILE(SCHEMA) SKIP(2) EDIT ('AUTHOR.',AUTHOR) (COL(8),A,COL(40),A);
PUT FILE(SCHEMA) EDIT (AUTHOR_PHONE,'.') (COL(40),A,COL(52),A);
END SCHMDES;
```

```
/*  GETDATE 3.1.1  */
/*********************************************************************
 *  DATE: 20 SEP 83                                                  *
 *  VERSION: 1.0                                                     *
 *  NAME:  GETDATE                                                   *
 *  MODULE NUMBER: 3.1.1                                             *
 *  FUNCTION: GET THE CURRENT DATE.                                  *
 *  INPUTS: TODATE                                                   *
 *  OUTPUTS: TODATE                                                  *
 *  GLOBAL VARIABLES USED:                                           *
 *  GLOBAL VARIABLES CHANGED:                                        *
 *  GLOBAL TABLES USED:                                              *
 *  FILES READ:                                                      *
 *  FILES WRITTEN: SCHEMA                                            *
 *  MODULES CALLED:                                                  *
 *  CALLING MODULES: SCHMDES                                         *
 *  AUTHOR: CAPT. JERRY OWENS                                        *
 *  HISTORY:                                                         *
 *********************************************************************/
/*

*/
GETDATE: PROCEDURE(TODATE);
   DCL TODAY CHAR(6);
   DCL TODATE CHAR(9);
TODAY = DATE;
TODATE = SUBSTR(TODAY,3,2) :: '/' :: SUBSTR(TODAY,5,2) :: '/' ::
        SUBSTR(TODAY,1,2) :: '.';
PUT FILE(SCHEMA) EDIT (TODATE)(COL(40),A);
END GETDATE;
```

```
/*  FILEDES 3.2  */
/*********************************************************************
 *  DATE: 10 SEP 83                                                  *
 *  VERSION: 1.1                                                     *
 *  NAME:  FILEDES                                                   *
 *  MODULE NUMBER: 3.2                                               *
 *  FUNCTION: BUILD THE FILE DESCRIPTION SECTION OF THE IDMS         *
 *           SCHEMA.                                                 *
 *  INPUTS: FILE_NAME                                                *
 *  OUTPUTS: FILE_NAME                                               *
 *  GLOBAL VARIABLES USED:                                           *
 *  GLOBAL VARIABLES CHANGED:                                        *
 *  GLOBAL TABLES USED:                                              *
 *  FILES READ: USRDEFS                                              *
 *  FILES WRITTEN: SCHEMA                                            *
 *  MODULES CALLED: IDMSOUT                                          *
 *  CALLING MODULES:                                                 *
 *  AUTHOR: CAPT. JERRY OWENS                                        *
 *  HISTORY:                                                         *
 *********************************************************************/
/*

*/
FILEDES: PROCEDURE(FILE_NAME);
   DCL FILE_NAME CHAR(8);
   DCL FILE_DEVICE CHAR(5);
GET FILE(USRDEFS) EDIT (FILE_NAME)(COL(1),A(8));
GET FILE(USRDEFS) EDIT (FILE_DEVICE)(COL(1),A(8));
PUT FILE(SCHEMA) EDIT ('*')(COL(7),A);
PUT FILE(SCHEMA) EDIT
        ('*    ************************************************')
        (COL(7),A);
PUT FILE(SCHEMA) EDIT ('********')(COL(62),A);
PUT FILE(SCHEMA) EDIT
       ('*    *              FILE DESCRIPTION STATEMENTS')
       (COL(7),A);
PUT FILE(SCHEMA) EDIT ('             *')(COL(55),A);
PUT FILE(SCHEMA) EDIT
        ('*    ************************************************')
        (COL(7),A);
PUT FILE(SCHEMA) EDIT ('********')(COL(62),A);
PUT FILE(SCHEMA) EDIT ('* ')(COL(7),A);
PUT FILE(SCHEMA) EDIT ('FILE DESCRIPTION.')(COL(8),A);
PUT FILE(SCHEMA) SKIP(2) EDIT ('FILE NAME IS '<FILE_NAME,
                               'ASSIGN TO ',FILE_NAME)
                           (COL(8),A,COL(21),A,COL(40),A,COL(50),A);
PUT FILE(SCHEMA) EDIT ('DEVICE TYPE IS ',FILE_DEVICE,'.')
                           (COL(40),A,COL(55),A,COL(60),A);
END FILEDES;
```

```
/*  AREADES 3.3  */
/*****************************************************************
 *  DATE: 10 SEP 83                                              *
 *  VERSION: 1.0                                                 *
 *  NAME:  AREADES                                               *
 *  MODULE NUMBER: 3.3                                           *
 *  FUNCTION: BUILD THE AREA DESCRIPTION SECTION OF THE IDMS     *
 *           SCHEMA.                                             *
 *  INPUTS: AREA_NAME, FILE_NAME                                 *
 *  OUTPUTS: AREA_NAME                                           *
 *  GLOBAL VARIABLES USED:                                       *
 *  GLOBAL VARIABLES CHANGED:                                    *
 *  GLOBAL TABLES USED:                                          *
 *  FILES READ: USRDEFS                                          *
 *  FILES WRITTEN: SCHEMA                                        *
 *  MODULES CALLED:                                              *
 *  CALLING MODULES: IDMSOUT                                     *
 *  AUTHOR: CAPT. JERRY OWENS                                    *
 *  HISTORY:                                                     *
 *****************************************************************/
/*

*/
AREADES: PROCEDURE(AREA_NAME,FILE_NAME);
   DCL AREA_NAME CHAR(16);
   DCL FILE_NAME CHAR(8);
   DCL RANGE_START FIXED BIN(31);
   DCL RANGE_END FIXED BIN(31);
   DCL LIMIT FIXED BIN(15);
GET FILE(USRDEFS) EDIT (AREA_NAME)(COL(1),A(16));
GET FILE(USRDEFS) EDIT (RANGE_START)(COL(1),F(8));
GET FILE(USRDEFS) EDIT (RANGE_END)(COL(1),F(8));
PUT FILE(SCHEMA) EDIT ('*')(COL(7),A);
PUT FILE(SCHEMA) EDIT
        ('*    ************************************************')
        (COL(7),A);
PUT FILE(SCHEMA) EDIT ('********')(COL(62),A);
PUT FILE(SCHEMA) EDIT
       ('*    *              AREA DESCRIPTION STATEMENTS')
       (COL(7),A);
PUT FILE(SCHEMA) EDIT ('             *')(COL(55),A);
PUT FILE(SCHEMA) EDIT
        ('*    ************************************************')
        (COL(7),A);
PUT FILE(SCHEMA) EDIT ('********')(COL(62),A);
PUT FILE(SCHEMA) EDIT ('* ')(COL(7),A);
PUT FILE(SCHEMA) EDIT ('AREA DESCRIPTION.')(COL(8),A);
PUT FILE(SCHEMA) SKIP(2) EDIT ('AREA NAME IS ',AREA_NAME)
                         (COL(8),A,COL(21),A);
PUT FILE(SCHEMA) EDIT ('RANGE IS ',RANGE_START,' THRU',RANGE_END
                    (COL(19),A,COL(28),F(8),COL(36),A,COL(42),F(8));
PUT FILE(SCHEMA) EDIT ('WITHIN FILE ',FILE_NAME)
                     (COL(40),A,COL(52),A);
```

```
LIMIT = (RANGE_END - RANGE_START) + 1;
PUT FILE(SCHEMA) EDIT ('FROM 1 THRU ',LIMIT,'.')
                      (COL(44),A,COL(59),F(8),COL(67),A);
END AREADES;
```

```
/*  BILDREC 3.4  */
/*****************************************************************
 *  DATE: 13 SEP 83                                              *
 *  VERSION: 1.2                                                 *
 *  NAME:  BILDREC                                               *
 *  MODULE NUMBER: 3.4                                           *
 *  FUNCTION: BUILD THE RECORD DESCRIPTION SECTION OF THE IDMS   *
 *          SCHEMA.                                              *
 *  INPUTS: AREA_NAME, REC_DESC, SET_DESC, RECNUM, SETNUM        *
 *  OUTPUTS: REC_DESC, SET_DESC, RECNUM, SETNUM                  *
 *  GLOBAL VARIABLES USED:                                       *
 *  GLOBAL VARIABLES CHANGED:                                    *
 *  GLOBAL TABLES USED:                                          *
 *  FILES READ:                                                  *
 *  FILES WRITTEN: SCHEMA                                        *
 *  MODULES CALLED: RECINFO, ITMINFO                             *
 *  CALLING MODULES: IDMSOUT                                     *
 *  AUTHOR: CAPT. JERRY OWENS                                    *
 *  HISTORY:                                                     *
 *****************************************************************/
/*
        .

*/
BILDREC: PROCEDURE(AREA_NAME,REC_DESC,SET_DESC,RECNUM,SETNUM);
   DCL AREA_NAME CHAR(16);
   DCL RECNUM FIXED BIN(15);
   DCL SETNUM FIXED BIN(15);
   DCL 1 REC_DESC (*) ,
         2 NAME_REC CHAR(16),
         2 NEXT_PTR FIXED BIN(15),
         2 PRIOR_PTR FIXED BIN(15);
   DCL 1 SET_DESC (*),
         2 SET_NAME CHAR(16),
         2 SET_OWNER CHAR(16),
         2 SET_MEMBER CHAR(16);
PUT FILE(SCHEMA) EDIT ('*')(COL(7),A);
PUT FILE(SCHEMA) EDIT
        ('*    ************************************************')
        (COL(7),A);
PUT FILE(SCHEMA) EDIT ('********')(COL(62),A);
PUT FILE(SCHEMA) EDIT
       ('*    *            RECORD DESCRIPTION STATEMENTS')
       (COL(7),A);
PUT FILE(SCHEMA) EDIT ('            *')(COL(55),A);
PUT FILE(SCHEMA) EDIT
        ('*    ************************************************')
        (COL(7),A);
PUT FILE(SCHEMA) EDIT ('********')(COL(62),A);
PUT FILE(SCHEMA) EDIT ('* ')(COL(7),A);
PUT FILE(SCHEMA)  EDIT ('RECORD DESCRIPTION.')(COL(8),A);
DO WHILE ('1'B);
  CALL RECINFO(AREA_NAME,REC_DESC(RECNUM),SET_DESC(SETNUM),
              SETNUM,RECNUM);
```

```
  CALL ITMINFO;
END;
END BILDREC;
```

```
/*  RECINFO 3.4.1  */
/*********************************************************************
 *  DATE: 18 SEP 83                                                  *
 *  VERSION: 1.2                                                     *
 *  NAME:  RECINFO                                                   *
 *  MODULE NUMBER: 3.4.1                                             *
 *  FUNCTION: GET RECORD NAME, RECORD ID, AND LOCATION MODE FOR      *
 *            A GIVEN RECORD.                                        *
 *  INPUTS: AREA_NAME, REC_DESC, SET_DESC, SETNUM, RECNUM            *
 *  OUTPUTS: REC_DESC, SET_DESC, SETNUM, RECNUM                      *
 *  GLOBAL VARIABLES USED:                                           *
 *  GLOBAL VARIABLES CHANGED:                                        *
 *  GLOBAL TABLES USED:                                              *
 *  FILES READ: IDMSMDL                                              *
 *  FILES WRITTEN: SCHEMA                                            *
 *  MODULES CALLED: LOADREC, LOADSET                                 *
 *  CALLING MODULES: BILDREC                                         *
 *  AUTHOR: CAPT. JERRY OWENS                                        *
 *  HISTORY:                                                         *
 *********************************************************************/
/*

*/
RECINFO: PROCEDURE(AREA_NAME,REC_DESC,SET_DESC,SETNUM,RECNUM);
   DCL RECNUM FIXED BIN(15);
   DCL SETNUM FIXED BIN(15);
   DCL 1 REC_DESC,
         2 NAME_REC CHAR(16),
         2 NEXT_PTR FIXED BIN(15),
         2 PRIOR_PTR FIXED BIN(15);
   DCL 1 SET_DESC,
         2 SET_NAME CHAR(16),
         2 SET_OWNER CHAR(16),
         2 SET_MEMBER CHAR(16);
   DCL REC_NAME CHAR(16) INIT ((16)' ');
   DCL REC_ID CHAR(4) INIT ((4)' ');
   DCL SET_TYPE CHAR(4) INIT ((4)' ');
   DCL SET_NAME CHAR(16) INIT ((16)' ');
   DCL VIA_NAME CHAR(16) INIT ((16)' ');
   DCL CALC_NAME CHAR(16) INIT ((16)' ');
   DCL AREA_NAME CHAR(16);
GET FILE(IDMSMDL) EDIT (REC_NAME,REC_ID) (COL(3),A(16),COL(20),A(4));
PUT FILE(SCHEMA) SKIP(2) EDIT ('RECORD NAME IS ',REC_NAME,'.')
                        (COL(8),A,COL(23),A,COL(39),A);
PUT FILE(SCHEMA) EDIT ('RECORD ID IS ',REC_ID,'.')
                        (COL(8),A,COL(21),A,COL(25),A);
CALL LOADREC(REC_DESC,REC_NAME,RECNUM);
GET FILE(IDMSMDL) EDIT (SET_TYPE)(COL(1),A(1));
IF SET_TYPE = 'C' THEN DO;
  GET FILE(IDMSMDL) EDIT (CALC_NAME)(COL(5),A(32));
  PUT FILE(SCHEMA) EDIT ('LOCATION MODE IS CALC ','USING ',CALC_NAME)
                         (COL(8),A,COL(40),A,COL(46),A);
  PUT FILE(SCHEMA) EDIT ('DUPLICATES ARE NOT ALLOWED.')
```

```
                        (COL(40),A);
END;
   ELSE DO;
     GET FILE(IDMSMDL) EDIT (VIA_NAME)(COL(5),A(16));
     PUT FILE(SCHEMA) EDIT ('LOCATION MODE IS VIA ',VIA_NAME,'SET.')
                           (COL(8),A,COL(29),A,COL(46),A);
     CALL LOADSET(SET_DESC,VIA_NAME,SETNUM);
   END;
PUT FILE(SCHEMA) EDIT ('WITHIN ',AREA_NAME,' AREA.')
                      (COL(8),A,COL(15),A,COL(31),A);
PUT FILE(SCHEMA) SKIP;
END RECINFO;
```

```
/*  LOADREC 3.4.1.1  */
/*********************************************************************
 *  DATE: 14 SEP 83                                                  *
 *  VERSION: 1.0                                                     *
 *  NAME:  LOADREC                                                   *
 *  MODULE NUMBER: 3.4.1.1                                           *
 *  FUNCTION: STORE THE NEEDED RECORD INFO FOR FUTURE USE            *
 *  INPUTS: REC_DESC, REC_NAME, RECNUM                               *
 *  OUTPUTS: REC_DESC, RECNUM                                        *
 *  GLOBAL VARIABLES USED:                                           *
 *  GLOBAL VARIABLES CHANGED:                                        *
 *  GLOBAL TABLES USED:                                              *
 *  FILES READ:                                                      *
 *  FILES WRITTEN:                                                   *
 *  MODULES CALLED:                                                  *
 *  CALLING MODULES: RECINFO                                         *
 *  AUTHOR: CAPT. JERRY OWENS                                        *
 *  HISTORY:                                                         *
 *********************************************************************/
/*

*/
LOADREC: PROCEDURE(REC_DESC,REC_NAME,RECNUM);
   DCL REC_NAME CHAR(16);
   DCL RECNUM FIXED BIN(15);
   DCL 1 REC_DESC,
         2 NAME_REC CHAR(16),
         2 NEXT_PTR FIXED BIN(15),
         2 PRIOR_PTR FIXED BIN(15);
REC_DESC.NAME_REC = REc_NAME;
REC_DESC.NEXT_PTR = 1;
REC_DESC.PRIOR_PTR = 2;
RECNUM = RECNUM + 1;
END LOADREC;
```

```
/* LOADSET 3.4.1.2 */
/*********************************************************************
 *  DATE: 15 SEP 83                                                  *
 *  VERSION: 1.0                                                     *
 *  NAME:  LOADSET                                                   *
 *  MODULE NUMBER: 3.4.1.2                                           *
 *  FUNCTION: STORE THE SET INFO FOR FUTURE USE.                     *
 *  INPUTS: SET_DESC, VIA_NAME, SETNUM                               *
 *  OUTPUTS: SET_DESC, SETNUM                                        *
 *  GLOBAL VARIABLES USED:                                           *
 *  GLOBAL VARIABLES CHANGED:                                        *
 *  GLOBAL TABLES USED:                                              *
 *  FILES READ: IDMSMDL                                              *
 *  FILES WRITTEN:                                                   *
 *  MODULES CALLED:                                                  *
 *  CALLING MODULES: RECINFO                                         *
 *  AUTHOR: CAPT. JERRY OWENS                                        *
 *  HISTORY:                                                         *
 *********************************************************************/
/*

*/      .
LOADSET: PROCEDURE(SET_DESC,VIA_NAME,SETNUM);
   DCL VIA_NAME CHAR(16);
   DCL SETNUM FIXED BIN(15);
   DCL 1 SET_DESC,
         2 SET_NAME CHAR(16),
         2 SET_OWNER CHAR(16),
         2 SET_MEMBER CHAR(16);
SET_DESC.SET_NAME = VIA_NAME;
GET FILE(IDMSMDL) EDIT (SET_DESC.SET_OWNER)(COL(25),A(16));
GET FILE(IDMSMDL) EDIT (SET_DESC.SET_MEMBER)(COL(45),A(16));
SETNUM = SETNUM + 1;
END LOADSET;
```

```
/*  ITMINFO 3.4.2  */
/*********************************************************************
 *  DATE: 15 SEP 83                                                  *
 *  VERSION: 1.01                                                    *
 *  NAME:  ITMINFO                                                   *
 *  MODULE NUMBER: 3.4.2                                             *
 *  FUNCTION: GET ALL OF THE ITEMS FOR A GIVEN RECORD.               *
 *  INPUTS:                                                          *
 *  OUTPUTS:                                                         *
 *  GLOBAL VARIABLES USED:                                           *
 *  GLOBAL VARIABLES CHANGED:                                        *
 *  GLOBAL TABLES USED:                                              *
 *  FILES READ: IDMSMDL                                              *
 *  FILES WRITTEN: SCHEMA                                            *
 *  MODULES CALLED:                                                  *
 *  CALLING MODULES: BILDREC                                         *
 *  AUTHOR: CAPT. JERRY OWENS                                        *
 *  HISTORY:                                                         *
 *********************************************************************/
/*

*/
ITMINFO: PROCEDURE;
   DCL ITEM_NAME CHAR(32);
   DCL ITEM_PIC CHAR(28);
   DCL ITEM_TYPE CHAR(1) INIT (' ');
GET FILE(IDMSMDL) EDIT (ITEM_TYPE)
                       (COL(1),A(1));
DO WHILE (ITEM_TYPE = 'I');
  GET FILE(IDMSMDL) EDIT (ITEM_NAME,ITEM_PIC)
                         (COL(8),A(32),COL(41),A(28));
  PUT FILE(SCHEMA) EDIT ('03',ITEM_NAME,ITEM_PIC,'.')
                         (COL(8),A,COL(12),A,COL(44),A,COL(72),A);
  GET FILE(IDMSMDL) EDIT (ITEM_TYPE)
                         (COL(1),A(1));
END;
END ITMINFO;
```

```
/*  BILDSET 3.5  */
/*********************************************************************
 *  DATE: 16 SEP 83                                                  *
 *  VERSION: 1.3                                                     *
 *  NAME:  BILDSET                                                   *
 *  MODULE NUMBER: 3.5                                               *
 *  FUNCTION: BUILD THE SET DESCRIPTION PORTION OF THE IDMS SCHEMA   *
 *  INPUTS: REC_DESC, SET_DESC, SETNUM                               *
 *  OUTPUTS:                                                         *
 *  GLOBAL VARIABLES USED:                                           *
 *  GLOBAL VARIABLES CHANGED:                                        *
 *  GLOBAL TABLES USED:                                              *
 *  FILES READ:                                                      *
 *  FILES WRITTEN: SCHEMA                                            *
 *  MODULES CALLED:  DBKEY                                           *
 *  CALLING MODULES:  IDMSOUT                                        *
 *  AUTHOR: CAPT. JERRY OWENS                                        *
 *  HISTORY:                                                         *
 *********************************************************************/
/*

*/
BILDSET: PROCEDURE(REC_DESC,SET_DESC,SETNUM);
   DCL SETNUM FIXED BIN(15);
   DCL I,J FIXED BIN(15);
   DCL 1 REC_DESC (*),
         2 NAME_REC CHAR(16),
         2 NEXT_PTR FIXED BIN(15),
         2 PRIOR_PTR FIXED BIN(15);
   DCL 1 SET_DESC (*),
         2 SET_NAME CHAR(16),
         2 SET_OWNER CHAR(16),
         2 SET_MEMBER CHAR(16);
PUT FILE(SCHEMA) EDIT ('*')(COL(7),A);
PUT FILE(SCHEMA) EDIT
        ('*    ************************************************')
        (COL(7),A);
PUT FILE(SCHEMA) EDIT ('********')(COL(62),A);
PUT FILE(SCHEMA) EDIT
       ('*    *              SET DESCRIPTION STATEMENTS ')
       (COL(7),A);
PUT FILE(SCHEMA) EDIT ('            *')(COL(55),A);
PUT FILE(SCHEMA) EDIT
        ('*    ************************************************')
        (COL(7),A);
PUT FILE(SCHEMA) EDIT ('********')(COL(62),A);
PUT FILE(SCHEMA) EDIT ('* ')(COL(7),A);
PUT FILE(SCHEMA)  EDIT ('SET DESCRIPTION.')(COL(8),A);
DO I = 1 TO SETNUM - 1;
  PUT FILE(SCHEMA) SKIP(2) EDIT ('SET NAME IS ',
                                 SET_DESC(I).SET_NAME,'.')
                          (COL(8),A,COL(20),A,COL(37),A);
  PUT FILE(SCHEMA) EDIT ('ORDER IS NEXT.')(COL(8),A);
```

```
  PUT FILE(SCHEMA) EDIT ('MODE IS CHAIN','LINKED TO PRIOR.')
                        (COL(8),A,COL(40),A);
  CALL DBKEY(REC_DESC,SET_DESC(I));
  PUT FILE(SCHEMA) EDIT ('MANDATORY AUTOMATIC.')(COL(40),A);
  PUT FILE(SCHEMA) SKIP(2);
END BILDSET;
```

```
/*  DBKEY 3.5.1  */
/*********************************************************************
 *  DATE: 16 SEP 83                                                  *
 *  VERSION: 1.0                                                     *
 *  NAME:  DBKEY                                                     *
 *  MODULE: 3.5.1                                                    *
 *  FUNCTION: GET THE OWNER AND MEMBER OF THE CURRENT SET, ALONG     *
 *           WITH THE CURRENT NEXT AND PRIOR POINTER OF EACH.        *
 *  INPUTS: REC_DESC, SET_DESC                                       *
 *  OUTPUTS:                                                         *
 *  GLOBAL VARIABLES USED:                                           *
 *  GLOBAL VARIABLES CHANGED:                                        *
 *  GLOBAL TABLES USED:                                              *
 *  FILES READ:                                                      *
 *  FILES WRITTEN: SCHEMA                                            *
 *  MODULES CALLED:                                                  *
 *  CALLING MODULES: BILDSET                                         *
 *  AUTHOR: CAPT. JERRY OWENS                                        *
 *  HISTORY:                                                         *
 *********************************************************************/
/*

*/
DBKEY: PROCEDURE(REC_DESC,SET_DESC);
   DCL 1 REC_DESC (*),
         2 NAME_REC CHAR(16),
         2 NEXT_PTR FIXED BIN(15),
         2 PRIOR_PTR FIXED BIN(15);
   DCL 1 SET_DESC,
         2 SET_NAME CHAR(16),
         2 SET_OWNER CHAR(16),
         2 SET_MEMBER CHAR(16);
   DCL I FIXED BIN(15) INIT (1);
DO WHILE (REC_DESC(I).NAME_REC := SET_DESC.SET_OWNER);
  I = I + 1;
END;
PUT FILE(SCHEMA) EDIT ('OWNER IS ',REC_DESC(I).NAME_REC,
                  'NEXT DBKEY POSITION IS ',REC_DESC(I).NEXT_PTR)
                  (COL(8),A,COL(17),A,COL(40),A,COL(63),F(3));
REC_DESC(I).NEXT_PTR = REC_DESC(I).PRIOR_PTR + 1;
PUT FILE(SCHEMA) eDIT ('PRIOR DBKEY POSITION IS ',
       REC_DESC(I).PRIOR_PTR,'.')(COL(40),A,COL(64),F(3),COL(67),A);
REC_DESC(I).PRIOR_PTR = REC_DESC(I).NEXT_PTR + 1;
I = 1;
DO WHILE (REC_DESC(I).NAME_REC := SET_DESC.SET_MEMBER);
  I = I + 1;
END;
PUT FILE(SCHEMA) EDIT ('MEMBER IS ',REC_DESC(I).NAME_REC,
                  'NEXT DBKEY POSITION IS ',REC_DESC(I).NEXT_PTR)
                  (COL(8),A,COL(18),A,COL(40),A,COL(63),F(3));
REC_DESC(I).NEXT_PTR = REC_DESC(I).PRIOR_PTR + 1;
PUT FILE(SCHEMA) EDIT ('PRIOR DBKEY POSITION IS ',
       REC_DESC(I).PRIOR_PTR)(COL(40),A,COL(64),F(3));
```

```
REC_DESC(I).PRIOR_PTR = REC_DESC(I).NEXT_PTR + 1;
END DBKEY;
```

```
/*  DMCL 3.6  */
/*********************************************************************
 *  DATE: 20 SEP 83                                                  *
 *  VERSION: 1.1                                                     *
 *  NAME:  DMCL                                                      *
 *  MODULE NUMBER: 3.6                                               *
 *  FUNCTION: CREATE THE ASSOCIATED DMCL INPUT FOR THE SCHEMA        *
 *  INPUTS: SCHEMA_NAME, SCHEMA_VERSION, AUTHOR, TODATE, AREA_NAME,  *
 *          DMCL_NAME                                                *
 *  OUTPUTS: DMCL_NAME                                               *
 *  GLOBAL VARIABLES USED:                                           *
 *  GLOBAL VARIABLES CHANGED:                                        *
 *  GLOBAL TABLES USED:                                              *
 *  FILES READ: USRDEFS                                              *
 *  FILES WRITTEN: DMCL                                              *
 *  MODULES CALLED:                                                  *
 *  CALLING MODULES: IDMSOUT                                         *
 *  AUTHOR: CAPT. JERRY OWENS                                        *
 *  HISTORY:                                                         *
 *********************************************************************/
/*

*/
DMCL: PROCEDURE (SCHEMA_NAME,SCHEMA_VERSION,AUTHOR,TODATE,
                 AREA_NAME,DMCL_NAME);
   DCL SCHEMA_NAME CHAR(8);
   DCL DMCL_NAME CHAR(8);
   DCL SCHEMA_VERSION CHAR(3);
   DCL TODATE CHAR(9);
   DCL AREA_NAME CHAR(16);
   DCL AUTHOR CHAR(30);
GET FILE(USRDEFS) EDIT (DMCL_NAME)(COL(1),A(8));
PUT FILE(DMCLFLE) EDIT ('DEVICE-MEDIA DESCRIPTION.')
                       (COL(8),A);
PUT FILE(DMCLFLE) EDIT
    ('DEVICE-MEDIA NAME IS',DMCL_NAME,'OF SCHEMA NAME',
                        SCHEMA_NAME,'VERSION',SCHEMA_VERSION,'.')
                       (COL(8),A,COL(29),A,COL(38),A,COL(53),A,
                        COL(62),A,COL(71),A,COL(75),A);
PUT FILE(DMCLFLE) SKIP(2) EDIT ('AUTHOR.',AUTHOR,'.')
                       (COL(8),A,COL(30),A,COL(60),A);
PUT FILE(DMCLFLE) EDIT ('DATE.',TODATE)(COL(8),A,COL(30),A);
PUT FILE(DMCLFLE) SKIP(2) EDIT ('INSTALLATION.','NAS 7000 - 2')
                       (COL(8),A,COL(30),A);
PUT FILE(DMCLFLE) EDIT ('WRIGHT - PATTERSON AFB, OHIO.')(COL(30),A);
PUT FILE(DMCLFLE) SKIP(2) EDIT
                       ('REMARKS.','THIS IS A GENERAL FORM OF THE')
                       (COL(8),A,COL(30),A);
PUT FILE(DMCLFLE) EDIT ('DMCL. PLEASE MAKE THE DESIRED CHANGES.')
                       (COL(30),A);
PUT FILE(DMCLFLE) SKIP(2) EDIT ('BUFFER SECTION.')(COL(8),A);
PUT FILE(DMCLFLE) EDIT ('BUFFER NAME IS GENERAL-BUFFER')(COL(12),A);
```

```
PUT FILE(DMCLFLE) EDIT ('PAGE CONTAINS 4060 CHARACTERS')(COL(16),A);
PUT FILE(DMCLFLE) EDIT ('BUFFER CONTAINS 8 PAGES.')(COL(16),A);
PUT FILE(DMCLFLE) SKIP(2) EDIT ('AREA SECTION.')(COL(8),A);
PUT FILE(DMCLFLE) EDIT ('COPY',AREA_NAME,'AREA')
                       (COL(12),A,COL(17),A,COL(34),A);
PUT FILE(DMCLFLE) EDIT ('FROM SCHEMA NAME',SCHEMA_NAME,'VERSION',
                        SCHEMA_VERSION)(COL(16),A,COL(33),A,
                        COL(42),A,COL(50),A);
PUT FILE(DMCLFLE) EDIT ('BUFFER IS GENERAL-BUFFER.')
                       (COL(16),A);
PUT FILE(DMCLFLE) SKIP(2) EDIT ('JOURNAL SECTION.')(COL(8),A);
PUT FILE(DMCLFLE) EDIT ('JOURNAL BLOCK CONTAINS 1000 CHARACTERS.')
                       (COL(12),A);
PUT FILE(DMCLFLE) EDIT
    ('FILE NAME IS TAPE-JOURNAL ASSIGN TO SYSJRNL')
    (COL(12),A);
PUT FILE(DMCLFLE) EDIT ('DEVICE TYPE IS 2400.')(COL(33),A);
CLOSE FILE(DMCLFLE);
END DMCL;
```

```
/*  SUBSCHM 3.7  */
/*********************************************************************
 *  DATE: 20 SEP 83                                                  *
 *  VERSION: 1.0                                                     *
 *  NAME:  SUBSCHM                                                   *
 *  MODULE NUMBER: 3.7                                               *
 *  FUNCTION: CREATE THE ASSOCIATED SUBSCHEMA FOR THE CURRENT        *
 *            SCHEMA.                                                *
 *  INPUTS: SCHEMA_NAME, REC_DESC, SET_DESC, RECNUM, SETNUM          *
 *  OUTPUTS:                                                         *
 *  GLOBAL VARIABLES USED:                                           *
 *  GLOBAL VARIABLES CHANGED:                                        *
 *  GLOBAL TABLES USED:                                              *
 *  FILES READ: USRDEFS                                              *
 *  FILES WRITTEN: SUBS                                              *
 *  MODULES CALLED:                                                  *
 *  CALLING MODULES: IDMSOUT                                         *
 *  AUTHOR: CAPT. JERRY OWENS                                        *
 *  HISTORY:                                                         *
 *********************************************************************/
/*

*/
SUBSCHM: PROCEDURE(SCHEMA_NAME,REC_DESC,SET_DESC,RECNUM,SETNUM);
   DCL SCHEMA_NAME CHAR(8);
   DCL SUB_NAME CHAR(8);
   DCL RECNUM FIXED BIN(15);
   DCL SETNUM FIXED BIN(15);
   DCL 1 REC_DESC (*),
         2 NAME_REC CHAR(16),
         2 NEXT_PTR FIXED BIN(15),
         2 PRIOR_PTR FIXED BIN(15);
   DCL 1 SET_DESC (*),
         2 SET_NAME CHAR(16),
         2 SET_OWNER CHAR(16),
         2 SET_MEMBER CHAR(16);
GET FILE(USRDEFS) EDIT (SUB_NAME)(COL(1),A(8));
PUT FILE(SUBS) EDIT ('DELETE SUBSCHEMA NAME IS',SUB_NAME,'.')
                    (COL(12),A,COL(38),A,COL(46),A);
PUT FILE(SUBS) EDIT ('ADD SUBSCHEMA NAME IS',SUB_NAME)
                    (COL(12),A,COL(34),A);
PUT FILE(SUBS) EDIT ('OF SCHEMA NAME IS',SCHEMA_NAME)
                    (COL(17),A,COL(35),A);
PUT FILE(SUBS) EDIT ('DMCL NAME IS',DMCL_NAME,'.')
                    (COL(19),A,COL(32),A,COL(41),A);
PUT FILE(SUBS) EDIT ('ADD AREA',AREA_NAME,'.')
                    (COL(12),A,COL(21),A,COL(37),A);
DO I = 1 TO RECNUM - 1;
  PUT FILE(SUBS) EDIT ('ADD RECORD',REC_DESC(I).NAME_REC,'.')
                      (COL(12),A,COL(23),A,COL(40),A);
END;
DO I = 1 TO SETNUM - 1;
  PUT FILE(SUBS) EDIT ('ADD SET',SET_DESC(I).SET_NAME,'.')
```

```
                                (COL(12),A,COL(20),A,COL(36),A);
    END;
    PUT FILE(SUBS) EDIT ('GENERATE.')(COL(12),A);
    END SUBSCHM;
    END IDMSOUT;
```

Appendix F

Automated Hierarchical - CODASYL Database Interface

Schema Translator TSO CLIST Source Code

CLIST Program Name: Menu **Author: Capt. Jerry Owens**

```
PROC 0
/*  This CLIST allows the user to execute programs developed
/*  for the s2k to idms schema translation
WRITE
WRITE ****************** s2k to idms translator menu ********************
WRITE *                                                                 *
WRITE *    1)  run usrprap              3)  run idmsout                 *
WRITE *   .2)  run rdschm              >3)  exit menu                   *
WRITE *                                                                 *
WRITE ***********************************************************************
WRITENR     Enter menu number:
READ &SELECT
DOSELECT: IF &SELECT = 1 THEN +
     EXEC jgo.clist(usrprap)
IF &SELECT = 2 THEN +
     EXEC jgo.clist(rdschm)
IF &SELECT = 3 THEN +
     EXEC jgo.clist(idmsout)
IF &SELECT < 4 THEN +
     EXEC jgo.clist(menu)
```

CLIST Program Name: Usrprmp Author: Capt. Jerry Owens

```
PROC 1 USERDEFS PRINT(DA(*))
/*  This clist allows the Usrprmp module to execute in the foreground */
CONTROL NOMSG
FREE F(USERIN,SYSPRINT,USEROUT,SCHMIN,SCHMOUT)
ALLOC F(USEROUT) DA('&SYSPREF..&USERDEFS') SHR
ALLOC F(USERIN) DA(*)
ALLOC F(SYSPRINT) DA(*)
CALL 'A740308.J60.LOAD(USRPRMP)'
FREE F(USERIN,SYSPRINT,USEROUT)
FREE DSN('A740308.J60.LOAD')
```

CLIST Program Name: Rdschm Author: Capt. Jerry Owens

```
PROC 6 S2KSCHM S2KTEMP IDMSMODL STARTNUM INCNUM MSGCLASS
CONTROL NOMSG
/*                                                                    */
/**********************************************************************/
/*  The Rdschm clist builds the necessary JCL to execute the rdschm */
/*  PL/1 module in the background.                                    */
/**********************************************************************/
/*                                                                    */
/**********************************************************************/
/*          BUILD JOB CARD INFORMATION                                */
/**********************************************************************/
/*                                                                    */
SET &JOBCARD1 = &STR(//)&SYSUID&STR(TRNS JOB )&SUBSTR(1:11,(4308,210),)
SET &JOBCARD2 = &SYSUID&STR(,CLASS=S,MSGCLASS=)&MSGCLASS&STR(,)
SET &JOBCARD3 = &STR(// NOTIFY=)&SYSUID,&STR(MSGLEVEL=(1,1),TIME=5,)
SET &JOBCARD4 = &STR(REGION=500K)
SET &JOBCARDA = &STR(&JOBCARD1&JOBCARD2)
SET &JOBCARDB = &STR(&JOBCARD3&JOBCARD4)
SET &JOBCARDC = &STR(/*JOBPARM S=NAS1)
/*                                                                    */
/**********************************************************************/
/*          BUILD STEPLIB INFORMATION                                 */
/**********************************************************************/
/*                                                                    */
SET &THESIS1 = &STR(//RDSCHM EXEC PGM=RDSCHM)
SET &STEPLIB = &STR(//STEPLIB DD DSN=A740308.J60.LOAD,DISP=SHR)
SET &SLASH1 = &STR(/*)
SET &SYSPRINT = &STR(//GO.SYSPRINT DD SYSOUT=*)
/*                                                                    */
/**********************************************************************/
/*          BUILD INPUT FILE(S) INFORMATION                           */
/**********************************************************************/
/*                                                                    */
SET &INPT1 = &STR(//GO.S2KSCHM DD DISP=SHR,DSN=)
SET &INPT2 = &SYSPREF&STR(.)&S2KSCHM
SET &INPTCARD = &STR(&INPT1&INPT2)
SET &OUPT1 = &STR(//GO.S2KTEMP DD DISP=SHR,DSN=)
SET &OUPT2 = &SYSPREF&STR(.)&S2KTEMP
SET &OUPTCARD = &STR(&OUPT1&OUPT2)
SET &INFL1 = &STR(//GO.IDMSMDL DD DISP=SHR,DSN=)
SET &INFL2 = &SYSPREF&STR(.)&IDMSMODL
SET &INFLCARD = &STR(&INFL1&INFL2)
SET &DDCARD = &STR(//GO.SYSIN DD *)
SET &INPTDATA = &STARTNUM&STR( )&INCNUM
```

```
/*                                                                    */
/**********************************************************************/
/*          BUILD IEBGENER JCL CARDS                                  */
/**********************************************************************/
/*                                                                    */
SET &GENER = &STR(//STEP2 EXEC PGM=IEBGENER)
SET &GENSYS = &STR(//SYSPRINT DD SYSOUT=*)
SET &DUMMY = &STR(//SYSIN DD DUMMY)
SET &SYSUT1A = &STR(//SYSUT1 DD DISP=SHR,DSN=)
SET &SYSUT1B = &SYSPREF&STR(.)&IDMSMODL
SET &SYSUT1 = &STR(&SYSUT1A&SYSUT1B)
SET &SYSUT2 = &STR(//SYSUT2 DD SYSOUT=*)
SET &SLASHEND = &STR(//)
/*                                                                    */
/**********************************************************************/
/*       BUILD THE OUTPUT FILE USING THE PREVIOUSLY CARDS             */
/*   CLIST IS EXPECTING TO FIND A DATASET WITH USERID.IDMS.DATA       */
/**********************************************************************/
/*                                                                    */
FREE F(JCL)
ALLOC F(JCL) DA(&SYSUID..IDMS.DATA) OLD
OPENFILE JCL OUTPUT
SET &JCL = &STR(&JOBCARDA)
PUTFILE JCL
SET &JCL = &STR(&JOBCARDB)
PUTFILE JCL
SET &JCL = &STR(&JOBCARDC)
PUTFILE JCL
SET &JCL = &STR(&THESIS1)
PUTFILE JCL
SET &JCL = &STR(&STEPLIB)
PUTFILE JCL
SET &JCL = &STR(&SLASH1)
PUTFILE JCL
SET &JCL = &STR(&SYSPRINT)
PUTFILE JCL
SET &JCL = &STR(&INPTCARD)
PUTFILE JCL
SET &JCL = &STR(&OUPTCARD)
PUTFILE JCL
SET &JCL = &STR(&INFLCARD)
PUTFILE JCL
SET &JCL = &STR(&DDCARD)
PUTFILE JCL
SET &JCL = &STR(&INPTDATA)
PUTFILE JCL
SET &JCL = &STR(&SLASH1)
PUTFILE JCL
SET &JCL = &STR(&GENER)
PUTFILE JCL
SET &JCL = &STR(&GENSYS)
PUTFILE JCL
SET &JCL = &STR(&DUMMY)
```

```
PUTFILE JCL
SET &JCL = &STR(&SYSUT1)
PUTFILE JCL
SET &JCL = &STR(&SYSUT2)
PUTFILE JCL
SET &JCL = &STR(&SLASH1)
PUTFILE JCL
SET &JCL = &STR(&SLASHEND)
PUTFILE JCL
CLOSFILE JCL
/*                                                                    */
/**********************************************************************/
/*              SUBMIT THE NEWLY CREATED FILE                         */
/**********************************************************************/
/*                                                                    */
CONTROL MSG
SUBMIT &SYSUID..IDMS.DATA
CONTROL NOMSG
FREE F(JCL)
EXIT
```

CLIST Program Name: Idmsout Author: Capt. Jerry Owens

```
PROC 6 USERDEFS SCHEMA IDMSMODL DMCL SUBSCHEM MSGCLASS
CONTROL NOMSG
/*                                                                  */
/*  The Idmsout Clist builds the necessary JCL to execute the       */
/*  Idmsout PL/1 module in the background                           */
/*                                                                  */
/*******************************************************************/
/*          BUILD JOB CARD INFORMATION                             */
/*******************************************************************/
/*                                                                  */
SET &JOBCARD1 = &STR(//)&SYSUID&STR(IDMS JOB )&SUBSTR(1:11,(4308,210),)
SET &JOBCARD2 = &SYSUID&STR(,CLASS=S,MSGCLASS=)&MSGCLASS&STR(,)
SET &JOBCARD3 = &STR(// NOTIFY=)&SYSUID,&STR(MSGLEVEL=(1,1),TIME=5,)
SET &JOBCARD4 = &STR(REGION=500K)
SET &JOBCARDA = &STR(&JOBCARD1&JOBCARD2)
SET &JOBCARDB = &STR(&JOBCARD3&JOBCARD4)
SET &JOBCARDC = &STR(/*JOBPARM S=NAS1)
/*                                                                  */
/*******************************************************************/
/*          BUILD STEPLIB INFORMATION                              */
/*******************************************************************/
/*                                                                  */
SET &IDMSOUT = &STR(//IDMSOUT EXEC PGM=IDMSOUT)
SET &STEPLIB = &STR(//STEPLIB DD DSN=A740308.JGO.LOAD,DISP=SHR)
SET &SLASH1 = &STR(/*)
SET &SYSPRINT = &STR(//GO.SYSPRINT DD SYSOUT=*)
/*                                                                  */
/*******************************************************************/
/*          BUILD INPUT FILE(S) INFORMATION                        */
/*******************************************************************/
/*                                                                  */
SET &USER1 = &STR(//GO.USRDEFS DD DISP=SHR,DSN=)
SET &USER2 = &SYSPREF&STR(.)&USERDEFS
SET &USERCARD = &STR(&USER1&USER2)
SET &OUT1 = &STR(//GO.SCHEMA DD DISP=SHR,DSN=)
SET &OUT2 = &SYSPREF&STR(.)&SCHEMA
SET &OUTCARD = &STR(&OUT1&OUT2)
SET &INFL1 = &STR(//GO.IDMSMDL DD DISP=SHR,DSN=)
SET &INFL2 = &SYSPREF&STR(.)&IDMSMODL
SET &INFLCARD = &STR(&INFL1&INFL2)
SET &DMCL1 = &STR(//GO.DMCLFLE DD DISP=SHR,DSN=)
SET &DMCL2 = &SYSPREF&STR(.)&DMCL
SET &DMCLCARD = &STR(&DMCL1&DMCL2)
SET &SUBS1 = &STR(//GO.SUBS DD DISP=SHR,DSN=)
SET &SUBS2 = &SYSPREF&STR(.)&SUBSCHEM
SET &SUBSCARD = &STR(&SUBS1&SUBS2)
```

```
/*****************************************************************/
/*          BUILD IEBGENER JCL CARDS                             */
/*****************************************************************/
/*                                                               */
SET &GENER2 = &STR(//STEP2 EXEC PGM=IEBGENER)
SET &GENER3 = &STR(//STEP3 EXEC PGM=IEBGENER)
SET &GENER4 = &STR(//STEP4 EXEC PGM=IEBGENER)
SET &GENSYS = &STR(//SYSPRINT DD SYSOUT=*)
SET &DUMMY = &STR(//SYSIN DD DUMMY)
SET &SYSUT1A = &STR(//SYSUT1 DD DISP=SHR,DSN=)
SET &SYSUT1B = &SYSPREF&STR(.)&SCHEMA
SET &SYSUT1C = &SYSPREF&STR(.)&DMCL
SET &SYSUT1D = &SYSPREF&STR(.)&SUBSCHEM
SET &GEN2 = &STR(&SYSUT1A&SYSUT1B)
SET &GEN3 = &STR(&SYSUT1A&SYSUT1C)
SET &GEN4 = &STR(&SYSUT1A&SYSUT1D)
SET &SYSUT2 = &STR(//SYSUT2 DD SYSOUT=*)
SET &SLASHEND = &STR(//)
/*                                                               */
/*****************************************************************/
/*       BUILD THE OUTPUT FILE USING THE PREVIOUSLY CARDS        */
/*  CLIST IS EXPECTING TO FIND A DATASET WITH USERID.IDMS.DATA   */
/*****************************************************************/
/*                                                               */
FREE F(JCL)
ALLOC F(JCL) DA(&SYSUID..IDMS.DATA) OLD
OPENFILE JCL OUTPUT
SET &JCL = &STR(&JOBCARDA)
PUTFILE JCL
SET &JCL = &STR(&JOBCARDB)
PUTFILE JCL
SET &JCL = &STR(&JOBCARDC)
PUTFILE JCL
SET &JCL = &STR(&IDMSOUT)
PUTFILE JCL
SET &JCL = &STR(&STEPLIB)
PUTFILE JCL
SET &JCL = &STR(&SLASH1)
PUTFILE JCL
SET &JCL = &STR(&SYSPRINT)
PUTFILE JCL
SET &JCL = &STR(&USERCARD)
PUTFILE JCL
SET &JCL = &STR(&OUTCARD)
PUTFILE JCL
SET &JCL = &STR(&INFLCARD)
PUTFILE JCL
SET &JCL = &STR(&DMCLCARD)
PUTFILE JCL
SET &JCL = &STR(&SUBSCARD)
PUTFILE JCL
SET &JCL = &STR(&SLASH1)
PUTFILE JCL
```

```
SET &JCL = &STR(&GENER2)
PUTFILE JCL
SET &JCL = &STR(&GENSYS)
PUTFILE JCL
SET &JCL = &STR(&DUMMY)
PUTFILE JCL
SET &JCL = &STR(&GEN2)
PUTFILE JCL
SET &JCL = &STR(&SYSUT2)
PUTFILE JCL
SET &JCL = &STR(&SLASH1)
PUTFILE JCL
SET &JCL = &STR(&SLASH1)
PUTFILE JCL
SET &JCL = &STR(&GENER3)
PUTFILE JCL
SET &JCL = &STR(&GENSYS)
PUTFILE JCL
SET &JCL = &STR(&DUMMY)
PUTFILE JCL
SET &JCL = &STR(&GEN3)
PUTFILE JCL
SET &JCL = &STR(&SYSUT2)
PUTFILE JCL
SET &JCL = &STR(&SLASH1)
PUTFILE JCL
SET &JCL = &STR(&GENER4)
PUTFILE JCL
SET &JCL = &STR(&GENSYS)
PUTFILE JCL
SET &JCL = &STR(&DUMMY)
PUTFILE JCL
SET &JCL = &STR(&GEN4)
PUTFILE JCL
SET &JCL = &STR(&SYSUT2)
PUTFILE JCL
SET &JCL = &STR(&SLASH1)
PUTFILE JCL
SET &JCL = &STR(&SLASHEND)
PUTFILE JCL
CLOSFILE JCL
/*                                                                  */
/********************************************************************/
/*              SUBMIT THE NEWLY CREATED FILE                       */
/********************************************************************/
/*                                                                  */
CONTROL MSG
SUBMIT &SYSUID..IDMS.DATA
CONTROL NOMSG
FREE F(JCL)
EXIT
```

Appendix G

Automated Hierarchical - CODASYL Database Interface

Schema Translator Sample Output

File Name: USERDEFS

```
AFITSCHM
1
CAPT JERRY OWENS
513-255-6321
AFITFILE
3330B
AFIT-REGION
4000
4099
AFITDMCL
AFITSUB1
```

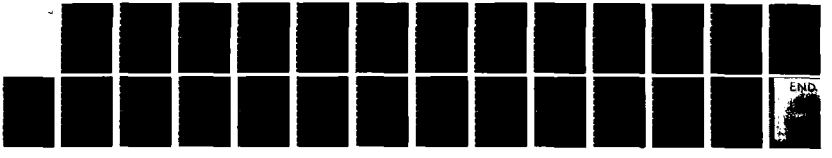
File Name: S2kschem

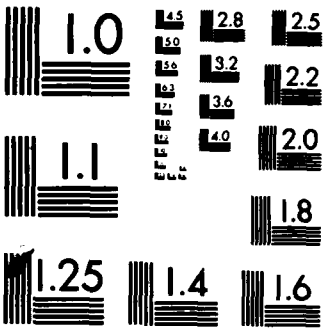
```
SYSTEM RELEASE NUMBER  10.1
DATA BASE NAME IS      EMPLOYEE
DEFINITION NUMBER           9
DATA BASE CYCLE NUMBER      15
     1* EMPLOYEE NUMBER (INTEGER NUMBER 9999)
     2* LAST NAME (CHAR X(10) WITH  FEW FUTURE OCCURRENCES )
     3* FORENAME (NON-KEY CHAR X(20))
     4* HIRE DATE (DATE)
     5* BIRTHDAY (DATE)
     6* SOCIAL SECURITY NUMBER (NON-KEY CHAR X(11))
     7* SEX (CHAR X(6) WITH MANY FUTURE OCCURRENCES )
     8* ETHNIC ORIGIN (CHAR X(9) WITH SOME FUTURE OCCURRENCES )
     9* EMPLOYEE STATUS (CHAR X(9) WITH MANY FUTURE OCCURRENCES )
    10* OFFICE-EXTENSION (NON-KEY CHAR X(9))
    11* ACCRUED VACATION (NON-KEY DECIMAL NUMBER 999.99)
    12* ACCRUED SICK LEAVE (NON-KEY DECIMAL NUMBER 999.99)
    13* SECURITY CLEARANCE (INTEGER NUMBER 999 WITH MANY FUTURE OCCURR
        ENCES )
    14* STREET ADDRESS (NON-KEY CHAR X(20))
    15* CITY-STATE (NON-KEY CHAR X(15))
    16* ZIP CODE (CHAR X(5) WITH  FEW FUTURE OCCURRENCES )
   100* POSITION WITHIN COMPANY (RECORD)
     101* POSITION TITLE (NON-KEY CHAR X(10) IN 100)
     102* DEPARTMENT (CHAR X(14) IN 100 WITH SOME FUTURE OCCURRENCES )

     103* MANAGER (C.  XXX IN 100 WITH  FEW FUTURE OCCURRENCES )
     104* POSITION TYPE (CHAR X(12) IN 100 WITH SOME FUTURE OCCURRENCE
          S )
     105* START DATE (DATE IN 100)
     106* END DATE (NON-KEY DATE IN 100)
     110* SALARY WITHIN POSITION (RECORD IN 100)
       111* PAY RATE (MONEY $9999.99 IN 110)
       112* PAY SCHEDULE (CHAR X(7) IN 110)
       113* EFFECTIVE DATE (DATE IN 110)
       114* CURRENT DEDUCTION (NON-KEY MONEY $9999.99 IN 110)
       120* MONTHLY PAYROLL ACCOUNTING (RECORD IN 110)
         121* PAYROLL MONTH (DATE IN 120)
         122* REGULAR HOURS (NON-KEY DECIMAL NUMBER 999.99 IN 120)
         123* OVERTIME HOURS (NON-KEY DECIMAL NUMBER 999.99 IN 120)
         124* GROSS PAY (NON-KEY MONEY $9999.99 IN 120)
         125* FEDERAL TAX DEDUCTION (NON-KEY MONEY $9999.99 IN 120)
         126* NET PAY (NON-KEY MONEY $9999.99 IN 120)
     130* ADDITIONAL INFORMATION (RECORD IN 100)
       131* LINE NUMBER (DECIMAL NUMBER 99.9 IN 130)
       132* COMMENT TEXT (NON-KEY TEXT X(7) IN 130)
   200* JOB SKILLS (RECORD)
     201* SKILL TYPE (CHAR X(12) IN 200 WITH SOME FUTURE OCCURRENCES )
     202* PROFICIENCY (NON-KEY CHAR X(5) IN 200)
```

```
  203* YEARS OF EXPERIENCE (NON-KEY INTEGER NUMBER 99 IN 200)
300* PERSONAL INTERESTS (RECORD)
  301* INTEREST (CHAR X(12) IN 300 WITH  FEW FUTURE OCCURRENCES )
  302* AFFILIATION (NON-KEY CHAR X(5) IN 300)
  303* COMMENT (NON-KEY TEXT X(5) IN 300)
400* EDUCATIONAL BACKGROUND (RECORD)
  410* EDUCATION (RECORD IN 400)
    411* SCHOOL (CHAR X(15) IN 410)
    412* DEGREE/CERTIFICATE (CHAR X(7) IN 410 WITH  FEW FUTURE OCCU
         RRENCES )
    413* DATE COMPLETED (DATE IN 410)
    414* MAJOR FIELD (NON-KEY CHAR X(16) IN 410)
    415* MINOR FIELD (NON-KEY CHAR X(12) IN 410)
  420* TRAINING (RECORD IN 400)
    421* SOURCE (NON-KEY CHAR X(12) IN 420)
    422* CLASS NAME (CHAR X(12) IN 420 WITH  FEW FUTURE OCCURRENCES
          )
    423* DATE ACCOMPLISHED (DATE IN 420)
```

AD-A138 012 AUTOMATED HIERARCHICAL TO CODASYL (CONFERENCE ON DATA SYSTEMS LANGUAGES) D..(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. J G OWENS 3/3

UNCLASSIFIED 16 DEC 83 AFIT/GCS/EE/83D-17 F/G 9/2 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

File Name: Rdschm Log Report

```
*****************************************************************************************
*       DATA BASE NAME IS     EMPLOYEE                                                  *
*                                                                                       *
*      0       EMPLOYEE                          REC                                    *
*****************************************************************************************


*****************************************************************************************
*             1*  EMPLOYEE NUMBER (INTEGER NUMBER 9999)                                *
*                                                                                       *
*       1       EMPLOYEE-NUMBER              0    PIC  9999          COMP-3            *
*****************************************************************************************


*****************************************************************************************
*             2*  LAST NAME (CHAR X(10) WITH  FEW FUTURE OCCURRENCES )                 *
*                                                                                       *
*       2       LAST-NAME                    0    PIC  X(10)                           *
*****************************************************************************************


*****************************************************************************************
*             3*  FORENAME (NON-KEY CHAR X(20))                                        *
*                                                                                       *
*       3       FORENAME                     0    PIC  X(20)                           *
*****************************************************************************************


*****************************************************************************************
*             4*  HIRE DATE (DATE)                                                     *
*                                                                                       *
*       4       HIRE-DATE                    0    PIC  9(7)          COMP-3            *
*****************************************************************************************
```

```
*************************************************************************************************
*             5*  BIRTHDAY (DATE)                                                                  *
*                                                                                                 *
*       5       BIRTHDAY                          0    PIC  9(7)           COMP-3                *
*************************************************************************************************

*************************************************************************************************
*             6*  SOCIAL SECURITY NUMBER (NON-KEY CHAR X(11))                                     *
*                                                                                                 *
*       6       SOCIAL-SECURITY-NUMBER            0    PIC  X(11)                               *
*************************************************************************************************

*************************************************************************************************
*             7*  SEX (CHAR X(6) WITH MANY FUTURE OCCURRENCES )                                   *
*                                                                                                 *
*       7       SEX                               0    PIC  X(6)                                *
*************************************************************************************************

*************************************************************************************************
*             8*  ETHNIC ORIGIN (CHAR X(9) WITH SOME FUTURE OCCURRENCES )                         *
*                                                                                                 *
*       8       ETHNIC-ORIGIN                     0    PIC  X(9)                                *
*************************************************************************************************

*************************************************************************************************
*             9*  EMPLOYEE STATUS (CHAR X(9) WITH MANY FUTURE OCCURRENCES )                       *
*                                                                                                 *
*       9       EMPLOYEE-STATUS                   0    PIC  X(9)                                *
*************************************************************************************************

*************************************************************************************************
*            10*  OFFICE-EXTENSION (NON-KEY CHAR X(9))                                            *
*                                                                                                 *
*       10      OFFICE-EXTENSION                  0    PIC  X(9)                                *
*************************************************************************************************
```

```
*****************************************************************************************
*            11*  ACCRUED VACATION (NON-KEY DECIMAL NUMBER 999.99)                      *
*                                                                                       *
*       11       ACCRUED-VACATION              0    PIC  999V99          COMP-3        *
*****************************************************************************************

*****************************************************************************************
*            12*  ACCRUED SICK LEAVE (NON-KEY DECIMAL NUMBER 999.99)                    *
*                                                                                       *
*       12       ACCRUED-SICK-LEAVE            0    PIC  999V99          COMP-3        *
*****************************************************************************************

*****************************************************************************************
*            13*  SECURITY CLEARANCE (INTEGER NUMBER 999 WITH MANY FUTURE OCCURRENCES )  *
*                                                                                       *
*       13       SECURITY-CLEARANCE            0    PIC  999             COMP-3        *
*****************************************************************************************

*****************************************************************************************
*            14*  STREET ADDRESS (NON-KEY CHAR X(20))                                   *
*                                                                                       *
*       14       STREET-ADDRESS                0    PIC  X(20)                         *
*****************************************************************************************

*****************************************************************************************
*            15*  CITY-STATE (NON-KEY CHAR X(15))                                       *
*                                                                                       *
*       15       CITY-STATE                    0    PIC  X(15)                         *
*****************************************************************************************

*****************************************************************************************
*            16*  ZIP CODE (CHAR X(5) WITH  FEW FUTURE OCCURRENCES )                    *
*                                                                                       *
*       16       ZIP-CODE                      0    PIC  X(5)                          *
*****************************************************************************************
```

```
*****************************************************************************************
*           100*  POSITION WITHIN COMPANY (RECORD)                                       *
*                                                                                       *
*       100     POSITION-WITHIN                 0    REC                                *
*****************************************************************************************


*****************************************************************************************
*             101*  POSITION TITLE (NON-KEY CHAR X(10) IN 100)                          *
*                                                                                       *
*       101     POSITION-TITLE                  100  PIC  X(10)                        *
*****************************************************************************************


*****************************************************************************************
*             102*  DEPARTMENT (CHAR X(14) IN 100 WITH SOME FUTURE OCCURRENCES )       *
*                                                                                       *
*       102     DEPARTMENT                      100  PIC  X(14)                        *
*****************************************************************************************


*****************************************************************************************
*             103*  MANAGER (CHAR XXX IN 100 WITH  FEW FUTURE OCCURRENCES )            *
*                                                                                       *
*       103     MANAGER                         100  PIC  XXX                           *
*****************************************************************************************


*****************************************************************************************
*             104*  POSITION TYPE (CHAR X(12) IN 100 WITH SOME FUTURE OCCURRENCES )     *
*                                                                                       *
*       104     POSITION-TYPE                   100  PIC  X(12)                        *
*****************************************************************************************


*****************************************************************************************
*             105*  START DATE (DATE IN 100)                                           *
*                                                                                       *
*       105     START-DATE                      100  PIC  9(7)          COMP-3         *
*****************************************************************************************
```

```
*****************************************************************************************
*              106*  END DATE (NON-KEY DATE IN 100)                                      *
*                                                                                       *
*        106      END-DATE                        100  PIC  9(7)          COMP-3       *
*****************************************************************************************


*****************************************************************************************
*              110*  SALARY WITHIN POSITION (RECORD IN 100)                              *
*                                                                                       *
*        110      SALARY-WITHIN-PO                100  REC                              *
*****************************************************************************************


*****************************************************************************************
*               111*  PAY RATE (MONEY $9999.99 IN 110)                                  *
*                                                                                       *
*        111      PAY-RATE                        110  PIC  9999V99        COMP-3       *
*****************************************************************************************


*****************************************************************************************
*               112*  PAY SCHEDULE (CHAR X(7) IN 110)                                   *
*                                                                                       *
*        112      PAY-SCHEDULE                    110  PIC  X(7)                        *
*****************************************************************************************


*****************************************************************************************
*               113*  EFFECTIVE DATE (DATE IN 110)                                      *
*                                                                                       *
*        113      EFFECTIVE-DATE                  110  PIC  9(7)          COMP-3       *
*****************************************************************************************


*****************************************************************************************
*               114*  CURRENT DEDUCTION (NON-KEY MONEY $9999.99 IN 110)                 *
*                                                                                       *
*        114      CURRENT-DEDUCTION               110  PIC  9999V99        COMP-3       *
*****************************************************************************************
```

```
*****************************************************************************************
*                120*  MONTHLY PAYROLL ACCOUNTING (RECORD IN 110)                       *
*                                                                                       *
*        120      MONTHLY-PAYROLL                 110  REC                              *
*****************************************************************************************

*****************************************************************************************
*                121*  PAYROLL MONTH (DATE IN 120)                                      *
*                                                                                       *
*        121      PAYROLL-MONTH                   120  PIC  9(7)          COMP-3       *
*****************************************************************************************

*****************************************************************************************
*                122*  REGULAR HOURS (NON-KEY DECIMAL NUMBER 999.99 IN 120)             *
*                                                                                       *
*        122      REGULAR-HOURS                   120  PIC  999V99        COMP-3       *
*****************************************************************************************

*****************************************************************************************
*                123*  OVERTIME HOURS (NON-KEY DECIMAL NUMBER 999.99 IN 120)            *
*                                                                                       *
*        123      OVERTIME-HOURS                  120  PIC  999V99        COMP-3       *
*****************************************************************************************

*****************************************************************************************
*                124*  GROSS PAY (NON-KEY MONEY $9999.99 IN 120)                        *
*                                                                                       *
*        124      GROSS-PAY                       120  PIC  9999V99       COMP-3       *
*****************************************************************************************

*****************************************************************************************
*                125*  FEDERAL TAX DEDUCTION (NON-KEY MONEY $9999.99 IN 120)            *
*                                                                                       *
*        125      FEDERAL-TAX-DEDUCTION           120  PIC  9999V99       COMP-3       *
*****************************************************************************************
```

```
*****************************************************************************************************
*                126*  NET PAY (NON-KEY MONEY $9999.99 IN 120)                                       *
*                                                                                                     *
*       126      NET-PAY                          120  PIC  9999V99          COMP-3                  *
*****************************************************************************************************


*****************************************************************************************************
*             130*  ADDITIONAL INFORMATION (RECORD IN 100)                                          *
*                                                                                                     *
*       130      ADDITIONAL-INFOR                 100  REC                                          *
*****************************************************************************************************


*****************************************************************************************************
*               131*  LINE NUMBER (DECIMAL NUMBER 99.9 IN 130)                                      *
*                                                                                                     *
*       131      LINE-NUMBER                      130  PIC  99V9            COMP-3                  *
*****************************************************************************************************


*****************************************************************************************************
*               132*  COMMENT TEXT (NON-KEY TEXT X(7) IN 130)                                       *
*                                                                                                     *
*       132      COMMENT-TEXT                     130  PIC  X(7)                                     *
*****************************************************************************************************


*****************************************************************************************************
*            200*  JOB SKILLS (RECORD)                                                              *
*                                                                                                     *
*       200      JOB-SKILLS                       0    REC                                          *
*****************************************************************************************************


*****************************************************************************************************
*             201*  SKILL TYPE (CHAR X(12) IN 200 WITH SOME FUTURE OCCURRENCES )                    *
*                                                                                                     *
*       201      SKILL-TYPE                       200  PIC  X(12)                                    *
*****************************************************************************************************
```

```
*****************************************************************************************
*             202*  PROFICIENCY (NON-KEY CHAR X(5) IN 200)                              *
*                                                                                       *
*       202     PROFICIENCY                   200  PIC  X(5)                           *
*****************************************************************************************

*****************************************************************************************
*             203*  YEARS OF EXPERIENCE (NON-KEY INTEGER NUMBER 99 IN 200)              *
*                                                                                       *
*       203     YEARS-OF-EXPERIENCE           200  PIC  99          COMP-3             *
*****************************************************************************************

*****************************************************************************************
*           300*  PERSONAL INTERESTS (RECORD)                                           *
*                                                                                       *
*       300     PERSONAL-INTERES              0    REC                                  *
*****************************************************************************************

*****************************************************************************************
*             301*  INTEREST (CHAR X(12) IN 300 WITH  FEW FUTURE OCCURRENCES )          *
*                                                                                       *
*       301     INTEREST                      300  PIC  X(12)                          *
*****************************************************************************************

*****************************************************************************************
*             302*  AFFILIATION (NON-KEY CHAR X(5) IN 300)                               *
*                                                                                       *
*       302     AFFILIATION                   300  PIC  X(5)                           *
*****************************************************************************************

*****************************************************************************************
*             303*  COMMENT (NON-KEY TEXT X(5) IN 300)                                   *
*                                                                                       *
*       303     COMMENT                       300  PIC  X(5)                           *
*****************************************************************************************
```

```
*****************************************************************************************************
*            400*  EDUCATIONAL BACKGROUND (RECORD)                                                     *
*                                                                                                      *
*       400      EDUCATIONAL-BACK              0    REC                                               *
*****************************************************************************************************


*****************************************************************************************************
*             410*  EDUCATION (RECORD IN 400)                                                         *
*                                                                                                      *
*       410      EDUCATION                     400  REC                                               *
*****************************************************************************************************


*****************************************************************************************************
*               411*  SCHOOL (CHAR X(15) IN 410)                                                     *
*                                                                                                      *
*       411      SCHOOL                        410  PIC  X(15)                                        *
*****************************************************************************************************


*****************************************************************************************************
*               412*  DEGREE/CERTIFICATE (CHAR X(7) IN 410 WITH  FEW FUTURE OCCURRENCES              *
*       )                                                                                              *
*       412      DEGREE-CERTIFICATE            410  PIC  X(7)                                         *
*****************************************************************************************************


*****************************************************************************************************
*               413*  DATE COMPLETED (DATE IN 410)                                                   *
*                                                                                                      *
*       413      DATE-COMPLETED                410  PIC  9(7)          COMP-3                         *
*****************************************************************************************************


*****************************************************************************************************
*               414*  MAJOR FIELD (NON-KEY CHAR X(16) IN 410)                                        *
*                                                                                                      *
*       414      MAJOR-FIELD                   410  PIC  X(16)                                        *
*****************************************************************************************************
```

```
*****************************************************************************************
*              415*  MINOR FIELD (NON-KEY CHAR X(12) IN 410)                            *
*                                                                                       *
*       415     MINOR-FIELD                    410  PIC  X(12)                         *
*****************************************************************************************


*****************************************************************************************
*             420*  TRAINING (RECORD IN 400)                                           *
*                                                                                       *
*       420     TRAINING                       400  REC                                *
*****************************************************************************************


*****************************************************************************************
*              421*  SOURCE (NON-KEY CHAR X(12) IN 420)                                 *
*                                                                                       *
*       421     SOURCE                         420  PIC  X(12)                         *
*****************************************************************************************


*****************************************************************************************
*              422*  CLASS NAME (CHAR X(12) IN 420 WiTH  FEW FUTURE OCCURRENCES )       *
*                                                                                       *
*       422     CLASS-NAME                     420  PIC  X(12)                         *
*****************************************************************************************


*****************************************************************************************
*              423*  DATE ACCOMPLISHED (DATE IN 420)                                    *
*                                                                                       *
*       423     DATE-ACCOMPLISHED              420  PIC  9(7)          COMP-3          *
*****************************************************************************************
```

File Name: S2ktemp

0	EMPLOYEE		REC		
1	EMPLOYEE-NUMBER	0	PIC	9999	COM
2	LAST-NAME	0	PIC	X(10)	
3	FORENAME	0	PIC	X(20)	
4	HIRE-DATE	0	PIC	9(7)	COM
5	BIRTHDAY	0	PIC	9(7)	COM
6	SOCIAL-SECURITY-NUMBER	0	PIC	X(11)	
7	SEX	0	PIC	X(6)	
8	ETHNIC-ORIGIN	0	PIC	X(9)	
9	EMPLOYEE-STATUS	0	PIC	X(9)	
10	OFFICE-EXTENSION	0	PIC	X(9)	
11	ACCRUED-VACATION	0	PIC	999V99	COM
12	ACCRUED-SICK-LEAVE	0	PIC	999V99	COM
13	SECURITY-CLEARANCE	0	XIC	999	COM
14	STREET-ADDRESS	0	PIC	X(20)	
15	CITY-STATE	0	PIC	X(15)	
16	ZIP-CODE	0	PIC	X(5)	
100	POSITION-WITHIN	0	REC		
101	POSITION-TITLE	100	PIC	X(10)	
102	DEPARTMENT	100	PIC	X(14)	
103	MANAGER	100	PIC	XXX	
104	POSITION-TYPE	100	PIC	X(12)	
105	START-DATE	100	PIC	9(7)	COM
106	END-DATE	100	PIC	9(7)	COM
110	SALARY-WITHIN-PO	100	REC		
111	PAY-RATE	110	PIC	9999V99	COM
112	PAY-SCHEDULE	110	PIC	X(7)	
113	EFFECTIVE-DATE	110	PIC	9(7)	COM
114	CURRENT-DEDUCTION	110	PIC	9999V99	COM
120	MONTHLY-PAYROLL	110	REC		
121	PAYROLL-MONTH	120	PIC	9(7)	COM
122	REGULAR-HOURS	120	PIC	999V99	COM
123	OVERTIME-HOURS	120	PIC	999V99	COM
124	GROSS-PAY	120	PIC	9999V99	COM
125	FEDERAL-TAX-DEDUCTION	120	PIC	9999V99	COM
126	NET-PAY	120	PIC	9999V99	COM
130	ADDITIONAL-INFOR	100	REC		
131	LINE-NUMBER	130	PIC	99V9	COM
132	COMMENT-TEXT	130	PIC	X(7)	
200	JOB-SKILLS	0	REC		
201	SKILL-TYPE	200	PIC	X(12)	
202	PROFICIENCY	200	PIC	X(5)	
203	YEARS-OF-EXPERIENCE	200	PIC	99	COM
300	PERSONAL-INTERES	0	REC		
301	INTEREST	300	PIC	X(12)	
302	AFFILIATION	300	PIC	X(5)	
303	COMMENT	300	PIC	X(5)	
400	EDUCATIONAL-BACK	0	REC		

410	EDUCATION	400	REC		
411	SCHOOL	410	PIC	X(15)	
412	DEGREE-CERTIFICATE	410	PIC	X(7)	
413	DATE-COMPLETED	410	PIC	9(7)	COM
414	MAJOR-FIELD	410	PIC	X(16)	
415	MINOR-FIELD	410	PIC	X(12)	
420	TRAINING	400	REC		
421	SOURCE	420	PIC	X(12)	
422	CLASS-NAME	420	PIC	X(12)	
423	DATE-ACCOMPLISHED	420	PIC	9(7)	COM

File Name: Idmsmodl

```
10  48
R EMPLOYEE          100 0          1
C   EMPLOYEE-NUMBER
I 1    EMPLOYEE-NUMBER                PIC  9999          COMP-3 0
I 2    LAST-NAME                      PIC  X(10)              0
I 3    FORENAME                       PIC  X(20)              0
I 4    HIRE-DATE                      PIC  9(7)          COMP-3 0
I 5    BIRTHDAY                       PIC  9(7)          COMP-3 0
I 6    SOCIAL-SECURITY-NUMBER         PIC  X(11)              0
I 7    SEX                            PIC  X(6)               0
I 8    ETHNIC-ORIGIN                  PIC  X(9)               0
I 9    EMPLOYEE-STATUS                PIC  X(9)               0
I 10   OFFICE-EXTENSION               PIC  X(9)               0
I 11   ACCRUED-VACATION               PIC  999V99        COMP-3 0
I 12   ACCRUED-SICK-LEAVE             PIC  999V99        COMP-3 0
I 13   SECURITY-CLEARANCE             PIC  999           COMP-3 0
I 14   STREET-ADDRESS                 PIC  X(20)              0
I 15   CITY-STATE                     PIC  X(15)              0
I 16   ZIP-CODE                       PIC  X(5)               0
R POSITION-WITHIN   105 100    0    17
V   RELATION-1          EMPLOYEE          POSITION-WITHIN
I 101  POSITION-TITLE                 PIC  X(10)            100
I 102  DEPARTMENT                     PIC  X(14)            100
I 103  MANAGER                        PIC  XXX              100
I 104  POSITION-TYPE                  PIC  X(12)            100
I 105  START-DATE                     PIC  9(7)          COMP-3 100
I 106  END-DATE                       PIC  9(7)          COMP-3 100
R SALARY-WITHIN-PO  120 110  100    23
V   RELATION-2          POSITION-WITHIN     SALARY-WITHIN-PO
I 111  PAY-RATE                       PIC  9999V99       COMP-3 110
I 112  PAY-SCHEDULE                   PIC  X(7)             110
I 113  EFFECTIVE-DATE                 PIC  9(7)          COMP-3 110
I 114  CURRENT-DEDUCTION              PIC  9999V99       COMP-3 110
R MONTHLY-PAYROLL   130 120  110    27
V   RELATION-3          SALARY-WITHIN-PO    MONTHLY-PAYROLL
I 121  PAYROLL-MONTH                  PIC  9(7)          COMP-3 120
I 122  REGULAR-HOURS                  PIC  999V99        COMP-3 120
I 123  OVERTIME-HOURS                 PIC  999V99        COMP-3 120
I 124  GROSS-PAY                      PIC  9999V99       COMP-3 120
I 125  FEDERAL-TAX-DEDUCTION          PIC  9999V99       COMP-3 120
I 126  NET-PAY                        PIC  9999V99       COMP-3 120
R ADDITIONAL-INFOR  140 130  100    33
V   RELATION-4          POSITION-WITHIN     ADDITIONAL-INFOR
I 131  LINE-NUMBER                    PIC  99V9          COMP-3 130
I 132  COMMENT-TEXT                   PIC  X(7)             130
R JOB-SKILLS        150 200  0      35
V   RELATION-5          EMPLOYEE          JOB-SKILLS
I 201  SKILL-TYPE                     PIC  X(12)            200
```

```
I 202  PROFICIENCY                    PIC  X(5)                  200
I 203  YEARS-OF-EXPERIENCE            PIC  99              COMP-3 200
R PERSONAL-INTERES  160 300  0      38
V   RELATION-6          EMPLOYEE          PERSONAL-INTERES
I 301  INTEREST                       PIC  X(12)                300
I 302  AFFILIATION                    PIC  X(5)                 300
I 303  COMMENT-1                      PIC  X(5)                 300
R EDUCATIONAL-BACK  170 400  0       0
V   RELATION-7          EMPLOYEE          EDUCATIONAL-BACK
I      EDUCATIONAL-BACK-DUMMY         PIC X(4)                  400
R EDUCATION         180 410  400     41
V   RELATION-8          EDUCATIONAL-BACK    EDUCATION
I 411  SCHOOL                         PIC  X(15)                410
I 412  DEGREE-CERTIFICATE             PIC  X(7)                 410
I 413  DATE-COMPLETED                 PIC  9(7)            COMP-3 410
I 414  MAJOR-FIELD                    PIC  X(16)                410
I 415  MINOR-FIELD                    PIC  X(12)                410
R TRAINING          190 420  400     46
V   RELATION-9          EDUCATIONAL-BACK    TRAINING
I 421  SOURCE-1                       PIC  X(12)                420
I 422  CLASS-NAME                     PIC  X(12)                420
I 423  DATE-ACCOMPLISHED              PIC  9(7)            COMP-3 420
```

File Name: Schema

```
*
*   ************************************************************
*   *            SCHEMA DESCRIPTION STATEMENTS                 *
*   ************************************************************
*
 SCHEMA DESCRIPTION.

 SCHEMA NAME IS AFITSCHM          VERSION 1  .

 DATE.                            11/18/83.

 INSTALLATION.                    NAS 7000 - 2
                                  WRIGHT-PATTERSON AFB, OHIO.

 REMARKS.                         THIS IS A SAMPLE IDMS SCHEMA
                                  DERIVED FROM THE AFITSCHM
                                  DATABASE SCHEMA CURRENTLY
                                  INSTALLED UNDER S2000.

 AUTHOR.                          CAPT JERRY OWENS
                                  513-255-6321.
*
*   ************************************************************
*   *              FILE DESCRIPTHON STATEMENTS                 *
*   ************************************************************
*
 FILE DESCRIPTION.

 FILE NAME IS AFITFILE            ASSIGN TO AFITFILE
                                  DEVICE TYPE IS 3330B.
*
*   ************************************************************
*   *              AREA DESCRIPTION STATEMENTS                 *
*   ************************************************************
*
 AREA DESCRIPTION.

 AREA NAME IS AFIT-REGION
          RANGE IS     4000 THRU     4099
                                  WITHIN FILE AFITFILE
                                     FROM 1 THRU         100.
```

```
*
*   ************************************************************
*   *            RECORD DESCRIPTION STATEMENTS                 *
*   ************************************************************
*
RECORD DESCRIPTION.

RECORD NAME IS EMPLOYEE        .
RECORD ID IS  100.
LOCATION MODE IS CALC          USING EMPLOYEE-NUMBER
                               DUPLICATES ARE NOT ALLOWED.
WITHIN AFIT-REGION      AREA.
03  EMPLOYEE-NUMBER               PIC  9999          COMP-3.
03  LAST-NAME                     PIC  X(10)              .
03  FORENAME                      PIC  X(20)              .
03  HIRE-DATE                     PIC  9(7)          COMP-3.
03  BIRTHDAY                      PIC  9(7)          COMP-3.
03  SOCIAL-SECURITY-NUMBER        PIC  X(11)              .
03  SEX                           PIC  X(6)               .
03  ETHNIC-ORIGIN                 PIC  X(9)               .
03  EMPLOYEE-STATUS               PIC  X(9)               .
03  OFFICE-EXTENSION              PIC  X(9)               .
03  ACCRUED-VACATION              PIC  999V99        COMP-3.
03  ACCRUED-SICK-LEAVE            PIC  999V99        COMP-3.
03  SECURITY-CLEARANCE            PIC  999           COMP-3.
03  STREET-ADDRESS                PIC  X(20)              .
03  CITY-STATE                    PIC  X(15)              .
03  ZIP-CODE                      PIC  X(5)               .

RECORD NAME IS POSITION-WITHIN .
RECORD ID IS  110.
LOCATION MODE IS VIA RELATION-1      SET.
WITHIN AFIT-REGION      AREA.
03  POSITION-TITLE                PIC  X(10)              .
03  DEPARTMENT                    PIC  X(14)              .
03  MANAGER                       PIC  XXX                .
03  POSITION-TYPE                 PIC  X(12)              .
03  START-DATE                    PIC  9(7)          COMP-3.
03  END-DATE                      PIC  9(7)          COMP-3.

RECORD NAME IS SALARY-WITHIN-PO.
RECORD ID IS  120.
LOCATION MODE IS VIA RELATION-2      SET.
WITHIN AFIT-REGION      AREA.
03  PAY-RATE                      PIC  9999V99       COMP-3.
03  PAY-SCHEDULE                  PIC  X(7)               .
03  EFFECTIVE-DATE                PIC  9(7)          COMP-3.
03  CURRENT-DEDUCTION             PIC  9999V99       COMP-3.
```

```
RECORD NAME IS MONTHLY-PAYROLL .
RECORD ID IS  130.
LOCATION MODE IS VIA RELATION-3      SET.
WITHIN AFIT-REGION      AREA.
03  PAYROLL-MONTH                   PIC  9(7)          COMP-3.
03  REGULAR-HOURS                   PIC  999V99        COMP-3.
03  OVERTIME-HOURS                  PIC  999V99        COMP-3.
03  GROSS-PAY                       PIC  9999V99       COMP-3.
03  FEDERAL-TAX-DEDUCTION           PIC  9999V99       COMP-3.
03  NET-PAY                         PIC  9999V99       COMP-3.

RECORD NAME IS ADDITIONAL-INFOR.
RECORD ID IS  140.
LOCATION MODE IS VIA RELATION-4      SET.
WITHIN AFIT-REGION      AREA.
03  LINE-NUMBER                     PIC  99V9          COMP-3.
03  COMMENT-TEXT                    PIC  X(7)          .

RECORD NAME IS JOB-SKILLS      .
RECORD ID IS  150.
LOCATION MODE IS VIA RELATION-5      SET.
WITHIN AFIT-REGION      AREA.
03  SKILL-TYPE                      PIC  X(12)         .
03  PROFICIENCY                     PIC  X(5)          .
03  YEARS-OF-EXPERIENCE             PIC  99            COMP-3.

RECORD NAME IS PERSONAL-INTERES.
RECORD ID IS  160.
LOCATION MODE IS VIA RELATION-6      SET.
WITHIN AFIT-REGION      AREA.
03  INTEREST                        PIC  X(12)         .
03  AFFILIATION                     PIC  X(5)          .
03  COMMENT-1                       PIC  X(5)          .

RECORD NAME IS EDUCATIONAL-BACK.
RECORD ID IS  170.
LOCATION MODE IS VIA RELATION-7      SET.
WITHIN AFIT-REGION      AREA.
03  EDUCATIONAL-BACK-DUMMY          PIC X(4)           .

RECORD NAME IS EDUCATION      .
RECORD ID IS  180.
LOCATION MODE IS VIA RELATION-8      SET.
WITHIN AFIT-REGION      AREA.
03  SCHOOL                          PIC  X(15)         .
03  DEGREE-CERTIFICATE              PIC  X(7)          .
03  DATE-COMPLETED                  PIC  9(7)          COMP-3.
03  MAJOR-FIELD                     PIC  X(16)         .
03  MINOR-FIELD                     PIC  X(12)         .
```

```
RECORD NAME IS TRAINING        .
RECORD ID IS  190.
LOCATION MODE IS VIA RELATION-9       SET.
WITHIN AFIT-REGION      AREA.
03  SOURCE-1                       PIC  X(12)                .
03  CLASS-NAME                     PIC  X(12)                .
03  DATE-ACCOMPLISHED              PIC  9(7)            COMP-3.
*
*    ************************************************************
*    *              SET DESCRIPTION STATEMENTS                  *
*    ************************************************************
*
SET DESCRIPTION.

SET NAME IS RELATION-1        .
ORDER IS NEXT.
MODE IS CHAIN                     LINKED TO PRIOR.
OWNER IS EMPLOYEE                 NEXT DBKEY POSITION IS   1
                                  PRIOR DBKEY POSITION IS   2.
MEMBER IS POSITION-WITHIN         NEXT DBKEY POSITION IS   1
                                  PRIOR DBKEY POSITION IS   2
          .                       MANDATORY AUTOMATIC.



SET NAME IS RELATION-2        .
ORDER IS NEXT.
MODE IS CHAIN                     LINKED TO PRIOR.
OWNER IS POSITION-WITHIN          NEXT DBKEY POSITION IS   3
                                  PRIOR DBKEY POSITION IS   4.
MEMBER IS SALARY-WITHIN-PO        NEXT DBKEY POSITION IS   1
                                  PRIOR DBKEY POSITION IS   2
                                  MANDATORY AUTOMATIC.



SET NAME IS RELATION-3        .
ORDER IS NEXT.
MODE IS CHAIN                     LINKED TO PRIOR.
OWNER IS SALARY-WITHIN-PO         NEXT DBKEY POSITION IS   3
                                  PRIOR DBKEY POSITION IS   4.
MEMBER IS MONTHLY-PAYROLL         NEXT DBKEY POSITION IS   1
                                  PRIOR DBKEY POSITION IS   2
                                  MANDATORY AUTOMATIC.
```

```
SET NAME IS RELATION-4          .
ORDER IS NEXT.
MODE IS CHAIN                       LINKED TO PRIOR.
OWNER IS POSITION-WITHIN            NEXT DBKEY POSITION IS   5
                                    PRIOR DBKEY POSITION IS  6.
MEMBER IS ADDITIONAL-INFOR          NEXT DBKEY POSITION IS   1
                                    PRIOR DBKEY POSITION IS  2
                                    MANDATORY AUTOMATIC.


SET NAME IS RELATION-5          .
ORDER IS NEXT.
MODE IS CHAIN                       LINKED TO PRIOR.
OWNER IS EMPLOYEE                   NEXT DBKEY POSITION IS   3
                                    PRIOR DBKEY POSITION IS  4.
MEMBER IS JOB-SKILLS                NEXT DBKEY POSITION IS   1
                                    PRIOR DBKEY POSITION IS  2
                                    MANDATORY AUTOMATIC.


SET NAME IS RELATION-6          .
ORDER IS NEXT.
MODE IS CHAIN                       LINKED TO PRIOR.
OWNER IS EMPLOYEE                   NEXT DBKEY POSITION IS   5
                                    PRIOR DBKEY POSITION IS  6.
MEMBER IS PERSONAL-INTERES          NEXT DBKEY POSITION IS   1
                                    PRIOR DBKEY POSITION IS  2
                                    MANDATORY AUTOMATIC.


SET NAME IS RELATION-7          .
ORDER IS NEXT.
MODE IS CHAIN                       LINKED TO PRIOR.
OWNER IS EMPLOYEE                   NEXT DBKEY POSITION IS   7
                                    PRIOR DBKEY POSITION IS  8.
MEMBER IS EDUCATIONAL-BACK          NEXT DBKEY POSITION IS   1
                                    PRIOR DBKEY POSITION IS  2
                                    MANDATORY AUTOMATIC.


SET NAME IS RELATION-8          .
ORDER IS NEXT.
MODE IS CHAIN                       LINKED TO PRIOR.
OWNER IS EDUCATIONAL-BACK           NEXT DBKEY POSITION IS   3
                                    PRIOR DBKEY POSITION IS  4.
MEMBER IS EDUCATION                 NEXT DBKEY POSITION IS   1
                                    PRIOR DBKEY POSITION IS  2
                                    MANDATORY AUTOMATIC.
```

```
SET NAME IS RELATION-9
ORDER IS NEXT.
MODE IS CHAIN                   LINKED TO PRIOR.
OWNER IS EDUCATIONAL-BACK       NEXT DBKEY POSITION IS  5
                                PRIOR DBKEY POSITION IS  6.
MEMBER IS TRAINING              NEXT DBKEY POSITION IS  1
                                PRIOR DBKEY POSITION IS  2
                                MANDATORY AUTOMATIC.
```

File Name: DMCL

```
DEVICE-MEDIA DESCRIPTION.
DEVICE-MEDIA NAME IS AFITDMCL OF SCHEMA NAME AFITSCHM VERSION  1    .

AUTHOR.              CAPT JERRY OWENS              .
DATE.                11/18/83.

INSTALLATION.        NAS 7000 - 2
                     WRIGHT - PATTERSON AFB, OHIO.

REMARKS.             THIS IS A GENERAL FORM OF THE
                     DMCL. PLEASE MAKE THE DESIRED CHANGES.

BUFFER SECTION.
   BUFFER NAME IS GENERAL-BUFFER
       PAGE CONTAINS 4060 CHARACTERS
       BUFFER CONTAINS 8 PAGES.

AREA SECTION.
   COPY AFIT-REGION      AREA
       FROM SCHEMA NAME AFITSCHM VERSION 1
       BUFFER IS GENERAL-BUFFER.

JOURNAL SECTION.
   JOURNAL BLOCK CONTAINS 1000 CHARACTERS.
   FILE NAME IS TAPE-JOURNAL ASSIGN TO SYSJRNL
                        DEVICE TYPE IS 2400.
```

File Name: Subs

```
DELETE SUBSCHEMA NAME IS  AFITSUB1.
ADD SUBSCHEMA NAME IS AFITSUB1
    OF SCHEMA NAME IS AFITSCHM
      DMCL NAME IS AFITDMCL .
ADD AREA AFIT-REGION     .
ADD RECORD EMPLOYEE        .
ADD RECORD POSITION-WITHIN  .
ADD RECORD SALARY-WITHIN-PO .
ADD RECORD MONTHLY-PAYROLL  .
ADD RECORD ADDITIONAL-INFOR .
ADD RECORD JOB-SKILLS       .
ADD RECORD PERSONAL-INTERES .
ADD RECORD EDUCATIONAL-BACK .
ADD RECORD EDUCATION        .
ADD RECORD TRAINING         .
ADD SET RELATION-1      .
ADD SET RELATION-2      .
ADD SET RELATION-3      .
ADD SET RELATION-4      .
ADD SET RELATION-5      .
ADD SET RELATION-6      .
ADD SET RELATION-7      .
ADD SET RELATION-8      .
ADD SET RELATION-9      .
GENERATE.
```

VITA

James G. Owens was born on 20 September 1956 in Norfolk, Virginia. He graduated from high school in Virginia Beach, Virginia, in 1974 and attended the University of South Carolina from which he received a B.S. in Computer Science in May 1978. Upon graduation, he received a commission in the USAF through the ROTC program and was assigned to the ASD Computer Center at Wright - Patterson AFB, Ohio, until 1 June 1982. On 16 December 1983, he graduated with a M.S. in Information Systems from the Air Force Institute of Technology.

Permanent Address: 4918 Ellendale Ct.
Dayton, Ohio 45431

8a. NAME OF FUNDING/SPONSORING ORGANIZATION	8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER

8c. ADDRESS (City, State and ZIP Code)	10. SOURCE OF FUNDING NOS.			
	PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UN NO.

11. TITLE (Include Security Classification)
See Box 19

12. PERSONAL AUTHOR(S)
James G. Owens, B.S., Capt, USAF

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
MS Thesis	FROM _____ TO _____	1983 December 16	215

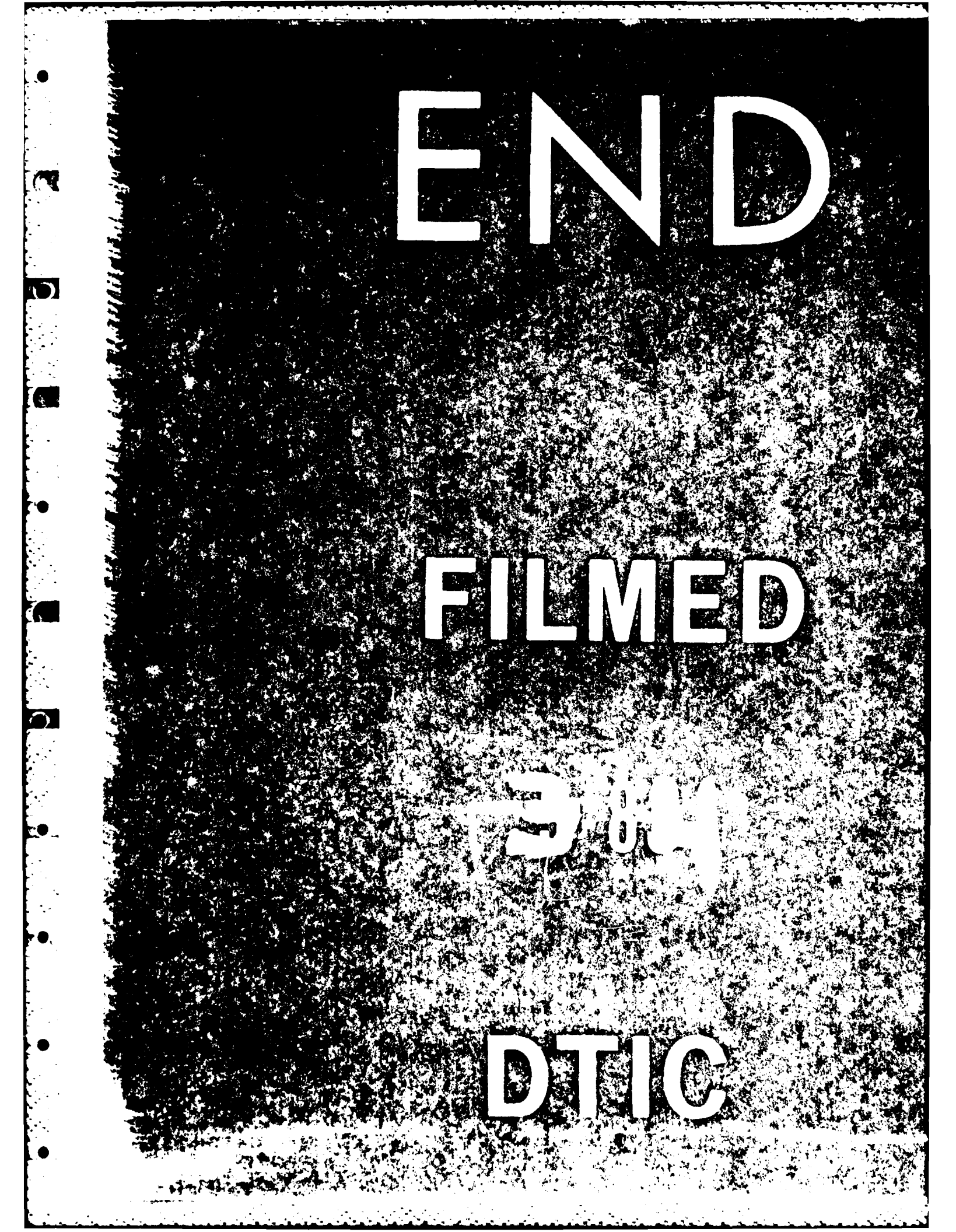
16. SUPPLEMENTARY NOTATION

Approved for public release: IAW AFR 190-17.
Lynn Wolaver 7 Feb 84
LYNN E. WOLAVER
Dean for Research and Professional Development
Air Force Institute of Technology (AIC)
Wright-Patterson AFB OH 45433

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB. GR.	Hierarchical Database, CODASYL Database, Schema Translator, IDMS, System 2000

An interactive hierarchical to CODASYL schema translator was designed and implemented with the goal of taking a System 2000 (hierarchical) schema as input and producing an equivalent IDMS (CODASYL) schema, storage schema (DMCL), and subschema. In addition, this system is to serve as part of an entire System 2000 to IDMS database translator system.

This thesis effort includes an analysis of the hierarchical and CODASYL models, an analysis of the System 2000 implementation of the hierarchical model, an analysis of the IDMS implementation of the CODASYL model, Data Flow Diagrams for the general hierarchical to CODASYL schema translator, structure charts for the System 2000 to IDMS schema translator, translator source code, and data dictionary.


END
FILMED
DTIC