1870

The Interactive Generation of Functional Dependencies

by

WILLIAM CLEN HUNT

B. G. S., University of Nebraska at Omaha
Omaha, Nebraska, 1972

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

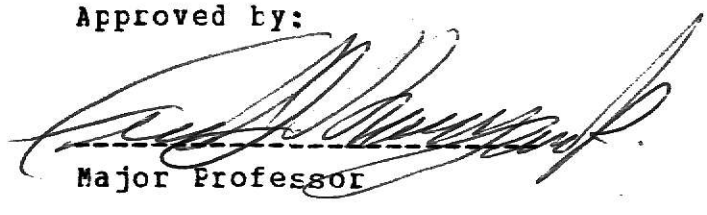
MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1978

Approved by:



Major Professor

Document
LD
2668
.R4
1978
H85
c.2

Acknowledagement

A special thanks tc Dr. Fred J. Maryanski fcr allowing me tc participate in the develcpment and implementation of the system fcr the automatic generation cf third normal fcrm relations, and for his professicnal attitude during its develcpment. Thanks tc Dr. Paul S. Fisher and Edwin W. Easham for serving as members cf my ccmmittee. The implementation was made possible thrcugh a grant frcm NCR, Inc.

Table of Content

1.0 Introduction.....1
2.0 Background.....4
2.1 The Problem of Third Normal Form.....5
2.2 Project.....6
2.3 FDGEN.....7
3.0 Functional Dependency Generator.....9
3.1 The FDGEN Algorithm.....10
3.2 Amplification of the FDGEN Algorithm.....17
4.0 Implementation.....41
4.1 Walk-Through.....41
4.2 Testing.....76
5.0 Conclusion.....94
5.1 Enhancement Considerations.....94

Reference and Bibliography.....97

Appendicies

Abstract Code A-1
Implementation - FDGEN Code - NCR Level II COBOL B-1

List of Figures

System Diagram.....8
Input File Function and Format.....11
Data Name Table Function and Format.....12
Functional Dependency Table Function and Format.....16
Output Format of Functional Dependencies.....17
Hierarchical Structure of Data-Name-Items.....19
Input File Example.....20
Data Name Table Example.....22
Sample Extracts of an Input File and Data Name Table
with status 'S' and 'N'.....25
Link Stack (Concatenation Stack) Format and Function....28
Storage Stack Format and Function.....30
Save Stack Format and Function.....31
Functional Dependency Output File Format and Function...39
Input File.....44
Concatenation Code Table.....46
Logical Internal View of the Data Name Table.....47
Update Section of the Data Name Table.....52
Update Logical View of the Data Name Table.....55
Logical View of the Concatenation and Storage Stack.....62
Logical View of the F-D-Table.....62
Current View of the Concatenation and Storage Stacks....64
Current View of the Save Stack and F-D-Table.....68
Current View of the Concatenation and Storage Stacks....70
Current Logical View of Internal Tables.....72
Final View of Program Generated and Updated Tables.....74
Functional Dependency Output Format.....75
Walk-Through Generated Functional Dependencies.....75

Input Test File 1.....77
Input Test File 2.....78
Input Test File 3.....80
Test Schema.....81
FDGEN Test Path Diagram.....82
User's Output of Test File 1.....86
User's Output of Test File 2.....87
Test Run 1 -- Input Test File 3.....88
Test Run 2 -- Input Test File 3.....89
Test Run 3 -- Input Test File 3.....90
Output File Generated from F-D-Table.....91

1.0 INTRODUCTION

Information is a prime commodity in today's hurried existence. Managers make decisions based on the best information available to them at the moment. Strategic planning is based on information compiled from many different sources. Every aspect of modern day life is influenced by information. Information is often related to power.

What is information? It is the meaningful interpretation of data which has been collected and intergrated.(1) To meet the needs of today's soceity data must be collected in great quantities. In order for massive amounts of data to be useful to a user, he must have real-time access to it and have a means of synthesizing the available data into meaningful collections so that it might be interpreted into useful information.

Systems have been developed which provide the user with mass storage of data, quick access to the data, and a means of systhesizing the accessed data. The systems, known as Data Base Management Systems (DBMS), allow a user to retrieve, manipulate, and store data without requiring him to know how the data is organized or stored within the storage device.

The user views the data in prespecified logical groups linked together in some organized structure. The more common DBMS structures are known as network and hierarchy. In such views, data may be represented in three ways:(2)

1. The content of the record (e.g., Maryanski's Dept = CS.);
2. The linkage between records (e.g., Maryanski's Dept record occurs in the hierarchy below the College Record for College of Arts and Science.); and
3. The ordering of records (e.g., All of the department records are stored in alphabetical order.).

Usage of such DBMS requires that the user have knowledge of the representation chosen (e.g., FIND NEXT RECORD of College-Dept Set).

The most recent addition to the DBMS family is represented by the Relational Data Model which makes possible the elimination of the representation-dependency from the user's interface. The relational model represents information at the user's interface only by data value. This representation-independence within a DBMS has generated enough interest within the industral community that research is being funded within the area.

Critical to the develcpment of a Relational Data Base is a user knowledgeable of the data-items which will comprise the data base. Data-items are the smallest logical unit of data within a data base. Data-items with the same characteristics are categorized by Data-Name-Item (Attribute). Each Data-Name-Item (DNI) has a domain within which are all data-items described by it. Sets of Data-Name-Items are combined to form relations or groups of

related Data-Name-Items. Each relation must possess one or more Data-Name-Item which can be used as a key to reference the other Data-Name-Items of the group. The association between the key Data-Name-Items and other Data-Name-Items in the group is referred to as functional dependence. The non-key Data-Name-Items are said to be functionally dependent upon the key Data-Name-Items. The set of Data-Name-Items is known as a Functional Dependency and the data base developer must be able to recognize them within his data.

The implementation of a DBMS based upon the Relational Data Model requires the development of a system which encompasses a minimum of three critical areas.(3)

1) A subsystem to accept the unsophisticated user's data and organize it for processing.

2) A subsystem to interact with the user, thus guiding him in the generation of Functional Dependencies.

3) A subsystem to automatically synthesize the generated Functional Dependencies into Third Normal Form (3NF).

This report presents the design, implementation and testing of the interative subsystem, Functional Dependency Generator (FDGEN), which enables the user to generate Functional Dependencies (FD).

2.0 BACKGROUND

A user interfaces with a DBMS through an application program or query language. The program provides the user with predefined views of data, independent of their stored configuration. The life expectancy of application programs is significantly increased when used with relational DBMS.(4)

A relational data base provides for the collecting of data-items in a relation and the formation of logical associations between the relations.

The purpose of a relational data base is fourfold:(5)

1. To free the collection of relations from undesirable insertion, update, and deletion dependencies;
2. To reduce the need for restructuring the collection of relations as new types of data are introduced, and thus increase the life span of application programs;
3. To make it feasible to represent any relation in the data base; and
4. To make the collection of relations neutral to the query statistics, where these statistics are liable to change as time goes by.

Before a Relational Data Base can fulfill its purpose its relations must be in Third Normal Form.

2.1 THE PROBLEM OF THIRD NORMAL FORM

To meet these requirements of a relational data base each data relation must be in Third Normal Form. Each must meet the following test: (6)

1. Each data-item in the relation must be based on a simple domain (it must be atomic);
2. Within each relation data-items are separated into key data-items and non-key data-items. Every non-key data-item must be fully dependent on every key data-item in the relation;
3. Every non-key data-item in the relation must be independent of all other non-key data-items in the relation; and
4. Every key data-item in the relation must be fully dependent on every other key data-item in the relation of which it is a part.

A data relation cannot be tested to see if it is in 3NF before its Functional Dependencies have been identified. A Functional Dependency among the data-items of a relation must be of the following form: (7)

A data-item D of relation C is functionally dependent upon data-item B if, at every instant of time, each B-value in C is associated with only one D-value. The relationship is expressed by the notation B --> D, and says that B determines D or D depends on B. Likewise,

a set of data-items in C may be functionally dependent on another data-item or set of data-items. The data-item (or set of data-items) on the left side of the arrow (B) is called the Determinant.

A user often experiences difficulty trying to determine the Functional Dependencies among the data-items of a data relation, thus increasing the probability that the schema designed would not meet the requirements of a relational data base.

2.2 PROJECT

The automatic generation of 3NF relations is a major step toward the development of relational data bases. NCR, Inc. provided a grant to fund the development of a prototype system which would interactively guide an unsophistocated user in the development of 3NF relations. The system design provided for four subsystems: (8)

1. The User Interface subsystem - An interactive program which reformats input data into hierarchical records, called reports, based on responses to queries from the system and produces an output file of records;
2. The FDGEN subsystem - An interactive program which inputs the data produced by the User Interface subsystem, interactively generates Functional Dependencies and produces as output a file of Functional Dependencies and a hardcopy listing of

Functional Dependencies for the user;

3. The Functional Dependency Analysier (FDA) subsystem - A program which inputs the data produced by the FDGEN subsystem and synthesis the Functional Dependencies into 3NF relations. It produces as output a hardcopy of the 3NF relations; and

4. The Editor subsystem - An interactive program which provides the user with the means to insert, delete, or change any portion of the original input file.

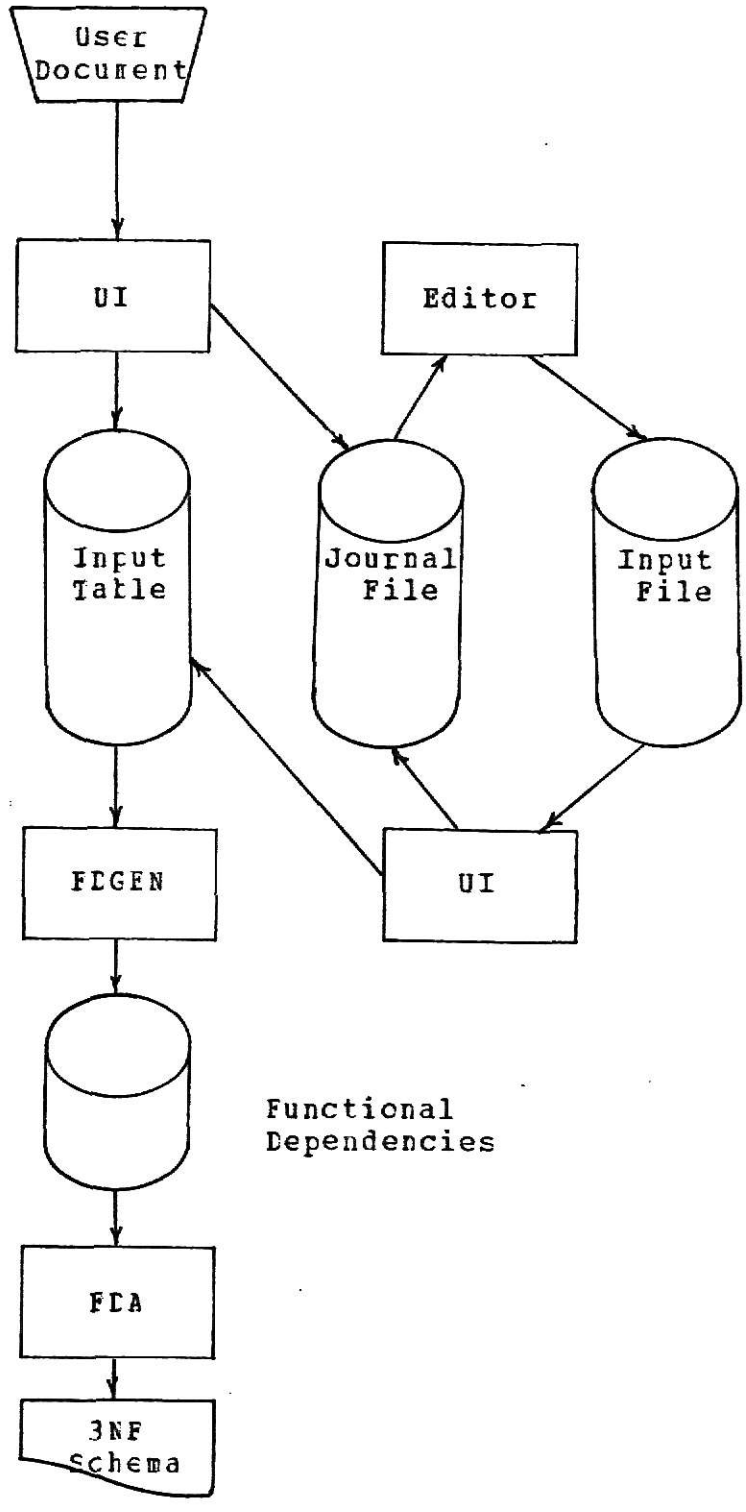
The diagram of the system is pictured in figure 2-1.(9)

2.3 FDGEN

The generation of Functional Dependencies (FD) is not an automatic process. It requires the interaction of a user who is familiar with the data-items input into the system. Data-items entered as a group must relate to each other so that a logical relation can be established. The logically related data-items become the input to the FDGEN. The subsystem defines terms, provides instructions, and directs the user in the manipulation of the related data-items enabling him to form the desired FD. The generated FD's are stored in an output file for later use as input for the FDA.

System Diagram

Figure 2-1



3.0 FUNCTIONAL DEPENDENCY GENERATOR

The Functional Dependency Generator (FDGEN) is an interactive program which accepts as input a file generated by the User Interface Program and produces as output a Functional Dependency Table (F-D-Table) which is stored on disc and used as input for the Functional Dependency Analyzer Program(FDA).(10)

The FDGEN processes the input data an constructs a Data Name Table (DNT) which is used throughout the remainder of the program. The processed data is presented to the interactive user along with the necessary instructions to guide him through the various phases of functional dependency generation. All processed data has associated with it a hierarchical code which allows distinctive data relations to be generated and presented to the user. The user's primary objective is to redefine the generated relations into logical Functional Dependencies(FD). Data manipulation facilities are provided which allow the necessary reconstruction of relations. A hardcopy of the generated FD's is provided to the user upon completion of data manipulation.

An example of a reconstructed relation is illustrated by the following relation with Data-Name-Items (DNI) OWNER, HOUSE, CAR, BUSINESS, and FARM. The user would manipulate the DNI's so that the following FD is generated and output.

OWNER --> HOUSE CAR BUSINESS FARM

The FD says that OWNER identifies the other Data-Name-items

within the relation.

3.1 THE FDGEN ALGORITHM

The algorithm for the generation of FD's is seperated into eleven logical steps.(11)

Step 1. Read the next record from the Input File into a buffer. The format and function of the Input File are provided in figure 3-1.

Step 2. Delete the first four characters (LINE #) from the buffer. Assign the fifth and sixth characters to Status and level respectively. Separate the remaining data in the buffer and assign each delimited set of characters (ATTRIBUTE) to a unique position within the DNT. Assign appropriate values to the associated fields. Link each level '0' position (REPORT NAME) to its successors level '0' position. Mark the end of the DNT. The format and function of the DNT are provided in figure 3-2.

Step 3. When all records have been read and all data has been stored in the DNT display each DNI one at a time. Ask the user to identify all Virtual-Data-Items. Enter a code '3' in the determ-flag field of the DNT for each DNI identified as a Virtual-Data-Item. A Virtual-Data-Item is a DNI which is

Input File Function and Format

Figure 3-1

Line no	Status	Level	Text
XXXX	X	X	X-70-X

Input File Format

Function of the Input File: To store user input text (DNI's) with their associated attributes (the level of the group, its status , and the group's position within the table (line no)).

Data Name Table Function and Format

Figure 3-2

Name	Level-no	Link-code	Determ-flag
X-10-X	X	XXX	X

Data Name Table Format

Function of the Data Name Table: Initially to store in the prescribed format the input data from the Input File as well as the associated special link codes (concatenation links) which are derived from the associated level-no and status. Updated codes will be entered as DNI's are defined through user interaction.

derivable from other DNI's within the data base. The Virtual-Data-Items selected will be eliminated from the report. Figure 4-4 pictures an example where the DNI PRICE has been identified as a Virtual-Data-Item and a code '3' has been entered.

Step 4. Identify potential Determinants by performing an intersection by DNI of each report with every other report in the DNI. For each intersected DNI enter a code of '2' in its associated determ-flag field in the DNT. An example of intersected DNI's is pictured in figure 4-5.

Step 5. Display for the user the list of all DNI's of the current report which have associated with them identical link codes. If a displayed DNI has code '2' in its associated determ-flag field display an asterisk to its right. The asterisk identifies a DNI as a potential Determinant of the displayed group. The user must identify one or more (possibly all) DNI's as Determinants of the group. An example of a marked intersected DNI is pictured with its group on page 63.

Step 6. (a) If the current group from which the Determinant was selected has a level greater than the levels of previously considered groups from the current report, then Determinants for the previously considered groups must be concatenated to the current Determinant to form the Determinant Set for

the current group. The concatenated Determinant Set MODEL, ENG-NC, STYLE, EXTERIOR, and INTERIOR is shown in the Concatenation Stack pictured in figure 4-8.

(b) If the group from which the Determinant was selected has a level greater than one but less than or equal to any level of a previous group from the current report, then those group Determinants whose levels are greater than ~~are~~ or equal to the current group's level must be discarded and the remaining Determinants concatenated to the Determinant of the current group. This situation can best be illustrated by example: Consider the Determinant set in figure 4-8. If the next Determinant selected by the user has associated with it a concatenation link code of '22' all Determinants with codes of '21' would be discarded from the Concatenation Stack and the new Determinant would be concatenated with MODEL to form the new Set.

Step 7. If two or more group Determinants have been concatenated, then display the Determinants and mark the most current with an asterisk. Provide the user with the following options:

a) Elect to leave the determinant configuration unchanged.

b). Elect to redefine all non-current Determinants as non-Determinants.

c). Elect to select one or more Determinants to be redefined as non-determinants.

An example of such a display is pictured on page 67.

Step 8. If two or more group Determinants have been concatenated and they identify themselves, then display the concatenated Determinants and mark with an asterisk those Determinants from the current group. Provide the user with the following options:

a). Elect not to change the determinant configuration.

b). Elect to redefine all non-current Determinants as non-determinants.

c). Elect to select one or more Determinant to be redefined as non-determinants.

The queries which provide the user options are shown on pages 65 and 66. The Determinants are marked the same as those displayed on page 67.

Step 9. Upon selection of an option by the user, add the new FD to the F-D-Table. The FD is added to the table in four steps as follows:

a) Enter the number of Determinants.

Functional Dependency Table Function and Format

Figure 3-3

List
no. of Determinants
Determinant Names
no. of non-determinants
non-determinant names
X-10-X

Functional Dependency Table Format

Function of the Functional Dependency Table: To store the Functional Dependencies generated by the program through interaction with the user. Storage will be sequential from the low index and will begin with a numeric (N), followed by (N) Determinant names, another numeric (M), followed by (M) DNI's. This sequence of events is repeated for each FD to be stored. It is the sole source of input for the FDTABLE output file, and the secondary output of the FDGEN program when written to the line printer.

b) Enter the Determinant names.

c) Enter the number of non-determinants.

d) Enter the non-determinant names.

The F-D-Table format and functions are provided in figure 3-3.

Step 10. Store the F-D-Table on disc for future use as input for the FDA.

Step 11. Restructure the contents of the F-D-Table by FD and output each FD in the format shown in figure 3-4.

OUTPUT FORMAT OF FUNCTIONAL DEPENDENCIES

FIGURE 3-4

DETERMINANT STRING ---> NON-DETERMINANT STRING

The abstract code for the FDGEN algorithm is provided in appendix A.

3.2 AMPLIFICATION OF THE FDGEN ALGORITHM

A detailed explanation of data flow, data manipulation, and user interactions are provided in amplification of the FDGEN Algorithm.

The FDGEN Program accepts as input a current maximum of 200 Data-Name-Items. These DNI's are extracted from the

input file which was produced by the User Interface Program. (12) The file is logically a table of records of four fields. The first field contains a four digit line number which is discarded since it serves no purpose in the FIGEN Program. The second field contains the status of the text field. The status consist of one alpha-character and is used to initiate link code generation. The third field contains the relative level number of the text in the text field. The level number is relative only within the current report. The level number is used to develop a hierarchy among the groups of DNI's within a report. It is analogous to a PL/1 structure or a CCECL record. A picture of a hierarchical structure of DNI's is provided in figure 3-5.

The last field is the text field which contains the DNI's. The data within the text field is delimited by spaces. At least one space must seperate each DNI; however, spaces may also precede the first DNI as well as follow the last one.

The record shown in figure 3-5 is depicted in the format of the Input File and pictured in figure 3-6.

The Input File is read one record at a time until all records have been read. Following the reading of each record the status and level number are extracted and saved for later use, then the DNI's are extracted one at a time from the text field.

Hierarchical Structure of Data-Name-Items

Figure 3-5

```
Record  Automobile
          01 Make
          01 Model
               02 Body-no
               02 Interior
               02 Exterior
               02 Style
               02 Eng-no
                    03 Weight
                    03 Cost
                    03 Price
```

Input File Example

Figure 3-6

LINE	S	L	TEXT
0000		0	Automobile
0010	Y	1	Make Model
0020	Y	2	Body-no Interior Exterior Style Eng-no
0030	Y	3	Weight Cost Price

LINE: Line Number.

S : Status.

L : Level Number.

Text: Data-Name-Items.

The data extracted from the Input File is used to build the DNT, a sample of which is pictured in figure 3-7. The DNT is the major data structure within the program. Data stored in this table is used to service all other structures of the program.

The name field is limited to ten characters; however, it is easily expanded. The level number field may contain any one digit integer while the concatenation link field may contain the characters 'NUL' ~~are~~ or any three digit integer. The determ-flag field is restricted to a one digit integer.

The DNT construction is continuous while data is being read from the Input File and the text field data is being separated. As each DNI is extracted it is placed in the name field of the DNT. If a DNI exceeds the allowable number of characters, an error message will be output and all excess characters will be truncated. The level number for the current record is placed in the level number field. The status does not become a part of the Data Name Table, but is used to initiate code updates.

Each status field may contain one of four characters: 'Y', 'C', 'S', or 'N'. The characters 'S' and 'N' alert the program that the normal hierarchical structuring is about to be interrupted. The character 'Y' simply allows the normal hierarchical structuring to continue while the 'C' causes the program to maintain the current level of structuring.(13) The character 'S' instructs the program to maintain the current level but

Sample Data Name Table

Figure 3-7

Name	Level Number	Concatenation Link	Determinant Flag
Automobile	0	NUL	0
Make	1	11	0
Model	1	11	0
Body-no	2	21	0
Interior	2	21	0
Exterior	2	21	0
Style	2	21	0
Eng-no	2	21	0
Weight	3	31	0
Cost	3	31	0
Price	3	31	3
END		XXX	

that the current record is not a continuation of the prior record. The 'N' causes the program to accept the level number provided by the user , but unless the level number is 1, not to consider the current record as a continuation of any prior records that might possess the same level number. These various conditions control the values that are placed in the concatenation link field.

When a DNI has a level number of 0 and no status, it is assumed to be a report name. The name and level number are entered into the table. The concatenation link field is used to link consecutive reports together. The concatenation link field associated with the current report name will receive the characters 'NUL'. If a predecessor report exists, then the address of the current report is placed in the concatenation link field associated with the predecessor report name. If however, the predecessor report exists in name only (no DNI's associated), an error message will be output and the user will be allowed to delete the erroneous report.

The remaining DNI's associated with the current report are entered in sequence of extraction and the associated concatenation link field is loaded from the Concatenation Code Table which is indexed by the level number. The table is reinitialized at the beginning of each new report. It has nine elements and each element contains a two digit integer. The first element contains 11 and each successive element receives the value of its predecessor plus 10.

For each occurrence of a status, 'S' or 'N', the Concatenation Code Table is updated. The update is equivalent to the movement of a pointer to the next sibling in an iteration traversal of a hierarchical tree structure. The table is updated at the current index (level number) and all succeeding elements by adding 1 to the integer value of each element. To illustrate the table's use, a sample Input File with status 'S' and 'N' and the corresponding DNT are pictured in figure 3-8.

The determ-flag field of each entry is initially set to zero. Following the insertion of the last entry, the end of the table is marked by insertion of an 'END' in the name field and 'XXX' in the concatenation link field. See figure 3-7 for an example of a closed table.

The program provides the user with a tutorial of Virtual-Data-Items and the necessary instruction to allow him to identify Virtual-Data-Items. The only user responses throughout this phase of the program are 'YES' and 'NO'.

The DNI's are read from the DNT one at a time and displayed for the user. Each DNI displayed is also identified by report name. This phase, in essence, provides the user with a sequential view of the DNT.

The user is ask to interact with the program by responding with a 'YES' or 'NO' to the question, is this DNI a Virtual-Data-Item. A 'NO' response allows the program to cycle to the next DNI in sequence;

Sample Extracts of an Input File and Data Name Table with Status of 'S' and 'N'

Figure 3-8

Line	S	L	TEXT
0060	Y	2	text1 text2
0070	Y	3	text3 text4
0080	S	3	text5 text6
0090	N	2	text7 text8

Extract of an Input File

Name	L	C-L	D-F
text1	2	21	0
text2	2	21	0
text3	3	31	0
text4	3	31	0
text5	3	32	0
text6	3	32	0
text7	2	22	0
text8	2	22	0

Extract of a corresponding Data Name Table Segment

however, if a 'YES' response is recorded, the value in the determ-flag field associated with the selected DNI is changed to '3'. The DNI is eliminated from futher consideration by virtue of the code '3'. Following the update of the code the program cycles to the next DNI.

When all DNI's have been listed an the user has responded, he is provided the opportunity to verify those selected Virtual-Data-Items. If a 'YES' response is recorded, the program searches the DNT in sequential order for DNI's whose associated determ-flag field contains the value '3'. For each code '3' found, the program displays the associated DNI and asks the question; 'is this a Virtual-Data-Item'. If the user's response is 'YES' the program continues its search until it reaches the end of the table, but if the response is 'NO' the value within the associated determ-flag field is changed to '0'. This facility allows a user to correct a possible selection error. If the user had elected not to verify the list of Virtual-Data-Items by entering 'NO' the program would have asked if the user desired to review a listing of the DNI's again. If a 'NO' response is recorded the program continues to the next phase, but if a 'YES' response is recorded the program returns to the beginning of the virtual identification phase and begins the cycle once again. The only new aspect to this cycle is that for each occurrence of a Virtual-Data-Item the program will inform the user that the displayed DNI has previously been selected as a Virtual-

Data-Item. The user may restore Virtual-Data-Items to DNI's only during the verify subphase of the program.

The intersection phase of the program preforms a pairwise comparison of every DNI in every report with the DNI's in every other report.(14) Each pairing of DNI's which produces a match causes the value in both associated determ-flag fields to be changed to '2'. DNI's with an associated determ-flag field value of '3' are not considered for pairing. The pairing for comparison is facilitated by the report linkage established in the DNT during the construction of the table. This linkage allows the DNI's of the first report to be paired and compared with the DNI's of all succeeding reports. Upon completion of each current report, its successor report becomes the current report and its DNI's are paired and compared with the DNI's of all succeeding reports. This process is continued until the last report becomes the current report. The last report is identified by having the characters 'NUL' in the determ-flag field associated with the report name. The end of the DNT is identified by the values stored in the name and determ-flag fields, 'END' and 'XXX' respectively.

Four data structures which are used during the determinant identification phase must be identified prior to presenting a detailed explanation of this phase.

1. The Concatenation Stack is a table of records with two fields. Its format and function are shown in figure 3-9.

Link Stack (Concatenation Stack) Format and Function

Figure 3-9

D-Name	Lev
X-10-X	XX

Link Stack Format

Function of the Link Stack: To store the selected Determinant and its level code. The level code is synonymous to the concatenation link code of the DNT. It stores the Determinant Set for the current report. It provides a source of input to the F-D-Table.

2. The Storage Stack is a table of elements. The format and function of this structure are provided in figure 3-10.

3. The Save Stack is identical to the Concatenation Stack except that its field names have been changed. Its format and function are presented in figure 3-11.

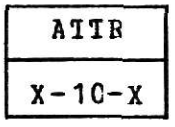
4. The Functional Dependency Table's function and format are provided in figure 3-3.

The determinant identification phase is highly interactive and requires input from the interacting user in order to generate the resultant FD's. The program cycles through each report and group in the same manner. Different codes associated with various groups may trigger different reactions to each group; however, each group is provided an equal opportunity to activate each facility within this phase.

The program begins this phase by displaying a brief tutorial on Determinants and asking the user if a futher explanation is desired. See display samples on page 56. A 'YES' response will provide the user

Storage Stack Format and Function

Figure 3-10



Storage Stack Format

Function of the Storage Stack: To store DNI's of a particular group as they are extracted from the DNI. To store a copy of the group Determinants as selected by the user. To provide a source of input to the F-D-Table.

Save Stack Format and Function

Figure 3-11

S-Name	Slev
X-10-X	XX

Save Stack Format

Function of the Save Stack: To store the Determinant Set asscciated with the predecessor group cf the report being considered. It provides a source of input to the Link Stack, but is seldom used.

with a view of an example of a Determinant as it identifies a set of DNI's. Upon a signal from the user the program initiates determinant selection. It begins by extracting information from the first report in the DNT. The report name displayed is followed by a set of instructions which explain how to use the facility. See sample displays on pages 57 and 58. To the left of the instructions are displayed all DNI's from the current report which have the same concatenation link code. A sample display is shown on page 59.

Link codes are generated in a hierarchical sequence beginning with an initial code of '11'. The program compares the generated code with that stored within the ~~determ-flag~~ CONC-LINK field of the DNT. When a match is found the DNI is extracted from the associated name field and displayed in addition to being pushed onto the Storage Stack. The program searches the report sequentially for other matching codes. Associated determ-flag fields are checked at each occurance of a matched code. If the determ-flag field value is 2 the associated name field values has been matched with another DNI of another report and an asterisk is displayed adjacent to the DNI. When all identical codes have been found and their associated DNI extracted, the user is asked to identify the Determinants which will uniquely identify the remaining DNI's of the group. The user is instructed to first consider as possible Determinants, all DNI's which are marked by an asterisk.

The user enters the selected Determinant names one at a time as directed by the program. DNI's spelled incorrectly will not be accepted and will produce an error message. When all Determinants have been entered, the user must respond to the next response by entering 'XX'. See example on page 63. This response signals the program to enter the next segment.

As each Determinant is entered it is pushed onto the Storage Stack and a count of their number is maintained. Additionally, the Determinant is pushed onto the Concatenation Stack following some minor investigations. The investigations consist of determining the level of any previously loaded Determinants. The process is as follows:

1. If no prior Determinant from this report is on the stack, then push the Determinant onto the stack along with its concatenation code(code);

2. If a Determinant has been loaded and its code was '1', then push the Determinant onto the stack with its code;

3. If a Determinant has been loaded and its code is less that or equal to the current Determinant's code, then push the Determinant onto the stack with its code; or

4. If a Determinant has been loaded and its code is greater than the current Determinant's code , then pop the stack until a Determinant is found with a level number value less than the level of the current

Determinant. Push the current Determinant onto the stack with its code.

Following the concatenation of Determinants in the Concatenation Stack, those Determinants which were pushed onto the Storage Stack are deleted along with all matching DNI's on the stack. The remaining DNI's represent the non-determinants of the current group. An example of a Storage Stack before and after deletion of Determinants is pictured in figure 4-8. Following the deletion of Determinants, if the Storage Stack is empty, a special situatuion exists. A case of all Determinants identifing themselves will have been detected. Two alternatives are presented to the user.

1) Elect to leave the Determinant configuration as it exists in which case the program will generate a unique symbol and push it onto the Storage Stack. This will identify the FD as one which is self-identifing. The symbol consist of a colon (:) followed by a unique two digit integer. The first integer will be '01'. If Determinants from only one group exist this option is automatically selected. See example in figure 4-20.

2) Elect to change some or all of the Determinants to non-determinants.

a) The decision to change all non-current Determinants to non-determinants is signified by the user entering 'YES'. This action initiates the movement of DNI's from the Concatenation Stack to the Storage Stack. The value of each Determinant's

code is compared with the current code. If they do not match the Determinant is removed from the Concatenation Stack to the Storage Stack. This process is repeated until all DNI's in the Concatenation Stack have been checked. Following the transfer, the remaining DNI's in the Concatenation Stack are compacted to the bottom of the stack.

b) The decision to individually select those Determinants which are to be restored to non-determinants is initiated simply by pressing the carriage return. This response is answered by a new set of instructions which tell the user how to enter the selected Determinant name and how to signal the program that no futher entries are to be made.

As each selected Determinant is entered, the program searches the Concatenation Stack from the bottom until the Determinant is found. If the Determinant is not found a spelling error has been detected and the user is asked to enter the name again.

The identified Determinant is deleted from the Concatenation Stack and pushed onto the Storage Stack. The Concatenation Stack is compacted to insure contiguous storage. The left side count, number of Determinants, is decreased by one and the right side count, number of non-determinants, is increased by one. This process is repeated until the

user enters 'XX'. This signals the program that no more entries are to be made.

A second special case will exist, if following the deletion of Determinants from the Storage Stack, the Storage Stack is not empty and the Concatenation Stack contains a set of concatenated Determinants. This special case may also introduce a third special case which will be discussed on page 37.

The second special case detects the fact that Determinants from more than one group have been concatenated. The program provides the user an opportunity to manipulate Determinants once again.

A list of all Determinants in the Concatenation Stack are displayed along with an asterisk to mark the most recently selected Determinants. The user is offered the same Determinant manipulation options as he was in the first special case. He is permitted to select the marked Determinants as well as the unmarked. See sample display on page 67. This facility allows the user to alter the hierarchical structuring of Determinants within the report by allowing him the option of selecting the Determinant Set for the current group without reguard to the generated hierarchical structure. The option is exercised by selecting from the list those Determinants which are to be restored to non-determinants. This action causes the selected DNI's to be transferred to the non-determinant side of the FD.

The third special case can be detected only within the constraints of the second special case. First there must exist a set of concatenated Determinants and secondly, the current group and its predecessor must be at the same level but have different codes. This case is representative of a tree structure where the parent node has two or more successor nodes. The successor nodes are at the same level but neither is a continuation of the other. The normal hierarchical structure is depicted by an n-ary tree structure, but there exists the possibility that the user elected, during the Determinant manipulation of the last group, to change the ancestors in the Determinant Set to non-determinants; breaking the hierarchical chain. This situation would mean that there would not be a hierarchical structuring of Determinants which could be concatenated to the current Determinant to form a complete Determinant Set. The hierarchy, if destroyed, is reconstructed when the third special case is detected. An example of the third special case can be observed in figure 4-21. NOTE: The third FD generated broke the hierarchical chain by defining 'A' as a non-determinant. The chain was restored and broken again in the fourth FD and reconstructed once again in the fifth. This example incorporates the first, second, and third special cases.

The program, upon detecting the third special case, compares the code values of the Determinants in the Save Stack with those of the Concatenation Stack. All

Determinants in the Save Stack with codes smaller than the smallest in the Concatenation Stack are moved to the bottom of the Concatenation Stack. They are moved one at a time to the Concatenation Stack which is now being used as a DEQUE. The current data in the stack is moved one possition higher for each addition. All counters are updated to reflect all changes. The hierarchical structure will have been reconstructed following the movement of the required Determinants, thus allowing a continuation of the second special case processing.

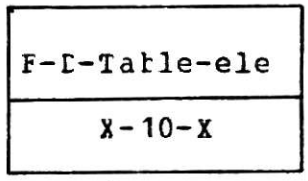
The user's interaction with the program ends following the special case processing. The program has continually added FD's to the F-D-Table as the user directed their construction. The end of the F-D-Table is marked by inserting a string of X's. See sample F-D-Table in figure 4-12.

The completed F-D-Table is copied to the Functional Dependency Output File on disc for future use as input for the FDA. The Output File format and function are provided in figure 3-12. The remaining task of FDGEN is to provide the user with a hardcopy of the FD's generated. See sample output in figure 4-14.

The program begins reading data sequentially from the low end of the F-D-Table. The first element read is an integer (N) which tells the program

Functicnal Dependency Cutput File Format and Function

Figure 3-12



Functional Deperdency Output File Format

Functicn of the Functional Dependency Output File: To store a copy of the Functicnal Dependencies stcred in the F-D-Table. It is the sole scurce of input to the FDA Program and the primary output of the FDGEN Program.

how many Determinants follcw. The Determinants are read and ccncatenated onto a string delimited ky one space. An arrow '--->' is concatenated cntc the string fcllcwing the Determinants. The F-D-Table pointer is incremented by one and the next element is read. Its value is an integer (M) which tells the program the number of non-determinants tc fcllow. If (N+M+1) is nct greater than twelve the non-determinants are read and concatenated onto the string which is then written to the line printer. If the value of (N+M+1) is greater than twelve the FD will exceed one line. The number of lines is determined based on a maximum of twelve elements per line. Each line is printed as it is generated from the F-D-Table. The program terminates when it encounters the string of X's which marks the end of the F-D-Table.

4.0 IMPLEMENTATION

The Functional Dependency Generation Program was implemented on an NCR 8250 System using NCR interactive CCBCL version 02.02.02 running under IMOS. A complete copy of the COBOL code is provided in appendix B. Included in the IMOS software package was the operating system, level II COBOL compiler, runtime interpreter, text editor, and interactive debug modules.(15)

Hardware:

The 8250 hardware consist of the following:

1 6850 processor

128K bytes MOS memory

1 cassette drive

2 Hawk 656 disc drives

2 796-101 CRT's

1 349-300 line printer

1 368 card reader

2 Asynchronous adapters (2400 baud)

1 battery back-up.

The program inputs data from an Input File, accepts input interactively, outputs data to a Functional Dependency Output File, and manipulates data in five internal tables. These tables were described in chapter 3.

4.1 WALK-THROUGH

The Functional Dependency Generator (FDGEN) was designed and inplemented as an interactive program. The minimum interactions a user might have with the program is

seven, plus one additional for each Data-Name-Item (DNI) entered. The maximum interactions would depend on the number and size of the reports entered as input. In order to provide a clear view of the program in action, a walk-through of an example will be conducted which will include all major phases of the program. The walk-through will also include system queries, user responses, views of internal structures, and the resultant Functional Dependencies (FD) produced. More complex examples will be presented during the explanation of testing. These examples will not be used to show the manipulation of the internal data structures.

The sample walk-through will use as input the following data which is depicted in a format that would be viewed by a user at a CRT.

```
      Report Name
0000  Automobile
0010  Model Make                                    Y
0020  Eng-no Style Exterior Interior Body-no        Y
0030  Body-no Year Weight Cost Price                Y
0040  END

      Report Name
0000  Engine
0010  Eng-no No-cyl Displac Eng-wt Eng-cost HP      Y
0020  END
```

The displayed reports are entered by the user one line at a time. These reports are used by the User Interface Program to generate the Input File. The Y's displayed at the extreme right are system generated based on user

responses tc system queries. They are used by the system to generate the status for the particular grcup of DNI's. A view of the Input File as generated from the above data is pictured in figure 4-1.

The FDGEN is executed by NCR 8250 JCL. The following is a complete walk-through of the FDGEN Program using the input data displayed in the Input File and picturing all system queries and user responses.

The first system query is in the form of an introductory instruction statement.

```
ANY INFORMATION THAT APPEARS ON THE SCREEN
WITHOUT ANY INSTRUCTICNS MAY BE REMCVED BY
DEPRESSING NEW LINE ** PRESS NEW LINE.
```

The user must respcnd by pressing 'NEW LINE', thus allowing execution to continue.

The Input File prcvides the seccnd input to the FDGEN. The file is automatically accessed and read a reccrd at a time until the entire file has been input. As each record is read the first four characters are discarded, leaving the group's status, level number, and associated DNI's. The program also discards all END's which are preceeded by twc blanks. The status is used to initiate the update of the Concatenation Code Table. The level becomes an entry in the

Input File

Figure 4-1

LN	S	L	TEXT
0000		0	Automobile
0010	Y	1	Model Make
0020	Y	2	Eng-No Style Exterior Interior Body-No
0030	Y	3	Body-No Year Weight Cost Price
0040			END
0000		0	Engine
0010	Y	1	Eng-No No-Cyl Displac Eng-wt Eng-cost HP
0020			END

LN: Line number

S: Status

I: Level Number

TEXT: Data-Name-Item

Data Name Table (DNT) and the current index to the Concatenation Code Table, from which is extracted the concatenation link entry to the DNT. A status of 'N' or 'S' is required to initiate the update of the Concatenation Code Table. The Concatenation Code Table is reinitialized at each determination of a new report. The initialized table is pictured at figure 4-2.

The DNT is constructed independently of user interaction. The end of the table is marked by inserting an 'END' in the last name field and 'XXX' in the last concatenation link field. The logical internal view of the table is pictured in figure 4-3.

If a report name or DNI has been entered that exceeds the maximum allowable number of characters the program will alert the user of the error by displaying the following error message.

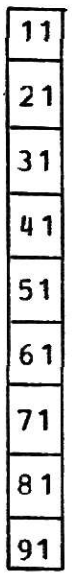
```
DATA NAME 'DATA-NAME-ITEM' EXCEEDS THE ALLOWABLE
NUMBER OF CHARACTERS, EXCESS CHARACTERS WILL BE
TRUNCATED.
```

The user's response is to press 'NEW LINE'. The program truncates the excess characters by assignment and continues.

If a report name was entered and was not followed by

Concatenation Code Table

Figure 4-2



Indexed by level number

Logical Internal View of the Data Name Table

Figure 4-3

Index	Name	Level-No	Conc-Link	Determ-Flag
1	Automobile	0	14	0
2	Model	1	11	0
3	Make	1	11	0
4	Eng-no	2	21	0
5	Style	2	21	0
6	Exterior	2	21	0
7	Interior	2	21	0
8	Body-no	2	21	0
9	Body-no	3	31	0
10	Year	3	31	0
11	Weight	3	31	0
12	Cost	3	31	0
13	Price	3	31	0
14	Engine	0	NUL	0
15	Eng-no	1	11	0
16	No-Cyl	1	11	0
17	Displac	1	11	0
18	Eng-wt	1	11	0
19	Eng-cost	1	11	0
20	HP	1	11	0
21	END		XXX	

at least one DNI an error condition will exist. The program detects the error and queries the user as follows.

'REPORT NAME'

ERROR -- THIS REPORT HAS NO ENTRIES. DO YOU WISH IT ELIMINATED - YES/NO.

The program is not designed to accept null reports. If the user responds by entering

NO

the program will generate erronous information as output; whereas, if the user's response is

YES

the null report is eliminated from the DNT by overlaying.

The DNT is the primary sourse of information for the remainder of the program. The DNT is first used during the interactive identification of Virtual-Data-Items (VDI). The following user instructions are provided.

USER INSTRUCTICNS
****

VIRTUAL DATA ITEM IDENTIFICATICN PHASE

THE USER MUST IDENTIFY ALL VIRTUAL DATA ITEMS WHICH HAVE BEEN ENTERED AS DATA-NAME-ITEMS. THIS WILL BE DONE BY ENTERING A YES FOR EACH IDENTIFIED VIRTUAL DATA ITEM. THE SYSTEM WILL DISPLAY ONE POTENTIAL VIRTUAL DATA ITEM AT A TIME. YOU MUST ENTER A YES(Y) IF IT IS A VDI AND A NO (N) IF NOT.

DO YCU DESIRE AN EXPLANATION OF A VIRTUAL DATA ITEM? ENTER YES OR NC.

If the user's response is 'NO' the program will continue, cn the other hand, a 'YES' response will produce the following description and example of a VDI.

A VIRTUAL DATA ITEM IS A DATA-NAME-ITEM WHICH IS DERIVAELE BY COMPUTATICN INVOLVING DATA-NAME-ITEMS FRCM WITHIN THE SAME DATA BASE. THIS SAYS THAT THE DATA-NAME-ITEMS CAN BE DERIVED FROM CTHER DATA-NAME-ITEMS THAT ARE WITHIN THE SAME DATA BASE OR POSSIBLY THE SAME REPCRT.

AN EXAMPLE OF A VIRTUAL DATA ITEM WOULD BE TOTAL-HOURS-WORKED
WHEN WITHIN THE SAME REPORT THE FOLLOWING DATA-NAME-ITEMS ARE AVAILABLE:
HOURS-WORKED-PER-WEEK
WEEKS-WORKED-PER-YEAR

IF YCU REQUIRE ADDITIONAL ASSISTANCE ENTER HELP

If the user enters 'HELP' the program will respond by providing the phone number and name of the development coordinator for the system, otherwise the user pressed 'NEW LINE' and the system will respond by displaying this message.

WE ARE READY TO START

The user simply presses 'NEW LINE' when he is ready to begin the VDI identification phase.

The program begins by displaying the report name associated with the DNI's to be presented to the user for possible selection. The user must respond to each newly presented DNI. The program cycles until all DNI's in the DNT have been presented. The following sequence is repeated for each DNI presented to the user.

REPORT TO BE CHECKED FOR VIRTUAL DATA ITEMS IS 'REPORT NAME'

'DATA-NAME-ITEM'

IS THIS DATA-NAME-ITEM A VDI? (YES or NO)

A user response of 'NO' allows the process to continue,

but a response of 'YES' interrupts the cycling process long enough to tag the seleceted VDI. The tagging is accomplished by inserting the code '3' into the asscciated determ-flag field for the selected DNI. In our example the DNI 'PRICE' will be selected as a VDI. The updated section of the DNI is pictured in figure 4-4.

After setting the tag, the program continues the cycling process. Upon completion, the user is presented with the following query.

```
ALL REPORTS AND THEIR RESPECTIVE DATA-NAME-ITEMS
HAVE BEEN DISPLAYED. DO YOU WISH TO VERFIY THE
LIST OF VIRTUAL DATA ITEMS. ENTER 'YES' OF 'NO'.
```

A user response of 'YES' will produce the following results.

```
                    PRICE

IF THIS IS A VALID VIRTUAL DATA ITEM ENTER YES
OTHERWISE NO.
```

In the example only one VDI was selected. Had there been more, the above display would be repeated until all selected VDI's had been displayed and the user provided the opportunity to verify their status.

Updated Section of the Data Name Table

Figure 4-4

Index	Name	Level-No	Conc-Link	Determ-Flag
12	Cost	3	31	0
13	Price	3	31	3
14	Engine	0	NUL	0

If the user responds with a 'YES' the program continues; however, if the response is 'NO' the status of the VDI is changed to that of a non-VDI. This is accomplished by changing the tag, code '3', back to a code of '0'. This is the only place in the program where an earlier decision may be reversed. 'PRICE', will remain a VDI in this example.

When all selected VDI's have been displayed and the user has responded to all queries, the system will present its next query.

```
DO YOU WISH TO REVIEW THE LIST OF DATA-NAME-ITEMS
OF EACH REPORT AGAIN TO INSURE THAT ALL VIRTUAL
DATA ITEMS HAVE BEEN IDENTIFIED. ENTER YES OR NO.
```

A 'NO' response to this query will terminate the virtual identification phase. A 'YES' response will cause all pointers to be reset to the beginning and the user will be given the opportunity to repeat the initial identification process again. This recycling of events includes another opportunity to verify the selected list as well as another review of all DNI's. The only addition to additional cycles will be the system's identification of all previously selected VDI's. This is depicted through a display of the previously selected VDI 'PRICE' when it is presented to the user as follows.

```
REPORT TO BE CHECKED FOR VIRTUAL DATA ITEMS IS
'REPORT NAME'

                'PRICE'     IDENTIFIED AS A VDI

IS THIS DATA-NAME-ITEM A VDI? (YES OR NO)
```

A 'NO' response will not change an identified VDI to a non-VDI; whereas, a 'YES' response will change a non-VDI to a VDI. An identified VDI can only be changed to a non-VDI during the verification of the VDI's.

The intersection phase of the program is not interactive. The purpose of this phase is to identify those DNI's which appear in more than one report. The identified DNI's are tagged by entering a code '2' in the determ-flag field associated with the DNI. If the DNI has been selected as a VDI, it is not considered during intersection. A view of the updated DNT following intersection is pictured in figure 4-5.

The determinant identification phase depends heavily upon user interactions. Upon entering this phase the program displays the following user requirements.

Updated Logical View of the Data Name Table

Figure 4-5

Index	Name	Level-No	Conc-Link	Determ-Flag
1	Automobile	0	14	0
2	Model	1	11	0
3	Make	1	11	0
4	Eng-no	2	21	2
5	Style	2	21	0
6	Exterior	2	21	0
7	Interior	2	21	0
8	Body-no	2	21	0
9	Body-no	3	31	0
10	Year	3	31	0
11	Weight	3	31	0
12	Cost	3	31	0
13	Price	3	31	3
14	Engine	0	NUL	0
15	Eng-no	1	11	2
16	No-Cyl	1	11	0
17	Displac	1	11	0
18	Eng-wt	1	11	0
19	Eng-cost	1	11	0
20	HP	1	11	0
21	END		XXX	

USER REQUIREMENTS
****

DETERMINANT IDENTIFICATION PHASE

YOU MUST IDENTIFY FROM EACH GROUP OF DATA-NAME-ITEMS DISPLAYED - THOSE ITEMS WHICH WILL UNIQUELY IDENTIFY THE REMAINING ITEMS WITHIN THE GROUP. THOSE DATA-NAME-ITEMS WHICH IDENTIFY THE REMAINING ITEMS ARE CALLED DETERMINANTS. EACH DETERMINANT IDENTIFIED MUST BE ENTERED UPON REQUEST. SPELLING IS CRITICAL. ANY QUESTIONS? YES/NO

A user response of 'YES' will produce the following example and explanation.

THE FOLLOWING DATA-NAME-ITEMS ARE DISPLAYED***** WAREHOUSE-NO, SPACE-AVAILABLE, ITEMS-STORED, STORAGE-LEVEL***** THE SINGLE DATA ITEM WHICH COULD IDENTIFY THE OTHER ITEMS WOULD BE WAREHOUSE-NO *****IT IS THE DETERMINANT*****

a 'NO' response, or following the display, the following statement is presented to the user.

ENTER GO WHEN YOU ARE READY TO START.

When the user is ready to begin the identification phase he must press 'NEW LINE'. This action allows the

program to continue. It produces the following message.

GROUP TO BE DISPLAYED ARE FROM REPORT 'REPORT NAME'

Displayed immediately following the report name is a variable explanation of what is to come. The display is variable because the variables WORD, ARTICLE, ONE-ARE-MORE, and VERB are assigned the correct words to make the explanation grammatically correct. The explanation is displayed only during the listing of groups from the first report.

'WORD' OF THE LISTED DATA-NAME-ITEMS MATCHED OTHER DATA-NAME-ITEMS IN OTHER REPORTS. THIS MEANS THAT YOU HAVE 'ARTICLE' CLUE AS TO WHICH DATA-NAME-ITEMS MIGHT BE 'ONE-ARE-MORE'

The values the variables may receive are:

WORD: None, One, Several;

ARTICLE: no, a, several;

ONE-ARE-MORE: a Determinant, Determinant;

VERB: is, are.

The following is a continuation of the above. The first text is used if no DNI's were intersected in the current report, otherwise the second text is used.

First Text

PLEASE ENTER THE DATA-NAME-ITEM WHICH MEETS THE REQUIREMENTS OF A DETERMINANT FOR THE GRCUP. THERE CAN BE MORE THAN CNE DETERMINANT PER GROUP. FAILURE TO IDENTIFY AT LEAST CNE DETERMINANT WILL CAUSE AN ERROR CCNDITION. THE SYSTEM WILL REQUEST THAT YOU ENTER EACH SELECTED DETERMINANT NAME ONE AT A TIME WHEN INDICATED. IF ALL DETERMINANT NAMES HAVE BEEN ENTERED, ENTER XX WHEN THE NEXT REQUEST APPEARS. EACH NAME ENTERED MUST BE SPELLED EXACTLY AS IT APPEARS CN THE SCREEN.

Second Text

THE DATA-NAME-ITEMS WHICH ALSO APPEAR IN CTHER RECORTS 'VERB' MARKED BY *. WHEN CONSIDERING WHICH DATA-NAME-ITEMS IDENTIFY THE REMAINING ITEMS IN THE GROUP--FIRST CONSIDER THE MARKED ITEMS. THERE CAN BE MORE THAN ONE DETERMINANT PER GROUP. FAILURE TO IDENTIFY AT LEAST ONE DETERMINANT WILL CAUSE AN ERROR CCNDITION. THE SYSTEM WILL REQUEST THAT YOU ENTER EACH SELECTED DETERMINANT NAME ONE AT A TIME WHEN INDICATED. IF ALL DETERMINANT NAMES HAVE BEEN ENTERED, ENTER XX WHEN THE NEXT REQUEST APPEARS. EACH NAME ENTERED MUST BE SPELLED EXACTLY AS IT APPEARS ON THE SCREEN.

Within the current report the program searches and displays to the left of the above instructions the list of DNI's possessing the same concatenation link code. The program will generate matching codes in sequence, thus allowing for a sequential presentation of groups within the current report. When a generated code can not be matched, new codes are generated until a match is found, or until a logical number of levels beyond the last matched level have been searched an found not to exist. At this point the program will have completed a report cycle and will select the next

report as the current report to be considered.

The following view, less instructions, is seen by the user.

```
GROUP TO BE DISPLAYED ARE FROM
REPORT AUTOMOBILE

MODEL                    INSTRUCTIONS
MAKE                        NOT
                          SHOWN
```

Immediately following the above display, the system requests that the user select and enter a Determinant. The user enters 'MODEL'.

```
ENTER THE DETERMINANT NAME.  MODEL
```

If the user spells the entry incorrectly, the program detects the error and presents the following message.

```
YOU SPELLED IT WRONG**TRY AGAIN**PLEASE.
PRESS NEW LINE
```

The press 'NEW LINE' response causes the incorrect resoponse to be disregarded and a new request for a Determinant. If the user inadvertently enters a Determinant name twice the program detects the error and presents the following message.

DUPLICATE DETERMINANT ENTRY -- ENTER A DETERMINANT NAME ONLY ONCE PER UNIQUE GROUP. PRESS NEW LINE

The press 'NEW LINE' response causes the incorrect response to be disregarded and the system to ask for the next Determinant name.

A user wishing tc delete a group from a report may do so by not identifing an identifing Determinant fcr the group. This editing prccess is accomplished by entering 'XX' when asked to enter thc Determinant name. The 'XX' response causes the program to recycle to the beginning of the identification phase, thus by-passing the Load FDTable Section. The system detects the actions and displays this message.

A DETERMINANT HAS NOT BEEN IDENTIFIED FOR THE LAST GROUP--GROUP IS DISCARDED

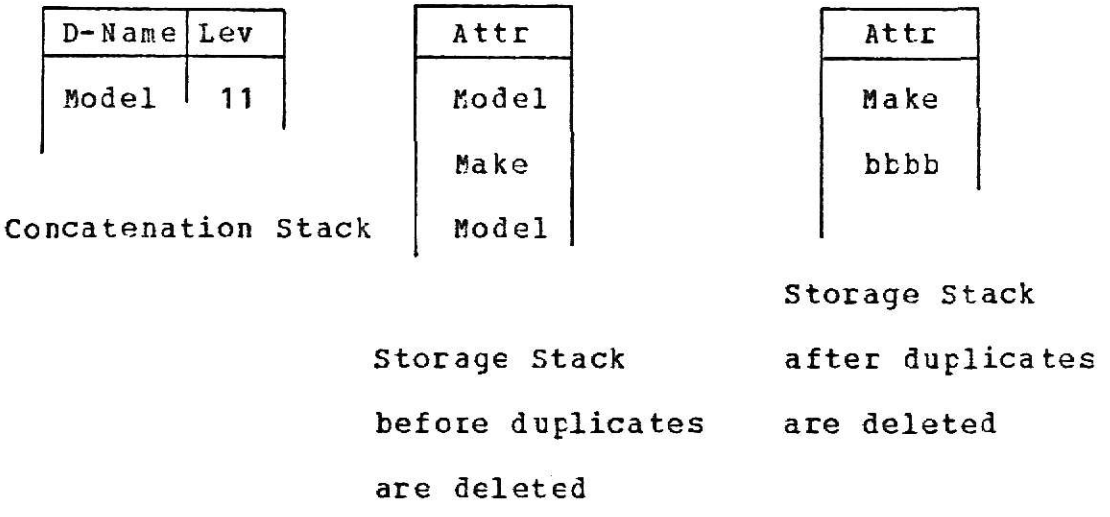
At the same time as the DNI's were being displayed on the screen they were being pushed onto the Storage Stack. The selected Determinant, 'MODEL', was pushed onto the Storage Stack as well as the Concatenation Stack. Their internal logical view is pictured in figure 4-6.

Duplicate DNI's are deleted from the Storage Stack and the first FD is loaded into the F-D-Table. A picture of its logical view is at figure 4-7.

The program recycles to the beginning of the determinant identificatiion phase, clears the Storage Stack, generates the next sequential search code '21', and begins its sequential search within the current report area of the DNT. The next view presented to the user is the currennt report name, a report of the instruction since report one is still current, and a listing of the next group in the sequence.

Logical View of the Concatenation and Storage Stacks

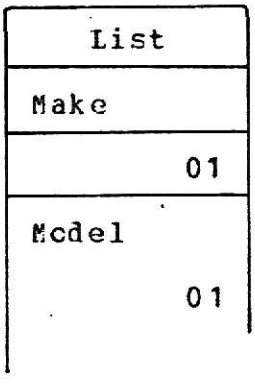
Figure 4-6



bbbb: Blanks

Logical View of the F-D-Table

Figure 4-7



```
GROUPS TO BE DISPLAYED ARE FROM REPORT
AUTOMOBILE

ENG-NO      *
STYLE                  INSTRUCTIONS
EXTERIOR
INTERIOR                   NOT
BODY-NO
                           SHOWN

          ENTER THE DETERMINANT NAME.  ENG-NO

          ENTER THE DETERMINANT NAME.  STYLE

          ENTER THE DETERMINANT NAME.  EXTERIOR

          ENTER THE DETERMINANT NAME.  INTERIOR

          ENTER THE DETERMINANT NAME.  XX
```

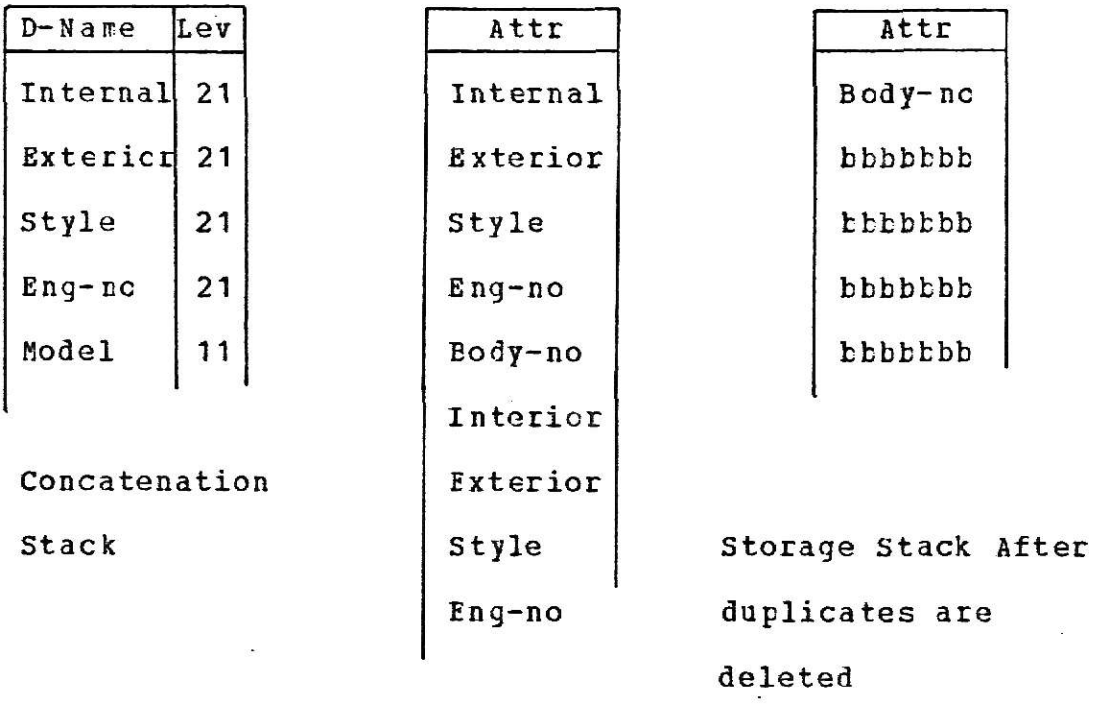
Each 'ENTER THE DETERMINANT NAME.' required a user response. The tables built and updated thus far during this cycle are pictured in figure 4-8.

Prior to updating the F-D-Table the program checks for two special cases which if detected will allow the user the opportunity to restructure the current FD.

The first special case exists when the Determinant Set does not identify any non-determinants. The example being used does not activate this facility; however, the user's options are the same as those provided in the second special case. An explanation and example of the first special case was presented on pages 34, 35, and 36. The queries for this case are shown on page 65.

Current View of the Concatenation and Storage Stacks

Figure 4-8



The second special case is detected when there exists a Determinant Set which consist of Determinants from more than one group. This case is detected as can be seen by referring to the Concatenation Stack pictured in figure 4-8. If the second special case is detected the program immediately checks for the third special case. This case does not exist in the example but was explained on page 37.

The program responds to a first special case situation by quering the user as follows.

SPECIAL SITUATION

THE FOLLOWING DATA-NAME-ITEMS HAVE BEEN CONCATENATED TO FORM A DETERMINANT. THE DETERMINANT AS STRUCTURED IS SELF-IDENTIFYING.

LIST OF
DETERIMNANTS
NOT SHOWN

IF THE DATA-NAME-ITEMS MARKED * UNIQUELY IDENTIFIES ANY OF THE OTHER LISTED ITEMS
ENTER YES IF NO -- THE RELATION IS ASSUMED CORRECT--PRESS NEW LINE.

A 'YES' response continues the above query.

ARE ALL UNMARKED DETERMINANTS IDENTIFIED BY THE MARKED DETERMINANTS? ENTER YES ** OTHERWISE PRESS NEW LINE.

A 'YES' response will transfer all unmarked Determinants from the Concatenation Stack to the Storage Stack, update all counters, and load the generated FD into the F-D-Table. A response of 'NEW LINE' produces the following continuation to the query.

```
LIST THOSE DATA-NAME-ITEMS WHICH ARE IDENTIFIED.
ENTER XX TO SIGNAL NO MORE ENTRIES ARE TO BE MADE.
```

Following these instructions the user enteries the selected Determinants and the program transfers each entry from the Concatenation Stack to the Storage Stack. Upon entry of 'XX', the generated FD is loaded into the F-D-Table. Any entry that is spelled incorrectly will cause an error condition to exist. The program responds with the following error message.

```
YOU SPELLED IT WRONG**TRY AGAIN**PLEASE
PRESS NEW LINE
```

The second special case, when detected, causes a user query similar to the first special case. The walk-through will provide several examples of this case.

```
ARE ANY OF THE FOLLOWING DETERMINANTS UNIQUELY
IDENTIFIED BY THE DETERMINANTS MARKED WITH THE *.
ENTER 'YES'**OTHERWISE PRESS NEW LINE.

        * INTERIOR
        * EXTERIOR
        * STYLE
        * ENG-NO
          MODEL
```

The response selected is 'NEW LINE'. This response generates nc futher queries of the user. The program saves the set of concatenated Determinants in the Save Stack and loads the F-D-Table. An internal view of the Save Stack and an updated view of the F-D-Table is pictured in figure 4-9.

The program updates its search code to '31' and recycles to the beginning of the current report in the DNT. The Storage Stack is cleared and the user is queried again.

```
GROUP TO BE DISPLAYED ARE FROM
REPORT AUTOMOBILE

BODY-NO              INSTRUCTIONS
YEAR
WEIGHT                   NOT
COST
                         SHOWN

ENTER THE DETERMINANT NAME.  BODY-NO

ENTER THE DETERMINANT NAME.  XX
```

Current View of the Save Stack and F-D-Table

Figure 4-9

S-Name	Slev
Interior	21
Exterior	21
Style	21
Eng-no	21
Model	11

Save Stack

List
Body-no
01
Interior
Exterior
Style
Eng-no
Model
05
Make
01
Model
01

F-D-Table

The user selected 'BODY-NC' to be the Determinant for this group while keeping in mind that all higher level Determinants for the current report are being concatenated to form the Determinant Set. The internal data structures as created and updated are pictured in figure 4-10.

The second special case is once again detected. The program immediately queries the user as before.

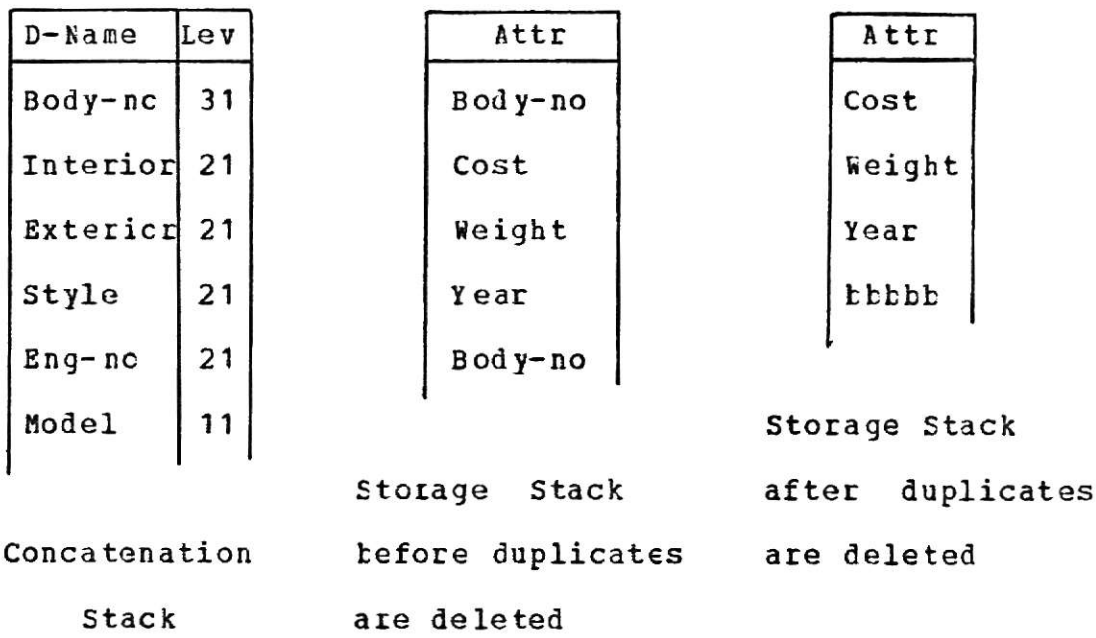
```
ARE ANY OF THE FOLLOWING DETERMINANTS UNIQUELY
IDENTIFIED BY THE DETERMINANTS MARKED WITH THE *.
ENTER 'YES'**OTHERWISE NEW LINE.
              * BODY-NO
                INTERIOR
                EXTERIOR
                STYLE
                ENG-NO
                MODEL
```

The selected response is 'YES'. This causes the query to be continued.

```
ARE ALL UNMARKED DETERMINANTS IDENTIFIED BY
THE MARKED DETERMINANTS?  ENTER 'YES'**
OTHERWISE PRESS NEW LINE.
```

Currect View of the Ccncatenation and Storage Stack

Figure 4-10



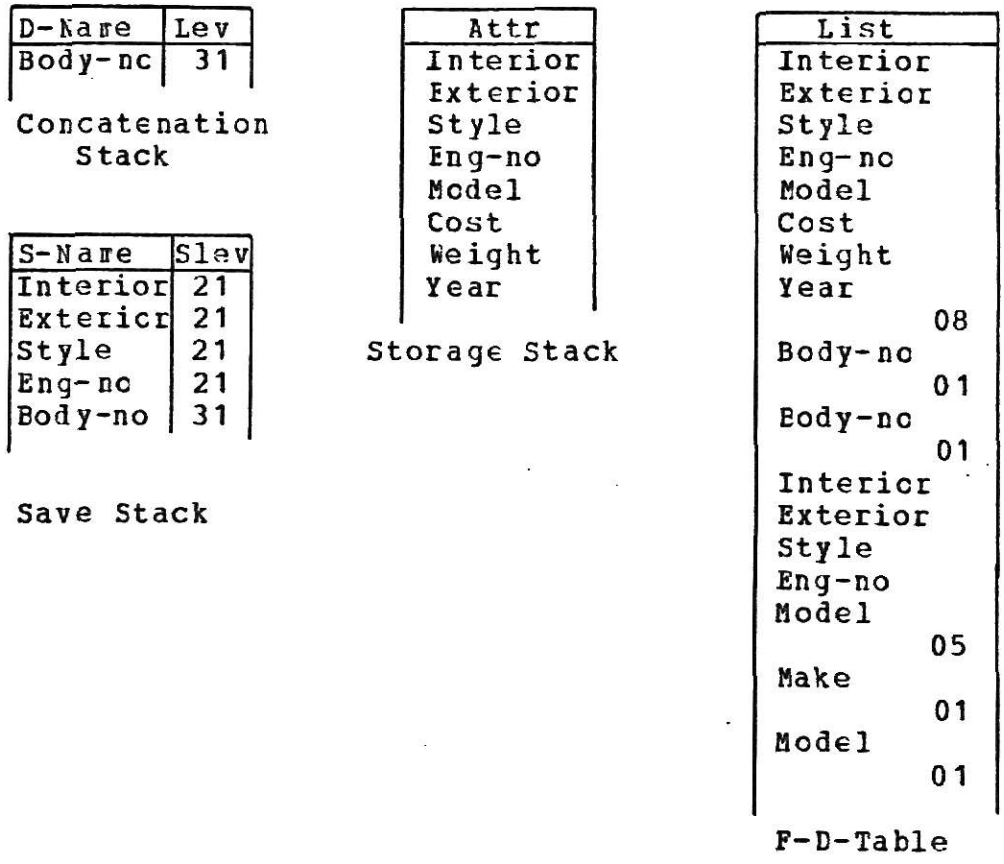
The selected response is 'YES'. This response causes all unmarked Determinants to be transferred from the Concatenation Stack to the Storage Stack. The Concatenation Stack contents are compacted to the low end and its contents saved in the Save Stack. The generated FD is loaded into the F-D-Table. The logical views of the internal tables (less DNT) are pictured in figure 4-11.

The program recycles to the beginning of the determinant identification section; is unable tc match a search code; clears the Save, Concatenate, and Storage Stacks; and enters the next sequential report within the DNT. The new search code selected is '11', and the user is queried as follows.

```
                GROUPS TO BE DISPLAYED ARE FROM
                REPORT ENGINE


ENG-NC      *
NO-CYL
DISPLAC
ENG-WT
ENG-COST
HP

                ENTER THE DETERMINANT NAME.  ENG-NO

                ENTER THE DETERMINANT NAME.  XX
```

Current Logical View of Internal Tables

Figure 4-11



No special cases are detected. The program generated and updated tables are pictured in figure 4-12. The program is unable to match additional search codes within the last report of the DNT. This condition terminates the determinant identification phase.

The program's last section is the print F-D-Table. Within this section each FD within the F-D-Table is extracted one at a time, reformated as pictured in figure 4-13 and written to the line printer. This output provides the user with a readable hardcopy of the generated FD. The hardcopy may be used by the user to implement changes to the Input File an regenerate the FD's. The hardcopy out of the user generated FD's for the example are pictured in figure 4-14.

Final View of Program Generated and Updated Tables

Figure 4-12

D-Name	Lev
Eng-no	11

Concatenation Stack

S-Name	Slev
Empty	00

Save Stack

Attr
Eng-no
HP
Eng-cost
Eng-wt
Displac
No-Cyl
Eng-no

Storage Stack before duplicates are deleted

Attr
HP
Eng-cost
Eng-wt
Displac
No-Cyl
bbbbbbbb

Storage Stack after duplicates are deleted

List
XXXXXXXXXX
HP
Eng-cost
Eng-wt
Displac
No-Cyl
05
Eng-no
01
Interior
Exterior
Style
Eng-no
Model
Cost
Weight
Year
08
Body-no
01
Body-no
01
Interior
Exterior
Style
Eng-no
Model
05
Make
01
Model
01

F-D-Table

Functional Dependency Output Format

Figure 4-13

DETERMINANT SET ---> NCN-DETERMINANT SET

Walk-Through Generated Functional Dependencies

Figure 4-14

Model ---> Make

Model Eng-no Style Exterior Interior ---> Body-nc

Body-nc ---> Year Weight Cost Model Eng-no Style Exterior Interior

Eng-nc ---> No-Cyl Displac Eng-wt Eng-cost HP

4.2 TESTING

The FDGEN Program was designed in a top-down configuration. The top-down design enhanced modular implementation which simplified the syntactical debugging process and enhanced modular testing. Testing was conducted in three phases; syntactical debug testing, program testing, and system testing.

The syntactical debug phase was conducted immediately following insertion of each module into the system. The only tool used during this phase was the COBOL compiler which provided a limited set of diagnostic outputs. During the program testing phase two input files were used. They are pictured in figure 4-15 and figure 4-16.

Test file 1 and 2 were used to debug the logical construction of the program prior to insertion of the special case modules. The COBOL compiler was once again the major debugging tool used during this phase. The complier diagnostics were augmented by an abundance of output statements. The statements were positioned so as to provide the tester with a complete internal view of all data at and around the point of concern. This procedure was continued until all paths had been tested for which the test data had been designed and all errors had been corrected.

The system test phase incorporated test files 1 and 2 in addition to

Input Test File 1

Figure 4-15

```
 LN SL                    TEXT

0000 0Inventory
0010Y1Product+ Product-Description
0020C1Quantity Manufacturer Shipping-Weight
0030  END
0040 0Customer
0050Y1Customer+ Customer-Name Address Phone
0060  END
0070 0Invoice
0080Y1Invoice+ Customer+ Data-of-Invoice
0090C1Gross-Amount
0100Y2Product+ Product-Description
0110C2Cost
0120  END
0130 0Accounts-Receivable
0140Y1Customer+
0150Y2Invoice+ Amount-Receivable
0160N1Customer-Total
0170  END
0180 0Backlog
0190Y1Product+
0200Y2Customer+
0210Y3Invoice+ Amount-Backordered
0220N2Bl-for-Customer
0230N1Total-for-Product
0240  END
0250  END
```

+: Number

Input Test File 2

Figure 4-16

```
 LN SL   TEXT
0000 0Areport
0010Y1A
0020Y2B1 B2
0030S2C C2
0040N1D1
0050Y2E1 E2
0060N1F
0070Y2G1
0080  END
0090 0Breport
0100Y1H I
0110  END
0120 0Creport
0130Y1I J
0140  END
0150 0Dreport
0160Y1H J
0170  END
0180 0Ereport
0190Y1X1
0200Y2X2 A D
0210  END
0220 0Freport
0230Y1C A
0240Y2D X1 X2
0250  END
0260 0Greport
```

```
 LN SL   TEXT
0270Y1A X1 B
0280  END
0290 0Hreport
0300Y1B
0310Y2C X2
0320  END
0330 0Ireport
0340Y1K1
0350C1K2
0360C1K3
0370C1K4
0380C1K5
0390C1K6
0400Y2L1
0410C2K4
0420C2K5
0430  END
0440 0Error
0450Y1A
0460  END
0470 0Strange
0480Y1M
0490Y2N
0500N10
0510  END
0520  END
```

test file 3 which is pictured in figure 4-17. The system test phase was designed to test the flow and correct manipulation of data through the four subsystems; User Interface, Editor, FDGEN, and FDA. The primary test objectives of the FDGEN subsystem were that the data placed into the Input File was correctly read; that the input data was accurately presented to the interactive user; that the data as defined by the user was correctly manipulated by the subsystem; and that the F-D-Table producted (Output File) was properly generated as output, both as user output as well as input for the FDA.

The FDGEN test schema is presented at figure 4-18. It insures that each path through which data may traverse is tested for each condition which is allowed to occur along that path. The test path diagram is pictured at figure 4-19. Each test case consist of a group of test segments. Each segment consist of one or more routines which are to be tested using one or more conditions. Segments without stated conditions are tested by virtue of the fact that the data must pass through their routines. The test segments are as follows:

SEGMENT	ROUTINE	CONDITIONS
A	Parse-Record Up-Level Build-D-N-Table Fill-it	1. Reports with groups with status of C, N, Y, and S. 2. DNI's with more than ten characters.
B	Virtual-elim	1. Select, verify, and review VDI's. 2. Select, verify, restore, and review VDI's. 3. Select, review, and select

Input Test File 3

Figure 4-17

```
 LN SL                          TEXT

0000 0Test1
0010Y1SSAN-ID STR-ADDRESS CITY-ADDRESS ZIP-CDOE TELEPHONE
0020C1MAR-STATUS VEH-STATE LIC-NO MAKE YEAR DECAL-NC
0030C1PRC-FR-ORG SOC-FR-ORG REL-EMPLMT EMP-ON-CAM EMP-OF-CAM
0040C1TUIT-ASST SVC-BRANCH TOT-SAL
0050Y2THIS IS A JUNK TEST GROUP
0060  END
0070 0TEST2
0080Y1SSAN-ID STU-NAME CITY-BIRTH ST-BRITH DATE-BIRTH
0090C1HS-NAME HS-CITY HS-STATE DR-GRAD NOK-NAME NOK-ADD
0100C1NOK-CITY NOK-STATE NCK-ZIP NCK-TELE DT-ENROLL
0110C1PRCJ-GR-DT ST-COLLEGE LEVEL-ID PRI-MAJOR SEC-MAJOR
0120C1ADV-NAME DEG-SOUGHT NO-SIST NC-BRCTH SCC-FR-ORG
0130C1PRC-FR-ORG TUIT-ASST MAR-STATUS
0140Y2STU-NAME F-NAME M-NAME
0150  END
0160 0TEST3
0170Y1SPECIAL TEST TC SHOW
0180Y2THAT
0190Y3THE WCRKS
0200N1THAT WHAT HAPPENS IS
0210  END
0220 0TEST7
0230Y1MCDEL NAKE
0240Y2ENG-NO STYLE EXTERIOR INTERIOR BODY-NC
0250Y3BCDY-NO YEAR WEIGHT COST PRICE
0260  END
0270 0TEST9
0280Y1ENG-NO NO-CYL DISPL ENG-WT ENG-COST HP
0290  END
0300 0TEST46
0310Y1STYLE EXTERIOR INTERIOR
0320Y2EXTERIOR NO-DOORS PAINT-CCDE TRIM-PK
0330S2INTERIOR CARGO-SP UPHOLSTERY SEATS ASSARY-PK
0340  END
0350 0TEST58
0360Y1PAINT-CODE COLOR-1 ICNE-NC
0370Y2TCNE-NO COLOR-2
0380  END
0390  END
```

Test Schema

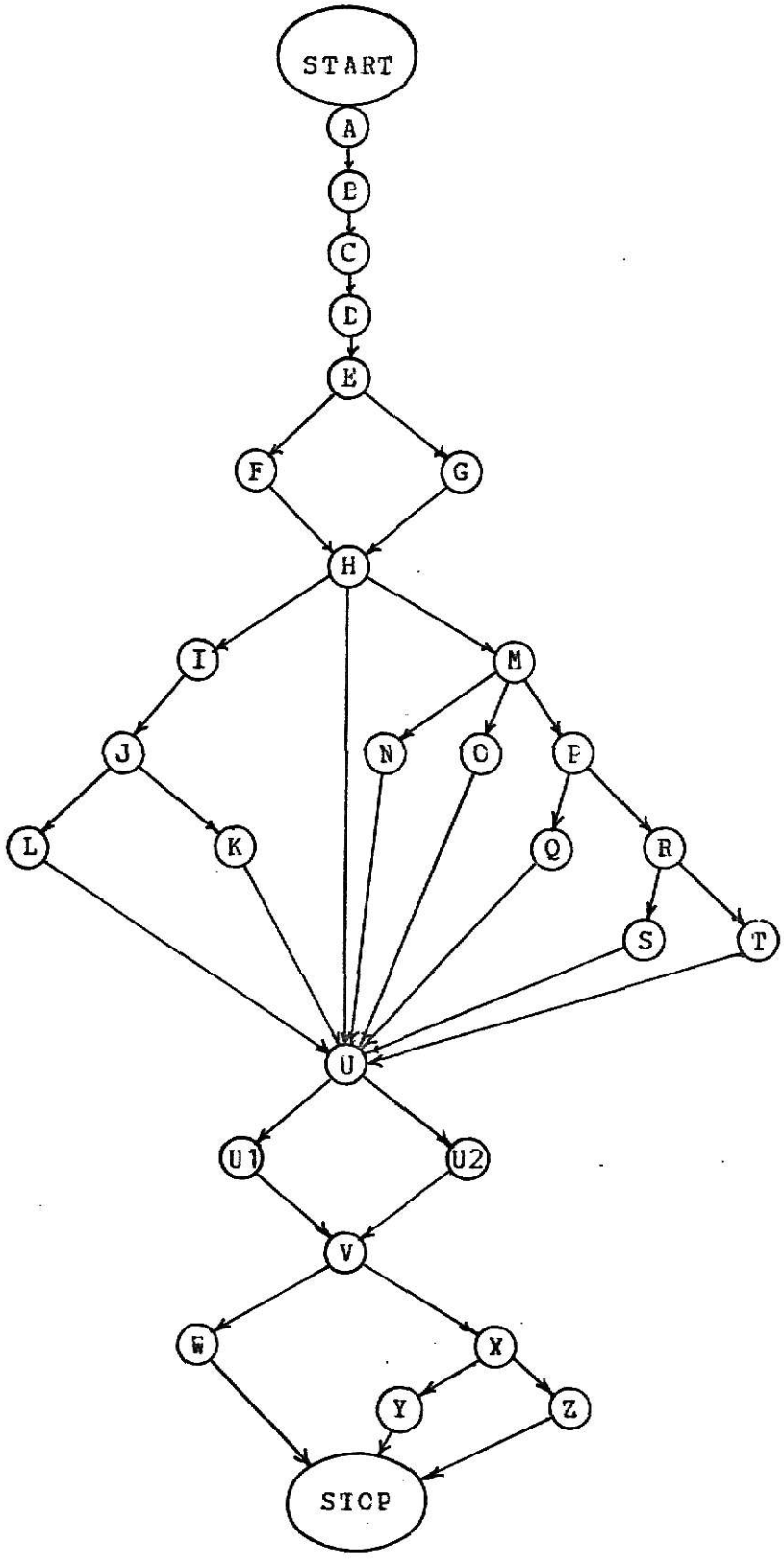
Figure 4-18

TEST CASES

Case No	Segments
1	A12 E1 C D E12 F H I J K U U2 V X Z
2	A1 B3 C D E1 F H I J K U U2 V X Y
3	A1 E1 C D E1 G H I J L U U1 V W
4	A1 B1 C D E1 F H M N U U2 V W
5	A1 E1 C D E1 F H M O U U2 V W
6	A1 B1 C D E1 F H M P Q U U2 V W
7	A1 E1 C D E1 F H M P R S U U2 V W
8	A1 B1 C D E1 F H M P R T U U2 V W
9	A1 E1 C D E1 F H U U2 V W

FDGEN Test Path Diagram

Figure 4-19



VDI's.

C	Report-Intersect Intersect Loop2	One or more reports with intersectable DNI's.
D	Determ-identify CleadFD Make-null Cat-Clear	
E	Savclear Fill-it Check-Name Check-DName Set-Key	1. Two or more reports with two or more groups. 2. Misspell Determinants. 3. Enter duplicate Determinants for the same group. 4. Non-selection of Determinant for one group.
F	Load-Cat-Stack Push-Stack Cat-union	1.Y Two groups with at least one Determinants each. The level of the second must be greater than the first. 2.N Two Determinants of the same level but seperated by Determinants of different levels. 3.C Continued group.
G	Load-Cat-Stack Pop-Stack Push-Stack Cat-union	1.S Two groups with levels not equal to one, both groups seperate but of the same level. 2.N Two groups - one with a level greater than one, the second Determinant selected with a level less than the first.
H	Build-F-D-Table De-lete Remove Check-List	Select a Determinant.
I	Spec-Case Disp-Det	Determinants identify themselves. The set remains unchanged.
J	Next-Q	Elect to change the set configuration.
K	B-New-Relation L-to-R Pack-it-Down Move-Det2	Elect to move all Determinants which have been concatenated to the last group's Determinant.
L	Sel-items	Elect to move one or more of

	Move-Det-Rt Search-Stack Compact	the Determinants which were concatenated to the last group's Determinant. Vary the number moved.
M	Spec-Case2	Two or more Determinants of different groups concatenated together.
N	Sav-More Transfer-Det Make-Room Savclear New-Union-Main New-union Compact	The current Determinant level must equal the successor level. Use status S and levels 1 and 2 intermixed.
O	Sav-Last	The current Determinant level is not equal to the successor level.
P	Disp-Det	The current Determinant level is not equal to the successor level.
Q	Sav-Last	Elect not to change the set configuration.
R	Sav-Last Next-Ques	Elect to change the set configuration.
S	Sel-items Move-Det-Rt Search-Stack Compact	Elect to move one or more of the Determinants which was concatenated to the last group's Determinants. Vary the size of the set moved.
T	B-New-Relation L-to-R Pack-it-D Move-Deter	Elect to move all Determinants not of the current group.
U	Load1	
U1		Self identifing Determinant.
U2	Load2	Non-determinants exist.
V	Next-Proc Load-Table Last-Proc Print-F-D-Table Genout	

	Build-L-S	
W	Build-R-S	Less than twelve Determinants and non-determinants.
X	Build-R-S More-Lines	More than twelve Determinants and non-determinants.
Y	Print-it Build-R-S Build-R-S	More than 24 Determinants and non-determinants.
Z	Build-R-S	Twelve or less Determinants and non-determinants.

The user's output of input test file 1 is pictured in figure 4-20. The file specifically tested the following segments and associated test conditions: A1(C,N,Y), A2, B1, B2, B3, C, D, E1, F1, F3, G2, H, I, L, M, P, R, S, U, U1, U2, V, and W.

The user's output of input test file 2, as manipulated to test segments A1(C,N,S,Y), C, D, E2, F1, F2, F3, G1, G2, H, I, J, M, N, O, P, Q, R, S, T, U, U1, U2, V, and W, is pictured in figure 4-21.

Input test file 3 is seperated into test reports. Each report name specifies the specific test case to be tested. The file must be input three times inorder to test all conditions. The user's output is presented in figures 4-22, 4-23, and 4-24. The Output File generated from the F-D-Table is pictured in figure 4-25. This file is stored on disc an is later used as input for the FDA subsystem.

User's Output of Test File 1

Figure 4-20

```
PRCDUCT+    --->   PRCLUCT-DE QUANTITY  MANUFACTUR SHIPPING-W
CUSTOMER+   --->    CUSTOMER-ADDRESS   PHONE
INVOICE+    --->   CUSTCMER+  DATE-OF-IN GROSS-AMOU
PRODUCT+    --->   PRODUCT-DE COST       INVCICE+
CUSTCMER+   --->   :01
CUSTOMER+ INVOICE+      --->   AMOUNT-REC
PRCDUCT+    --->   :02
PRODUCT+  CUSTOMER+     --->   :03
CUSTCMER+ INVOICE+    --->   AMOUNT-BAC PRODUCT+
BL-FOR-CUS   --->   :04
```

User's Output of Test File 2

Figure 4-21

```
A              --->  L1          F
A          B1                --->  B2
C            --->  C2            A
E1           --->  E2            A
A          G1              --->  :01
H           --->  I
I           --->  J
H           --->  J
X1          --->  :02
X2          --->  A           D          X1
C           --->  A
C        D              --->  X1          X2
A           --->  X1          B
B           --->  :03
C           --->  X2
K3          K2          K1                --->   K4          K5
K6
K5          K4          L1                --->   K3          K2
K1
A           --->  :04
M           --->  O
N           --->  M
```

Test Run 1 -- Input Test File 3

Figure 4-22

```
SSAN-ID    --->  STR-ADDRES  CITY-ADDRE ZIP-CODE   TELEPHONE
MAR-STATUS VEH-STATE  LIC-NO   MAKE    YEAR    DECAL-NO PRO-
FR-ORG SOC-FR-ORG REL-EMPLMT EMP-ON-CAM EMP-OF-CAM TUIT-ASST
SVC-BRANCH TOT-SAL
IS  THIS   --->  SSAN-ID   TEST
SSAN-ID    --->  STU-NAME   CITY-BIRTH ST-BIRTH   DATE-BIRTH
HS-NAME HS-CITY  HS-STATE  DR-GRAD   NOK-NAME  NOK-ADD  NOK-
CITY NOK-STATE NOK-ZIP   NOK-TELE  DT-ENROLL  PROJ-GR-DT ST-
COLLEGE LEVEL-ID  PRI-MAJOR  SEC-MAJOR  ADV-NAME  DEG-SOUGHT
NC-SIST  NC-BROTH   SOC-FR-ORG  PRO-FR-ORG   TUIT-ASST  MAR-
STATUS
STU-NAME  --->  F-NAME  M-NAME  SSAN-ID
SPECIAL  TEST   --->  TO      SHOW THAT WHAT HAPPENS IS
SPECIAL TEST THAT   --->  :01
THAT   THE   --->  SPECIAL  TEST    WORKS
MODEL  --->   MAKE
MODEL   BODY-NO   --->   ENG-NO STYLE  EXTERIOR  INTERIOR
MODEL  BODY-NO   --->   YEAR   WEIGHT  COST   PRICE
ENG-NO   --->   NO-CYL DISPL  ENG-WT   ENG-COST   HP
EXTERIOR  STYLE   --->  INTERIOR
EXTERIOR  STYLE   --->  NO-DOORS  PAINT-CODE  TRIM-PK
STYLE    EXTERIOR   INTERIOR   --->    CARGO-SP   UPHOLSTERY
SEATS   ASSARY-PK
PAINT-CODE   --->   COLOR-1   TONE-NO
TONE-NO   --->   COLOR-2   PAINT-CODE
```

Test Run 2 -- Input Test File 3

Figure 4-23

```
SSAN-ID     --->  STR-ADDRES  CITY-ADDRE ZIP-CODE   TELEPHONE
MAR-STATUS  VEH-STATE  LIC-NO   MAKE   YEAR    DECAL-NO PRO-
FR-ORG SOC-FR-ORG REL-EMPLMT EMP-ON-CAM EMP-OF-CAM TUIT-ASST
SVC-BRANCH TOT-SAL
SSAN-ID     --->  STU-NAME  CITY-BIRTH  ST-BIRTH   DATE-BIRTH
HS-NAME HS-CITY  HS-STATE  DR-GRAD  NOK-NAME  NOK-ADD   NOK-
CITY NOK-STATE  NOK-ZIP  NOK-TELE   DT-ENROLL PROJ-GR-DT ST-
COLLEGE LEVEL-ID  PRI-MAJOR  SCF-MAJOR  ADV-NAME  DEG-SOUGHT
NO-SIST NO-BROTH SOC-FR-ORG PRO-FR-ORG TUIT-ASST MAR-STATUS
STU-NAME    --->  F-NAME  M-NAME  SSAN-ID
SPECIAL  TEST     --->  TO SHOW THAT WHAT   HAPPENS  IS
SPECIAL  TEST     THAT      --->  :01
THAT   THE        --->  SPECIAL TEST   WORKS
MODEL   --->  MAKE
MODEL  INTERIOR  EXTERIOR  STYLE   ENG-NO     --->  BODY-NO
MODEL   INTERIOR  EXTERIOR   STYLE    BODY-NO     --->   YEAR
WEIGHT COST   PRICE   ENG-NO
ENG-NO   --->  NO-CYL   DISPL   ENG-WT   ENG-COST  HP
STYLE   --->  EXTERIOR  INTERIOR
STYLE   EXTERIOR   --->  NO-DOORS  PAINT-CODE TRIM-PK
STYLE     INTERIOR      --->   CARGO-SP    UPHOLSTERY   SEATS
ASSARY-PK
PAINT-CODE   --->  COLOR-1   TONE-NO
TONE-NO    --->  COLOR-2  PAINT-CODE
```

Test Run 3 -- Input Test File 3

Figure 4-24

```
SSAN-ID    ---> STR-ADDRES  CITY-ADDRE ZIP-CODE   TELEPHONE
MAR-STATUS VEH-STATE  LIC-NO   MAKE   YEAR   DECAL-NO   PRO-
FR-ORG SCC-FR-ORG REL-EMPLMT EMP-ON-CAM EMP-OF-CAM TUIT-ASST
SVC-BRANCH TOT-SAL
THIS      ---> IS     A       JUNK    TEST    GROUP     SSAN-ID
STU-NAME  SSAN-ID      ---> CITY-BIRTH  ST-BIRTH  DATE-BIRTH
HS-NAME    HS-CITY     HS-STATE  DR-GRAD   NOK-NAME   NOK-ADD
NOK-CITY  NOK-STATE  NOK-ZIP  NOK-TELE   DT-ENRCLL  PROJ-GR-
DT ST-CCLLEGE LEVEL-ID  PRI-MAJOR   SEC-MAJOR ADV-NAME  DEG-
SOUGHT NO-SIST  NO-BROTH  SCC-FR-ORG PRC-FR-CRG TUIT-ASST
SPECIAL  TEST    ---> TO       SHOW    THAT   WHAT   HAPPENS
IS
SPECIAL   TEST     THAT       --->   :C1
THAT   THE    ---> WORKS    TEST    SPECIAL
MCDEL    ---> MAKE
MODEL   STYLE    ENG-NO    ---> EXTERIOR  INTERIOR  BODY-NO
BCDY-NO   ---> YEAR   WEIGHT   COST    PRICE    MODEL STYLE
ENG-NO
ENG-NO    ---> NO-CYL   DISPL    ENG-WT   ENG-COST  HP
INTERIOT  EXTERIOR   ---> STYLE
EXTERIOR   ---> NO-DCORS  PAINT-CODE TRIM-PK   INTERIOR
EXTERIOR   INTERIOR     --->    CARGC-SP   UPHCISTERY  SEATS
ASSARY-EK
PAINT-CODE   ---> COLOR-1   TCNE-NC
TCNE-NO    ---> COLCR-2   PAINT-CODE
```

Output File Generated From F-D-Table

Figure 4-25

```
        001
SSAN-ID
        018
STR-ADDRES
CITY-ADDRE
ZIP-CODE
TELEPHONE
MAR-STATUS
VEH-STATE
LIC-NO
MAKE
YEAR
DECAL-NO
PRO-FR-ORG
SOC-FR-ORG
REL-EMPLMT
EMP-CN-CAM
EMP-OF-CAM
TUIT-ASST
SVC-BRANCH
TOT-SAL
        002
IS
THIS
        002
SSAN-ID
TEST
        001
SSAN-ID
        028
STU-NAME
CITY-BIRTH
ST-BIRTH
DATE-BIRTH
HS-NAME
HS-CITY
HS-STATE
DR-GRAD
NOK-NAME
NOK-ADD
NOK-CITY
NOK-STATE
NOK-ZIP
NOK-TELE
DT-ENRCLL
PROJ-GR-DT
ST-COLLEGE
```

Figure 4-25 Continued

```
LEVEL-ID
PRI-MAJOR
SEC-MAJOR
ADV-NAME
DEG-DOUGHT
NO-SIST
NO-BROTH
SOC-FR-ORG
PRO-FR-ORG
TUIT-ASST
MAR-STATUS
        001
STU-NAME
        003
F-NAME
M-NAME
SSAN-ID
        002
SPECIAL
TEST
        006
TO
SHOW
THAT
WHAT
HAPPENS
IS
        003
SPECIAL
TEST
THAT
        001
:01
        002
THAT
THE
        003
SPECIAL
TEST
WORKS
        001
MODEL
        001
MAKE
        002
MODEL
BODY-NO
        004
ENG-NO
STYLE
```

Figure 4-25 Continued

```
EXTERIOR
INTERIOR
        002
MODEL
BODY-NO
        004
YEAR
WEIGHT
COST
PRICE
        001
ENG-NO
        005
NO-CYL
DISPL
ENG-WT
ENG-COST
HP
        002
EXTERIOR
STYLE
        001
INTERIOR
        002
EXTERIOR
STYLE
        003
NO-DOORS
PAINT-CODE
TRIM-PK
        003
STYLE
EXTERIOR
INTERIOR
        004
CARGO-SP
UPHOLSTERY
SEATS
ASSARY-PK
        001
PRINT-CODE
        002
COLOR-1
TONE-NO
        001
TONE-NO
        002
COLOR -2
PAINT-CODE
```

5.0 CONCLUSION

To the unsophistocated user a Relational Data Model is difficult to conceptualize. It presents him with a unique task of data relatien development which normally would be beyond his capability. The FDGEN as implemented provides the user with a sophistocated tool to aid him in the development of a Relational Data Base. The user is able to enter his data in logical segments and transform it into sets of related data which can be manipulated into Functional Dependencies. The manipulation of the data is guided by the FDGEN Program but directed by the user. This process allows for maximum interaction with the user, the one who knows most about the data and its interrelationship. The final product is a set of FD's which is provided to the user and is made available for synthesis into 3NF Relations by the FDA Program of the Third Normal Form Synthesis System. (16)

5.1 ENHANCEMENT CONSIDERATIONS

The FDGEN was designed to provide its user with a copy of its final product - a list of the Functional Dependencies generated. A user after reviewing the list of FD's and discovering that incorrect FD's were generated might not be able to recall the interactive query responses which produced the incorrect FD's. This would mean that the user would have to regenerate the FD's and hope that he did not make the same error again.

The possiblity of trial and error manipulaticns by a user can be eliminated by making a few modifications to the FDGEN design. The essence of the modification would be to provide the user with a ccpy of each query and his response to that query. This modification would allow the user to map a new course of actien prior to regeneration of the FD's.

The modifications can be accomplished by assigning each query as a constant to an elementry data item; replacing each query in the Prccedure Division with its equivalent elementry data item; and following each query, insert the assignment of the query tc the output file for output to the line printer. All query responses are output to the line printer alsc. The modificatien can be implemented without increasing the current program size.

A second enhancement which is applicable for inclusion in the FDGEN Subsystem is a data relatien preprocessor module. The preprocessor module would extend the capability of the unsophistocated user with respect to the generation of FD's. It would allow him to enter all related data as logical sets. This means that data could be entered while in First Normal Form (1NF). In order for the data to be in 1NF all Data-Name-Items in the relatien must be atomic. The current system restricts each relation to those Data-Name-Items which will be used for the censtruction of one FD. Included in the FD may be concatenated Determinants from other relations within the hierarchical report. The

relation as used is synonymous to a group.

The addition of a preprocessor module to the FDGEN Program would require major design modifications to the input modules of the program. The preprocessor would be interactive and would replace the current Parse-Record module and its submodules.

Reference

1. Tsichritzis, D. C. and Lochovsky, F. H., Data Base Management Systems , Academic Press, Inc., New York, New York, 1977, pp. 3-5.

2. Chamberlin, D. D., 'Relational Data Base Management Systems'. Computing Surveys, volume Eight 1 (March, 1976) : 43.

3. Maryanski, F, J., Buser, R. H., Hoflich, M. A., Hunt, W. C., Kusnyer, S. K., and Stevens, T. J. 'Automatic Generation of Third Normal Form Relations'. Technical report number CS 77-21, Conputer Science Department, Kansas State University, Manhattan, Kansas, 1977, pp. 1-20.

4. Codd, E. F. 'Futher Normalization of the Data Base Relational Model', Data Base Systems , ed. Randall Rustin, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1972, pp. 33.

5. Tsichritzis, pp. 42.

6. Tsichritzis, pp. 47-53.

7. Chamberlin, pp. 48.

8. Maryanski, pp. 2.

9. Maryanski, pp. 4.

10. Maryanski, pp. 12.

11. Maryanski, pp. 12-17.

12. Maryanski, pp. 8.

13. Maryanski, pp. 6-8.

14. Maryanski, pp. 14.

15. Maryanski, pp. 2-5.

16. Maryanski, pp. 20-29.

Bibliography

1. Tsichritzis, D. C. and Lochovsky, F. H., Data Base Management Systems , Academic Press, Inc., New York, New York, 1977.

2. Chamberlin, D. D., 'Relational Data Base Management Systems'. Computing Surveys, Volume Eight 1 (March, 1976): 43-66.

3. Maryanski, F. J., Buser, R. H., Hoflich, M. A., Hunt, W. O., Kusnyer, S. K., and Stevens, T. J. 'Automatic Generation of Third Normal Form Relations'. Technical report number CS 77-21, Computer Science Department, Kansas State University, Manhattan, Kansas, 1977.

4. Codd, E. F. 'Futher Normalization of the Data Base Relational Model', Data Base Systems , ed. Randall Rustin, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1972, pp. 33-64.

APPENDIX-A FDGEN ABSTRACT CODE

The FDGEN is of modular construction. All major modules are supported by submodules. All variables are global to all modules.

Standard Declarations are assumed.

Main Module:

FDGEN:

Begin.

Do until all records have been read.

Read a record. Discard line number. If record ¬= END then parse remainder of record.

END.

Insert end marker at end of Data-Name-Table. Mark all Virtual-Data-Items. Mark all Data-Name-Items which appear in more than one report. Identify Determinants. Print output table.

END.

END FDGEN.

Sub-Module:

-Parse remainder of record:

Parse-Record:

Begin.

Separate record into status, level-number, and remainder of the record.

If report has no entries than error. If end of record then return. If level-number is

equal tc cr less than the last level-number and is not a continuation then increase the concatenation code by 1 for all remaining levels.

Do until end of record.

Check length of first Data-Name-Item in record. If length > 10 then error, truncate remaining characters. Remove first Data-Name-Item from record. Continue construction of the Data-Name-Table.

END.

END.

END Parse-Record.

-Increase concatenation code:

Up-level:

Begin.

With level-number as index. From current index--increment each concatenation code by 1 - To end of table.

END.

END up-level.

-Continue construction of the Data-Name-Table:

Build-D-N-Table:

Begin.

```
          Increment table index by 1.  Assign Data-Name-Item
          to  name  field  (fld).  Assign  level-number  to
          level-no fld.  Assign 0 to Determ-flag fld.
               Begin.
                    Reinitialize concatenation-list.  Assign
                    NUL to  concatenation-link fld.  If not
                    first report  then assign  index of  new
                    report to  last report's  concatenation-
                    link fld ELSE  assign concatenation-code
                    to concatenation-link fld.
               END.
     END.
END Build-D-N-Table.

-Reinitialize concatenation-list:
Fill-it:
     Begin.
          Index <-- 1.
               Assign 11 to first indexed location.
               Do 8 times.
                    Increment index by  1.  Increment assign
                    value  by  10.  Assign  new  value  to
                    concatenation-code (index).
               END.
          END.
END Fill-it.
```

```
-Mark all Virtual-Data-Items:
Virtual-Elim:
     Begin.
          Index <-- 1.  Display  instructions and a tutorial
          of a Virtual-Data-Item.
          Do until end of Data-Name-Table.
               Display report name.
               Do until report name changes.
                    If  level-no =  0  then error.  Display
                    value of name fld.  If Determ-Flag fld =
                    3 then name  previously identified.  Ask
                    if name is a  Virtual-Data-Item.  If yes
                    then  assign   3  to   Determ-Flag  fld.
                    Increment index by 1.
               END.
          END.
     END.
END Virtual-Elim.

-Mark  all Data-Name-Items  which appear  in  more than  one
report:
Report-Intersect:
     Begin:
          Select first report.
          Do until selected report's conc-link fld = NUL.
               If level-no fld  = 0 then conc-link  is index
               to next report. Compare each  name fld of the
```

```
                current report with each name fld of all
                subsequent reports within the Data-Name-
                Table. Select the next report in sequence as
                the current report.
            END.
        END.
END Report-Intersect.

-Compare each name field of the current report...:
Intersect:
        Begin.
            Select name fld to be compared. If name fld
            previously matched then return. Select beginning
            index from which to begin comparing. Compare name
            flds until end of table.
        END.
END Intersect.

-Compare name fields until end of table:
Loop2:
        Begin.
            Select name fld to be compared.
            Do until end of table.
                If name flds match and selected name is not a
                report name or a Virtual-Data-Item then
                assign 2 to the Determ-Flag fld. Select next
                name fld to be compared.
```

```
          END.
     END.
END Loop2.

-Identify Determinants:
Determ-Identify:
     Begin.
          Display instructions and a tutorial of a
          Determinant. Clear the Functional Dependency
          Table, S-Stack Table, and the L-Stack Table.
          Initialize indices.
          Do until end of Data-Name-Table.
               Reinitialize concatenation-list.
               Do until end of current report.
                    Increment index by 1. If level-no fld =
                    0 then conc-link fld is index to next
                    report. Display report name.
                    Do until all unique conc-link fld values
                    have been found.
                         Display all name fld values of
                         current report with the same conc-
                         link fld value. Assign each
                         selected name fld value to the S-
                         Stack Table. If name fld value is
                         an intersected item then display
                         asterisk by name.
                         END.
```

```
                    Display instructions as to Determinant
                    selection.
                    Do until no more selections.
                        Accept selected determinant.  If
                        spelled incorrectly then error ELSE
                        If duplicate entry  then error ELSE
                        assign reply to S-Stack Table.  Set
                        Determ-Flag fld to 1.
                    END.
                    If no Determinant selected for displayed
                    group then error--discard group ELSE
                    assign reply to L-Stack  Table and count
                    all new entries.  Build next part of the
                    Functional Dependency Table (FDT).
                END.
            END.
            Prepare to write the FDT to disk.
        END.
    END Determ-Identify.

    -Duplicate entry:
    Check-DName:
        Begin.
            If  reply  is  =  to  any  previous reply  for  the
            displayed group then error.
        END.
    END Check-DName.
```

```
-Spelled incorrectly:
Check-Name.
     Begin.
          If reply does not match any ATTR fld value in S-
          Stack then error.
     END.
END Check-Name.

-Prepare to write the FDT to disk:
Load-table:
     Begin.
          Assign one record at a time from the FDTABLE in
          Working Storage to a file name. Write the file to
          disk until all files have been written.
     END.
END Load-table.

-Assign reply to L-Stack Table and count the number of new
entries:
Load-Cat-Stack:
     Begin.
          Digit gets first digit of conc-link fld value.
               Case.
                    Digit > last key: Push all Determinants
                    on L-Stack.
                    Digit = 1: Push all Determinants on L-
                    Stack.
```

```
                    Digit = last key and last key ¬= 1:
                         Pop Determinants off  L-Stack until
                         Lev fld  value <  digit.  Push  all
                         Determinants on L-Stack.
                    Digit < last key:
                         Pop Determinants off  L-Stack until
                         Lev fld  value <  digit.  Push  all
                         Determinants on L-Stack.
               END Case.
          Last key <-- digit.
     END.
END Load-Cat-Stack.
   NOTE: Last key is initially 1.

-Push all Determinants on L-Stack:
Push-Stack:
     Begin.
          Do count times.
               Increment index of L-Stack  by 1.  D-Name fld
               <-- Attr fld  value.  Decrement  index of  S-
               Stack by 1.
          END.
     END.
END Push-Stack.

-Pop  all Determinants  off L-Stack  until Lev  fld value  <
digit:
```

```
Pop-Stack:
     Begin.
          Do until Lev fld value < digit or value = 1.
          Decrement L-Stack index by 1.
          END.
     END.
END Pop-Stack.

-Set Determ-Flag field to 1:
Set-Key:
     Begin.
          Set index to beginning of current report.
          Do until all name flds of current report have been
          compared.
               Increment index of Data-Name-Table by 1. If
               not a Virtual-Data-Item and name fld value
               matches Attr field value of S-Stack then
               assign 1 to Determ-Flag fld.
          END.
     END.
END Set-Key.

-Build next part of the Functional Dependency Table:
Build-F-D-Table:
     Begin.
          Status <-- 0. Save L-Stack and S-Stack indices.
          Delete Determinants from S-Stack. Save the number
```

of Data-Name-Items in Hold-no. If the last level number was greater than 1 then special-case2 Save the last level number referenced. Increment F-D-Table index by 1. Assign number of Determinants in L-Stack tc list fld.

(The number must be right justified)

Lcad that number of Determinants into F-D-Table. If a Determinant identifies itself then build a unique symbol ccnsisting of a colon (:) followed by a unique 2-digit number (i.e., :01).

Increment indcx by 1. Assign number cf Data-Name-Items in S-Stack to F-D-Table. Load that number of Data-Name-Items into F-D-Table. If S-Stack is empty then lcad the special symbol.

END.

END Build-F-D-Table.

-Special-case1:

Spec-Case:

Begin.

Display tutcrial. Initialize L-Stack. Display L-Stack. Display instructions. Ask questions. If yes then ask next question.

END.

END Spec-Case.

-Display I-Stack:

```
Disp-Det:
     Begin.
          Display each Determinant.  Count the number of
          Determinants.
     END.
END Disp-Det.

-Ask next question:
Next-Q:
     Begin.
          Ask question.  If yes then move all Determinants
          whose level number does not match the saved level
          number to the right side.  Compact L-Stack.
          If no then select the Determinant to be moved.
     END.
END Next-Q.

-Move all Determinants whose level number...:
B-New-Relation:
     Begin.
          Initialize L-Stack index to 1.  Determine the
          number of Determinants to be moved.  Move that
          many Determinants to the right side.
     END.
END B-New-Relation.

-Move that many Determinants to the right side:
```

```
L-to-R:
     Begin.
          Increment S-Stack index by 1.  ATTR fld <-- D-Name
          fld value.  Increment L-Stack index by 1.
     END.
END L-to-R.

-Compact L-Stack:
Pack-it-down:
     Begin.
          Move all  remaining D-Name fld  values to  the low
          end of L-Stack.
     END.
END Pack-it-down.

-Move all remaining D-Name...:
Move-Det2:
     Begin.
          Compact to the low end of L-Stack all values in L-
          Stack.  Level fld <-- 11.
     END.
END Move-Det2.

-Select the Determinant to be moved.
Sel-items:
     Begin.
          Display instructions.
```

Do until no more Determinants are selected. ATTR fld <-- D-Name fld value selected.

END.

END.

END Sel-items.

-ATTR field <-- D-Name field value selected:

Move-det-rt:

Begin.

Set a flag. L-Stack index <-- 1. Search L-Stack for the selected Determinant. If flag still set then spelling error. L-Stack index <-- 1. Compact L-Stack again. Increment the size of S-Stack by 1.

END.

END Move-det-rt.

-Search L-Stack for the selected Determinant:

Search-stack:

Begin.

If D-Name fld value selected then ATTR fld <-- D-Name fld value Change flag setting. Increment L-Stack index by 1. ELSE increment L-Stack index by 1.

END.

END Search-Stack.

-Compact L-Stack again:

Compact:

Begin.

Move all remaining D-Name and Level fld values to the low end of L-Stack.

END.

END Compact.

-Special-case2:

Spec-case2:

Begin.

If the level saved is equal to the last level referenced and that level is not 1 then STATUS <-- S. If STATUS is S then move all Determinants from Save-Stack tc L-Stack which have higher levels than the highest level listed in Level fld. L-stack index <-- 0. Union all D-Name fld values. If only one Determinant exists then Save L-Stack in Save-Stack ELSE Ask question.

Display L-Stack. If yes then save L-Stack in Save-Stack. Ask next question in sequence. ELSE save L-Stack in Save-Stack.

END.

END Spec-case2.

-Move all Determinants frcm Save-Stack...:

Sav-move:

```
     Begin.
          Save stack index <-- 1.  Transfer all Determinants
          which meet the requirements.  Save-Stack index <--
          1.  Clear Save-Stack.
     END.
END Sav-move.

-Transfer all Determinants which meet the requirements:
Transfer-det:
     Begin.
          Set  L-Stack index  to top  of stack.  Save-Stack
          index <--  1.  If S-Name  has a higher  level than
          the tested  D-Name then  Move all  L-Stack  values
          one position higher.  L-Stack index <-- 1.  D-Name
          fld <-- S-Name fld value.  ELSE  check the next S-
          Name fld values level.
     END.
END Transfer-det.

-Move all L-Stack values cne postion higher:
Make-room:
     Begin.
          Move each  element of L-Stack one  positicn higher
          so that the low position will be vacant.
     END.
END Make-rocm.
```

```
-Clear Save-Stack:
Saveclear:
     Begin.
          Fill Save-Stack with spaces.
     END.
END Savclear.

-Union all D-Name field values:
New-union-main:
     Begin.
          Increment L-Stack index by 1. Match <-- D-Name
          fld value. Flag <-- 1. Perform union operation.
          If flag is 3 then compact L-Stack again ELSE
          decrement L-Stack index by 1.
     END.
END New-unicn-main.

-Preform union operation:
New-unicn:
     Begin.
          Increment L-Stack index by 1.  If match equals D-
          Name fld value than flag gets 3.
     END.
END New-union.

-Save L-Stack in Save-Stack:
Sav-last:
```

```
     Begin.
          Save-Stack <-- L-Stack.
     END.
END Sav-last.

-Ask next question in sequence:
Next-Ques:
     Begin.
          Display instructions.  If yes then move all
          Determinants whose level number does not match the
          saved level number to the right side.
          Pack L-Stack again.  If no then select the
          Determinant to be mcved.
     END.
END Next-Ques.

-Pack L-Stack again:
Pack-It-D:
     Begin.
          Mcve all remaining D-Name and  Level fld values to
          the low end of L-Stack.
     END.
END Pack-It-D.

-Move all remaining D-Name and Level field value...:
Move-Determ:
     Begin.
```

```
          Compact to the low end of L-Stack all values in L-
          Stack.
     END.
END Move-Determ.

-Load that number of Determinants into F-D-Table:
Load1:
     Begin.
          L-Stack index <-- 1.
          Do until L-Stack is empty.
               Increment F-D-Table index by 1.  List fld <--
               D-Name fld value.  Increment L-Stack index by
               1.
          END.
     END.
END Load1.

-Delete Determinants from S-Stack:
De-lete:
     Begin.
          Delete each  Determinant assigned to  L-Stack from
          S-Stack twice.
     END.
END De-lete.  (Delete includes Remove and Check-List)

-Load that number of Data-Name-Items into F-D-Table:
Load2:
```

```
     Begin.
          S-Stack index <-- 1.
          Do until S-Stack is empty.
               Increment F-D-Table index by 1.  List fld <--
               ATTR fld  value.  Increment S-Stack  index by
               1.
          END.
     END.
END load2.

-Print output file:
Print-F-D-Table:
     Begin.
          F-D-Table index <-- 1.  Do until end of F-D-Table.
               Build  left side  of  output.  Add arrow  to
               output.  Build  right side of  output.  Print
               output.
          END.
     END.
END Print-F-D-Table.

-Build left side of output:
Build-L-S:
     Begin.
          Increment F-D-Table index by 1.
          Do list fld value times.
               Generate a  string of  Determinants separated
```

```
                    by blanks.  Increment F-D-Table index by 1.
               END.
          END.
END Build-L-S.

-Build right side of output:
Build-R-S:
          Begin.
               Increment F-D-Table index by 1.
               Do list fld value times.
                    Continue generating the  string consisting of
                    Determinants and  the arrow  by adding  Data-
                    Name-Items to  the  string.  Increment  F-D-
                    Table index by 1.
               END.
          END.
END Build-R-S.
```

FDGEN CODE - NCR LEVEL II COBOL

```
 1*  000010 IDENTIFICATION DIVISION.
 2*  000020 PROGRAM-ID. F-D-GENERATOR.
 3*  000030 ENVIRONMENT DIVISION.
 4*  000040 CONFIGURATION SECTION.
 5*  000050 SOURCE-COMPUTER. NCR-CENTURY-8200.
 6*  000060 OBJECT-COMPUTER. NCR-CENTURY-8200.
 7*  000070 INPUT-OUTPUT SECTION.
 8*  000080 FILE-CONTROL.
 9*  000090
10*  000100     SELECT USER1-FILE ASSIGN TO DISC.
11*  000110
12*  000120     SELECT F-D-TABLE ASSIGN TO DISC.
13*  000130
14*  000140     SELECT F-D-PRINT ASSIGN TO PRINTER.
15*  000150
16*  000160 DATA DIVISION.
17*  000170 FILE SECTION.
18*  000180 FD USER1-FILE BLOCK CONTAINS 6 RECORDS, RECORD CONTAINS 76
19*  000190         CHARACTERS, LABEL RECORDS ARE STANDARD.
20*  000200 01  USER1-RECORD PIC X(76).
21*  000210
22*  000220 FD  F-D-TABLE BLOCK CONTAINS 51 RECORDS, RECORD CONTAINS 10
23*  000230         CHARACTERS, LABEL RECORDS ARE STANDARD.
24*  000240 01  F-D-TABLE-ELE PIC X(10).
25*  000250 FD  F-D-PRINT LABEL RECORDS ARE OMITTED.
26*  000260 01  PRINT-LINE PIC X(132).
27*  000270 WORKING-STORAGE SECTION.
28*  000280 01  FDTABLE.
29*  000290     02 LIST PIC X(10) OCCURS 250 TIMES INDEXED BY F-D-INDEX.
30*  000300 01  DATA-NAME-TABLE.
31*  000310     03 DATA-N OCCURS 200 TIMES INDEXED BY ANINDEX.
32*  000320          05  NAME          PIC X(10).
33*  000330          05  LEVEL-NO      PIC 9.
34*  000340          05  CONC-LINK     PIC XXX.
35*  000350          05  DETERM-FLAG   PIC 9.
36*  000360 01  S-STACK.
37*  000370
```

```
000380     02  ATTR PIC X(10) OCCURS 50 TIMES INDEXED BY STACK-INDEX.
000390 01  L-STACK.
000400     02  CAT-STACK OCCURS 20 TIMES INDEXED BY CAT-INDEX.
000410     04  D-NAME  PIC X(10).
000420     04  LEV     PIC 99.
000430 01  CONC-LIST.
000440     02 CONC-CODE PIC 99 OCCURS 9 TIMES.
000450 01  SAVE-STACK.
000460     02  SA-SK OCCURS 20 TIMES INDEXED BY S-INDEX.
000470     04  S-NAME PIC X(10).
000480     04  SLEV PIC 99.
000490 77  RECONST-RECORD PIC X(76).
000500 77  HOLD PIC X(72).
000510 77  INTERMED-STRING PIC X(70).
000520 77  CHAR-COUNT PIC 999.
000530 77  STAT-HOLD PIC X.
000540 77  LEVEL-HOLD PIC 9.
000550 77  ATTRIBUTE-STRING PIC X(70).
000560 77  BREAK PIC X VALUE IS SPACES.
000570 77  COUNT-1 PIC 99.
000580 77  COUNT-2 PIC 99.
000590 77  WORD-COUNT PIC 9.
000600 77  RESIDUE PIC X(69).
000610 77  ATTRIBUTE PIC X(10).
000620 77  NEXT-LOC PIC 999 VALUE 0.
000630 77  FIND PIC 999 VALUE 1.
000640 77  LOOK PIC 9 VALUE 1.
000650 77  HOLD-LEVEL PIC 9.
000660 77  MAIN-LEVEL PIC 99 VALUE 11.
000670 77  LEVEL-CODE PIC 99.
000680 77  COWNT PIC 999 VALUE 0.
000690 77  KOUNT PIC 999 VALUE 0.
000700 77  REPLY PIC X(11).
000710 77  NEXT-RPT PIC XXX VALUE "1".
000720 77  THAT-MANY PIC 999 VALUE 0.
000730 77  CK-NEXT PIC 999.
000740 77  KNT PIC 999.
```

```
 75*  000750 77  MATCH PIC X(10).
 76*  000760 77  DO-IT PIC 999.
 77*  000770 77  SUB-ANINDEX PIC 999 VALUE 0.
 78*  000780 77  SET-FLAG PIC 9 VALUE 0.
 79*  000790 77  LINE-NO PIC 99 VALUE 4.
 80*  000800 77  FIRSTN1 PIC X.
 81*  000810 77  NEXTN1 PIC X(8).
 82*  000820 77  FIRSTN2 PIC X.
 83*  000830 77  NEXTN2 PIC X(8).
 84*  000840 77  KEY1 PIC XX.
 85*  000850 77  WORD PIC X(7).
 86*  000860 77  ARTICAL PIC XX.
 87*  000870 77  ONE-OR-MORE PIC X(13).
 88*  000880 77  VERB PIC XXX.
 89*  000890 77  L-S-COUNT PIC 9 VALUE 0.
 90*  000900 77  LAST-LEV PIC 9 VALUE 1.
 91*  000910 77  NUMBER1 PIC X(9).
 92*  000920 77  NUMBER2 PIC X(9).
 93*  000930 77  TOPP PIC 999.
 94*  000940 77  DOUBLE-CK PIC 9.
 95*  000950 77  CONVERT1 PIC 999.
 96*  000960 77  NUMBER3 PIC 999.
 97*  000970 77  NUMBER4 PIC 999.
 98*  000980 77  TOP-STACK PIC 99.
 99*  000990 77  THIS-ATTR PIC X(10).
100*  001000 77  LEFT-SIDE PIC 9(10).
101*  001010 77  RIGHT-SIDE PIC 9(10).
102*  001020 77  ARROW PIC XXX VALUE "-->".
103*  001030 77  HOW-MANY PIC 99 VALUE 0.
104*  001040 77  SPEC-SY PIC XXX.
105*  001050 77  COLON PIC X VALUE ":".
106*  001060 77  HOLD-NO PIC 999.
107*  001070 77  NUM-PART PIC 99 VALUE 1.
108*  001080 77  KEY2 PIC 9.
109*  001090 77  MOVERT PIC X(10) JUSTIFIED RIGHT.
110*  001100 77  K-TRACK PIC 99.
111*  001110 77  SET-TRAP PIC 9 VALUE IS 0.
```

```
112*   001120 77  THREE3 PIC 9 VALUE 3.
113*   001130 77  TWO2 PIC 9 VALUE 2.
114*   001140 77  KEY3 PIC XX.
115*   001150 77  KEY4 PIC X.
116*   001160 77  KEY5 PIC X.
117*   001170 77  NUMBER5 PIC 99.
118*   001180 77  SET-CHECK PIC 9 VALUE 0.
119*   001190 77  ADD-ON PIC 9.
120*   001200 77  DET-COUNT PIC 99.
121*   001210 77  HOLD-DET-COUNT PIC 99.
122*   001220 77  DET-KOUNT PIC 99.
123*   001230 77  LEV-HOLD PIC 99.
124*   001240 77  SAVETOP PIC 99 VALUE 0.
125*   001250 77  KNTT PIC 99.
126*   001260 77  KNTTY PIC 99.
127*   001270 77  KOUNTY PIC 99.
128*   001280 77  HOL-LEV PIC 9.
129*   001290 77  SAVE-LEVEL PIC 9 VALUE 0.
130*   001300 PROCEDURE DIVISION.
131*   001310 BEGIN1.
132*   001320     DISPLAY "ANY INFORMATION THAT APPEARS ON THE SCREEN WITHOUT
133*   001330-                "ANY INSTRUCTIONS MAY BE", LINE 10, ERASE.
134*   001340     DISPLAY "REMOVED BY DEPRESSING NEW LINE.**PRESS NEW LINE.".
135*   001350
136*   001360     ACCEPT REPLY.
137*   001370     IF REPLY IS NOT EQUAL TO "DKUJRK" DISPLAY " ", ERASE.
138*   001380     OPEN INPUT USERI-FILE.
139*   001390     OPEN OUTPUT F-D-PRINT.
140*   001400 RE. READ USERI-FILE INTO RECONST-RECORD AT END GO TO V-E.
141*   001410     MOVE 5 TO CHAR-COUNT.
142*   001420     UNSTRING RECONST-RECORD INTO HOLD WITH POINTER CHAR-COUNT.
143*   001430     IF HOLD IS EQUAL TO "  END" GO TO RE.
144*   001440     PERFORM PARSE-RECORD THRU NEXT-RECORD.
145*   001450     GO TO RE.
146*   001460
147*   001470 V-E. CLOSE USERI-FILE.
148*   001480
```

```
149*  001490      SET ANINDEX UP BY 1.
150*  001500      MOVE "END" TO NAME(ANINDEX).
151*  001510      MOVE "XXX" TO CONC-LINK(ANINDEX).
152*  001520      PERFORM VIRTUAL-ELIM THRU VIRTUAL-END.
153*  001530
154*  001540      PERFORM REPORT-INTERSECT THRU THE-BOTTUM.
155*  001550
156*  001560      OPEN OUTPUT F-D-TABLE.
157*  001570      PERFORM DETERM-IDENTIFY THRU LAST-PROC.
158*  001580
159*  001590
160*  001600
161*  001610      PERFORM PRINT-F-D-TABLE THRU THATS-ALL.
162*  001620
163*  001630      CLOSE F-D-TABLE.
164*  001640      CLOSE F-D-PRINT.
165*  001650      STOP RUN.
166*  001660
167*  001670
168*  001680 PARSE-RECORD.
169*  001690      UNSTRING HOLD INTO STAT-HOLD, LEVEL-HOLD, INTERMED-STRING.
170*  001700      MOVE 0 TO COUNT-1.
171*  001710      INSPECT INTERMED-STRING TALLYING COUNT-1 FOR LEADING BREAK.
172*  001720      IF COUNT-1 IS EQUAL TO 0
173*  001730      MOVE INTERMED-STRING TO ATTRIBUTE-STRING
174*  001740           ELSE ADD 1 TO COUNT-1
175*  001750                MOVE COUNT-1 TO CHAR-COUNT
176*  001760                UNSTRING INTERMED-STRING INTO ATTRIBUTE-STRING
177*  001770                WITH POINTER CHAR-COUNT.
178*  001780      MOVE 1 TO CHAR-COUNT, K-TRACK.
179*  001790      IF LEVEL-HOLD IS EQUAL TO 0 IF LEVEL-HOLD IS EQUAL TO
180*  001800          SET-TRAP MOVE 1 TO SET-TRAP
181*  001810              ELSE DISPLAY NAME(ANINDEX), POSITION 35
182*  001820                   DISPLAY "ERROR--THIS REPORT HAS NO ENTRIES."
183*  001830                           POSITION 23
184*  001840                   DISPLAY "DO YOU WISH IT ELIMINATED-YES/NO."
185*  001850                           POSITION 23
```

```
186*  001860                ACCEPT REPLY. POSITION 40
187*  001870                IF REPLY IS EQUAL TO "Y" OR
188*  001880                      REPLY IS EQUAL TO "YE" OR
189*  001890                           REPLY IS EQUAL TO "YES"
190*  001900                             SET ANINDEX DOWN BY 1.
191*  001910           IF LEVEL-HOLD IS GREATER THAN 0 MOVE 0 TO SET-TRAP.
192*  001920     IF LEVEL-HOLD IS EQUAL TO " " OR ATTRIBUTE-STRING
193*  001930          IS EQUAL TO "END" GO TO NEXT-RECORD.
194*  001940     IF LEVEL-HOLD IS GREATER THAN 0
195*  001950          IF STAT-HOLD IS EQUAL TO "N" OR STAT-HOLD IS EQUAL TO
196*  001960       "S"   SUBTRACT LEVEL-HOLD FROM 10 GIVING THAT-MANY
197*  001970             PERFORM UP-LEVEL THAT-MANY TIMES
198*  001980              MOVE 11 TO CONC-CODE(1)
199*  001990             SUBTRACT THAT-MANY FROM 10 GIVING LEVEL-HOLD.
200*  002000 SEPERATE.
201*  002010
202*  002020     UNSTRING ATTRIBUTE-STRING DELIMITED BY BREAK INTO RESIDUE.
203*  002030     COUNT IN COUNT-1. WITH POINTER K-TRACK.
204*  002040        IF COUNT-1 IS LESS THAN 11 GO TO NEXT-LINE.
205*  002050
206*  002060
207*  002070         DISPLAY "DATA-NAME". LINE 5. POSITION 5. ERASE.
208*  002080         DISPLAY RESIDUE.   LINE 5. POSITION 15.
209*  002090         DISPLAY "EXCEEDS THE ALLOWABLE NUMBER OF CHARACTERS."
210*  002100                 LINE 5. POSITION 35.
211*  002110         DISPLAY "EXCESS CHARACTERS WILL BE   TRUNCATED.", LINE 7,
212*  002120                  POSITION 5.
213*  002130     ACCEPT REPLY.
214*  002140     IF REPLY IS NOT EQUAL TO "ZYJPU" DISPLAY " ", ERASE.
215*  002150 NEXT-LINE.
216*  002160     MOVE 0 TO WORD-COUNT.
217*  002170     UNSTRING ATTRIBUTE-STRING DELIMITED BY ALL BREAK INTO
218*  002180     ATTRIBUTE. COUNT IN COUNT-1. WITH POINTER CHAR-COUNT
219*  002190          TALLYING IN WORD-COUNT.
220*  002200     IF COUNT-1 IS EQUAL TO 0 OR WORD-COUNT IS EQUAL TO 0
221*  002210                  GO TO NEXT-RECORD.
222*  002220     PERFORM BUILD-D-N-TABLE THRU TAIL-END.
```

```
223*  002230      GO TO SEPERATE.
224*  002240-NEXT-RECORD.
225*  002250      EXIT.
226*  002260 UP-LEVEL.
227*  002270      ADD 1 TO CONC-CODE(LEVEL-HOLD).
228*  002280      ADD 1 TO LEVEL-HOLD.
229*  002290 BUILD-D-N-TABLE.
230*  002300      ADD 1 TO NEXT-LOC.
231*  002310      SET ANINDEX TO NEXT-LOC.
232*  002320      MOVE ATTRIBUTE TO NAME(ANINDEX).
233*  002330      MOVE LEVEL-HOLD TO LEVEL-NO(ANINDEX).
234*  002340      MOVE 0 TO DETERM-FLAG(ANINDEX).
235*  002350
236*  002360
237*  002370      IF LEVEL-HOLD IS NOT EQUAL TO 0 GO TO NEXT-STEP.
238*  002380          MOVE 11 TO MAIN-LEVEL.
239*  002390          MOVE 1 TO LEVEL-HOLD.
240*  002400              PERFORM FILL-IT 9 TIMES.
241*  002410      MOVE 0 TO LEVEL-HOLD.
242*  002420
243*  002430          MOVE "NUL" TO CONC-LINK(ANINDEX).
244*  002440          SET ANINDEX TO FIND.
245*  002450
246*  002460          IF NEXT-LOC IS NOT EQUAL TO 1 MOVE NEXT-LOC TO
247*  002470                                        CONC-LINK(ANINDEX).
248*  002480          MOVE NEXT-LOC TO FIND.
249*  002490          SET ANINDEX TO NEXT-LOC.
250*  002500      GO TO TAIL-END.
251*  002510 NEXT-STEP.
252*  002520
253*  002530          MOVE CONC-CODE(LEVEL-HOLD) TO CONC-LINK(ANINDEX).
254*  002540 TAIL-END.
255*  002550      EXIT.
256*  002560*THE END OF BUILD-DATA-NAME-TABLE.
257*  002570 FILL-IT.
258*  002580      MOVE MAIN-LEVEL TO CONC-CODE(LEVEL-HOLD).
259*  002590      ADD 1 TO LEVEL-HOLD.
```

```
260*  002600      ADD 10 TO MAIN-LEVEL.
261*  002610
262*  002620
263*  002630
264*  002640 VIRTUAL-ELIM.
265*  002650      DISPLAY "USER INSTRUCTIONS", LINE 4, POSITION 32, ERASE.
266*  002660      DISPLAY "******************", LINE 5, POSITION 32.
267*  002670      DISPLAY "VIRTUAL DATA ITEM IDENTIFICATION PHASE", LINE 7,
268*  002680               POSITION 21.
269*  002690      DISPLAY "THE USER MUST IDENTIFY ALL VIRTUAL DATA ITEMS WHICH
270*  002700-              "HAVE BEEN", LINE 10, POSITION 10.
271*  002710      DISPLAY "ENTERED AS DATA-NAME-ITEMS.  THIS WILL BE DONE BY EN
272*  002720-              "TERING", LINE 11,  POSITION 10.
273*  002730      DISPLAY "A YES FOR EACH IDENTIFIED VIRTUAL DATA ITEM.  THE SY
274*  002740-              "STEM WILL", LINE 12, POSITION 10.
275*  002750      DISPLAY "DISPLAY ONE POTENTIAL VIRTUAL DATA ITEM AT A TIME.
276*  002760-              "YOU",  LINE 13, POSITION 10.
277*  002770      DISPLAY "MUST ENTER A YES (Y) IF IT IS A VDI AND A NO (N) IF
278*  002780-              "NOT.", LINE 14, POSITION 10.
279*  002790
280*  002800
281*  002810      DISPLAY "DO YOU DESIRE AN EXPLANATION OF A VIRTUAL DATA ITEM?
282*  002820-              "ENTER YES OR NO", LINE 17, POSITION 10.
283*  002830      ACCEPT REPLY, LINE 19, POSITION 5.
284*  002840
285*  002850      IF REPLY IS EQUAL TO "Y" OR
286*  002860           REPLY IS EQUAL TO "YE" OR
287*  002870                REPLY IS EQUAL TO "YES"
288*  002880          DISPLAY "A VIRTUAL DATA ITEM IS A DATA-NAME-ITEM WHICH IS
289*  002890-                 " DERIVABLE", LINE 5, POSITION 5, ERASE
290*  002900          DISPLAY "BY COMPUTATION INVOLVING DATA-NAME-ITEMS FROM WI
291*  002910-                  "THIN THE ", LINE 6, POSITION 5
292*  002920          DISPLAY "SAME DATA BASE. THIS SAYS THAT THE DATA-NAME-ITE
293*  002930-                  "M CAN", LINE 7, POSITION 5
294*  002940          DISPLAY "BE DERIVED FROM OTHER DATA-NAME-ITEMS THAT ARE
295*  002950-                 "WITHIN THE", LINE 8, POSITION 5
296*  002960          DISPLAY "SAME DATA BASE OR POSSIBLY THE SAME REPORT."
```

```
297*  002970                LINE 9, POSITION 5
298*  002980          DISPLAY "AN EXAMPLE OF A VIRTUAL DATA ITEM WOULD BE TOTAL
299*  002990-                 "-HOURS-WORKED", LINE 12, POSITION 5
300*  003000          DISPLAY "WHEN WITHIN THE SAME REPORT THE FOLLOWING DATA-
301*  003010-                 "NAME-ITEMS ARE AVAILABLE:", LINE 13, POSITION 5
302*  003020          DISPLAY "HOURS-WORKED-PER-WEEK", LINE 15, POSITION 10
303*  003030          DISPLAY "WEEKS-WORKED-PER-YEAR", LINE 16, POSITION 10
304*  003040
305*  003050      DISPLAY "IF YOU REQUIRE ADDITIONAL ASSISTANCE ENTER HELP."
306*  003060                LINE 20, POSITION 15
307*  003070      ACCEPT REPLY, LINE 21, POSITION 5.
308*  003080      IF REPLY IS EQUAL TO "H" OR
309*  003090           REPLY IS EQUAL TO "HE" OR
310*  003100                REPLY IS EQUAL TO "HEL" OR
311*  003110                     REPLY IS EQUAL TO "HELP"  DISPLAY
312*  003120                     "PLEASE CALL RICH AT 8-316-687-6339"
313*  003130                        LINE 5, POSITION 10, ERASE
314*  003140          ELSE DISPLAY "WE ARE READY TO START.", LINE 5
315*  003150                                    POSITION 29, ERASE.
316*  003160      ACCEPT REPLY.
317*  003170      IF REPLY IS NOT EQUAL TO "JDTIFH" DISPLAY " ", ERASE.
318*  003180 BEGIN2.
319*  003190      SET ANINDEX TO 1.
320*  003200 STRT.
321*  003210      IF CONC-LINK(ANINDEX) IS EQUAL TO "XXX" GO TO FINISH.
322*  003220      DISPLAY "REPORT TO BE CHECKED FOR VIRTUAL DATA ITEMS IS ",
323*  003230                LINE 5, POSITION 12, ERASE.
324*  003240      DISPLAY NAME(ANINDEX), LINE 5, POSITION 59.
325*  003250      SET ANINDEX UP BY 1.
326*  003260 VIRT-CHECK.
327*  003270      IF LEVEL-NO(ANINDEX) IS EQUAL TO 0
328*  003280      DISPLAY "ERROR-REPORT WITH NO ERTRIES HAS BEEN FOUND"
329*  003290      ACCEPT REPLY
330*  003300      IF REPLY IS NOT EQUAL TO "KDEKFOG" DISPLAY " ", ERASE
331*  003310                GO TO STRT.
332*  003320      DISPLAY NAME(ANINDEX), LINE 7, POSITION 35.
333*  003330           DISPLAY "                    ", LINE 7, POSITION 50.
```

```
334*  003340      IF DETERM-FLAG(ANINDEX) IS EQUAL TO 3 DISPLAY "THIS IS A VDI"
335*  003350                                          , LINE 7, POSITION 50.
336*  003360      DISPLAY "IS THIS DATA-NAME-ITEM A VDI? (YES OR NO)", LINE 9,
337*  003370                                 POSITION 10.
338*  003380      ACCEPT REPLY, LINE 9, POSITION 52.
339*  003390      IF REPLY IS EQUAL TO "Y" OR
340*  003400            REPLY IS EQUAL TO "YE" OR
341*  003410                 REPLY IS EQUAL TO "YES"
342*  003420                 MOVE 3 TO DETERM-FLAG(ANINDEX)
343*  003430                                  ADD 1 TO COWNT
344*  003440                                  DISPLAY " ", LINE 1, ERASE.
345*  003450          SET ANINDEX UP BY 1.
346*  003460
347*  003470          IF LEVEL-NO(ANINDEX) IS EQUAL TO 0 GO TO STRT.
348*  003480          IF CONC-LINK(ANINDEX) IS NOT EQUAL TO "XXX"
349*  003490              GO TO VIRT-CHECK.
350*  003500 FINISH.
351*  003510      MOVE 0 TO KOUNT.
352*  003520
353*  003530      DISPLAY "ALL REPORTS AND THEIR RESPECTIVE DATA-NAME-ITEMS HAV
354*  003540-                "E BEEN", LINE 5, POSITION 10, ERASE.
355*  003550      DISPLAY "DISPLAYED.  DO YOU WISH TO VERIFY THE LIST OF VIRTUA
356*  003560-                    "L DATA ", LINE 6, POSITION 10.
357*  003570      DISPLAY "ITEM, ENTER YES OR NO.", LINE 7, POSITION 10.
358*  003580      ACCEPT REPLY LINE 7, POSITION 35.
359*  003590
360*  003600      IF RFPLY IS EQUAL TO "N" OR
361*  003610            REPLY IS EQUAL TO "NO" GO TO LAST-CHANCE.
362*  003620
363*  003630      SET ANINDEX TO 1.
364*  003640 LOOP.
365*  003650      IF COWNT IS EQUAL TO KOUNT GO TO LAST-CHANCE.
366*  003660      IF CONC-LINK(ANINDEX) IS EQUAL TO "XXX" GO TO LAST-CHANCE.
367*  003670      IF DETERM-FLAG(ANINDEX) IS EQUAL TO THREE3
366*  003680       DISPLAY NAME(ANINDEX), LINE 10 POSITION 35, ERASE
369*  003690
370*  003700      DISPLAY "IF THIS IS A VALID VIRTUAL DATA ITEM ENTER YES OTHER
```

```
371*  003710-                    "WISE NO." LINE 12, POSITION 10
372*  003720      ACCEPT REPLY LINE 14, POSITION 20
373*  003730      IF REPLY IS EQUAL TO "N" OR
374*  003740            REPLY IS EQUAL TO "NO" MOVE 0 TO DETERM-FLAG(ANINDEX)
375*  003750                                  DISPLAY " ", ERASE
376*  003760                                  ELSE DISPLAY " ", ERASE
377*  003770      ADD 1 TO KOUNT
378*  003780      SET ANINDEX UP BY 1
379*  003790      GO TO LOOP.
380*  003800      SET ANINDEX UP BY 1.
381*  0038
382*  003820      GO TO LOOP.
383*  003830
384*  003840 LAST-CHANCE.
385*  003850      DISPLAY "DO YOU WISH TO REVIEW THE LIST OF DATA-NAME-ITEMS
386*  003860-           "OF EACH REPORT", LINE 5, POSITION 10, ERASE.
387*  003870      DISPLAY "AGAIN TO INSURE  THAT ALL VIRTUAL-DATA-ITEMS HAVE
388*  003880-           "BEEN IDENTIFIED.", LINE 6, POSITION 10.
389*  003890      DISPLAY "ENTER YES OR NO", LINE 7, POSITION 10.
390*  003900      ACCEPT REPLY LINE 7, POSITION 30.
391*  003910
392*  003920      IF REPLY IS EQUAL TO "Y" OR
393*  003930            REPLY IS EQUAL TO "YE" OR
394*  003940                  REPLY IS EQUAL TO "YES" GO TO BEGIN2.
395*  003950 VIRTUAL-END.
396*  003960      EXIT.
397*  003970*END OF VIRTUAL-IDENTIFY
398*  003980 REPORT-INTERSECT.
399*  003990      DISPLAY " ", ERASE.
400*  004000      SET ANINDEX TO 1.
401*  004010 LOOP-1.
402*  004020      IF LEVEL-NO(ANINDEX) IS EQUAL TO 0 MOVE CONC-LINK(ANINDEX)
403*  004030          TO  NEXT-RPT.
404*  004040      IF NEXT-RPT IS EQUAL TO "NUL" GO TO THE-BOTTUM.
405*  004050      SET SUB-ANINDEX TO ANINDEX.
406*  004060      MOVE NEXT-RPT TO CONVERT1.
407*  004070      SUBTRACT SUB-ANINDEX , 1 FROM CONVERT1 GIVING THAT-MANY.
```

```
408*  004080      MOVE NEXT-RPT TO CK-NEXT.
409*  004090      ADD 1 TO CK-NEXT.
410*  004100      SET KNT TO ANINDEX.
411*  004110      PERFORM INTERSECT THRU INTERSECT-END THAT-MANY TIMES.
412*  004120      SET ANINDEX TO CONVERT1.
413*  004130      GO TO LOOP-1.
414*  004140 THE-BOTTUM.
415*  004150      EXIT.
416*  004160 INTERSECT.
417*  004170      ADD 1 TO KNT.
418*  004180      SET ANINDEX TO KNT.
419*  004190      IF DETERM-FLAG(ANINDEX) IS EQUAL TO TWO2 GO TO INTERSECT-END.
420*  004200      MOVE NAME(ANINDEX) TO MATCH.
421*  004210
422*  004220      SET ANINDEX TO CK-NEXT.
423*  004230
424*  004240      PERFORM LOOP-2 UNTIL CONC-LINK(ANINDEX) IS EQUAL TO "XXX".
425*  004250 INTERSECT-END.
426*  004260      EXIT.
427*  004270 LOOP-2.
428*  004280      IF MATCH IS EQUAL TO NAME(ANINDEX) AND LEVEL-NO(ANINDEX)
429*  004290          IS NOT EQUAL TO 0 AND DETERM-FLAG(ANINDEX) IS NOT EQUAL
430*  004300              TO THREE3 MOVE TWO2 TO DETERM-FLAG(ANINDEX)
431*  004310                   SET NEXT-LOC TO ANINDEX
432*  004320                   SET ANINDEX TO KNT
433*  004330                   MOVE TWO2 TO DETERM-FLAG(ANINDEX)
434*  004340                   SET ANINDEX TO NEXT-LOC
435*  004350                   SET ANINDEX UP BY 1
436*  004360              ELSE SET ANINDEX UP BY 1.
437*  004370*END LOOP-2.
438*  004380*END INTERSECT.
439*  004390*END REPORT-INTERSECT.
440*  004400
441*  004410 DETERM-IDENTIFY.
442*  004420      DISPLAY "USER REQUIREMENTS", LINE 5, POSITION 31, ERASE.
443*  004430      DISPLAY "*****************", LINE 6, POSITION 31.
444*  004440      DISPLAY "DETERMINANT IDENTIFICATION PHASE", LINE 8, POSITION
```

```
445*  004450      25.
446*  004460      DISPLAY "YOU MUST IDENTIFY FROM EACH GROUP OF DATA-NAME-ITEMS
447*  004470-           " DISPLAYED-THOSE", LINE 10, POSITION 5.
448*  004480      DISPLAY "ITEMS WHICH WILL UNIQUELY IDENTIFY THE REMAINING ITE
449*  004490-             "MS WITHIN", LINE 11, POSITION 5.
450*  004500      DISPLAY "THE GROUP. THOSE DATA-NAME-ITEMS WHICH IDENTIFY THE
451*  004510-             "REMAINING ITEMS", LINE 12, POSITION 5.
452*  004520      DISPLAY "ARE CALLED DETERMINANTS. EACH DETERMINANT IDENTIFIED
453*  004530-           " MUST BE", LINE 13, POSITION 5.
454*  004540      DISPLAY "ENTERED UPON REQUEST. SPELLING IS CRITICAL. ANY QUES
455*  004550-             "TIONS? YES/NO", LINE 14, POSITION 5.
456*  004560      ACCEPT REPLY, LINE 15, POSITION 5.
457*  004570      IF REPLY IS EQUAL TO "N" OR
458*  004580           REPLY IS EQUAL TO "NO" GO TO SKIP-LAST.
459*  004590      DISPLAY "THE FOLLOWING DATA-NAME-ITEMS ARE DISPLAYED*****
460*  004600-     "WAREHOUSE-NO, "  LINE 5, POSITION 10, ERASE.
461*  004610      DISPLAY "SPACE-AVAILABLE, ITEMS-STORED, STORAGE-LEVEL*****"
462*  004620            LINE 6, POSITION 10.
463*  004630      DISPLAY "THE SINGLE DATA ITEM WHICH COULD IDENTIFY THE OTHER
464*  004640-           " ITEMS WOULD", LINE 8, POSITION 10.
465*  004650      DISPLAY " BE WAREHOUSE-NO******IT IS THE DETERMINANT******"
466*  004660            LINE 9, POSITION 10.
467*  004670
468*  004680 SKIP-LAST.
469*  004690      DISPLAY "ENTER GO WHEN YOU ARE READY TO BEGIN.", LINE 16,
470*  004700            POSITION 15.
471*  004710      ACCEPT REPLY, LINE 17 POSITION 15.
472*  004720      IF REPLY IS EQUAL TO "G" OR
473*  004730         REPLY IS EQUAL TO "GO" DISPLAY " ", ERASE.
474*  004740      SET F-D-INDEX TO 1.
475*  004750      PERFORM CLEARFD 250 TIMES.
476*  004760      SET F-D-INDEX TO 0.
477*  004770      SET ANINDEX TO 1.
478*  004780      SET STACK-INDEX TO 1.
479*  004790      PERFORM MAKE-NULL 50 TIMES.
480*  004800      SET CAT-INDEX TO 1.
481*  004810      PERFORM CAT-CLEAR 20 TIMES.
```

```
482*  004820 MAIN-LOOP.
483*  004830     SET CAT-INDEX TO 0.
484*  004840     SET S-INDEX TO 1.
485*  004850     PERFORM SAVCLEAR 20 TIMES.
486*  004860          MOVE 0 TO SAVETOP.
487*  004870
488*  004880     MOVE 1  TO LAST-LEV.
489*  004890     IF CONC-LINK(ANINDEX) IS EQUAL TO "XXX"
490*  004900                    MOVE "END" TO NAME(ANINDEX)
491*  004910                         SET F-D-INDEX UP BY 1
492*  004920                         MOVE "XXXXXXXXXX" TO LIST(F-D-INDEX)
493*  004930                         PERFORM NEXT-PROC
494*  004940                         GO TO LAST-PROC.
495*  004950
496*  004960     MOVE 11 TO MAIN-LEVEL.
497*  004970     MOVE 1 TO LEVEL-HOLD.
498*  004980     PERFORM FILL-IT 9 TIMES.
499*  004990     MOVE 1 TO LEVEL-HOLD.
500*  005000
501*  005010     SET TOPP TO ANINDEX.
502*  005020     IF LEVEL-NO(ANINDEX) IS EQUAL TO 0 MOVE CONC-LINK
503*  005030         (ANINDEX) TO NEXT-RPT.
504*  005040     DISPLAY "GROUPS TO BE DISPLAYED ARE FROM REPORT", LINE 3
505*  005050             POSITION 16, ERASE.
506*  005060     DISPLAY NAME(ANINDEX), LINE 3, POSITION 55.
507*  005070 SECOND-LOOP.
508*  005080     MOVE 0 TO KOUNT, DOUBLE-CK, SET-FLAG, SET-CHECK.
509*  005090     MOVE 7 TO LINE-NO.
510*  005100 SECOND-PART.
511*  005110
512*  005120      MOVE CONC-CODE(LEVEL-HOLD) TO KEY1.
513*  005130     SET STACK-INDEX TO 0.
514*  005140 THIRD-LOOP.
515*  005150     ADD 1 TO LINE-NO.
516*  005160 THIRD-LOOP1.
517*  005170     SET ANINDEX UP BY 1.
518*  005180*LOOKING FOR END OF TABLE
```

```
519*  005190      SET KNT TO ANINDEX.
520*  005200      IF KNT IS EQUAL TO NEXT-RPT OR
521*  005210              LEVEL-NO(ANINDEX) IS EQUAL TO 0 OR
522*  005220                   CONC-LINK(ANINDEX) IS EQUAL TO "XXX"
523*  005230                        GO TO TEST-GROUP.
524*  005240*GROUP EXIT
525*  005250      IF CONC-LINK(ANINDEX) IS NOT EQUAL TO KEY1 GO TO THIRD-LOOP1.
526*  005260*SKIP VIRTUAL DATA ITEM
527*  005270      IF DETERM-FLAG(ANINDEX) IS EQUAL TO THREE3 GO TO THIRD-LOOP1.
528*  005280           MOVE 0 TO DOUBLE-CK, SET-CHECK.
529*  005290                MOVE 1 TO SET-FLAG.
530*  005300           DISPLAY NAME(ANINDEX), LINE LINE-NO, POSITION 5.
531*  005310*LOAD STACK
532*  005320           SET STACK-INDEX UP BY 1.
533*  005330           MOVE NAME(ANINDEX) TO ATTR(STACK-INDEX).
534*  005340*CHECK FOR INTERSECTED DATA-NAME-ITEMS
535*  005350      IF DETERM-FLAG(ANINDEX) IS EQUAL TO TWO2 DISPLAY "*".
536*  005360           LINE LINE-NO, POSITION 16.
537*  005370                ADD 1 TO KOUNT.
538*  005380      GO TO THIRD-LOOP.
539*  005390 TEST-GROUP.
540*  005400      MOVE 0 TO L-S-COUNT.
541*  005410      IF SET-FLAG IS EQUAL TO 1 GO TO NEXT-GROUP.
542*  005420           ADD 1 TO LEVEL-HOLD.
543*  005430                IF LEVEL-HOLD IS GREATER THAN 8 GO TO SET-GROUP.
544*  005440      SET ANINDEX TO TOPP.
545*  005450      MOVE 0 TO SET-FLAG.
546*  005460      GO TO SECOND-PART.
547*  005470 SET-GROUP.
548*  005480      ADD 1 TO SET-CHECK.
549*  005490      MOVE 2 TO LEVEL-HOLD.
550*  005500      MOVE 8 TO THAT-MANY.
551*  005510      PERFORM UP-LEVEL THAT-MANY TIMES.
552*  005520      MOVE 2 TO LEVEL-HOLD.
553*  005530      IF SET-CHECK IS EQUAL TO 4 GO TO MAIN-LOOP.
554*  005540      SET ANINDEX TO TOPP.
555*  005550      GO TO SECOND-PART.
```

```
556*   005560
557*   005570*BUILD WORDING FOR DISPLAY TO FOLLOW.
558*   005580 NEXT-GROUP.
559*   005590     IF TOPP IS GREATER THAN 1 GO TO GET-NAME.
560*   005600     SET ANINDEX TO TOPP.
561*   005610     IF KOUNT IS GREATER THAN 1 MOVE 3 TO KOUNT.
562*   005620     PERFORM FIX-WORD1 UNTIL KOUNT NOT EQUAL TO 0.
563*   005630     PERFORM FIX-WORD2 UNTIL KOUNT NOT EQUAL TO 1.
564*   005640     PERFORM FIX-WORD3 UNTIL KOUNT NOT EQUAL TO 3.
565*   005650     GO TO EXPLAN1.
566*   005660 FIX-WORD1.
567*   005670     MOVE "NONE" TO WORD.
568*   005680     MOVE "NO" TO ARTICAL.
569*   005690     MOVE "DETERMINANTS" TO ONE-OR-MORE.
570*   005700     MOVE 99 TO KOUNT.
571*   005710 FIX-WORD2.
572*   005720     MOVE "A" TO ARTICAL.
573*   005730     MOVE "A DETERMINANT" TO ONE-OR-MORE.
574*   005740     MOVE "ONE" TO WORD.
575*   005750     MOVE "IS" TO VERB.
576*   005760     MOVE 100 TO KOUNT.
577*   005770 FIX-WORD3.
578*   005780     MOVE "A" TO ARTICAL.
579*   005790     MOVE "ARE" TO VERB.
580*   005800     MOVE "SEVERAL" TO WORD.
581*   005810     MOVE "DETERMINANT" TO ONE-OR-MORE.
582*   005820     MOVE 100 TO KOUNT.
583*   005830 EXPLAN1.
584*   00
585*   005850
586*   005860     DISPLAY WORD, LINE 5, POSITION 20.
587*   005870     DISPLAY "OF THE LISTED DATA-NAME-ITEMS MATCHED OTHER DATA-"
588*   005880                 LINE 5, POSITION 28.
589*   005890     DISPLAY "NAME-ITEMS IN OTHER REPORTS. THIS MEANS THAT YOU
590*   005900-                "HAVE", LINE 6, POSITION 20.
591*   005910     DISPLAY ARTICAL, LINE 7, POSITION 20.
592*   005920     DISPLAY "CLUE AS TO WHICH DATA-NAME-ITEM/S MIGHT BE", LINE 7
```

```
593*  005930                POSITION 23.
594*  005940       DISPLAY ONE-OR-MORE, LINE 8, POSITION 20.
595*  005950*CHECK THE NUMBER OF DATA-NAME-ITEMS MARKED.
596*  005960
597*  005970       IF KOUNT IS EQUAL TO 100 GO TO MORE-THAN-ONE.
598*  005980       DISPLAY "PLEASE ENTER THE DATA-NAME-ITEM WHICH MEETS THE"
599*  005990                LINE 10, POSITION 20.
600*  006000       DISPLAY "REQUITEMENTS OF A DETERMINANT FOR THIS GROUP. THERE"
601*  006010                LINE 11, POSITION 20.
602*  006020       DISPLAY "CAN BE MORE THAN ONE DETERMINANT PER GROUP. FAILURE"
603*  006030                LINE 12, POSITION 20.
604*  006040       DISPLAY "TO IDENTIFY AT LEAST ONE DETERMINANT WILL CAUSE AN"
605*  006050                LINE 13, POSITION 20.
606*  006060       DISPLAY "ERROR CONDITION. THE SYSTEM WILL REQUEST THAT YOU"
607*  006070                LINE 14, POSITION 20.
608*  006080       DISPLAY "ENTER EACH SELECTED DETERMINANT NAME ONE AT A TIME"
609*  006090                LINE 15, POSITION 20.
610*  006100       DISPLAY "WHEN INDICATED. IF ALL DETERMINANT NAMES HAVE BEEN"
611*  006110                LINE 16, POSITION 20.
612*  006120       DISPLAY "ENTERED, ENTER XX WHEN THE NEXT REQUEST APPEARS."
613*  006130                LINE 17, POSITION 20.
614*  006140       DISPLAY "EACH NAME ENTERED MUST BE SPELLED EXACTLY AS IT"
615*  006150                LINE 18, POSITION 20.
616*  006160       DISPLAY "APPEARS ON THE SCREEN.", LINE 19, POSITION 20.
617*  006170*MORE THAN ONE DATA-NAME-ITEM WAS UNDERLINED.
618*  006180 MORE-THAN-ONE.
619*  006190*A CHECK TO INSURE THAT THE CORRECT MESSAGE WAS DISPLAYED
620*  006200       IF KOUNT IS EQUAL TO 99 GO TO GET-NAME.
621*  006210       DISPLAY "THE DATA-NAME-ITEM/S WHICH ALSO APPEAR IN OTHER REPO
622*  006220-                  "RTS", LINE 10, POSITION 20.
623*  006230       DISPLAY VERB, LINE 11, POSITION 20.
624*  006240       DISPLAY "MARKED BY *. WHEN CONSIDERING WHICH DATA-NAME-ITEMS"
625*  006250                  LINE 11, POSITION 20.
626*  006260       DISPLAY "IDENTIFY THE REMAINING ITEMS IN THE GROUP--FIRST"
627*  006270                  LINE 12, POSITION 20.
628*  006280       DISPLAY "CONSIDER THE MARKED ITEMS. THERE CAN BE MORE"
629*  006290                  LINE 13, POSITION 20.
```

```
630*  006300      DISPLAY "THAN ONE DETERMINANT PER GROUP. FAILURE TO "
631*  006310                  LINE 14, POSITION 20.
632*  006320      DISPLAY "IDENTIFY AT LEAST ONE DETERMINANT WILL CAUSE AN"
633*  006330                  LINE 15, POSITION 20.
634*  006340      DISPLAY "ERROR CONDITION. THE SYSTEM WILL REQUEST THAT YOU"
635*  006350                  LINE 16, POSITION 20.
636*  006360      DISPLAY "ENTER EACH SELECTED DETERMINANT NAME ONE AT A TIME"
637*  006370                  LINE 17, POSITION 20.
638*  006380      DISPLAY "WHEN INDICATED. IF ALL DETERMINANT NAMES HAVE BEEN"
639*  006390                  LINE 18, POSITION 20.
640*  006400      DISPLAY "ENTERED, ENTER XX WHEN THE NEXT REQUEST APPEARS."
641*  006410                  LINE 19, POSITION 20.
642*  006420      DISPLAY "EACH NAME ENTERED MUST BE SPELLED EXACTLY AS IT"
643*  006430                  LINE 20, POSITION 20.
644*  006440      DISPLAY "APPEARS ON THE SCREEN.", LINE 21, POSITION 20.
645*  006450*ACCEPT DETERMINANT NAMES FROM CRT
646*  006460
647*  006470 GET-NAME.
648*  006480      DISPLAY "ENTER THE DETERMINANT NAME.", POSITION 20.
649*  006490      ACCEPT REPLY, POSITION 48.
650*  006500      IF REPLY IS EQUAL TO "X" OR
651*  006510            REPLY IS EQUAL TO "XX" OR
652*  006520                 REPLY IS EQUAL TO "XXX" DISPLAY " ", ERASE
653*  006530                      GO TO OUT-OF-NAME.
654*  006540           SET TOP-STACK TO STACK-INDEX.
655*  006550           MOVE 1 TO LOOK.
656*  006560           SET STACK-INDEX TO 1.
657*  006570           PERFORM CHECK-NAME TOP-STACK TIMES.
658*  006580           SET STACK-INDEX TO TOP-STACK.
659*  006590           IF LOOK IS NOT EQUAL TO 7
660*  006600                DISPLAY "YOU SPELLED IT WRONG**TRY AGAIN**PLEASE."
661*  006610                     , POSITION 20
662*  006620                DISPLAY "PRESS NEW LINE", POSITION 20
663*  006630                ACCEPT REPLY
664*  006640                IF REPLY IS NOT EQUAL TO "KSHTKT" DISPLAY " "
665*  006650                GO TO GET-NAME.
666*  006660      PERFORM CHECK-DNAME L-S-COUNT TIMES.
```

```
567*  006670      SET STACK-INDEX TO TOP-STACK.
568*  006680      IF LOOK IS EQUAL TO 8
669*  006690         DISPLAY "DUPLICATE DETERMINANT ENTRY--ENTER A DETERMINANT"
670*  006700              POSITION 20
671*  006710      DISPLAY "NAME ONLY ONCE PER UNIQUE GROUP. PRESS NEW LINE"
672*  006720            POSITION 20
673*  006730          ACCEPT REPLY
674*  006740           IF REPLY IS NOT EQUAL TO "KDFOEJ" DISPLAY " "
675*  006750            GO TO GET-NAME.
676*  006760*LOAD HOLDING STACK
677*  006770      SET STACK-INDEX UP BY 1.
678*  006780          MOVE REPLY TO ATTR(STACK-INDEX).
679*  006790          ADD 1 TO L-S-COUNT.
680*  006800      SUBTRACT TOPP    FROM KNT GIVING NUMBER5.
681*  006810      SET ANINDEX TO TOPP.
682*  006820      PERFORM SET-KEY NUMBER5 TIMES.
683*  006830          SET ANINDEX TO TOPP.
684*  006840          GO TO GET-NAME.
685*  006850*NO MORE DETERMINANTS IDENTIFIED IN THAT GROUP.
686*  006860 OUT-OF-NAME.
687*  006870      SET NEXT-LOC TO ANINDEX.
688*  006880      SET ANINDEX TO TOPP.
689*  006890      IF NEXT-LOC IS NOT EQUAL TO TOPP OR L-S-COUNT IS EQUAL TO 0
690*  006900          DISPLAY "A DETERMINANT HAS NOT BEEN IDENTIFIED FOR THE
691*  006910-                                  "LAST GROUP--GROUP IS DESCARDED.",
692*  006920                              LINE 5, POSITION 5
693*  006930                    ACCEPT REPLY
694*  006940                    IF REPLY IS NOT EQUAL TO "WLDHUK"
695*  006950                       DISPLAY " ", ERASE
696*  006960      ADD 1 TO LEVEL-HOLD
697*  006970           GO TO SECOND-LOOP.
696*  006980
699*  006990      ADD 1 TO LEVEL-HOLD.
700*  007000*CONCATENATE GROUP DETERMINANTS WHICH BELONG AND BUILD THE
701*  007010*FUNCTIONAL DEPENDENCY TABLE.
702*  007020      SET TOP-STACK TO STACK-INDEX.
703*  007030
```

```
704*  007040      PERFORM LOAD-CAT-STACK.
705*  007050
706*  007060                SET STACK-INDEX TO TOP-STACK.
707*  007070               PERFORM BUILD-F-D-TABLE.
706*  007080      GO TO SECOND-LOOP.
709*  007090 NEXT-PROC.
710*  007100      SET F-D-INDEX TO 1.
711*  007110      PERFORM LOAD-TABLE UNTIL LIST(F-D-INDEX) IS EQUAL TO
712*  007120          "XXXXXXXXXX".
713*  007130 LAST-PROC.
714*  007140      EXIT.
715*  007150 CHECK-DNAME.
716*  007160      IF REPLY IS EQUAL TO ATTR(STACK-INDEX) MOVE 8 TO LOOK.
717*  007170      SET STACK-INDEX DOWN BY 1.
718*  007180 CHECK-NAME.
719*  007190       IF REPLY IS EQUAL TO ATTR(STACK-INDEX) MOVE 7 TO LOOK.
720*  007200       SET STACK-INDEX UP BY 1.
721*  007210 LOAD-TABLE.
722*  007220      WRITE F-D-TABLE-ELE FROM LIST(F-D-INDEX).
723*  007230      SET F-D-INDEX UP BY 1.
724*  007240*THE END OF DETERM-IDENTIFY.
725*  007250 MAKE-NULL.
726*  007260      MOVE SPACES TO ATTR(STACK-INDEX).
727*  007270      SET STACK-INDEX UP BY 1.
728*  007280 CLEARFD.
729*  007290      MOVE SPACES TO LIST(F-D-INDEX).
730*  007300      SET F-D-INDEX UP BY 1.
731*  007310 SAVCLEAR.
732*  007320      MOVE SPACES TO S-NAME(S-INDEX).
733*  007330      MOVE 00 TO SLEV(S-INDEX).
734*  007340      SET S-INDEX UP BY 1.
735*  007350 CAT-CLEAR.
736*  007360      MOVE SPACES TO D-NAME(CAT-INDEX).
737*  007370      MOVE 00 TO LEV(CAT-INDEX).
738*  007380      SET CAT-INDEX UP BY 1.
739*  007390 LOAD-CAT-STACK.
740*  007400      UNSTRING KEY1 INTO FIRSTN1, FIRSTN2.
```

```
741*  007410      MOVE FIRSTN1 TO KEY2.
742*  007420      IF KEY2 IS GREATER THAN LAST-LEV
743*  007430              PERFORM PUSH-STACK L-S-COUNT TIMES.
744*  007440      IF KEY2 IS EQUAL TO 1
745*  007450              PERFORM PUSH-STACK L-S-COUNT TIMES.
746*  007460      IF KEY2 IS EQUAL TO LAST-LEV AND LAST-LEV IS NOT EQUAL TO 1
747*  007470         PERFORM POP-STACK THRU ENDBEGIN3.
748*  007480      IF KEY2 IS EQUAL TO LAST-LEV AND LAST-LEV IS NOT EQUAL TO 1
749*  007490         PERFORM PUSH-STACK L-S-COUNT TIMES.
750*  007500      IF KEY2 IS LESS THAN LAST-LEV
751*  007510              PERFORM POP-STACK THRU ENDBEGIN3.
752*  007520      IF KEY2 IS LESS THAN LAST-LEV
753*  007530              PERFORM PUSH-STACK L-S-COUNT TIMES.
754*  007540      MOVE KEY2 TO LAST-LEV.
755*  007550
756*  007560
757*  007570 PUSH-STACK.
758*  007580      SET CAT-INDEX UP BY 1.
759*  007590      MOVE ATTR(STACK-INDEX) TO D-NAME(CAT-INDEX).
760*  007600      MOVE KEY1 TO LEV(CAT-INDEX).
761*  007610         IF CAT-INDEX IS GREATER THAN 1 SET NUMBER4 TO CAT-INDEX
762*  007620             SUBTRACT 1 FROM NUMBER4
763*  007630             MOVE D-NAME(CAT-INDEX) TO MATCH
764*  007640             MOVE 1 TO LOOK
765*  007650             SET CAT-INDEX TO 1
766*  007660                 PERFORM CAT-UNION NUMBER4 TIMES
767*  007670                 IF LOOK IS EQUAL TO 5
768*  007680                     MOVE SPACES TO D-NAME(CAT-INDEX)
769*  007690                        MOVE 00 TO LEV(CAT-INDEX)
770*  007700                        MOVE 1 TO LOOK
771*  007710                        SET CAT-INDEX DOWN BY 1.
772*  007720      SET STACK-INDEX DOWN BY 1.
773*  007730 POP-STACK.
774*  007740 BEGIN3.
775*  007750      IF CAT-INDEX IS LESS THAN 1 GO TO ENDBEGIN3.
776*  007760      MOVE LEV(CAT-INDEX) TO KEY3.
777*  007770      UNSTRING KEY3 INTO KEY4, KEY5.
```

```
778*  007780      MOVE KEY4 TO HOLD-LEVEL.
779*  007790      IF HOLD-LEVEL IS NOT LESS THAN KEY2
780*  007800          MOVE SPACES TO D-NAME(CAT-INDEX)
781*  007810          MOVE 00 TO LEV(CAT-INDEX)
782*  007820              SET CAT-INDEX DOWN BY 1
783*  007830                  GO TO BEGIN3.
784*  007840
785*  007850 ENDBEGIN3.
786*  007860      EXIT.
787*  007870
788*  007880 CAT-UNION.
789*  007890      IF MATCH IS EQUAL TO D-NAME(CAT-INDEX) MOVE 5 TO LOOK.
790*  007900      SET CAT-INDEX UP BY 1.
791*  007910*SET THE DETERMINANT FLAG TO YES IF IDENTIFIED
792*  007920 SET-KEY.
793*  007930      SET ANINDEX UP BY 1.
794*  007940      IF DETERM-FLAG(ANINDEX) IS NOT EQUAL TO 3 AND
795*  007950          NAME(ANINDEX) IS EQUAL TO ATTR(STACK-INDEX) MOVE 1 TO
796*  007960              DETERM-FLAG(ANINDEX).
797*  007970 BUILD-F-D-TABLE.
798*  007980      MOVE "0" TO STAT-HOLD.
799*  007990      SET NUMBER4 TO CAT-INDEX.
800*  008000      SET NUMBER3 TO STACK-INDEX.
801*  008010      SUBTRACT L-S-COUNT FROM NUMBER3.
802*  008020          PERFORM DE-LETE THRU LETE-DE.
803*  008030      SUBTRACT L-S-COUNT FROM NUMBER3 GIVING HOLD-NO.
804*  008040          IF NUMBER4 IS GREATER THAN 1 AND
805*  008050              HOLD-NO IS NOT GREATER THAN 0 PERFORM SPEC-CASE.
806*  008060      IF LAST-LEV IS GREATER THAN 1 PERFORM SPEC-CASE2 THRU
807*  008070      CASE2END.
808*  008080              MOVE LAST-LEV TO SAVE-LEVEL.
809*  008090      SET F-D-INDEX UP BY 1.
810*  008100          MOVE NUMBER4 TO MOVERT.
811*  008110          MOVE MOVERT TO LIST(F-D-INDEX).
812*  008120              SET CAT-INDEX TO 1.
813*  008130              PERFORM LOAD1 NUMBER4 TIMES.
814*  008140                  SET CAT-INDEX TO NUMBER4.
```

```
815*  008150                    SET F-D-INDEX UP BY 1.
816*  008160
817*  008170       IF HOLD-NO IS NOT GREATER THAN 0 MOVE 1 TO HOLD-NO
816*  008180                                  ADD 1 TO HOW-MANY
819*  008190                                  MOVE HOW-MANY TO NUM-PART
820*  008200                          STRING COLON, NUM-PART DELIMITED BY SIZE
821*  008210                              INTO SPEC-SY
822*  008220                              MOVE HOLD-NO TO MOVERT
823*  008230                              MOVE MOVERT  TO LIST(F-D-INDEX)
824*  008240                              SET F-D-INDEX UP BY 1
825*  008250                              MOVE SPEC-SY TO LIST(F-D-INDEX)
826*  008260                              MOVE 0 TO HOLD-NO
827*  008270                              SET FIND TO F-D-INDEX
828*  008280                     ELSE     MOVE HOLD-NO TO MOVERT
829*  008290                           MOVE MOVERT TO LIST(F-D-INDEX)
830*  008300                           SET TOP-STACK TO STACK-INDEX
831*  008310                           SET STACK-INDEX TO 1
832*  008320                               PERFORM LOAD2 TOP-STACK TIMES
833*  008330                               SET FIND TO F-D-INDEX.
834*  008340
835*  008350* SPECIAL CASE OPTION  SELF IDENTIFING DETERMINANT CHANGE ROUTINE
836*  008360 SPEC-CASE.
837*  008370       DISPLAY "SPECIAL SITUATION", LINE 5, POSITION 31, ERASE.
838*  008380       DISPLAY "THE FOLLOWING DATA-NAME-ITEMS HAVE BEEN", LINE 7,
839*  008390            POSITION  20.
840*  008400       DISPLAY "CONCATINATED TO FORM A DETERMINANT.  THE", LINE 8,
841*  008410            POSITION 20.
842*  008420       DISPLAY "DETERMINANT AS STRUCTURED IS SELF-IDENTIFING.", LINE
843*  008430            9, POSITION 20.
844*  008440            MOVE 12 TO LINE-NO.
845*  008450            MOVE 0 TO DET-COUNT.
846*  008460            SET CAT-INDEX TO 1.
847*  008470                 PERFORM DISP-DET NUMBER4 TIMES.
848*  008480                 SET CAT-INDEX TO NUMBER4.
849*  008490                 ADD 2 TO LINE-NO.
850*  008500       DISPLAY "IF THE DATA-NAME-ITEMS MARKED * UNIQUELY IDENTIFIES"
851*  008510            LINE LINE-NO, POSITION 20.
```

```
852*  008520          ADD 1 TO LINE-NO.
853*  008530      DISPLAY "ANY OF THE OTHER LISTED ITEMS **ENTER YES** IF NO--"
854*  008540          LINE LINE-NO, POSITION 20.
855*  008550          ADD 1 TO LINE-NO.
856*  008560      DISPLAY "THE RELATION IS ASSUMED CORRECT--PRESS NEWLINE.",
857*  008570          LINE LINE-NO, POSITION 20.
858*  008580      ACCEPT REPLY, POSITION 20.
859*  008590      IF REPLY IS EQUAL TO "Y" OR
860*  008600          REPLY IS EQUAL TO "YE" OR
861*  008610              REPLY IS EQUAL TO "YES" SET STACK-INDEX TO 0
862*  008620                  PERFORM NEXT-Q
863*  008630                      DISPLAY " ", ERASE.
864*  008640
865*  008650
866*  008660* EDIT OPTION TO ALLOW USER TO MOVE DETERMINANTS FROM LEFT
867*  008670* SIDE OF A RELATION TO THE RIGHT SIDE AS A DATA-NAME-ITEM.
868*  008680 SPEC-CASE2.
869*  008690      IF SAVE-LEVEL IS EQUAL TO LAST-LEV AND LAST-LEV IS NOT
870*  008700      EQUAL TO 1 MOVE "S" TO STAT-HOLD.
871*  008710      IF STAT-HOLD IS EQUAL TO "S" PERFORM SAV-MOVE THRU S-MEND
872*  008720      SET CAT-INDEX TO 0
873*  008730      MOVE 0 TO KOUNTY
874*  008740      MOVE NUMBER4 TO KNTTY
875*  008750      PERFORM NEW-UNION-MAIN NUMBER4 TIMES
876*  008760      MOVE KNTTY TO NUMBER4
877*  008770      SET CAT-INDEX TO 1.
878*  008780      IF NUMBER4 IS LESS THAN 2 SET S-INDEX TO 1
879*  008790                               SET CAT-INDEX TO 1
880*  008800                               PERFORM SAV-LAST
881*  008810                               SET S-INDEX TO 1
882*  008820                               SET CAT-INDEX TO 1
883*  008830                               MOVE 1 TO SAVETOP
884*  008840                               GO TO CASE2END.
885*  008850      MOVE 12 TO LINE-NO.
886*  008860      MOVE 0 TO DET-COUNT.
887*  008870      SET CAT-INDEX TO 1.
888*  008880      DISPLAY "ARE ANY OF THE FOLLOWING DETERMINANTS UNIQUELY"
```

```
869*  008890           LINE 8, POSITION 20,
890*  008900              "IDENTIFIED BY THE DETERMINANTS MARKED WITH THE"
891*  008910           LINE 9, POSITION 20,
892*  008920              "*.  ENTER YES ** OTHERWISE PRESS NEWLINE."
893*  008930           LINE 10, POSITION 20.
894*  008940      PERFORM DISP-DET NUMBER4 TIMES.
895*  008950      SET CAT-INDEX TO 1.
896*  008960      ADD 2 TO LINE-NO.
897*  008970      ACCEPT REPLY, LINE LINE-NO, POSITION 20.
898*  008980      IF REPLY IS EQUAL TO "Y" OR
899*  008990           REPLY IS EQUAL TO "YE" OR
900*  009000                REPLY IS EQUAL TO "YES"
901*  009010                      SET S-INDEX TO 1
902*  009020                      PERFORM SAV-LAST NUMBER4 TIMES
903*  009030                      SET S-INDEX DOWN BY 1
904*  009040                      SET SAVETOP TO S-INDEX
905*  009050                     PERFORM NEXT-QUES
906*  009060      ELSE
907*  009070      SET S-INDEX TO 1
908*  009080      PERFORM SAV-LAST NUMBER4 TIMES
909*  009090      SET S-INDEX DOWN BY 1
910*  009100      SET CAT-INDEX DOWN BY 1
911*  009110      SET SAVETOP TO S-INDEX.
912*  009120      DISPLAY " ", ERASE.
913*  009130 CASEZEND.
914*  009140      EXIT.
915*  009150
916*  009160
917*  009170 PACK-IT-D.
918*  009180      MOVE 1 TO KNTT.
919*  009190      ADD 1 TO NUMBER4.
920*  009200      PERFORM MOVE-DFTER L-S-COUNT TIMES.
921*  009210      MOVE KEY2 TO LAST-LEV.
922*  009220      SET NUMBER4 TO CAT-INDEX.
923*  009230      SUBTRACT L-S-COUNT FROM DET-COUNT.
924*  009240      ADD DET-COUNT TO HOLD-NO.
925*  009250
```

```
926*  009260
927*  009270 PACK-IT-DOWN.
928*  009280           MOVE 1 TO KNTT.
929*  009290           ADD 1 TO NUMBER4.
930*  009300           PERFORM MOVE-DET2 L-S-COUNT TIMES.
931*  009310           MOVE 1 TO LAST-LEV.
932*  009320           SET NUMBER4 TO CAT-INDEX.
933*  009330           SUBTRACT L-S-COUNT FROM DET-COUNT.
934*  009340           ADD DET-COUNT TO HOLD-NO.
935*  009350* DISPLAY THE CONCATINATED DETERMINANT.
936*  009360 DISP-DET.
937*  009370      DISPLAY D-NAME(CAT-INDEX), LINE LINE-NO, POSITION 35.
938*  009380      IF LEV(CAT-INDEX) IS EQUAL TO KEY1
939*  009390           DISPLAY "*", LINE LINE-NO, POSITION 33.
940*  009400      SET CAT-INDEX UP BY 1.
941*  009410      ADD 1 TO LINE-NO.
942*  009420      ADD 1 TO DET-COUNT.
943*  009430* BUILD THE NEW RELATION FROM THE CONCATINATED DETERMINANT.
944*  009440 B-NEW-RELATION.
945*  009450      SET CAT-INDEX TO 1.
946*  009460      SUBTRACT L-S-COUNT FROM NUMBER4.
947*  009470           PERFORM L-TO-R NUMBER4 TIMES.
948*  009480 B-N-RDONE.
949*  009490      EXIT.
950*  009500 L-TO-R.
951*  009510      SET STACK-INDEX UP BY 1.
952*  009520      MOVE D-NAME(CAT-INDEX) TO ATTR(STACK-INDEX).
953*  009530      SET CAT-INDEX UP BY 1.
954*  009540*LOAD THE LEFT SIDE DATA INTO THE FDTABLE.
955*  009550 LOAD1.
956*  009560      SET F-D-INDEX UP BY 1.
957*  009570      MOVE D-NAME(CAT-INDEX) TO LIST(F-D-INDEX).
958*  009580          SET CAT-INDEX UP BY 1.
959*  009590*DELETE IDENTICAL NAME ITEMS FROM THE STACK LS ITEMS IN RT SIDE.
960*  009600 DE-LETE.
961*  009610      PERFORM REMOVE L-S-COUNT TIMES.
962*  009620 LETE-DE.
```

```
963*  009630      EXIT.
964*  009640
965*  009650*FILL THE STACK WITH BLANKS.
966*  009660 REMOVE.
967*  009670      SET TOP-STACK TO STACK-INDEX.
968*  009680          MOVE ATTR(STACK-INDEX) TO THIS-ATTR.
969*  009690              SET STACK-INDEX TO NUMBER3.
970*  009700              PERFORM CHECK-LIST NUMBER3 TIMES.
971*  009710          SET STACK-INDEX TO TOP-STACK.
972*  009720          MOVE SPACES TO ATTR(STACK-INDEX).
973*  009730          SET STACK-INDEX DOWN BY 1.
974*  009740 CHECK-LIST.
975*  009750      IF ATTR(STACK-INDEX) IS EQUAL TO THIS-ATTR
976*  009760              MOVE SPACES TO ATTR(STACK-INDEX).
977*  009770                SET STACK-INDEX DOWN BY 1.
978*  009780
979*  009790
980*  009800 SAV-MOVE.
981*  009810      SET S-INDEX TO 1.
982*  009820      IF SLEV(S-INDEX) IS NOT EQUAL TO 00
983*  009830           PERFORM TRANSFER-DET SAVETOP TIMES.
984*  009840      SET S-INDEX TO 1.
985*  009850      PERFORM SAVCLEAR  SAVETOP  TIMES.
986*  009860      MOVE 0 TO SAVETOP.
987*  009870      SET S-INDEX TO 1.
988*  009880      SET CAT-INDEX TO NUMBER4.
989*  009890 S-MEND.
990*  009900      EXIT.
991*  009910
992*  009920
993*  009930 TRANSFER-DET.
994*  009940      SET CAT-INDEX TO NUMBER4.
995*  009950      MOVE SLEV(S-INDEX) TO KEY3.
996*  009960      UNSTRING KEY3 INTO KEY4, KEY5.
997*  009970      MOVE KEY4 TO HOL-LEV.
998*  009980      IF HOL-LEV    IS LESS THAN KEY2 PERFORM MAKE-ROOM NUMBER4
999*  009990              TIMES
```

```
1000*  010000      SET CAT-INDEX UP BY 1
1001*  010010      ADD 1 TO NUMBER4
1002*  010020      MOVE S-NAME(S-INDEX) TO D-NAME(CAT-INDEX)
1003*  010030      MOVE SLEV(S-INDEX) TO LEV(CAT-INDEX)
1004*  010040      MOVE SPACES TO S-NAME(S-INDEX)
1005*  010050      MOVE 00 TO SLEV(S-INDEX)
1006*  010060      SET S-INDEX UP BY 1
1007*  010070      ELSE SET S-INDEX UP BY 1.
1008*  010080
1009*  010090 MAKE-ROOM.
1010*  010100      MOVE D-NAME(CAT-INDEX) TO ATTRIBUTE.
1011*  010110      MOVE LEV(CAT-INDEX) TO LEV-HOLD.
1012*  010120      SET CAT-INDEX UP BY 1.
1013*  010130      MOVE ATTRIBUTE TO D-NAME(CAT-INDEX).
1014*  010140      MOVE LEV-HOLD TO LEV(CAT-INDEX).
1015*  010150      SET CAT-INDEX DOWN BY 2.
1016*  010160
1017*  010170 NEXT-QUES.
1018*  010180      DISPLAY "ARE ALL UNMARKED DETERMINANTS IDENTIFIED BY"
1019*  010190           POSITION 20,
1020*  010200              "THE MARKED DETERMINANTS?", POSITION 20,
1021*  010210              "ENTER YES  ** OTHERWISE PRESS NEWLINE."
1022*  010220           POSITION 20.
1023*  010230      ACCEPT REPLY, POSITION 20.
1024*  010240      IF REPLY IS EQUAL TO "Y" OR
1025*  010250           REPLY IS EQUAL TO "YE" OR
1026*  010260                REPLY IS EQUAL TO "YES"
1027*  010270                     DISPLAY " ", ERASE
1028*  010280                     PERFORM B-NEW-RELATION THRU B-N-RDONE
1029*  010290                     PERFORM PACK-IT-D
1030*  010300      ELSE PERFORM SEL-ITEMS THRU SEL-END
1031*  010310           DISPLAY " ", ERASE.
1032*  010320
1033*  010330
1034*  010340 SAV-LAST.
1035*  010350      MOVE D-NAME(CAT-INDEX) TO S-NAME(S-INDEX).
1036*  010360      MOVE LEV(CAT-INDEX) TO SLEV(S-INDEX).
```

```
1037*  010370      SET CAT-INDEX UP BY 1.
1038*  010380      SET S-INDEX UP BY 1.
1039*  010390
1040*  010400
1041*  010410
1042*  010420 NEXT-Q.
1043*  010430      DISPLAY "ARE ALL UNMARKED DATA-NAME-ITEMS IDENTIFIED"
1044*  010440      POSITION  20.
1045*  010450              "BY THE MARKED DATA-NAME-ITEMS?", POSITION 20,
1046*  010460              "ENTER YES ** OTHERWISE PRESS NEWLINE.", POSITION 20.
1047*  010470      ACCEPT REPLY. POSITION 20.
1048*  010480      IF REPLY IS EQUAL TO "Y" OR
1049*  010490           REPLY IS EQUAL TO "YE" OR
1050*  010500                REPLY IS EQUAL TO "YES"  DISPLAY " ", ERASE
1051*  010510           PERFORM B-NEW-RELATION THRU B-N-RDONE
1052*  010520      PERFORM PACK-IT-DOWN
1053*  010530      ELSE MOVE 1 TO LAST-LEV
1054*  010540           PERFORM SEL-ITEMS THRU SEL-END
1055*  010550                DISPLAY " ", ERASE.
1056*  010560
1057*  010570
1058*  010580
1059*  010590 MOVE-DETER.
1060*  010600      SET CAT-INDEX TO NUMBER4.
1061*  010610      MOVE D-NAME(CAT-INDEX) TO ATTRIBUTE.
1062*  010620      MOVE LEV(CAT-INDEX) TO LEV-HOLD.
1063*  010630      MOVE SPACES TO D-NAME(CAT-INDEX).
1064*  010640      MOVE 00 TO LEV(CAT-INDEX).
1065*  010650      SET CAT-INDEX TO KNTT.
1066*  010660      MOVE ATTRIBUTE TO D-NAME(CAT-INDEX).
1067*  010670      MOVE LEV-HOLD TO LEV(CAT-INDEX).
1068*  010680      ADD 1 TO NUMBER4.
1069*  010690      ADD 1 TO KNTT.
1070*  010700
1071*  010710 MOVE-DET2.
1072*  010720      SET CAT-INDEX TO NUMBER4.
1073*  010730      MOVE D-NAME(CAT-INDEX) TO ATTRIBUTE.
```

```
1074*  010740      MOVE SPACES TO D-NAME(CAT-INDEX).
1075*  010750      MOVE 00 TO LEV(CAT-INDEX).
1076*  010760      SET CAT-INDEX TO KNTT.
1077*  010770      MOVE ATTRIBUTE TO D-NAME(CAT-INDEX).
1078*  010780      MOVE 11 TO LEV(CAT-INDEX).
1079*  010790      ADD 1 TO NUMBER4.
1080*  010800      ADD 1 TO KNTT.
1081*  010810
1082*  010820
1083*  010830 SEL-ITEMS.
1084*  010840      DISPLAY "LIST THOSE DATA-NAME-ITEMS WHICH ARE IDENTIFIED."
1085*  010850          POSITION 20.
1086*  010860      DISPLAY "ENTER XX TO SIGNAL NO MORE ENTRIES TO BE MADE."
1087*  010870          POSITION 20.
1088*  010880 SEL-NEXT.
1089*  010890      ACCEPT REPLY, POSITION 20.
1090*  010900      IF REPLY IS EQUAL TO "X" OR
1091*  010910          REPLY IS EQUAL TO "XX" OR
1092*  010920              REPLY IS EQUAL TO "XXX" GO TO SEL-END.
1093*  010930      PERFORM MOVE-DET-RT THRU M-D-REND.
1094*  010940      GO TO SEL-NEXT.
1095*  010950 SEL-END.
1096*  010960      EXIT.
1097*  010970
1098*  010980
1099*  010990 MOVE-DET-RT.
1100*  011000      MOVE 4 TO LOOK.
1101*  011010      SET CAT-INDEX TO 1.
1102*  011020      PERFORM SEARCH-STACK NUMBER4 TIMES.
1103*  011030      IF LOOK IS EQUAL TO 4
1104*  011040          DISPLAY "YOU SPELLED IT WRONG**TRY AGAIN**PLEASE."
1105*  011050          POSITION 20.
1106*  011060              "PRESS NEWLINE", POSITION 20
1107*  011070          ACCEPT REPLY
1108*  011080          IF REPLY IS NOT EQUAL TO "JSHEROIEF" DISPLAY " "
1109*  011090              ELSE NEXT SENTENCE
1110*  011100      ELSE SET CAT-INDEX TO 1
```

```
1111*  011110           MOVE 1 TO KNTT
1112*  011120                ADD 3, L-S-COUNT TO NUMBER4
1113*  011130                PERFORM COMPACT THRU PACK-IT NUMBER4 TIMES
1114*  011140                SUBTRACT 4, L-S-COUNT FROM NUMBER4
1115*  011150                     ADD 1 TO HOLD-NO
1116*  011160           SET CAT-INDEX DOWN BY 1.
1117*  011170 M-D-REND.
1118*  011180     EXIT.
1119*  011190
1120*  011200
1121*  011210 SEARCH-STACK.
1122*  011220     IF REPLY IS EQUAL TO D-NAME(CAT-INDEX)
1123*  011230           MOVE 1 TO LOOK
1124*  011240           SET STACK-INDEX UP BY 1
1125*  011250           MOVE REPLY TO ATTR(STACK-INDEX)
1126*  011260           MOVE SPACES TO D-NAME(CAT-INDEX)
1127*  011270           MOVE 00 TO LEV(CAT-INDEX)
1128*  011280           SET CAT-INDEX UP BY 1
1129*  011290     ELSE SET CAT-INDEX UP BY 1.
1130*  011300
1131*  011310 NEW-UNION-MAIN.
1132*  011320     SET CAT-INDEX UP BY 1.
1133*  011330     MOVE  D-NAME(CAT-INDEX) TO MATCH.
1134*  011340     MOVE 1 TO LOOK.
1135*  011350     SUBTRACT 1 FROM KNTTY GIVING THAT-MANY.
1136*  011360     PERFORM NEW-UNION THAT-MANY TIMES.
1137*  011370     IF LOOK IS EQUAL TO 3
1138*  011380           SET CAT-INDEX TO KNTT
1139*  011390           MOVE SPACES TO D-NAME(CAT-INDEX)
1140*  011400           MOVE 00 TO LEV(CAT-INDEX)
1141*  011410           ADD 2 TO KNTTY
1142*  011420           PERFORM COMPACT THRU PACK-IT KNTTY TIMES
1143*  011430           SUBTRACT 3 FROM KNTTY
1144*  011440           ADD 1 TO KOUNTY
1145*  011450           SET CAT-INDEX TO KOUNTY
1146*  011460     ELSE SET CAT-INDEX DOWN BY 1.
1147*  011470
```

```
1148*  011480
1149*  011490 NEW-UNION.
1150*  011500      SET CAT-INDEX UP BY 1.
1151*  011510      IF MATCH IS EQUAL TO D-NAME(CAT-INDEX) AND MATCH IS
1152*  011520      NOT EQUAL TO SPACES MOVE 3 TO LOOK
1153*  011530           SET KNTT TO CAT-INDEX.
1154*  011540
1155*  011550
1156*  011560
1157*  011570
1158*  011580
1159*  011590 COMPACT.
1160*  011600      IF D-NAME(CAT-INDEX) IS NOT EQUAL TO SPACES
1161*  011610           MOVE D-NAME(CAT-INDEX) TO ATTRIBUTE
1162*  011620           MOVE LEV(CAT-INDEX) TO LEV-HOLD
1163*  011630           MOVE SPACES TO D-NAME(CAT-INDEX)
1164*  011640           MOVE 00 TO LEV(CAT-INDEX)
1165*  011650               SET CAT-INDEX TO KNTT
1166*  011660               MOVE ATTRIBUTE TO D-NAME(CAT-INDEX)
1167*  011670               MOVE LEV-HOLD TO LEV(CAT-INDEX)
1168*  011680               ADD 1 TO KNTT
1169*  011690               SET CAT-INDEX UP BY 1
1170*  011700           ELSE  SET CAT-INDEX UP BY 1.
1171*  011710 PACK-IT.
1172*  011720      EXIT.
1173*  011730*LOAD THE RT SIDE DATA ITEMS INTO THE FDTABLE AND BLANK THE STACK.
1174*  011740
1175*  011750 LOAD2.
1176*  011760      IF ATTR(STACK-INDEX) IS NOT EQUAL TO " "
1177*  011770               SET F-D-INDEX UP BY 1
1178*  011780               MOVE ATTR(STACK-INDEX) TO LIST(F-D-INDEX)
1179*  011790               MOVE SPACES TO ATTR(STACK-INDEX)
1180*  011800               SET STACK-INDEX UP BY 1
1181*  011810          ELSE SET STACK-INDEX UP BY 1.
1182*  011820*END OF BUILD-F-D-TABLE
1183*  011830
1184*  011840*OUTPUT THE F-D-TABLE AS L-S --> R-S.
```

```
1185*  011850
1186*  011860 PRINT-F-D-TABLE.
1187*  011870     SET F-D-INDEX TO 1.
1188*  011880         MOVE SPACES TO THIS-ATTR.
1189*  011890         PERFORM GENOUT UNTIL THIS-ATTR IS EQUAL TO "XXXXXXXXXX".
1190*  011900 THATS-ALL.
1191*  011910     EXIT.
1192*  011920 GENOUT.
1193*  011930     MOVE SPACES TO PRINT-LINE.
1194*  011940     MOVE LIST(F-D-INDEX) TO LEFT-SIDE.
1195*  011950     MOVE 1 TO CHAR-COUNT.
1196*  011960     PERFORM BUILD-L-S LEFT-SIDE TIMES.
1197*  011970     SET F-D-INDEX UP BY 1.
1198*  011980     STRING "   ", ARROW, "   " DELIMITED BY SIZE INTO PRINT-LINE
1199*  011990           WITH POINTER CHAR-COUNT.
1200*  012000         MOVE LIST(F-D-INDEX) TO RIGHT-SIDE.
1201*  012010     MOVE RIGHT-SIDE TO NUMBER5.
1202*  012020     ADD LEFT-SIDE, 1 TO NUMBER5.
1203*  012030     IF NUMBER5 IS GREATER THAN 12
1204*  012040          SUBTRACT LEFT-SIDE, 1 FROM 12 GIVING NUMBER4
1205*  012050          SUBTRACT NUMBER4 FROM RIGHT-SIDE
1206*  012060          MOVE RIGHT-SIDE TO NUMBER5
1207*  012070          MOVE NUMBER4 TO RIGHT-SIDE
1208*  012080               PERFORM BUILD-R-S RIGHT-SIDE TIMES
1209*  012090               WRITE PRINT-LINE
1210*  012100               PERFORM MORE-LINES
1211*  012110     ELSE PERFORM BUILD-R-S RIGHT-SIDE TIMES
1212*  012120          WRITE PRINT-LINE.
1213*  012130         SET F-D-INDEX UP BY 1.
1214*  012140         MOVE LIST(F-D-INDEX) TO THIS-ATTR.
1215*  012150          SET THAT-MANY TO F-D-INDEX.
1216*  012160          IF THAT-MANY IS GREATER THAN FIND MOVE "XXXXXXXXXX" TO
1217*  012170          THIS-ATTR.
1218*  012180
1219*  012190
1220*  012200 MORE-LINES.
1221*  012210     IF NUMBER5 IS LESS THAN 12
```

```
1222*  012220      MOVE SPACES TO PRINT-LINE
1223*  012230      MOVE 1 TO CHAR-COUNT
1224*  012240      PERFORM BUILD-R-S NUMBER5 TIMES
1225*  012250      WRITE PRINT-LINE
1226*  012260      ELSE DIVIDE 12 INTO NUMBER5 GIVING THAT-MANY
1227*  012270           REMAINDER NUMBER4
1228*  012280           MOVE 12 TO RIGHT-SIDE
1229*  012290           PERFORM PRINT-IT THAT-MANY TIMES
1230*  012300           MOVE 1 TO CHAR-COUNT
1231*  012310           MOVE SPACES TO PRINT-LINE
1232*  012320           PERFORM BUILD-R-S NUMBER4 TIMES
1233*  012330           WRITE PRINT-LINE.
1234*  012340 BUILD-L-S.
1235*  012350      SET F-D-INDEX UP BY 1.
1236*  012360          STRING LIST(F-D-INDEX), BREAK DELIMITED BY SIZE INTO
1237*  012370                 PRINT-LINE WITH POINTER CHAR-COUNT.
1238*  012380
1239*  012390 PRINT-IT.
1240*  012400      MOVE 1 TO CHAR-COUNT.
1241*  012410      MOVE SPACES TO PRINT-LINE.
1242*  012420      PERFORM BUILD-R-S RIGHT-SIDE TIMES.
1243*  012430      WRITE PRINT-LINE.
1244*  012440
1245*  012450 BUILD-R-S.
1246*  012460      SET F-D-INDEX UP BY 1.
1247*  012470          STRING LIST(F-D-INDEX), BREAK DELIMITED BY SIZE INTO
1248*  012480                 PRINT-LINE  WITH POINTER CHAR-COUNT.
```

The Interactive Generation of Functional Dependencies

by

WILLIAM CLEN HUNT

B. G. S., University of Nebraska at Omaha
Omaha, Nebraska, 1972

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirement for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhatten, Kansas

1978

Information is a prime commodity in today's diverse world. Its availability is enhanced through the use of Data Base Management Systems (DBMS) which control the automatic storage and retrieval of data. These systems allow users to view data independently of its physical storage configuration.

A type DBMS based on the Relational Data Model allows a user to view data in logical groups. It makes possible the elimination of the representation-dependency from the user's interface, thus simplifing the user's interface requirements. A prototype system to automatically generate Third Normal Form Relations (3NF) has been developed. The system developed produces 3NF Relations in three phases: (1) produces a hierarchical representation of the input data; (2) generates a set of Functional Dependencies; and (3) synthesizes the Functional Dependencies into 3NF Relations.

The report 'The Interactive Generation of Functional Dependencies', describes in detail the algorithm for the interactive generation of Functional Dependencies. The design, implementation, and testing of the algorithm are presented in detail, including a comprehensive walk-through of the tested program. The abstract code and NCR level II COBOL code are provided in appendices. Enhancements to the system which would expand and simplify the user's interactions with the system are discussed.