THE DESIGN AND IMPLEMENTATION OF A LIBRARY DATA BASE

by

CONSTANTINE PAPANASTASOPOULOS M. Sc.

A Project

Submitted to the School of Graduate Studies

in Partial Fulfilment of the Requirements

for the Degree

Master of Science

McMaster University

September 1978

A LIBRARY DATA BASE

MASTER OF SCIENCE (1978)
(Computer Science)

McMASTER UNIVERSITY
Hamilton, Ontario
Canada

TITLE: The Design and Implementation
of a Library Data Base

AUTHOR: Constantine Papanastasopoulos

B.Sc. Universite de Montreal(1971)
M.Sc. McMaster University(1972)

SUPERVISOR: Professor K. A. Redish

NUMBER OF PAGES: v , 181

ABSTRACT

This project will implement an interactive system that allows users to interrogate the hierarchic program Library Index.

Major components include transformation of the existing Index into the necessary Data Base and the Design and Implementation of the interactive query-answer system.

The system will include the SCHEMA, SUB-SCHEMAS and the required application Programs.

ACKNOWLEDGEMENTS

I wish to express my appreciation to my supervisor Professor K. A. Redish for his guidance and encouragement during this project.

I am especially grateful to Dr. P. Dunmore, the Program Librarian, who has given this project to me together with the related data and his valuable assistance throughout this project.

I would, also, like to thank Professor J. Masterson for his assistance during the first stages of this project and T. Snider for the fruitful discussions with him during the implementation of this project.

Finally I would like to thank my wife Setta for the program Key-punching and typing of this project.

TABLE OF CONTENTS

C H A P T E R I

I N T R O D U C T I O N

This project involves the design and the inplementation of a Data Base which will use the software modules of the CDC-DMS-170 system to achieve its purpose.

The data will be stored in records forming files which will be joined together in meaningful relationships.

The DMS-170 functional configuration is shown in fig. 1 together with all modules comprising the system (DDL, CDCS, etc.).

Before we state the problem and the method which we will use to attack it will be wise to state some definitions and introduce briefly some terms used in the design of a Data Base.

In CHAPTER-2 we will discuss the design of the Data Base for the particular problem and explain the reasons which led us to use that specific approach.

In CHAPTER-3 we will discuss the implementation of the Data Base, the programming techniques used and the difficulties faced in implementing the Data Base.

We start with a brief theoretical description of a Data Base.

A "Data Base" may be defined as a collection of interrelated data stored together without harmful or unnecessary redundancy to serve one or more applications in an optimal fashion; the data are stored so that they are independent of programs which use them.

The above definition introduces the following main attributes of a Data Base.

i) Data independence which implies that the data and the application programs which use them are independent so that either may be changed without changing the other.

ii) Real-time accessibility which implies that the data will be quickly (fast response) available to users at almost all times when they need them.

iii) Privacy which implies that unauthorised access to the data will be prevented.

iv) User-machine transparency which implies that the user need not use the lengthy operation of writing programs in any programing language to retrieve the data.

The methods to attain all these attributes will be discussed in ch. II, where the design techniques will be explained.

The design starts with a logical description of the Data Base, referred to as a Schema.

DDL: is a high-level COBOL-like language used to describe the schema, from the designer's chart to program-statements easily assimilated by the Computer, through the DDL-Compiler.

A Schema: is a chart of detailed description of the entire Data Base.

The description is generated by DDL statements that name the schema, organize it into addressable storage units (files) called areas, and describe each record together with the characteristics (mode, size) of the data comprising the record.

The description starts by giving unique names to all the above-mentioned addressable units. It also establishes and names the neaningful relationships which join the different files. Only one schema exists for the Data Base.

A Sub-Schema: is a chart of detailed descriptions of the portion of the Data Base that is available to a particular program.

The description is generated by DDL statements and replaces the DATA DIVISION of all application programs using that portion. Any number of application programs can refer to a sub-schema. Any number of sub-schemas can exist for a Data Base.

The format of the Library Index has been designed and discussed by K. A. Redish in a Paper: "Tree structures for a Program Library Index", where an explanation is given for the reason why the Tree structure classification was used to design a user-oriented program Library Index.

A very short summary of this procedure is given below.

According to this method, the various fields of interest (problems and sub-problems) for which there are available routines, have been given classification numbers and classified thus into successive levels in the form of tree structures (fig. 2).

The root of every Library tree, labels a discrete group of related problems; every child (a node of higher level) is referred to a subgroup of its father group, so the "pathchart" is effectively a tree and so ideally each path through this tree will end in a unique leaf where the routines (if there are available ones) will exist for a particular application.

If there is no routine(s) for a certain application, the Data Base facility will allow the insertion of a routine(s) as soon as it becomes available, without causing any "havoc" in the system or any modification of the existing application programs.

The source data to be stored in the Data Base have been provided by Dr. P. Dunmore, program Librarian, in the form of files containing information concerning the different available routines of the subprogram Library to be used by individuals who wanted to solve specific mathematical or scientific problems.

The parameters of the existing routines are stored in a separate file which is linked to the routine file, so anyone can retrieve information concerning any parameter through the routine-parameter relation if he knows the routine name.

There is finally a private file where information pertinent only to the program librarian is stored and it is only accessible by him through the routine-private relation.

The librarian can access this file if he wants to retrieve information about any routine; this file is inaccessible to the user.

The design of the Data Base has been done on the basis that it will be used by interactive programmers.

The strategy of attack is a top-down query of the Library tree down to the leaves, where the specific routines are retrieved. Then the user can get any information concerning their tasks, parameters, control cards etc.

In the design of the system the man-machine dialogue has

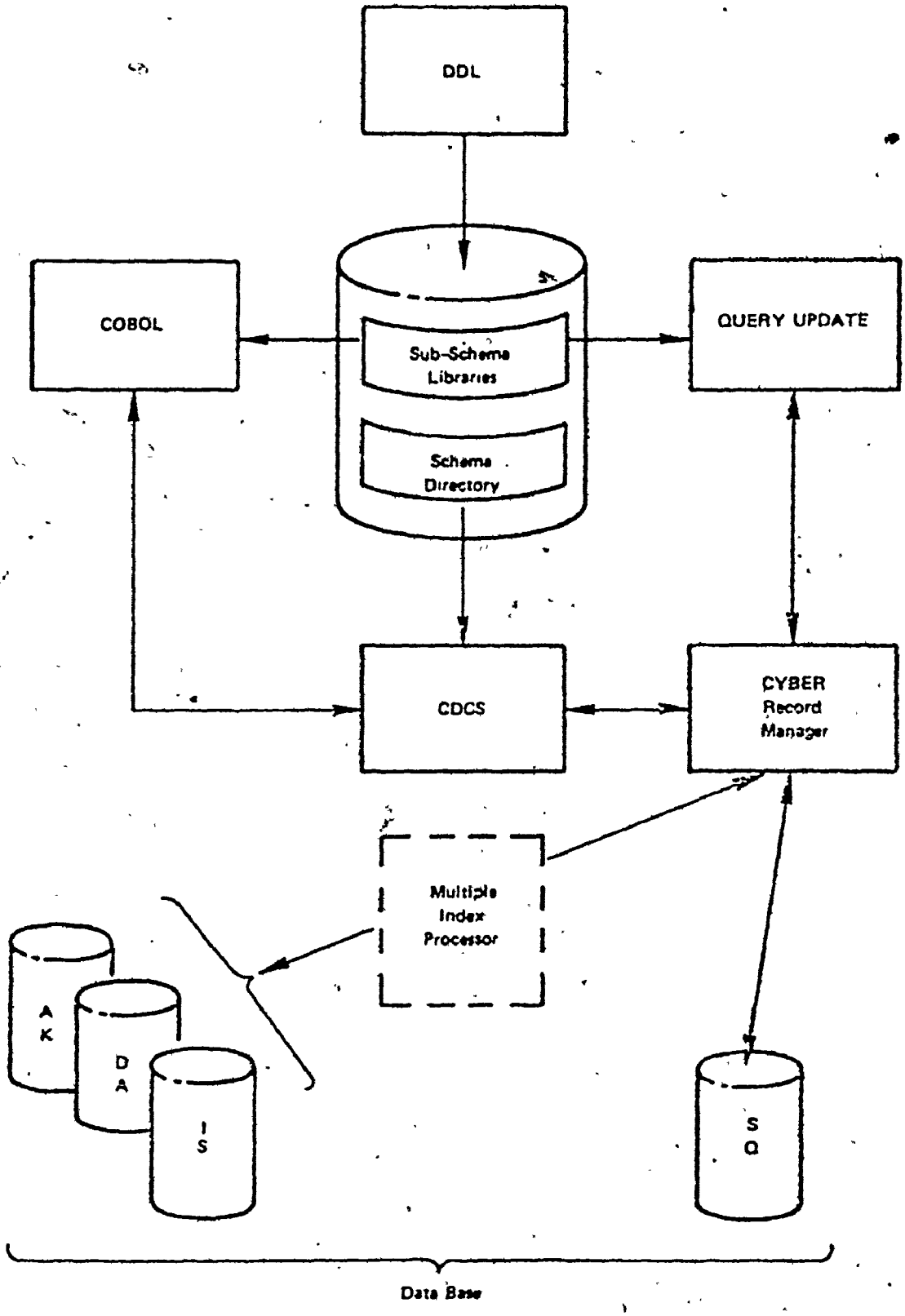


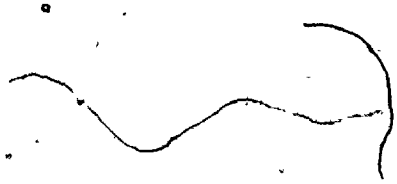
Figure 1 DMS-170 Functional Configuration

to conform with a sample dialogue proposed by P. Dunmore/for whom the system primarily has been designed.

The sample dialogue below is given to show from the very beginning one of the main objectives which the Data Base system has to fulfil.

USER-indicates the interactive user at the terminal.

CRT-indicates the system response on the TV-terminal.



SAMPLE DIALOGUE

USER: HELP, LIBRARY

CRT: THERE ARE AVAILABLE THE FOLLOWING AREAS IN THE LIBRARY
CHOOSE ONE:-

A. FACILITIES, SORTING, PLOTTING

B. STATISTICS

C. ENGINEERING

D. NUMERICAL ANALYSIS

USER: ? A

CRT: SELECT ONE OF:-

0003 TAPE HANDLING

0106 FILE ORGANIZATION

0109 TIMING

0302 COMPILERS

0306 PREPROCESSING AND EDITING

0601 SORTING

0606 CHARACTER AND SYMBOL MANIPULATION

0607 INFORMATION CLASSIFICATION, STORAGE AND RETRIEVAL

0704 INPUT

0804 OUTPUT-BCD

0806 PLOTTING

USER: ? 0003

CRT: SELECT ONE OF:-

0003A VOLUME

0003B PART

USER: ? C003A

CRT: SELECT ONE OF:-

0003AB COPY

USER: ? 0003AB

CRT: WE HAVE THE FOLLOWING ROUTINES:-

1. COPYST - UTILITY PROGRAM TO CONVERT A CODED SCOPE FILE TO A STRANGER TAPE-USEFUL IN TRANSFERRING DATA FILES TO OTHER INSTALLATIONS.
2. TPCON - UTILITY PROGRAM TO CONVERT A 7-TRACK STRANGER TAPE TO A FILE IN SCOPE FORMAT.

IF YOU WANT MORE INFORMATION ABOUT A PARTICULAR ROUTINE, PLEASE TYPE NAMES.

USER: ? TPCON

CRT: TPCON - CAN BE ACCESSED WITH CONTROL CARDS GRAB (MACLIB), LDSET (LIB MCLIB)

USAGE IS:

CALL TPCON (A,N,IA)

DO YOU WANT A DESCRIPTION OF ARGUMENTS OF THE ROUTINE? TYPE YES OR NO.

USER: ? YES

CRT: THE ARGUMENTS ARE AS FOLLOWS:**

A - NUMBER OF CHRS IN INPUT TAPE

N - NUMBER OF CHRS IN SCOPE TAPE

IA - FORMAT CODE

DO YOU WANT MORE INFORMATION FROM THE SYSTEM?

TYPE YES OR NO.

USER: ? NO

CRT: — OFF

During the design of the Data Base system the data have been stored in such a way, as to be able to conform with the above mentioned format of the man-machine dialogue.

Data concerning the parameter file have not been provided due to the fact that they were not available at that time, so the dialogue stops at the end of the routine level.

The schema and sub-schemas of the entire Data Base, encompass the parameter file, so only the application programs to create and retrieve that file need to be written in the future, when the data for the file exist.

** The argument description is ficticious for display purposes, since no argument information has been provided yet.

E43.04

NUMERICAL DIFFERENTIATION

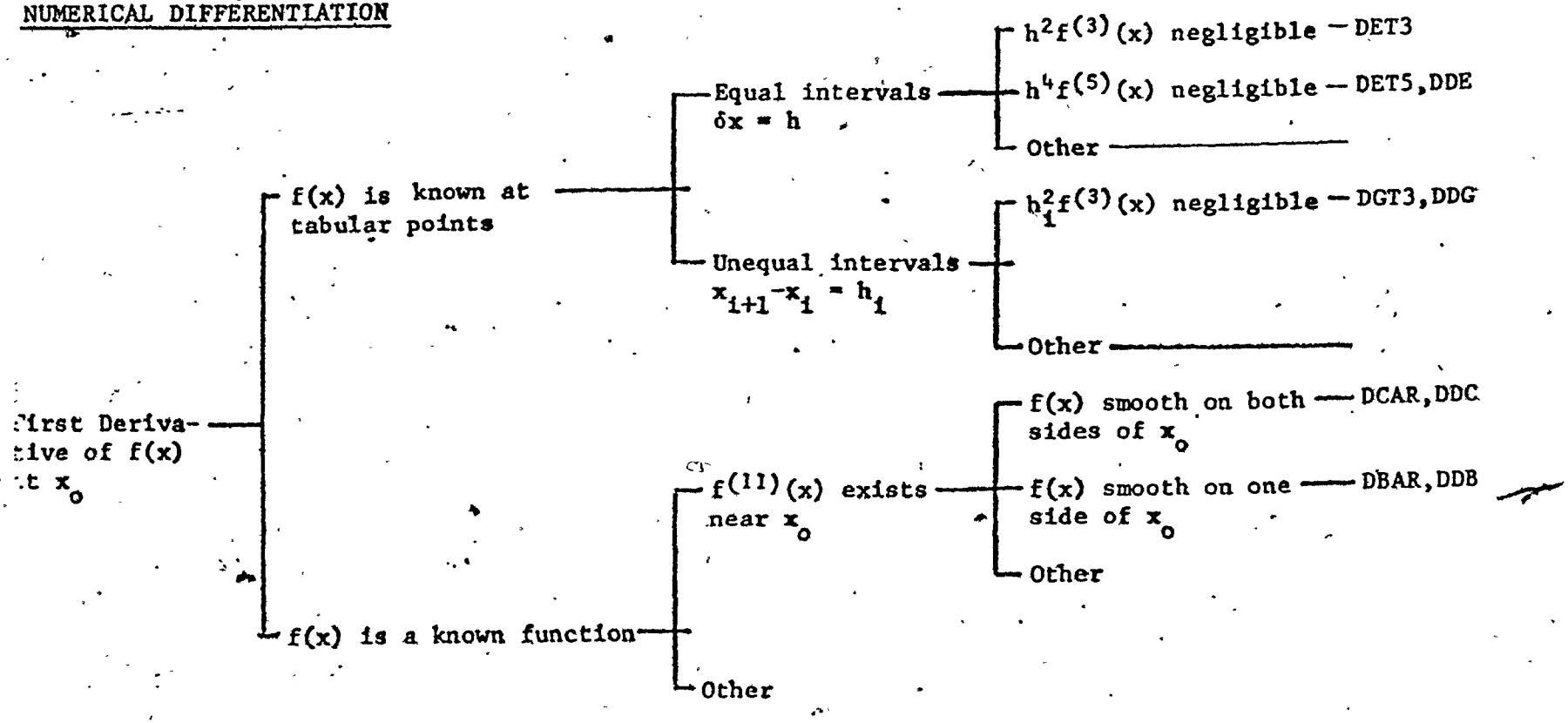


FIG. 2

C H A P T E R II

T H E P R O G R A M L I B R A R Y D A T A B A S E D E S I G N

S C H E M A

A system study originates with a recognition of a problem within an organization, then the analysis of the problem becomes a project.

There is no easy formula or checklist approach to making a systems study or development, since each case is different. Once the problem has been posed, as explained in ch. I, a design step-by-step bottom-up procedure will be followed to organize the supplied information in the form of data items, into aggregates, files which will compose the Data Base. A flowchart of the design procedure is given in fig. 3.

2.1. Problem Definition:

This is the most important and it has already been stated precisely and agreed upon all parties concerned.

A sample of the output format has been defined and given in ch. I.

2.2 Information Gathering:

As the first step we become familiar with the existing system, the history and backgroung of organization.

The design of the Index, the classification trees and the related routines attached to the leaves of each tree has already

been done and supplied to me by K. A. Redish.

The Data concerning the classification trees and the related subroutines already exist stored in direct access devices under certain item formats, and have been provided by Dr. P. A. Dunmore program Librarian in the form of two files concerning;

i) Data for the classification trees.

ii) Data for the routines related to these.

The format of these data items; the type, size, and the mode of each one have been specified to a certain extent. Of course these data will be reduced or expanded and reorganized so as to fit our design (record and file structure) without violating the required specification.

The specifications of data supplied were as follows:

2.2.1 Data concerning classification trees:

Here we present the data supplied, their size and type (A alphabetic, AN alphanumeric, N numeric) and their description.

1. Classification code: 12 AN chrs, of which the first four(4) are digits identifying (selecting) a particular Library tree out of Library Index; the remaining 8 chars are letters to select any one of the subordinate nodes of a higher level, filled with blanks to a total of 8 chrs. An example is given in fig. 4.

2. Library Index: 1 A chr, which is blank if this subject is included in the program Library Index, otherwise the letter A.

3. Note: 4 A, not in use for the time being but the space has been reserved by blanks for future use.

4. Subject description: A in

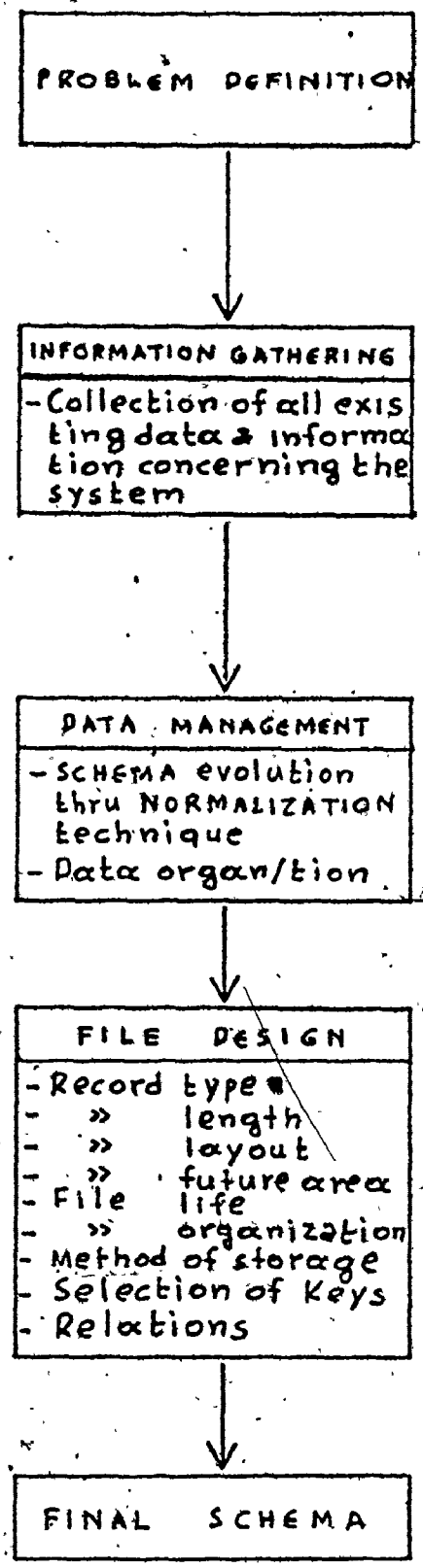


FIG. 3

Procedure followed for the data base design.

code, describing the particular subject.

2.2.2 Data concerning the subroutines:

The following data are specified for every subroutine available in the Library.

1. Routine name: 15 AN chrs, which name the particular routine; although it should be unique for any particular routine, there is a group of routines which have the same name (SPSS). For the time being no one routine has more than 7 chars, but it has been required to reserve space for future expansion.

2. Library name: 7 A chrs, it refers to the Library to which the subroutine belongs. This Library may be one of the following eight: MMUIMSL, SSPLIB, PLOTLIB, MACLIB, EISPACK, OPTISEP, BNDP, BMD.

3. Availability level: 1 N chr, in the range from 0 to 4; for the time being only three levels are used:

   1. means that the program is available on-line to all jobs. A main program can be executed by a single control card and this particular subroutine can be called by any FTN job.
   2. means that the program is available on-line, but that additional control cards are required to make it available to your job.
   3. means that the program is not available on-line, but that a sourcedeck may be obtained from the applications librarian at the Computer Centre.

4. Support level: 1 A chr, one of A/B/U/X. It indicates the

CREATION OF CLASSIFICATION CODE

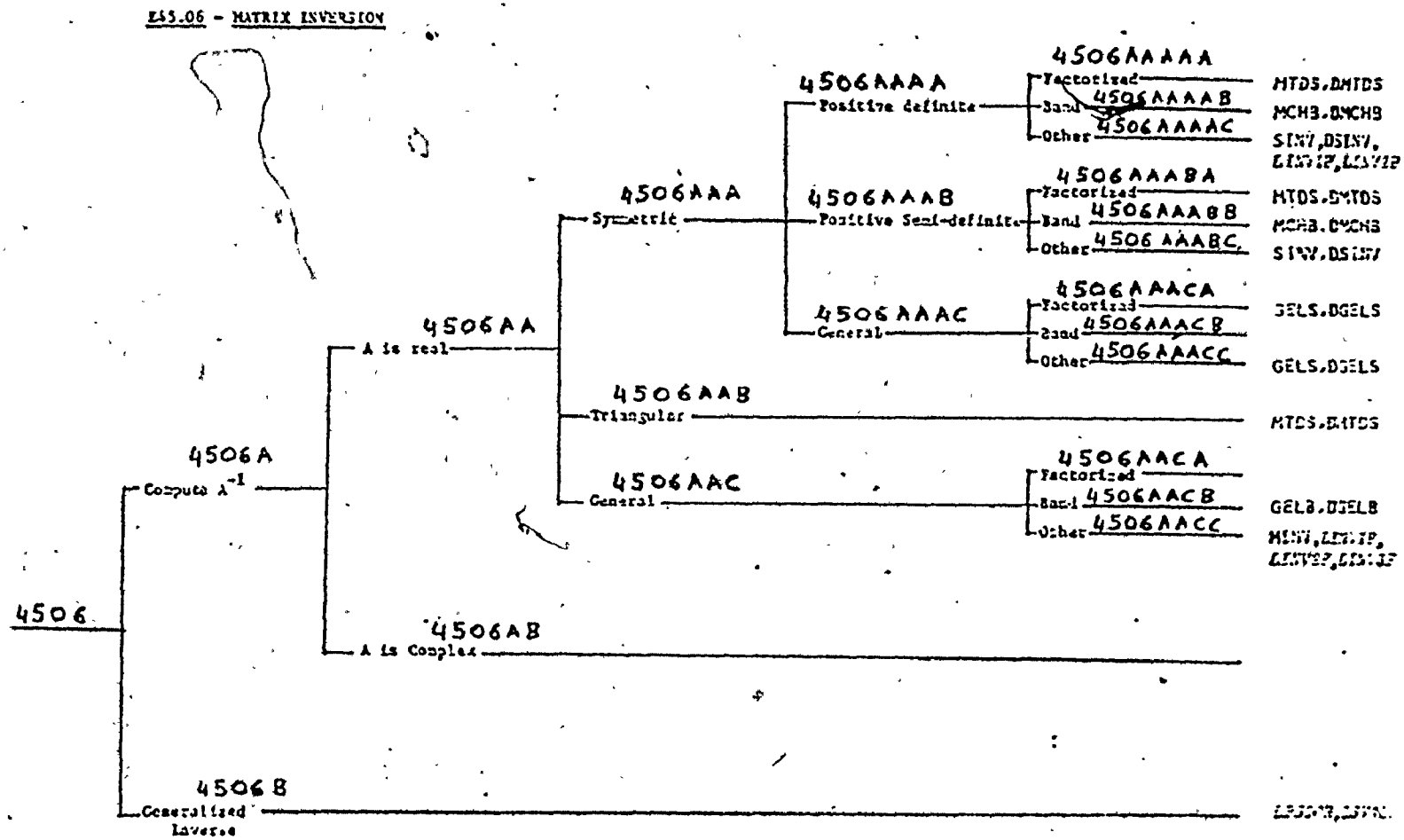


FIG. 4

support priority provided to this subroutine by the institution. It may be one of the following:

A. means that this routine is fully supported by the Computer Centre and full priority will be given in its debugging, by the programming assistant.

B. means that this routine is supported by the Computer Centre, in the sense that errors will be corrected when resources available to the library maintenance effort permit.

U. means that the routine is unsupported, and is made available to users on an "as is" basis.
Correction of any errors is the responsibility of the user.

X. means that the routine is fully supported by external sources at a level comparable to that of level A.

5. Documentation number: 8 AN chrs, a reference to the documentation of the program.

6. Classification code: 12 AN chrs, the same as the tree classification code. It means that a pointer in the node-record points to the routine-record having the same classification code. So several routines may have the same classification code (in case that more than one routine exists for a certain subject).

7. Description of the routine: An alphanumeric variable text in ASCII code, giving the description of the routine's task.

8. Argument names: 7 AN chrs, the names of the routine's parameters in the order they appear in the call statement.

The number of parameters for each routine is variable.

9. Type of argument: 1 A chr, one of R/I/D/C/H, that mean, Real, Integer, Double Precision, Complex, Hollerith respectively.

10. Input/Output: 2 A chrs, one of I/O/IO, that means, Input variable, Output variable or both respectively.

11. Description of argument: Alphanumeric variable text in ASCII code, describing the use of each parameter.

12. Pointers to equivalent arguments in other routines: For each argument there are several data items, indicating the name of another routine (mainly in matrix routines) where an equivalent argument is found.

13. Source language code: 1 A chr, which may be F/M/A/C, that specifies the language in which the source code of this subprogram has been written.

2.2.3 Data concerning private information:

There are data which either are of non-importance to the user or their access must be prevented for privacy reasons or both. On the contrary they are useful and important to the program Librarian (statistics, information reasons) so these data are stored in a different file.

1. Source tape: 7 AN chrs, stores the tape Label containing the binary code of the particular routine.

2. Deck name: 9 AN characters.

3. Monitoring name: 7 AN chrs.

4. File position within source: 1-100 digits.

5. Documentation source: 1 A chr, may be I/X/U.

6. Maintained by: 4 AN chrs. This data item can have up to a maximum of 4 occurrences.

2.3 Data Management:

Data management is the process of organizing data into a logical structure and providing methods of access to these data.

Here we shall face the problem of:

i) Structure of data.

ii) Organization of data.

The question of how data should be structured depends upon the hierarchy of information.

We proceed from the (bottom) lowest form of data to the highest (data item-record-file-data base).

The method which will be used is based upon the Normalization technique discussed by James Martin in his book Computer Data Base Organization.

2.3.1 Normalization technique:

Normalization technique solves the problem of complicated Data Base systems. As a Data Base grows, then the system becomes complicated, cumbersome, inflexible since the logical linkages become a net web. Entanglements built up in trees and plexes are solved by the Normalization technique. This technique is applied to the logical description of the data (user's view) and not to the physical representation.

Normalization: is a process which step-by-step replaces complicated relationships (as in networks) by a 2-dimensional tabular form (flat file), without loss of any information. This step-by-

step procedure is based on the principle;

"A file which is "flat" (fig. 5) except for a repeating group, can be normalized by removing the repeating group into a separate table or flat file without any loss of information".

The new file related to the previous one (as consisting a part of it) is given a new name.

2.3.2 Schema creation procedure:

The following procedure is nothing else than a mere "normalization of a tree structure" as our problem is. This procedure is going to provide a first approximation to a schema for the designed Data Base. Other considerations then, are going to establish the final form of the schema.

From first glance at the data which we have at our disposal we see that they can be easily divided into three discrete groups; the data concerning the classification trees, the data concerning the routines, and private information for every routine. So, based on the fact that, a record consists of all data items logically related to one particular object and a file is a collection of such records, then as a first approximation we should expect three (3) record types and the collection of each record type composes one file, so we have three (3) files.

In one file we store all the records concerning the classification trees, one record per node.

In the second file we store all records concerning the available routines, one record per routine containing all the data related to this routine.

The third file contains all the records containing

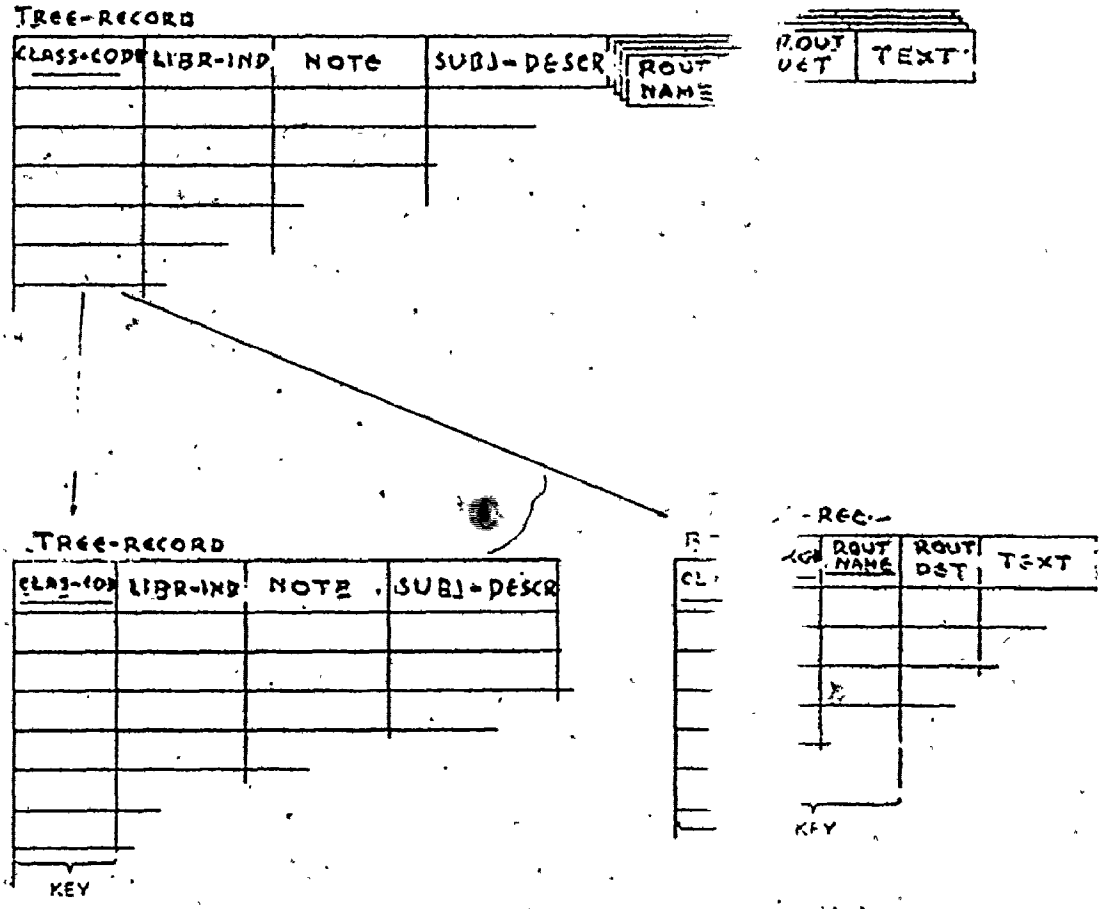


Figure 5 A repeating group is removed by splitting file into two relations.

te information concerning the available subroutines, one record per routine. This file will be inaccessible to users for privacy reasons. This observation at the begining may be a good hint for the future.

We try now to deduce the schema using the normalization technique, knowing from the above observation roughly what we should expect as a result. Our basic structure is a classification tree, where each node of this tree is of the same record type (one node for each subject) and every tree extends to a different level. Therefore the classification tree is a homogeneous tree of variable depth. So it can be represented by one file consisting of one type of record.

At a leaf of the tree the routines are attached from 0 (no routine) to any number of them. Since the routines attached to a node consist of repeating groups we normalize. We start by removing the repeating group (or vector) into a separate file incorporating as a key of the relation or as a joining item in the new file the key of the file above it. This procedure takes place at every level until all repeating groups (or vectors) have been removed and we end up with a flat file. With such a procedure the tree structures evolve into successive levels (a file per level). In fig. 6, 6a the successive diagrams show how the first approximation of the schema is derived through successive applications of the normalization technique at each level whenever a repeating group is encountered.

At the next level, in the routine file, every routine re-

cord contains the arguments of the routine, which vary in number together with the argument details. So there is a repeating group, which is removed and a new file is formed containing the arguments of each routine on a record type. Care should be taken in the new file to include as a linking item the key of the routine file (routine-name) together with each argument of this routine (one to many relationship).

Privacy requirements also dictate another record type to form another file for all private information. So, we maintain one record for each routine to contain all private information, comprising another file.

Certain restrictions are placed on variable occurrence data items in the COBOL sub-schema. If the COBOL sub-schema is a consideration when the schema is created, the following rules apply:

-Only one variable occurence data item can be specified for a record.

-This variable occurrence data item must be the last item in the record.

Based on these, we observe that in the record type of the argument file we have two variable occurrence data items (they are so specified in our supplied data); the variable text of the argument description and the variable number of the pointers to other routines. So we are obliged to normalize further, removing one of the two repeating items into a separate file. We can choose either the description or the pointers (fig. 6b).

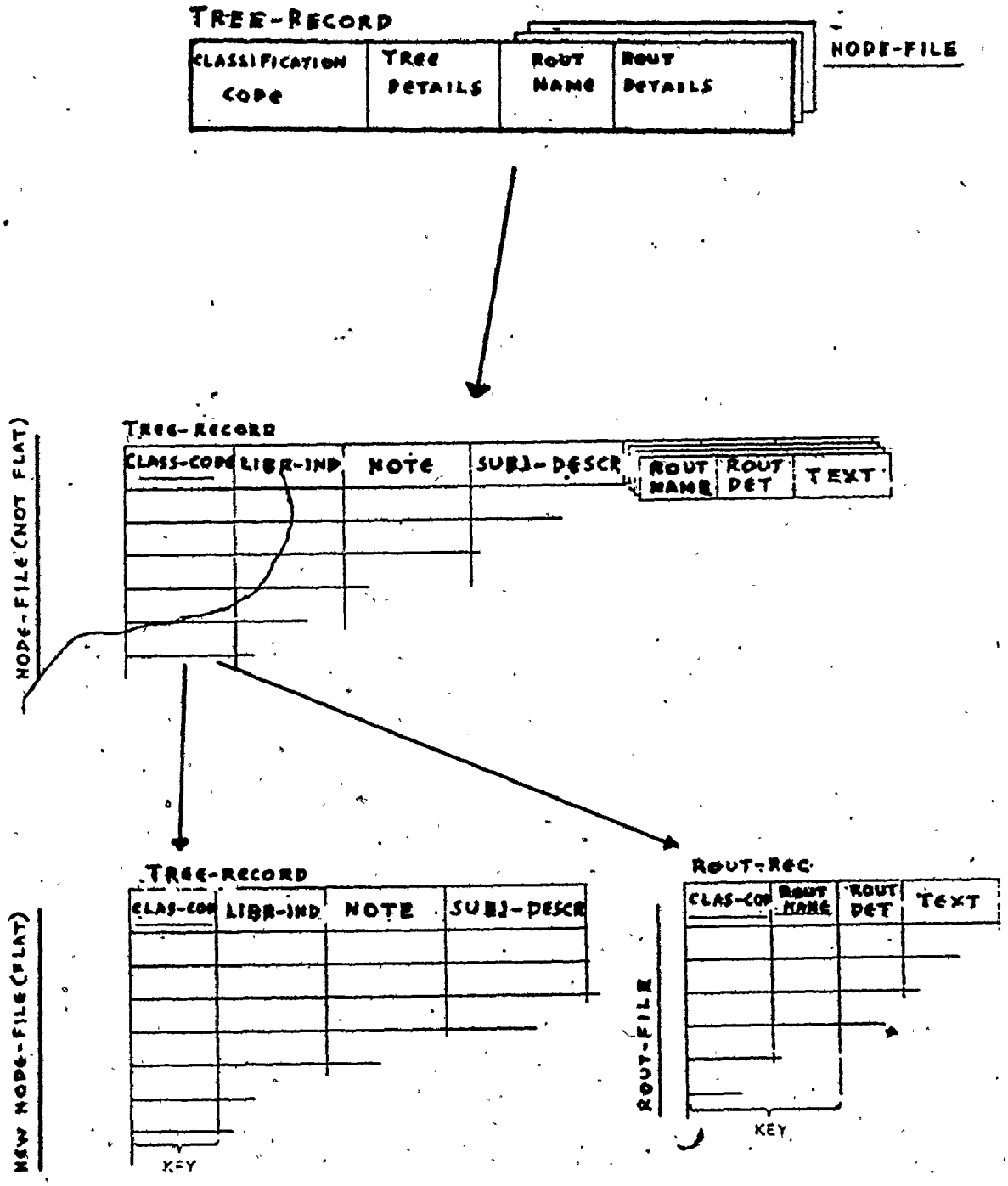


FIG.6 NORMALIZATION OF THE NODE-FILE
(Removal of the Routine repeating group).

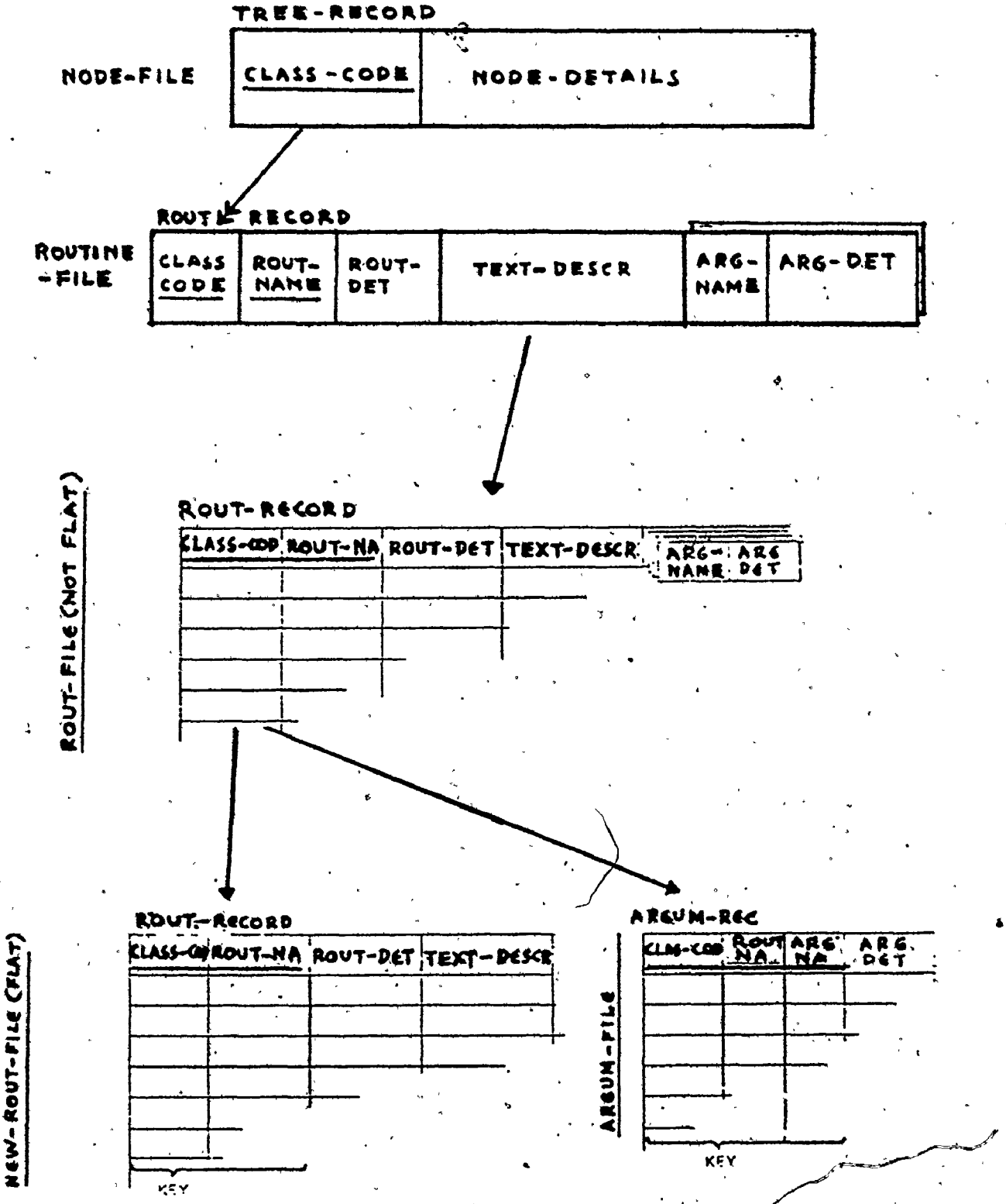


FIG. 6a NORMALIZATION OF THE ROUT-FILE
(Removal of the argument repeating group).

We prefer the first choice in order to have all the variable text in one file; so the first schema approximation is given in fig. 7.

2.3.3 File design:

File design requires the description of the characteristics of the file and the records which make it up, including each item withing each record and its position.

To refine the schema until the final form is achieved, some additional factors must be taken into consideration.

Record type: Since variable text exists in each file (node description, routine description argument description, pointers) a variable length record has to be chosen for each file. CYBER Record Manager (CRM) which performs execution-time I/O processing for DMS-170, supports a variety of record types from which we must choose one. These record types include fixed length (RT=F), zero byte (RT=Z) and character count (RZ=D). An easily handled length record is the trailer count (RT=T) which we choose for all the files, except the last one (private) which will be chosen to be of type fixed (RT=F) since it contains a vector of constant length (4 times).

RT=T-type records: consist of a fixed-length base and a variable number of fixed-length trailer items. A count field (a data item) in the fixed-length base specifies the number of fixed-lenght trailer items appended to each record.

COBOL sub-schema requires that:

-The data item controlling the number of occurences of the variable size vector should be the item just be-

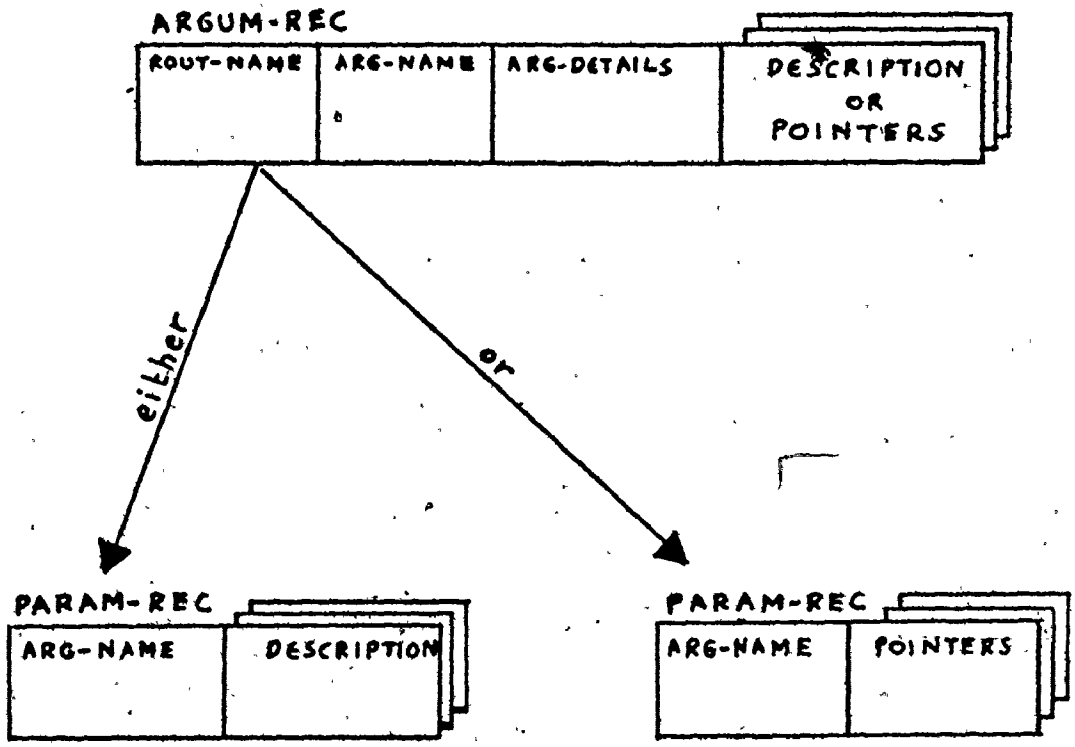
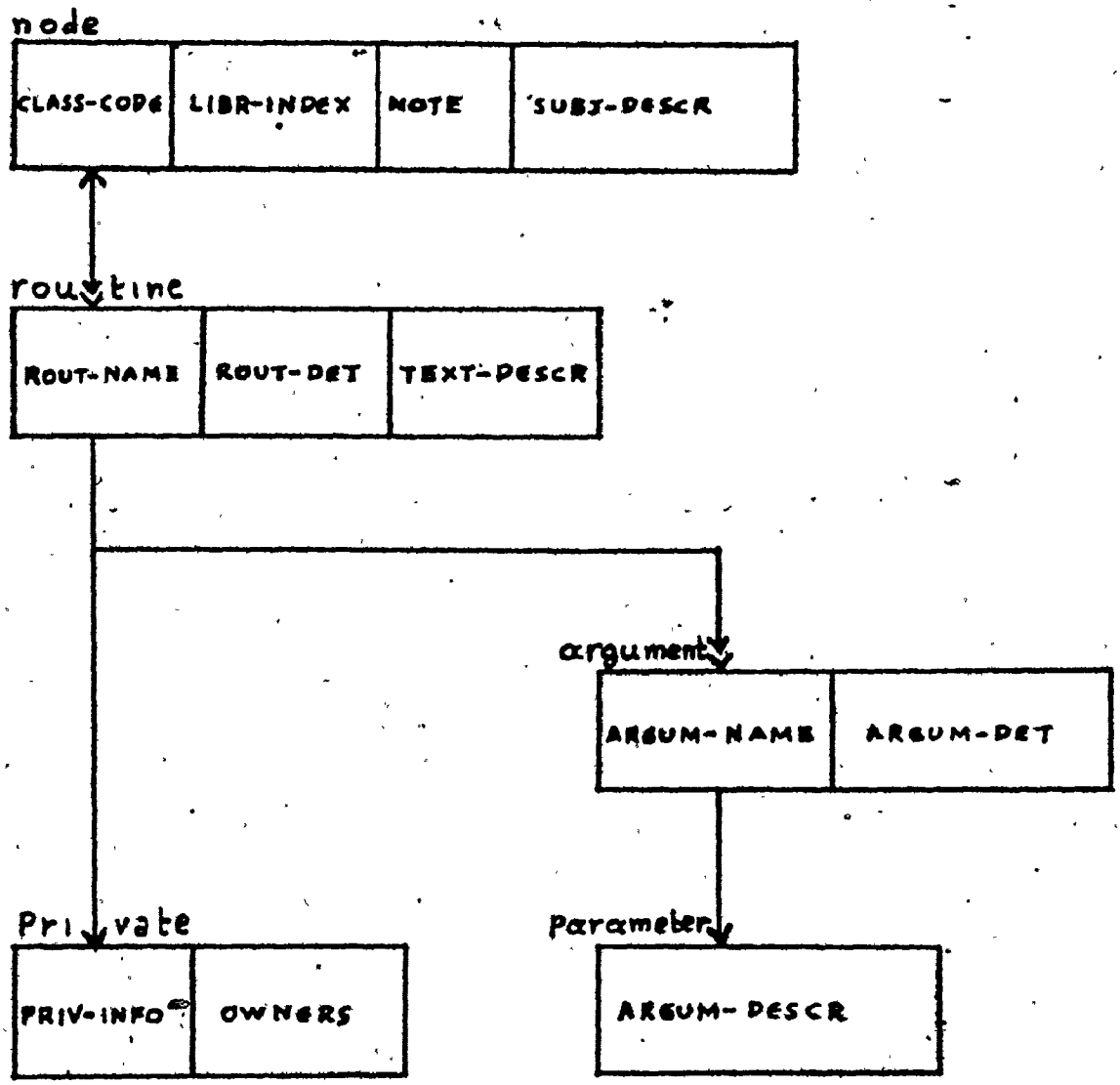


FIG. 6b Separation of more than one repeated items per record.



NORMALIZED FORM OF THIS SCHEMA

NODE (CLASS-CODE, LIBR-INDEX, NODE, SUBJ-DESCR)
ROUTINE (CLASS-CODE, ROUT-NAME, ROUT-DET, TEXT-DESCR)
ARGUMENT (CLASS-CODE, ROUT-NAME, ARGUM-NAME, ARGUM-DET)
PARAMETER (CLASS-CODE, ROUT-NAME, ARGUM-NAME, ARGUM-DESCR)
PRIVATE (CLASS-CODE, ROUT-NAME, PRIV-INFO, OWNERS)

The item prior to parentheses is the name of the relation.
The items inside the parentheses are the names of the data items.
The underlined items are the primary keys.

FIG. 7

fore the vector.

The format of a T-type record is shown in fig. 8. According to this format, another data item, the counter field, is added to each T-type record.

Layout: The position of data items in the records is not critical in most cases, but it should follow some logical sequence which will be easy to read and understand. Exceptions are the variable vectors, which must be the last items on each record and the count filed the last but one.

Actual record length: The total number of characters contained in the record should be specified.

For T-type records the formula is:

$$\text{Actual Total Length} = HL + (n \times TL)$$

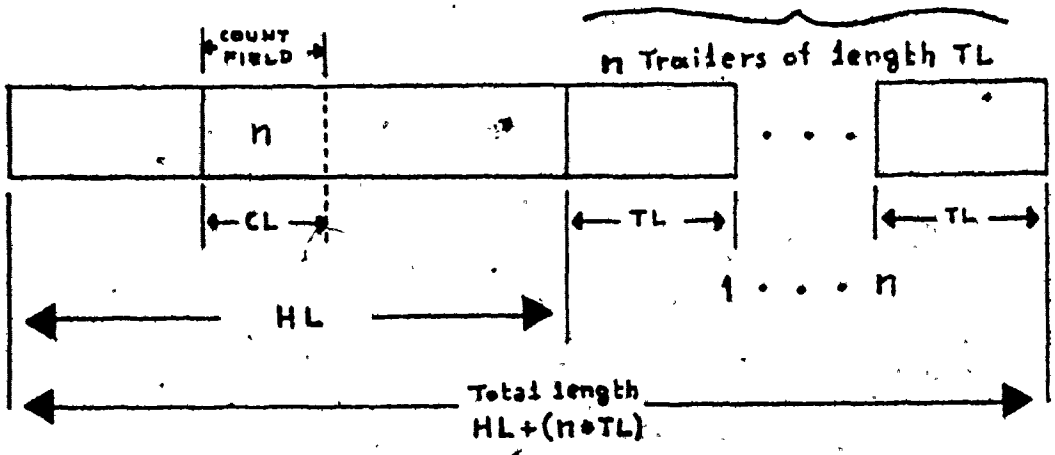


FIG. 8 THE FORMAT OF T-type RECORD

where HL Header length (sum of characters of all data items).

TL Trailer length (number of chrs in one trailer item).

n An integer from 0 to n.

To define the size of the integer n we consider the next condition.

Future area: To avoid having to change an entire file at some later instance, some blank space can be left which can be defined as "For Future Area Use", on any record type for which we expect that more data items will be needed for the future. For existing vectors, more trailers than we need now should be taken, for further text addition. This consideration affects the size of the integer n.

In our case a data item has been reserved in the node-record, now filled with blanks.

To define the maximum size of the count-field data item so as to reserve the necessary space in the schema, we made a test run of the supplied files containing the data for the classification trees and the routines. A COBOL program extracted the maximum number of the trailers from each file which was found to be 8 items for the node-file and 44 items for the routine-file (of 10 chrs each) containing the variable ASCII text.

To take care for the future addition of more record instances or possible expansion of the variable text, in order to give more information, we increase the number of trailers, so the schema will reserve the necessary space for the variable text in advance. We reserved respectively 25 items for the node-re-

cord and argument record, and 100 items for the routine-record and argument-record.

Life: The life of the data file should be specified in terms of the updating cycle (daily, weekly, monthly), retention periods etc. In our case, since the file is almost not volatile at all no consideration has been taken at all for the updating (addition, deletion) of the files.

Number of files: We have created five areas (files) with one record type per file, in order to avoid the use of record code necessary when multiple record types exist within a file.

TREEAREA- stores the data concerning Library trees.

ROUTAREA- stores the data concerning the routines.

ARGUMENTAREA- stores the data concerning routine arguments, except argument description.

PARAMETAREA- stores the variable text of arguments description.

PRIVATEAREA- stores information concerning only the program librarian, inaccessive to the users.

File organization: Four file organizations are supported by CYBER Record Manager and can be accessed by CDCS: sequential (SQ), indexed sequential (IS), direct access (DA) and actual key (AK).

The file choice is specified in the FILE control statement and is stored in the schema directory. The choice of the organization we specify for each area (file) depends upon the purpose of the Data Base and the way we want to use it. The purpose of our Data Base is to access data both randomly and sequentially from almost all files.

In the schema corresponding to a file in a program is the AREA which is the portion of mass storage, the schema definition reserves; it can be accessed in the same manner as a file.

Treearea: Both access methods will be used here; sequentially- if we want to retrieve one or more routines related to a particular subject, we must reach the leaf of the tree which describes the particular subject. To do this, we must traverse the tree from node-to-node, top-down through a successive query along the levels of the tree, leading us to the leaf where the routine or the routines will be attached for our particular choice. Randomly- since the whole trees representing the subjects have been divided into groups of interest, such as Science, Engineering, Statistics etc., so having sorted all relevant trees in group of trees in ascending order of classification code, in order not to read sequentially, passing through all groups if we want to access a certain group; we randomly access the root of the first tree of the particular group containing the relevant subject and then read sequentially the roots of all trees of the group until we choose the one which suits our requirements. We then read, sequentially, from top-to-bottom through the nodes of higher order levels until we get our routines. So the organization of this file will be indexed sequential (IS).

Routarea: This file is accessed through its parent file, by using the classification code of Treearea to retrieve the routine-records having the same classification code, attached to the leaf containing the subject of interest. This is a random access area.

If we want also to retrieve information concerning a particular routine, we use the routine-name as a Key to retrieve the particular record instance out of the routine file. So the organization easily could be DA, but since this file has been organized before as sequential with alphabetic order of routine-name for sequential listing, in order to satisfy this requirement we choose it to be also IS.

Argumentarea, Parametarea: Those areas (files) could easily be organized as sequential since they will almost never be accessed directly, but through their owner routine record of the routine file as will be seen in the relation description below.

To make these files more flexible for future use, in case someone wanted to access these files directly, let us say if one wants to get information concerning a parameter of a certain routine, or if he wanted to list the parameters of certain routine, it is chosen to be IS.

Privatearea: This is the same case as the previous two files, since it is accessed through its parent routine-file.

To make it independent, in case the program Librarian wanted to access it directly through the routine-number we choose it to be IS.

Methods of storage: Since our files are all IS we store all data on disk.

Selection of Keys: Since our files are indexed sequential, every record should have some means of identification which is unique to that record.

Keys are often obvious, however, they are sometimes hard to create.

Another fact which must be taken into consideration here, is that since we use COBOL sub-schema, using COBOL5 to write the application programs we must conform to the regulation that:-

No duplicate primary keys are supported by COBOL5.

The Key of the tree-record is obvious, the classification code which uniquely identifies a subject. For the routine file, the routine-name could be easily selected as a primary Key, in case it was unique (as expected), but strangely with the supplied data, there were routines with the same name, attached at different nodes. This is the case with the 18 routines with the same name SPSS, attached to nodes with classification code 0804, 1302, 1303. One way to make a unique Key is to concatenate both data items (routine-name and classif-code) and create one Key, as the normalization requires. Another way is to create a new Key; this is the way I prefer.

The unique Key easily constructed is the routine-number. Simply at a run of the existing file a counter generates numbers from 1 to the number of routines at steps of 10, to allow for future insertions, assigning a number to a created new data item on the routine-record.

In the argument-file, the argument-name is not eligible by itself to be chosen as a Key, since many routines could have the same argument for different purposes. Again as previously, a concatenation of routine-name (major Key) and argument-name

(minor Key) could make a unique name (this is what normalization requires), but still I dislike concatenation as lengthening data items, so I prefer to create a new unique primary Key, the argument-number as in the routine-file.

Parametarea, because it has been chosen IS, it has the argument-number as a unique primary Key, though it is accessed through its parent record parameter-record. The primary Key for Private-area is the routine-number.

Relationships: Here we shall establish the relationships and the joining items. There are three relationships joining our files into meaningful relations.

Routretrieve: The way we retrieve the particular routines for a certain subject is to search sequentially the tree-file until we encounter the leaf containing the subject for which we need a routine. Then using the classification code of this node-record as a Key, we randomly access the routine-file to retrieve all the routines under the same classification code (alternate Key for routine-file). But, if we do not want to use the classification code as an alternate Key in the routine-file we simply relate the two files, by joining any record instance of the tree-file with all the instances (one-to-many relation) of the record of the routine-file having the same classification code with its parent records of tree-file. This means that we join all the routines relevant to a subject to the node containing the subject (in other words, we attach the routines to the leaves of the trees). Accessing the tree-record containing the subject of interest,

we retrieve at the same time all the related routines. The relation is one to many, because for a certain node (subject) there may exist more than one routine.

Routparam: Under this relation there are related three areas (files), Routarea-argumentarea-parametarea.

Here we relate one particular routine with all its arguments, one-to-many relation, so we join all record instances of routine-file with those records of parameter-file having same routine-number.

In this way, through the first part of the relation, we retrieve all the arguments of a certain routine, as soon as we have accessed that routine and at the same time also through the second part of the relation (argumentarea-parametarea) we retrieve the instance of the record from the parameter-file containing the description of this particular argument, a one-to-one relation between these two files.

Routprivate: Here we join any one routine record of routine-file with one record instance of the private-file containing the information for that particular routine, a one-to-one relation.

Key modification of tree area: The physical layout of the data for the tree-file supplied by the program Librarian was the left-list layout in the top-down-left-right sequence (HSAM of IBM).

In this sequence (fig. 9) a sample tree (root clasif-code 1305) is represented physically in ascending order of the classification code key. This layout is of no use at all for our applications. Since we access this file sequentially in order to

spot the subject of our interest, the left-right layout will force the system to read almost every record before we select the one we want. If, let us say, we wanted the 1305 BCA record, we should read 14 records (fig. 9) before we reach it with the above approach. But, if we arrange our records, so that we access everytime all the records (nodes) of a successively higher level, this will reduce the searching time drastically. As the parents of the successive higher levels are collectively displayed for choice, at any successive display of a level, we shall eliminate whole branches (of higher levels) leaving only the ones of interest to search for.

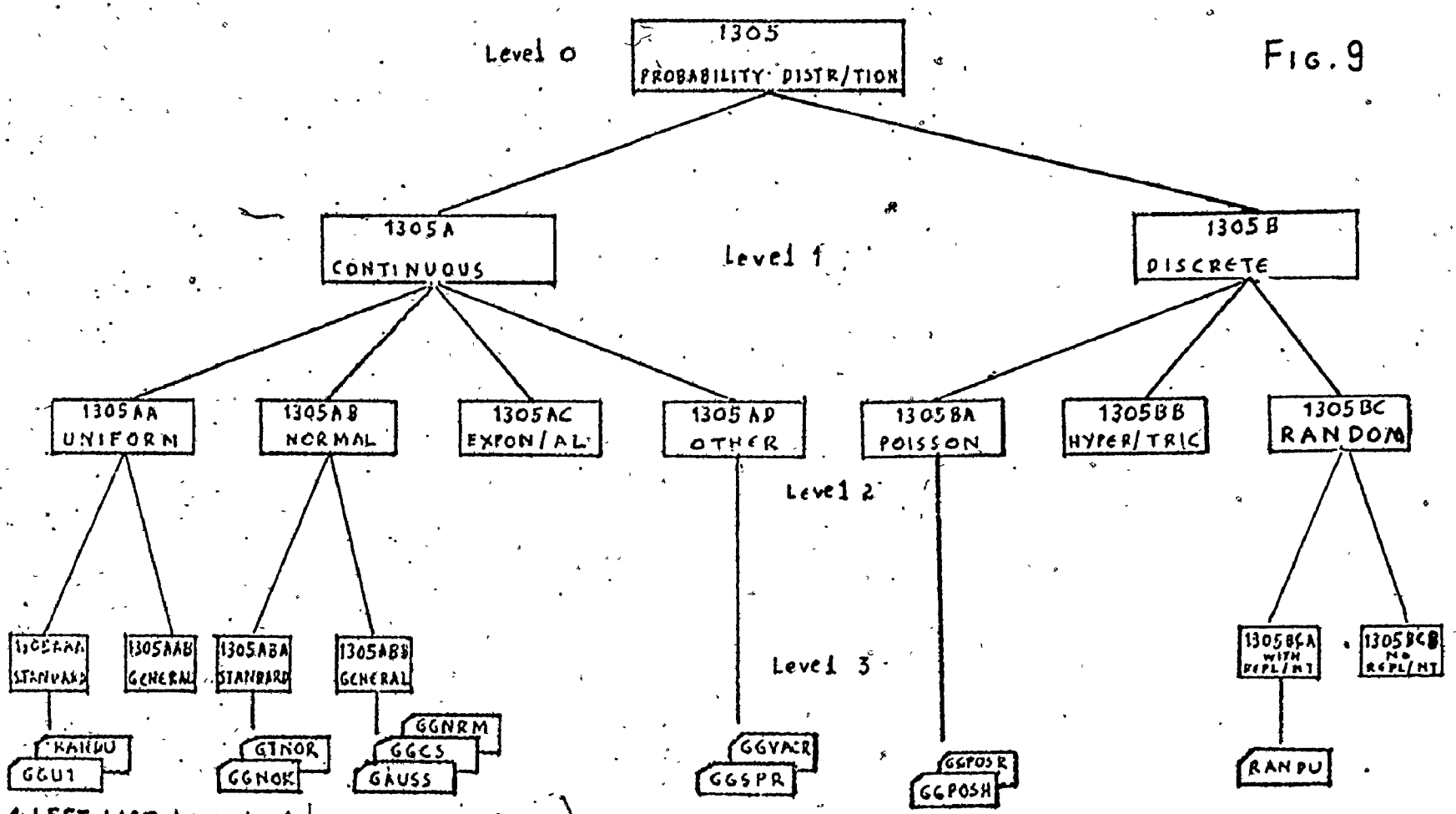
In our case the record 1305 BCA can be spotted in 3 displays and 6 records have to be read before we access it, as follows:

1st Display	2nd Display	3rd Display
1305A continuous	1305BA Poisson	1305BCA Random withrepl/ment
1305B discrete	1305BB Hyper/tric	1305BCB Random with
	1305BC Random	norepl/ment

Since we know in advance, that our problem will be discrete statistics concerning random variables with replacement from the 1st display we shall eliminate all the continuous branch, in the second display we shall eliminate all branches but Random variables, in the 3rd display we shall pickout our record.

In order to access records with a level-by-level sequence we must modify the classification code before we sort the file

FIG. 9



1. LEFT-LIST LAYOUT (PHYSICAL REPRES/TION)

1305	1305A	1305AA	1305AAA	1305AAB	1305AB	1305ABA	1305ABB	1305AC	1305AD	1305B	1305BA	1305BB	1305BC	1305BCA	1305BCB

2. LEVEL-BY-LEVEL LAYOUT (PHYSICAL REPRES/TION)

01305	11305A	11305B	21305AA	21305AB	21305AC	21305AD	21305BA	21305BB	21305BC	31305AAA	31305AAB	31305ABA	31305ABB	31305BCA	31305BCB

into ascending order of this key. To do this we add as a prefix to every key its level number (0-max level depth), so the key of every successive level will be of higher order from the key of its previous level, in every tree. Since the tree-file is a collection of trees, so all the roots of the trees (0 Level) will be in sequence together and all the levels of the same depth of the trees will be sequentially ordered, so the system in every successive access will display level-by-level the selected tree's levels until the leaf where the routines are, will be found.

To do this modification we increase the length of the classification code data item from 12 chrs to 13 chrs to add the level number prefix in each key before we create our data base file, from the supplied one.

The method of modification is an implementation technique, discussed in the next chapter, but here in the schema definition we reserve the necessary space for that. The completed schema with relations is shown in fig. 10. The logical name of the schema LIBRARY, where the schema directory has been stored, is HELPLIBRARY-DB and the Physical file name CYLISCH.

S U B - S C H E M A S

The sub-schema designs depend on the particular applications involving the stored data, and every application may use all or part of the data which have been stored in the devices.

If a group of applications use a particular portion of

HELPLIBRARY-DB SCHEMA

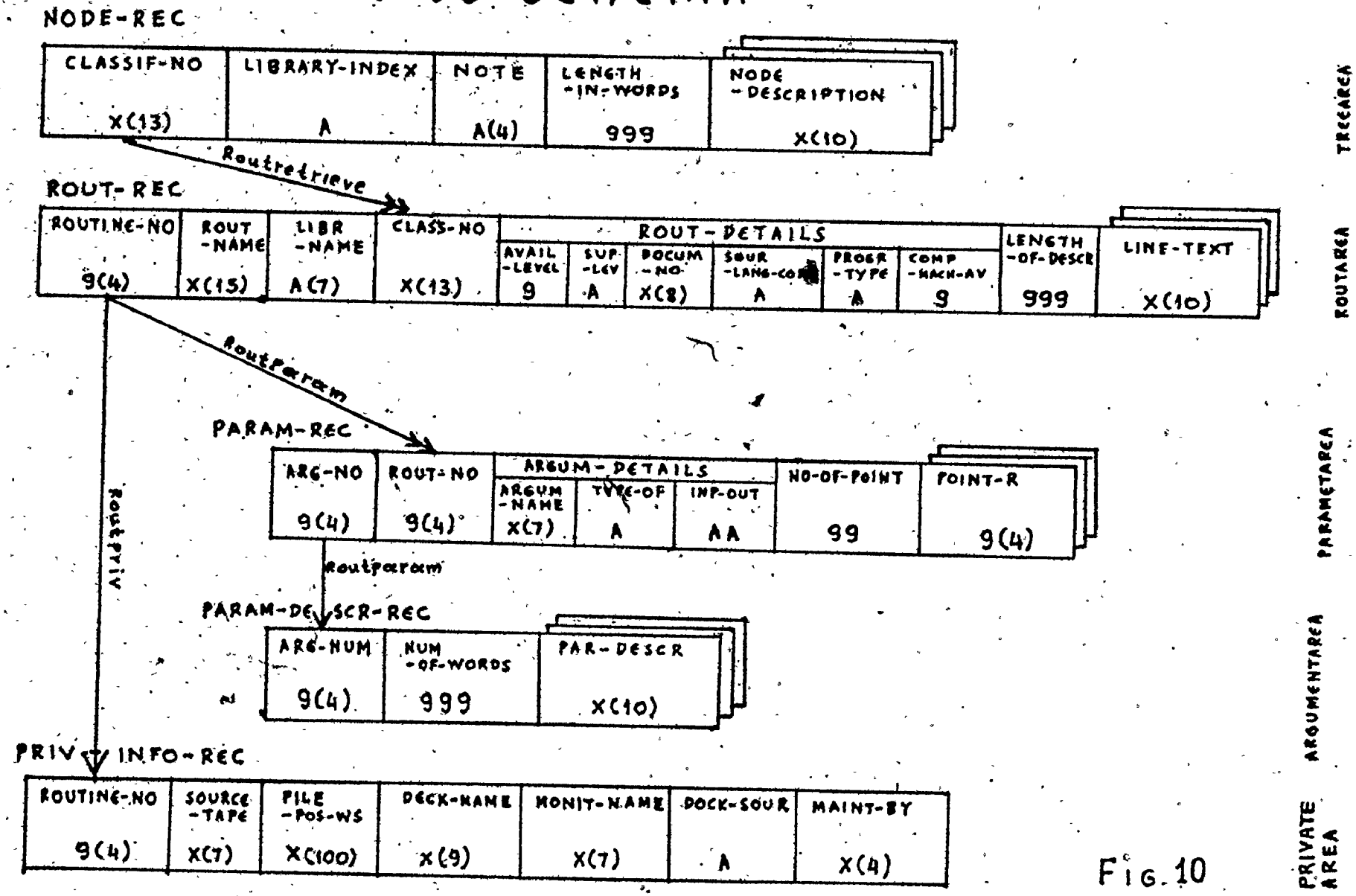


Fig. 10

TRECAREA
ROUTAREA
PARAMETAREA
ARGUMENTAREA
PRIVATE AREA

the Data Base, then one sub-schema can describe these data and can be accessed by any number of application programs using this portion of the data. Strictly speaking the schema is not accessed directly from the application programs, but through sub-schemas. The interface between schema and sub-schema is the CDCS module, part of DMS-170 (fig. 1, ch. I). CDCS intervenes between schema and sub-schema, interrogates schema, converts data, maps etc.

At the programming level, sub-schema record descriptions replace the COBOL application program record descriptions, in other words the COBOL application program DATA DIVISION concerning (describing) the Data Base files involved is included in the sub-schema which the particular application program accesses.

2.4 Sub-schema Compilation:

When the DDL-Compiler compiles a sub-schema it stores it in a LIBRARY; (fig. 11). This LIBRARY is automatically created when the first sub-schema is compiled and saved. Any other sub-schema after the first is ADDED to the LIBRARY.

Any sub-schema may be deleted by PURGE command of DDL from the LIBRARY.

Any sub-schema can replace another in the same LIBRARY.

2.4.1 Data Security:

In the DMS-170 modular Data Base data security is achieved not through passwords, but through different sub-schema LIBRARIES.

THE SUB-SCHEMA LIBRARY

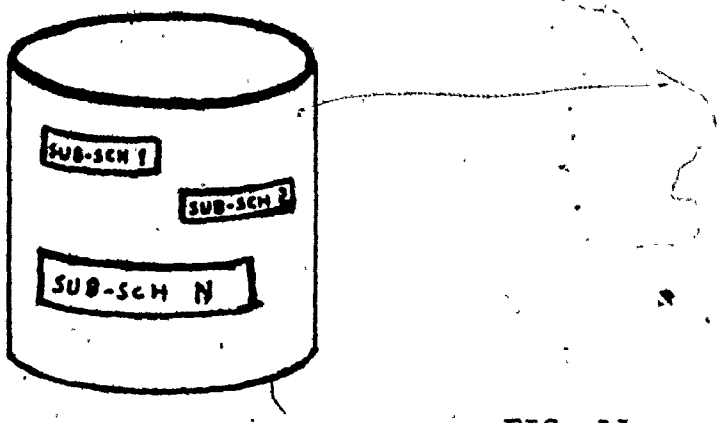


FIG. 11

In case we want to secure files from user access, then we create a separate LIBRARY, of which the name is used as a password and store on this LIBRARY the sub-schemas describing the Data Base portions (areas) which should be for private access. Sub-schemas on another sub-schema LIBRARY different from the one some application program accesses are not available to this program.

2.4.2 The Data Base creation sub-schemas:

The fact that not all files are available for the time being except the two first (Treearea, Routarea) and for training purposes, on sub-schemas I used 4 sub-schemas (instead of 2, one for all the files but the private and one for the private) for creation and one sub-schema for Retrieval.

The Creation sub-schemas for all the files except the private, were stored on a separate LIBRARY.

The sub-schema for creation and Retrieval of the private file, for security purposes, as has been discussed above, has been stored on another LIBRARY.

The logical design of the Creation sub-schemas has been drawn with the presuposition that every file is going to be created separately, so every sub-schema contains 1 file, TREEFILE-CREATIONSUB1, ROUTFILECREATIONSUB2, PRIVATEFILECREATIONSUB4 (fig. 12, 12a).

In PARAMFILECREATIONSUB3 -the case of the third Creation sub-schema for the argument-file -, at the same time the parameter-file (containing the variable length text of the arguments description) is created, so both files are created in one run. This sub-schema will contain both files, but not any relationship. Even though there is not any relation defined in the creation sub-schema of the two files (argument, parameter) which are supposed to be related by a named relationship (Routparam) as the schema indicates, the Data Base CDCS module will interrogate the schema and will join the indicated data items to relate the two files (fig. 13). It would be redundant to redefine a relationship in the sub-schema, though it can be done.

2.4.3 The Data Base Retrieval sub-schemas:

The fifth sub-schema, RETRIEVALSUB5 is stored on separate LIBRARY from the creation sub-schemas.

NODE-REC

CLASSIF-NO	LIBRARY-INDEX	NODE	NO-OF-WORDS	NODE-DESCR
'X(13)'	'A'	'A(4)'	'999'	'X(10)'

——— PRIMARY KEYS

----- ALTERNATE KEYS

Sub-schema for the NODE-FILE

ROUT-REC

ROUTINE-NO	ROUT-NAME	LIBR-NAME	CLASS-NO	ROUT-DETAILS						L-OF-D	TASK-OF-ROUT
				AVAIL-LEVEL	SUP-LEV	DOCUM-NO	S-L-C	PROGR-TYPE	C-M-AV		
'9(4)'	'X(15)'	'A(7)'	'X(13)'	'9'	'A'	'X(8)'	'A'	'A'	'9'	'999'	'X(10)'

SUB-SCHEMA
for the ROUT-FILE

FIG.12 CREATION SUB-SCHEMAS

PARAM-REC

ARG-NO	ROUT-NO	ARGUM-DETAILS			NO-OF-POINT	POINT-R
		ARGUM-NAME	TYPE-OF	INP-OUT		
'9(4)'	'9(4)'	'X(7)'	'A'	'AA'	'99'	'9(4)'

PARAM-DESCR-REC

ARG-NUM	NUM-OF-WORDS	PAR-DESCR
'9(4)'	'999'	'X(10)'

—— PRIMARY KEYS

------ ALTERNATE KEYS

FIG.13

Sub-schema for the ARGUM & PARAM-FILES

PRIV-INFO-REC

ROUTINE-NO	SOURCE-TAPE	F-P-W	DECK-NAME	MONIT-NAME	POC-SOUR	MAINT-BY
'9(8)'	'X(7)'	'X(100)'	'X(9)'	'X(7)'	'A'	'X(4)'

Sub-schema for the PRIVE-FILE

FIG.12a CREATION SUB-SCHEMAS

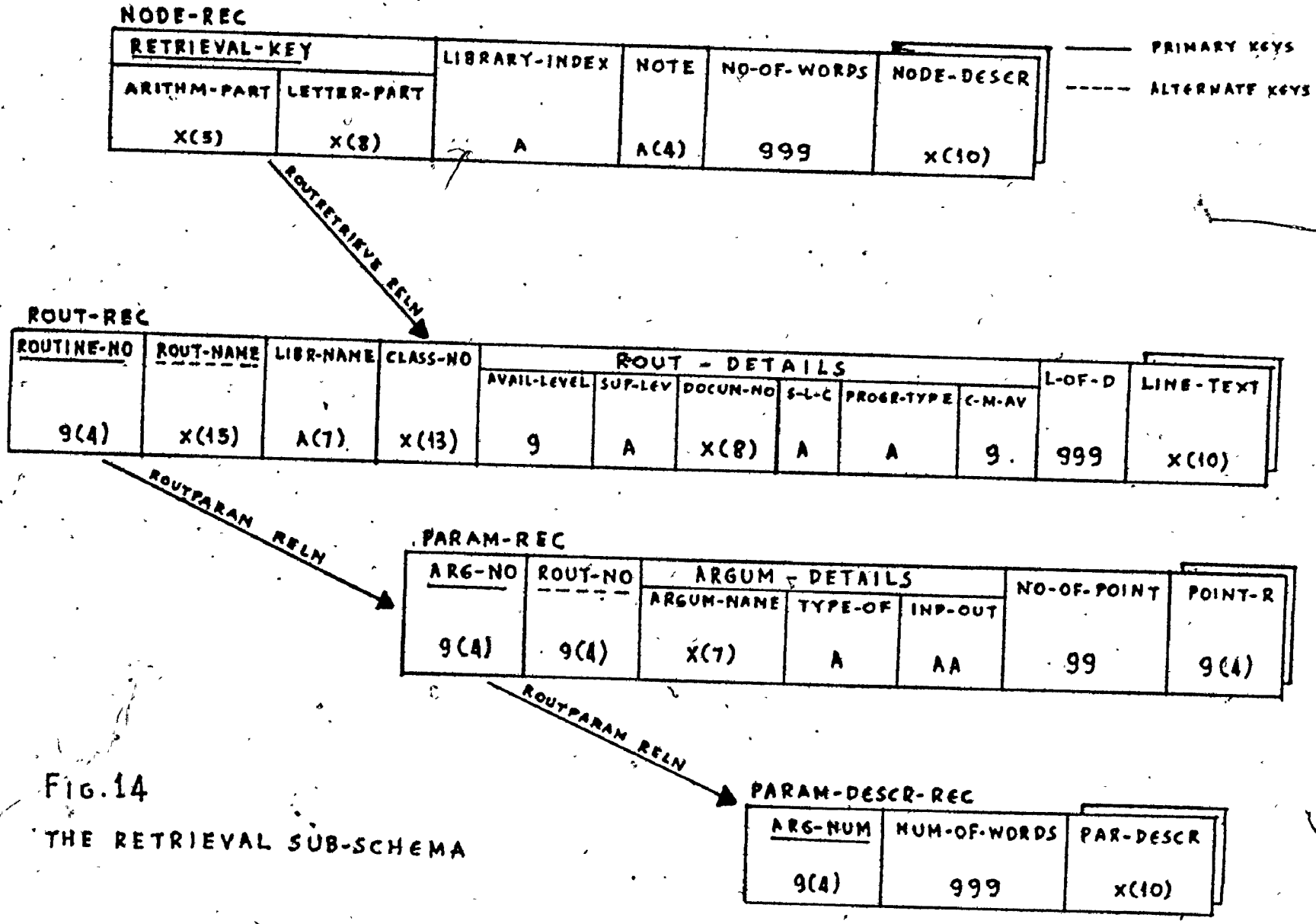


FIG. 14

THE RETRIEVAL SUB-SCHEMA

This sub-schema retrieves information from all files but the private, it will be exactly the same as the schema, with the only difference that the private file and the relationship ROUT-PRIV between Rout-file and Private-file will be absent. All the other relations will be indicated, as been used for the retrieval (fig. 14).

A Retrieval sub-schema for the private file has not been drawn.

Distribution of the sub-schemas between the different LIBRARIES

LIBRARY NAME	SUB-SCHEMA NAME
SUBSLIB (pfn:CYL2CRS)	-TREEFILECREATIONSUB1: sub-schema for the Creation of the NO-DE-FILE.
	-ROUTFILECREATIONSUB2: sub-schema for the Creation of the ROUT-FILE.
	-PARAMFILECREATIONSUB3: sub-schema for the Creation of the ARGU-MENT-FILE and PARAMETER-FILE.
PRIVLIB (pfn:CYL3PRS)	-PRIVATEFILECREATIONSUB4: sub-schema for the Creation of the PRIVATE-FILE.

: sub-

schema for the retrieval has not been drawn yet.

RETRLIB (pfn:CYL4RTS)

-RETRIEVALSUB5: sub-schema for the retrieval of information from all the files of the Data Base, except the private one.

CHAPTER III
IMPLEMENTATION OF THE DATA BASE

3.1 APPLICATION PROGRAMS

After the design of the schema and the several sub-schemas and their coding, the last step is the design of the application programs. These are the programs with which we shall use the Data Base facility. The application programs will be written in COBOL 5. There is another high level language, with which DMS-170 has an interface, the QUERY-UPDATE, which could have been used equally. I prefer COBOL5, because, QUERY-UPDATE is a machine dependent language (for CDC machines), while COBOL can be implemented (with possibly minor modifications) on any machine.

The COBOL5 program accesses the sub-schema which is the interface between it and the CDCS (fig. 15).

Any COBOL application program can refer only to one sub-schema, though a given sub-schema can be referred by any number of application programs. The coding of an application program is the same as a usual COBOL program, with some minor requirements and restrictions. These are found in the CDC-COBOL5 reference manual.

-A required sub-schema clause in the SPECIAL NAMES paragraph names the refered sub-schema.

-The DATA DIVISION does not include FD-entries for DB-fi-

les, because they are defined in the attached sub-schema.

-Two more statements, READ and WRITE relations are included in the repertoire and used in the PROCEDURE DIVISION to handle relations which do not exist in an ordinary COBOL program.

3.2 Creation of application programs:

In our case, the purpose of these programs is straightforward; to read sequential files as input, modify them and create IS-files as output to conform with the file structure and organization defined in the schema. The separate sub-schemas for every file, require one creation application program for every file. Since the format (record structure) of the supplied input files is not exactly the one which we have defined in our schema and sub-schemas, rearrangements and modifications have to be done by an application program, prior to creation of the Data Base IS-files.

One of the main purposes of the Data Base facility, as has been mentioned in Ch. I, is the independence of application programs from the structure of the existing data. To achieve this, and keep creation application programs independent and as simple as possible; instead of creating the Data Base files directly from the existing ones, we use an intermediate stage. We create temporary sequential files with the required format, to conform with the sub-schema definition, and from them we create the required Data Base IS-files. So UTILITY-COBOL programs will

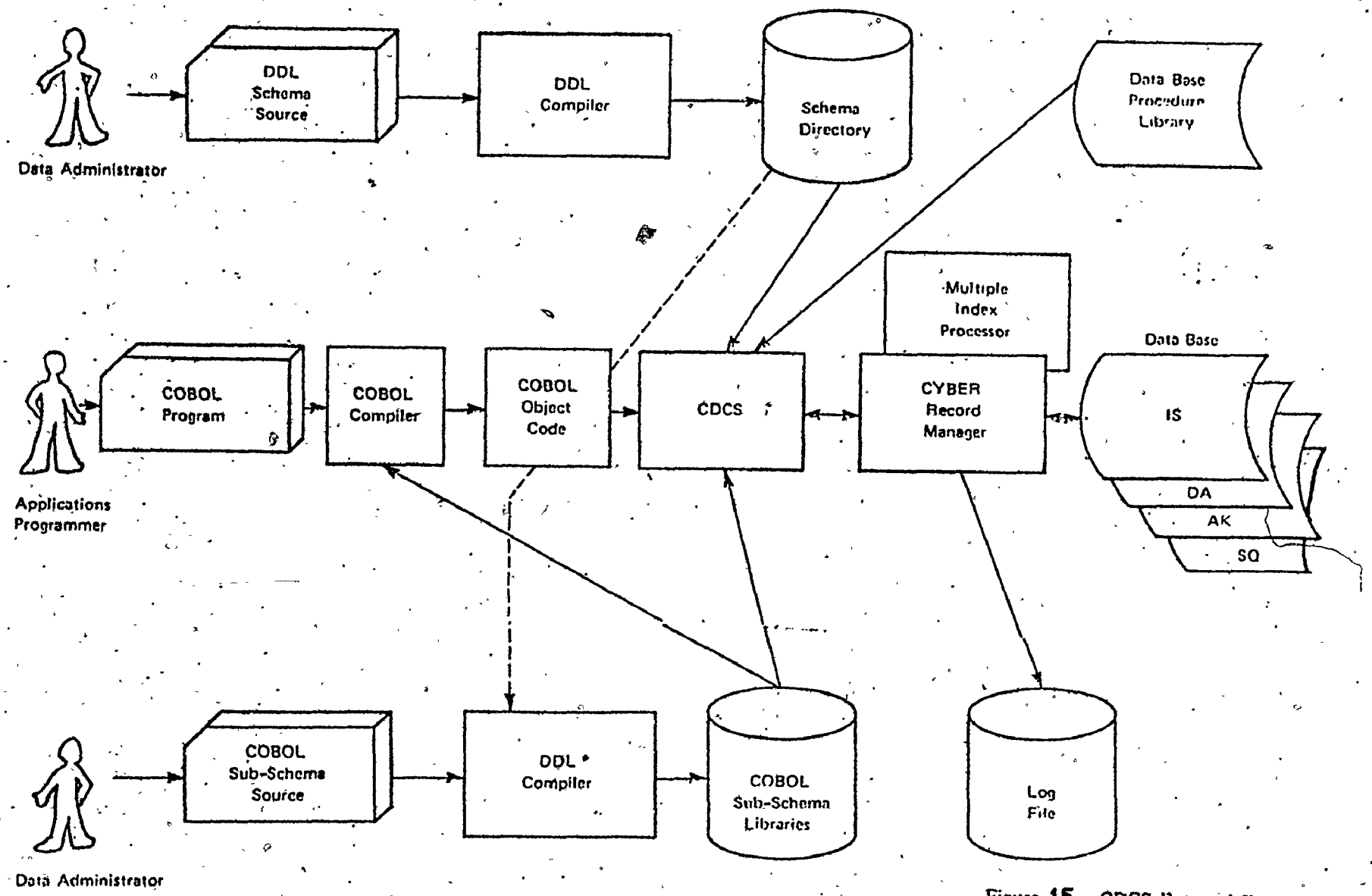


Figure 15 CDCS External Environment

make the rearrangements and modifications to the existing files, prior to creation application programs (fig. 16). So the creation application programs will only be used to transfer the records from the temporary files to Data Base files, record by record.

3.2.1 UTILITY PROGRAMS:

These COBOL5 programs will access the existing SQ-files containing the data, modify and rearrange the records to the required format defined in the schema and sub-schema and also sort the files to the sequence we prefer. The resulting output files will be stored as SQ-files which will be used as Input files to the db-creation application programs.

3.2.2 Temporary NODE-FILE construction:

Classification code modification: As has been discussed earlier, instead of left-list layout we shall use level-by-level layout, by prefixing the level number in front of the existing classification code. In this process, we define all the roots of the existing trees to be of Level-0.

The classification code item has been constructed to be a concatenation of 4-digits (identifying a particular tree, out of the forest of LIBRARY trees) and an alphabetic part filled up with spaces to a maximum of 8 characters (tree depth is 8 levels) to indicate the successive levels of a tree (Ch. II, fig. 4).

According to the above definition the Level-0 of a tree will have all spaces catenated to the digits, the Level-1 will have one letter and spaces catenated and so on. Based on that

fact, our program in order to find the Level number of a node, must count the number of catenated letter on the right of the 4-digit part, and assign that count as a Level number. Since there is a maximum of 8 character positions in the item, it is easier to count the number of spaces and calculate the Level number from the relation:

8 - number of spaces = number of letters = Level number

and this number (in the range 0-8) will be prefixed in an available 1 character data item ahead of the 4-digit number. To achieve this, we REDEFINE the classif-no data item as a group (fig. 17) and use the INSPECT verb of COBOL5 to count the number of spaces to the right of the 4-digit part. The resulting file, before it is stored as a temporary file, is sorted with respect to the new classif-code data item; after sorting, the records have been arranged in Level-by-Level sequence and thus they are accessed by our Retrieval application program.

During the creation run the classif-code is accomodated as a PIC X(13) data item (defined at the sub-schema).

The system chart of the whole procedure is shown in fig. 18.

3.2.3 Temporary ROUT-FILE constraction:

For the Routine-file, instead of using as a primary key the concatenated routine-name and classif-code, as normalization procedure requires, we have decided (as has been discussed in Ch. II) to create a new key, the Routine-number which uniquely iden-

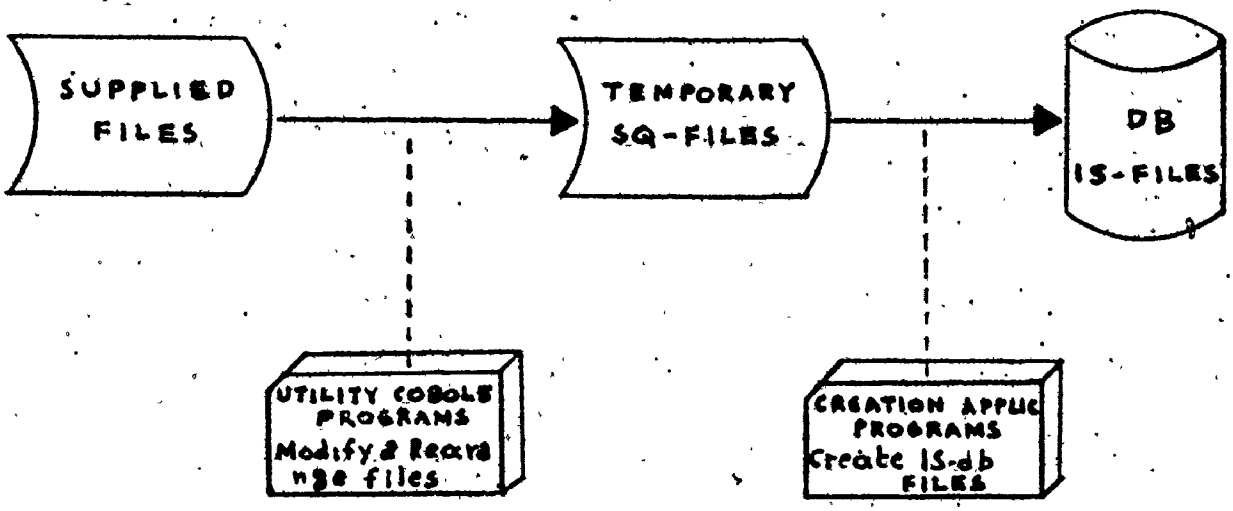


FIG.16

tifies each routine-record.

Here our UTILITY program CREATE-ROUTFI-TEMP2 accesses as Input the supplied file PROGPAP (by Program Librarian) modifies the classif-code, which is here the joining item to the NODE-FILE, by the same procedure as has been used for the NODE-FILE.

The only difference is that the ROUT-FILE, will not be sorted with respect to the classif-code item as was the NODE-FILE: it will remain sorted in the alphabetic order of the Routine-names, that is, the initial sequence (PROGPAP sequence) will not be altered.

During the run we shall also create the primary key of the file. As we transfer the records from the PROGPAP file to the TEMP2 file, a counter in steps of 10 digits will insert its count in a newly created data item; ROUTINE- NO, to be used as

a primary key.

With the 10 digit interval between two successive records we provide for future insertions of routines. With this technique also, the ROUTINE-file is ordered automatically in ascending order of its primary key. Nothing else special is done with this file. The Creation application programs are straight forward and the program comments in the listing are enough to explain what is done.

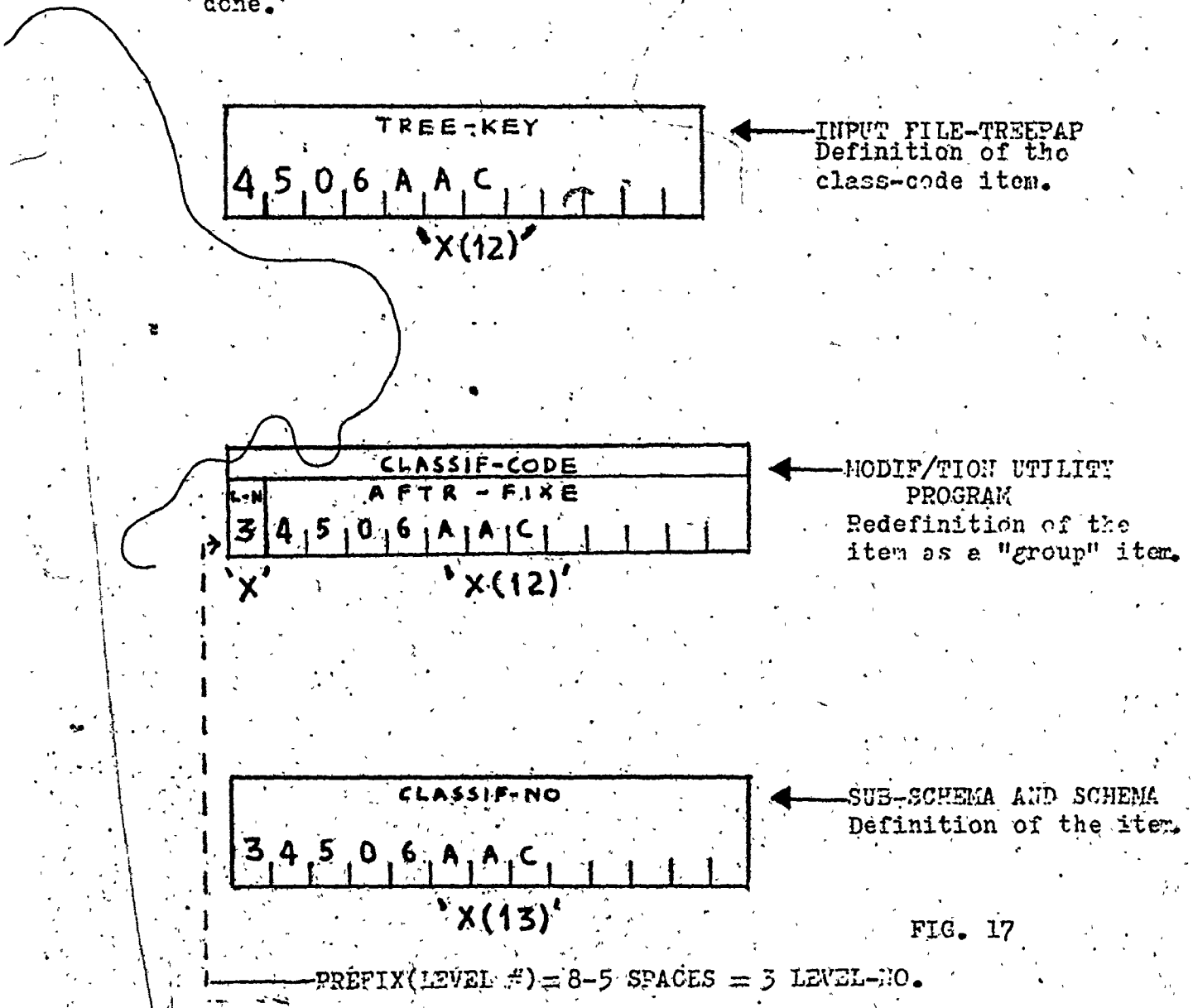


FIG. 17

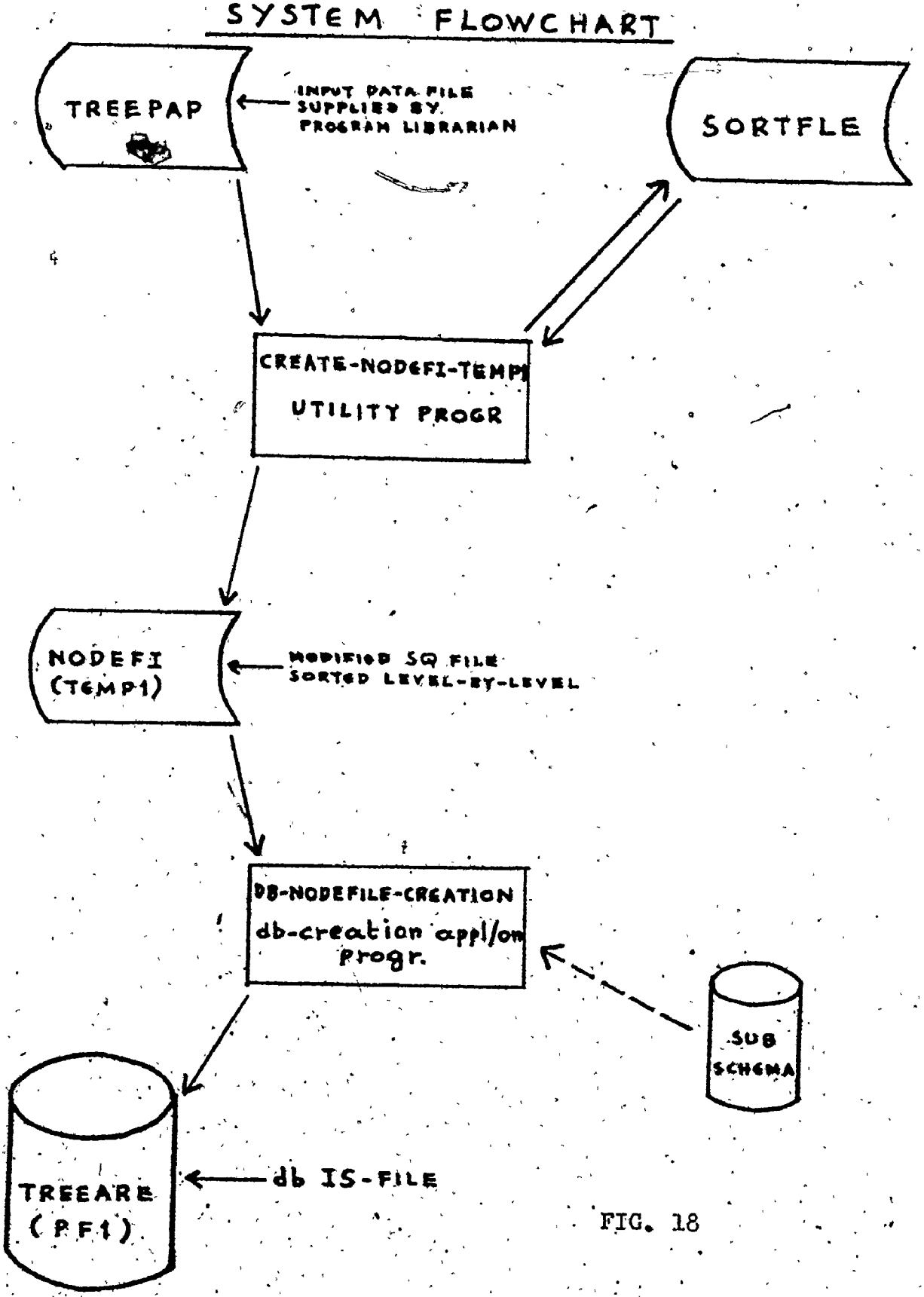

SYSTEM FLOWCHART
TREEPAP
INPUT DATA FILE SUPPLIED BY PROGRAM LIBRARIAN
SORTFLE
CREATE-NODEFI-TEMP1
UTILITY PROGR
NODEFI (TEMP1)
MODIFIED SQ FILE SORTED LEVEL-BY-LEVEL
DB-NODEFILE-CREATION
db-creation appl/on progr.
SUB SCHEMA
TREEARE (PF1)
db IS-FILE

FIG. 18

3.3 RETRIEVAL APPLICATION PROGRAM:

This is the most important and most complicated program of the project. It is an interactive COBOL5 program which carries out the dialogue with the user.

By successive interrogation, the program using the user's selection as a guide, builds up step-by-step the search key, to access and retrieve only those records from the file, which are required in the user application The procedure is explained in Ch. II.

The Retrieval program documentation explains in detail almost exhaustively, what every paragraph in the PROCEDURE DIVISION does. A Flowchart of this program is given in fig. 19-19c.

The most complicated techniques of this program will also be discussed here.

Group-choice: Professor K. A. Redish, who handed to me the Library trees, had already divided them into distinct groups, every group containing relevant subjects, based on the root (of every tree) subject description.

The first choice of the User is one of the displayed groups, which will include the tree which will contain the subject of its interest in one of its nodes.

When the User chooses a group, the system must display the roots (0-Level) of all the trees of this group, so the USER may be able to choose the root of the tree which most likely contains its subject. In case of wrong selection, the "Backup"

facility, explained later, will allow him to go back and choose again. If, for example, his subject is "Matrix transposition", the root "Matrices" has to be chosen.

To do this in the program, the lowest and the highest class-code item (key of the file) of 0-Level of every group have been stored as constant data items, so according to the group choice, the lowest class-code is moved to become the current key of search with which we start the file randomly. Then, since the keys are arranged in ascending order and the highest key of the group is also known, a seq/al reading retrieves all the records of the group, one by one, and the key of each one before being displayed is compared to the highest key (last record of the group) of the group, which is the condition to terminate the seq/tial reading, so reading ends when only the records of this particular group are desplayed.

Key-validity facility: Every time a record is displayed, its key (class-code) is saved in WORKING-STORAGE table CHECK-TABLE in successive locations CHECK-ITEM(I) with the statement:

MOVE CLAS-CODE OF NODE-LINE-REC TO CHECK-ITEM(I) in the CRT-DISPLAY-PAR.

Since there are possibilities of mispunching by the user when he chooses one of the displayed keys, the entered key before it becomes the current search key, is compared with all the saved keys in KEY-VALIDITY-PAR. If it is one of them, the system proceeds to access the next level records, else a message is displayed to the user to re-enter again one of the displayed keys.

THE FLOWCHART OF THE RETRIEVAL PROGRAM

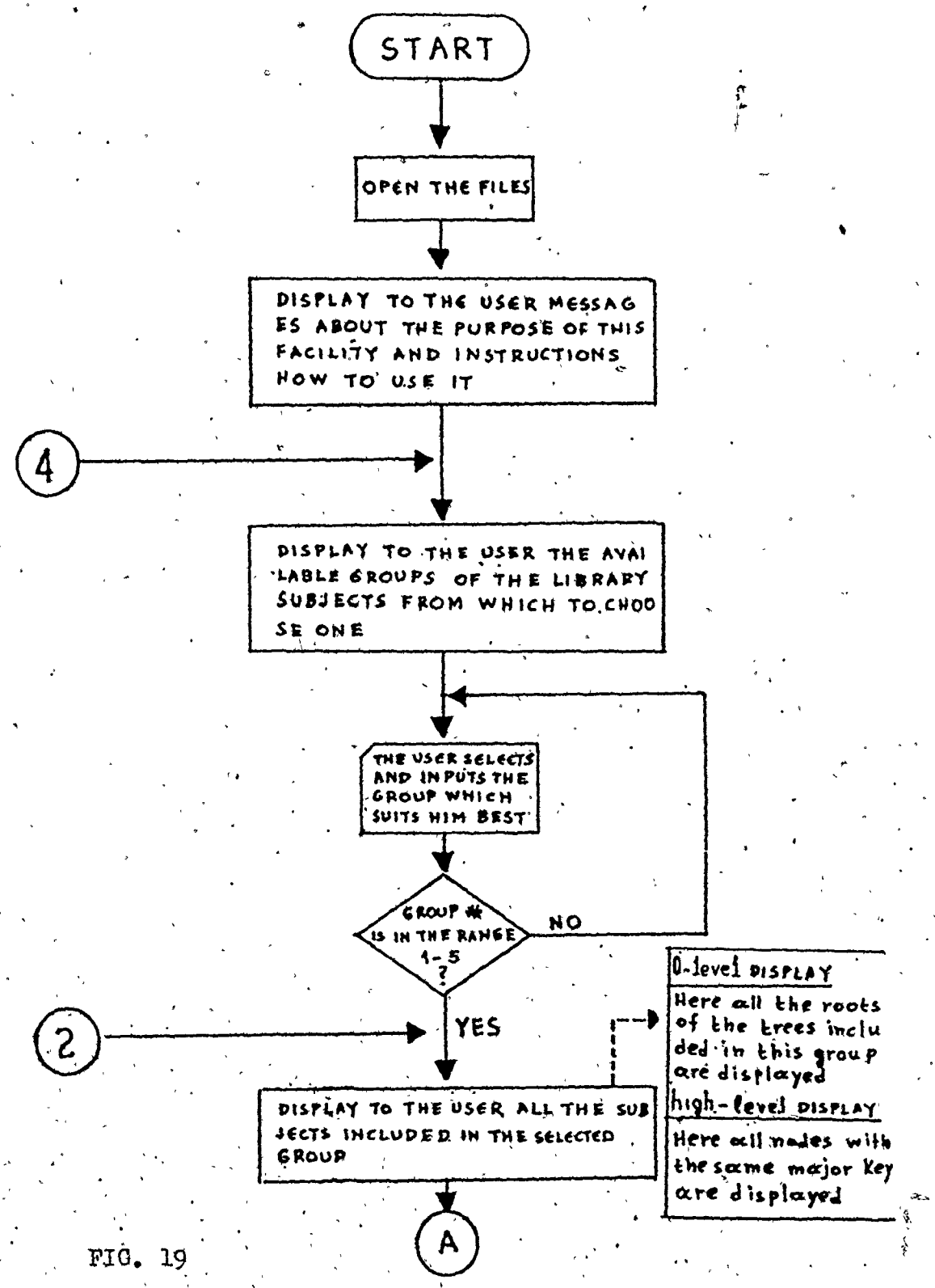


FIG. 19

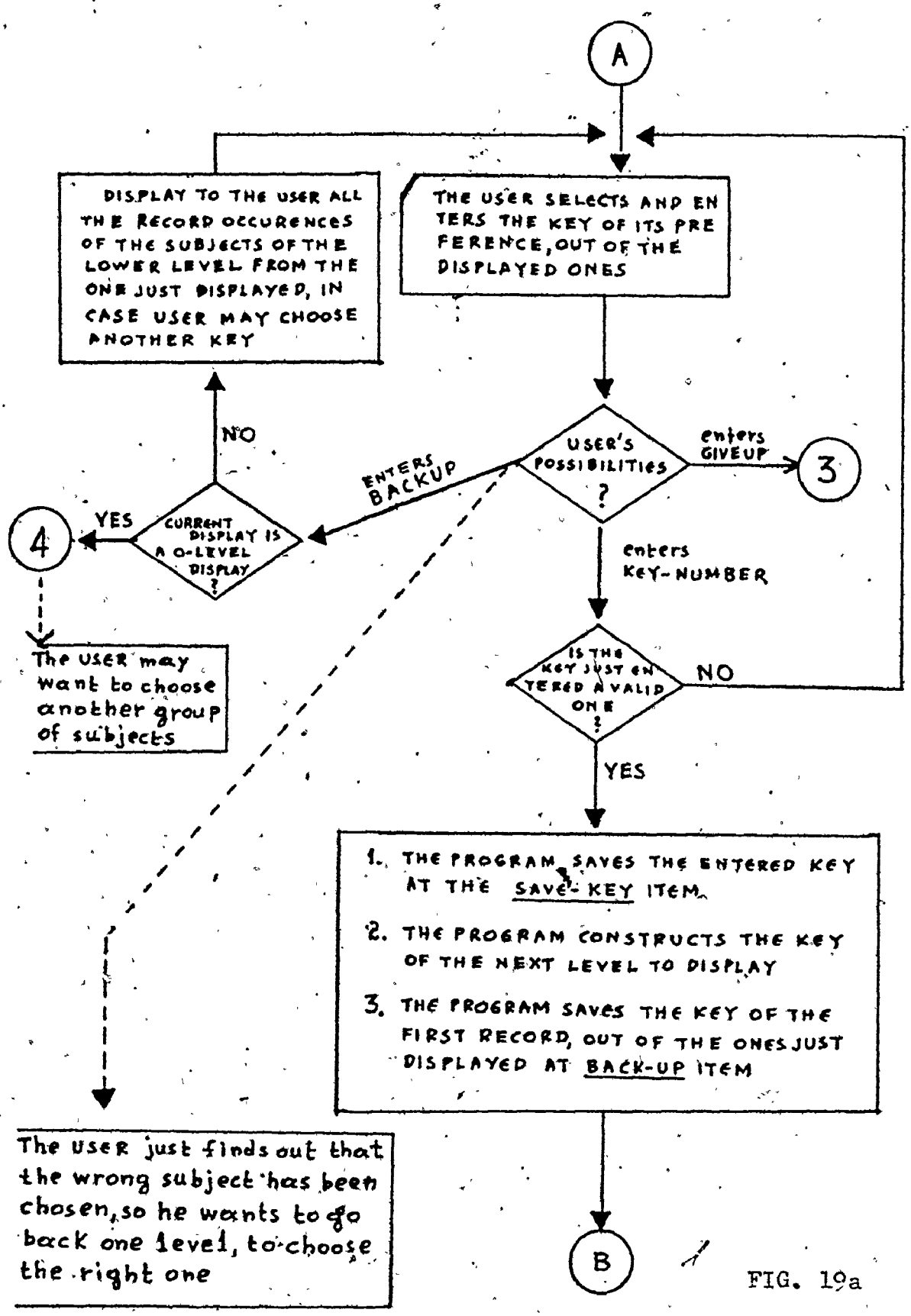

A
DISPLAY TO THE USER ALL THE RECORD OCCURENCES OF THE SUBJECTS OF THE LOWER LEVEL FROM THE ONE JUST DISPLAYED, IN CASE USER MAY CHOOSE ANOTHER KEY
THE USER SELECTS AND ENTERS THE KEY OF ITS PREFERENCE, OUT OF THE DISPLAYED ONES
NO
USER'S POSSIBILITIES ?
enters GIVEUP
3
ENTERS BACKUP
YES
4
CURRENT DISPLAY IS A 0-LEVEL DISPLAY ?
enters KEY-NUMBER
The USER may want to choose another group of subjects
IS THE KEY JUST ENTERED A VALID ONE ?
NO
YES
1. THE PROGRAM SAVES THE ENTERED KEY AT THE SAVE-KEY ITEM.
2. THE PROGRAM CONSTRUCTS THE KEY OF THE NEXT LEVEL TO DISPLAY
3. THE PROGRAM SAVES THE KEY OF THE FIRST RECORD, OUT OF THE ONES JUST DISPLAYED AT BACK-UP ITEM
The USER just finds out that the wrong subject has been chosen, so he wants to go back one level, to choose the right one
B

FIG. 19a

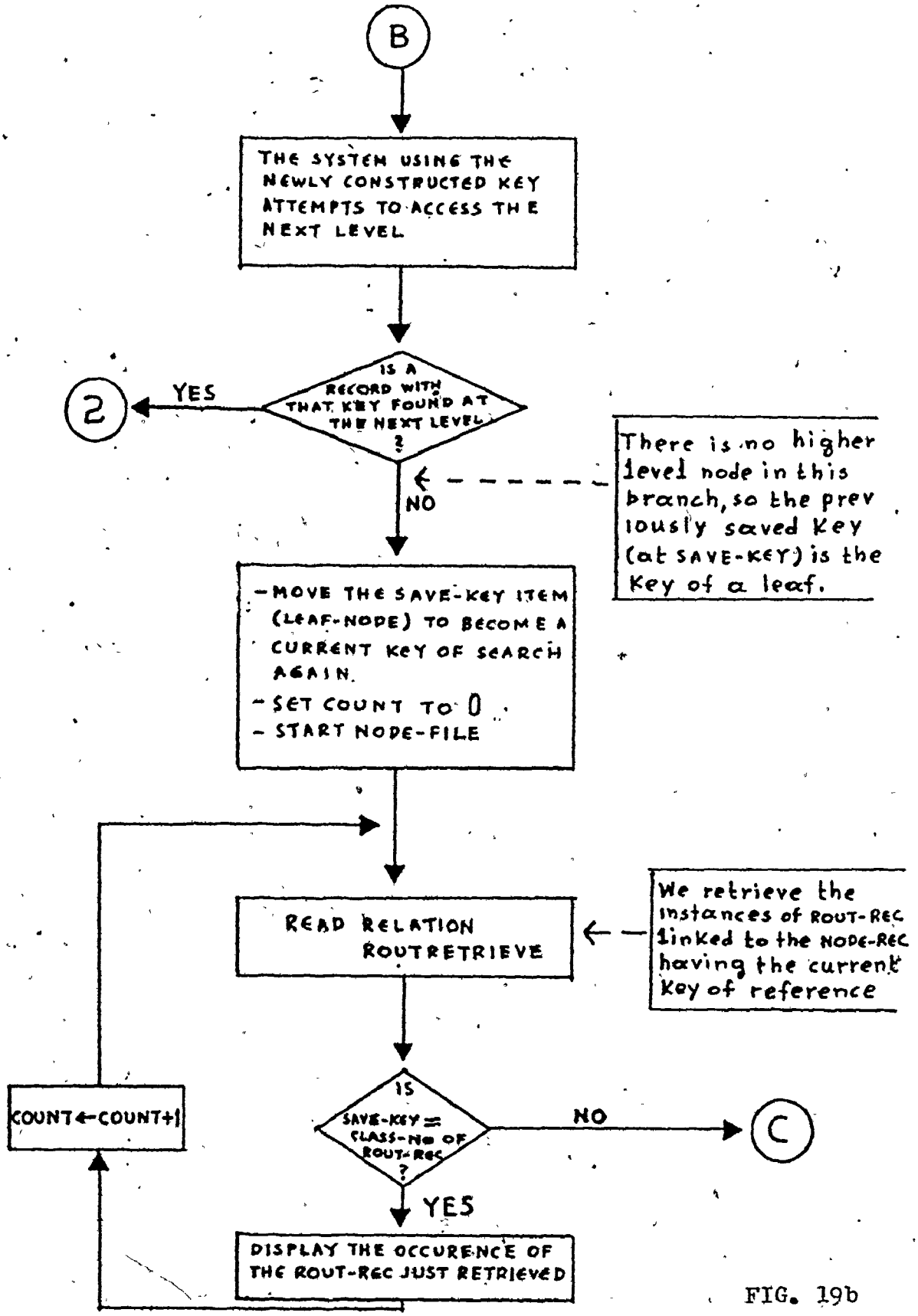

B
THE SYSTEM USING THE NEWLY CONSTRUCTED KEY ATTEMPTS TO ACCESS THE NEXT LEVEL
IS A RECORD WITH THAT KEY FOUND AT THE NEXT LEVEL ?
YES
2
NO
There is no higher level node in this branch, so the previously saved key (at SAVE-KEY) is the key of a leaf.
- MOVE THE SAVE-KEY ITEM (LEAF-NODE) TO BECOME A CURRENT KEY OF SEARCH AGAIN.
- SET COUNT TO 0
- START NODE-FILE
READ RELATION ROUTRETRIEVE
We retrieve the instances of ROUT-REC linked to the NODE-REC having the current key of reference
IS SAVE-KEY = CLASS-No OF ROUT-REC ?
NO
C
YES
COUNT ← COUNT+1
DISPLAY THE OCCURENCE OF THE ROUT-REC JUST RETRIEVED

FIG. 19b

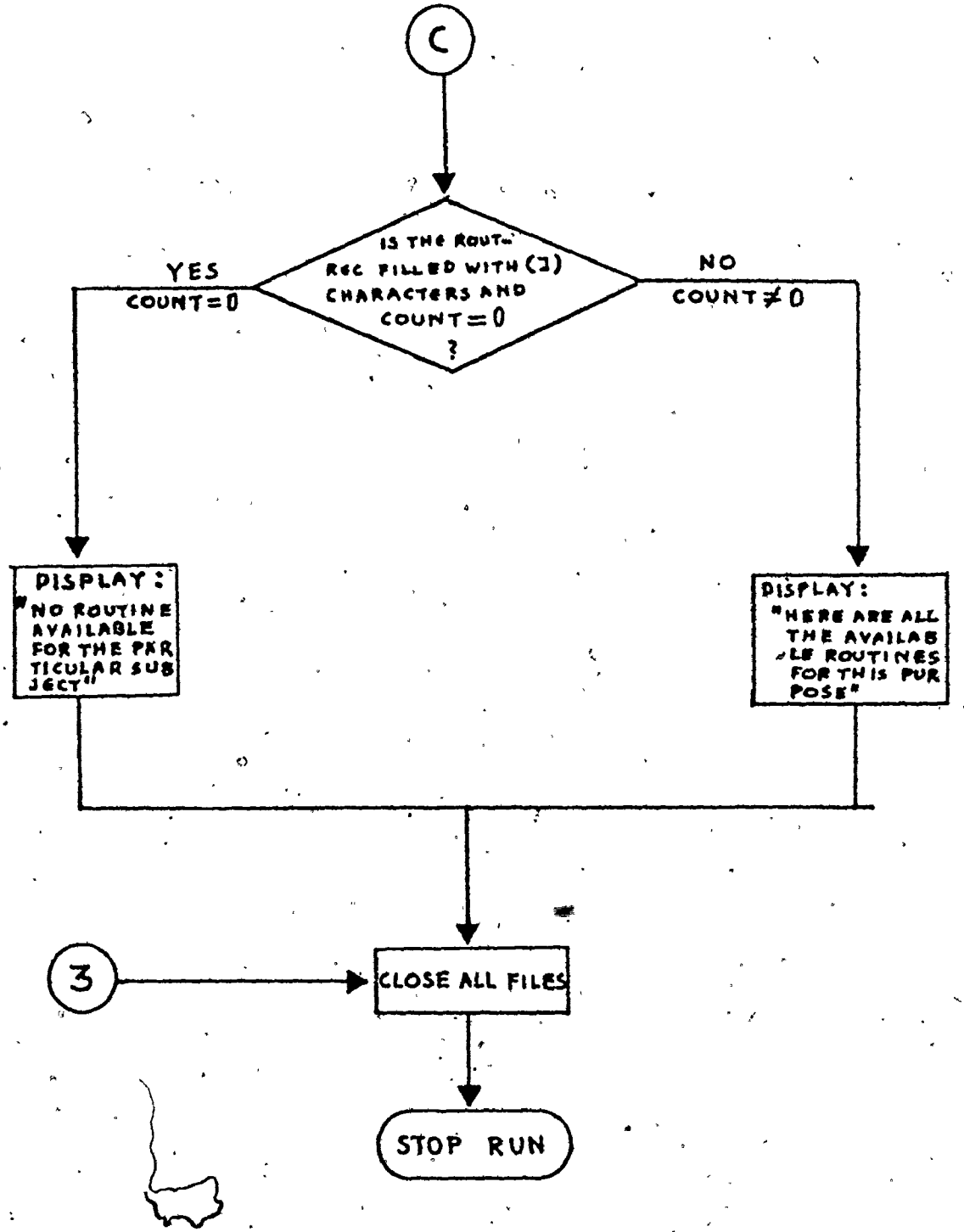

C
IS THE ROUT" REC FILLED WITH (1) CHARACTERS AND COUNT=0 ?
YES
COUNT=0
NO
COUNT≠0
DISPLAY: "NO ROUTINE AVAILABLE FOR THE PARTICULAR SUBJECT"
DISPLAY: "HERE ARE ALL THE AVAILABLE ROUTINES FOR THIS PURPOSE"
3
CLOSE ALL FILES
STOP RUN

FIG. 19c

Every time, only the most recently displayed keys are saved in CHECK-TABLE, overwriting the previously saved keys.

Backup facility: Any time the user selects a key which mostly suits him, and the display starts, the class-code key of the first of the displayed records is saved in the BACKUP-KEY data item with the statements:

MOVE LEVEL TO ITEM-1

MOVE CHECK-ITEM(1) TO ITEM-2 in the NODE-CHOICE-PAR.

The BACKUP-ITEM at any time accomodates the key of the first of the currently displayed records.

When the next higher level records are displayed, if the user finds out that he had made a wrong choice previously and wishes to choose again another key of the previous level, he just enters the keyword "BACKUP". The program then branches to DECISION-PAR.

If the previously displayed nodes were the roots of the trees (0-Level display) of a particular group, that is the user had chosen the wrong group, then the control is transfered to DIALOGUE-PAR to redisplay the available groups, so the user will choose another group.

If the previous display was Level-1 display, that is, the user had chosen the root of the wrong tree out of the trees of the group; then control is transfered to ZERO-LEVEL-PAR to redisplay the roots, so the user will choose another root; else the previously saved BACKUP-KEY becomes the current key of reference to start the node-file again at that record and the sequential

reading redisplays all the records from that node on. Since only the key of one previous level is saved at a time, the user cannot go more than one level back; in such case he must 'GIVEUP' terminating the job and logging in again from the beginning.

Routin-retrieval technique: Any time the user selects the key of his preference and after the key has been checked for validity (it is one of the displayed keys) control is transfered to PART-FOUND paragraph. There, before the level-number is modified by the statement:

ADD 1 TO LEVEL

so that, the chosen key becomes the new current key of search, the chosen key is saved in the SAVE-KEY item. Then, the next level is accessed. The user (or the system) does not know in advance if there is a next higher level for this record, that is, if the request may be further resolved (in other words, at the access time of each node it is not known if that particular node is a leaf or not).

Therefore, as the access arm, directed by the program, searches the higher level nodes randomly trying to locate the current key of search in the START-PAD, if it can not locate such a key, then this means that the previous node was a leaf and no higher level exists for this branch. In that case, the INVALID KEY option is activated transfering control to BACK-UP paragraph.

What is done here is obvious, the saved key from the

previous level becomes again the current key of reference with the statement:

MOVE SAVE-KEY TO RETRIEVAL-KEY

and the access arm is now positioned in NODE-FILE at that record (START NODE-FILE). Control is then passed to the READ-RELATION-PAR, where use is made of the Data Base RELATION facility.

The START statement in the BACK-UP PAR has already positioned the root file (here NODE-FILE) at the particular record, before the first sequential relation is read.

Then the READ ROUTRETRIEVE NEXT RECORD retrieves sequentially all the routine records joined to the node-file record (just positioned) having the class-code equal to save-key (classif-code of the leaf node).

SAVE-KEY data item, at this particular time, stores the class-code of a leaf. If there is no routine for the particular subject, then no routine record will be joined to the particular node-rec and the system (CDCS) will submit a NULL record occurrence to the buffer.

A NULL record consists of a display code right bracket ([ ) in each character position. If this NULL record is presented with the first access of the ROUT-FILE, then the data item PRLN-CNTR which stores the number of accessed routines and is advanced by one for every routine-rec retrieved will be equal to (0) if it is tested, so indicating that no routine-rec has been retrieved. In this case the message 'NO AVAILABLE ROUTINES FOR

THIS PURPOSE' will be displayed.

If REEN-CNTR≠0 when a NULL record has been encountered, it means that all related routine records have been retrieved and there are no more for this subject. In that case the message 'HERE ARE ALL THE AVAILABLE ROUTINES FOR THIS PURPOSE' will be displayed.

Choice of displayed records: The paragraph which chooses the records qualifying to be displayed, out of the records read, is the REC-DISPL-PAR which is called by READ-PAR when a condition for display is [redacted].

The technique used for that, "the chopping technique" (which will become apparent later) uses a number of data items of progressively increased length, differing by one character the one from the other. These items are defined in the WORKING STORAGE (items X-6 to X-12).

This technique works as follows: Using the level-by-level sequence to order the records on the storing media, the same level of every tree is stored in ascending order of the tree root number (fig. 20). Level-0 and Level-1 nodes read by the system are all qualified to be displayed. The Level-0 of a particular group are the roots of the included trees from which one has to be chosen. The whole Level-1 of a tree has to be displayed, so the user will choose a particular node. In Level-2 to Level-8 the records of every level flourish at a high rate and since not all of them are qualified for display (but only the children of the previously chosen node) as some of them have

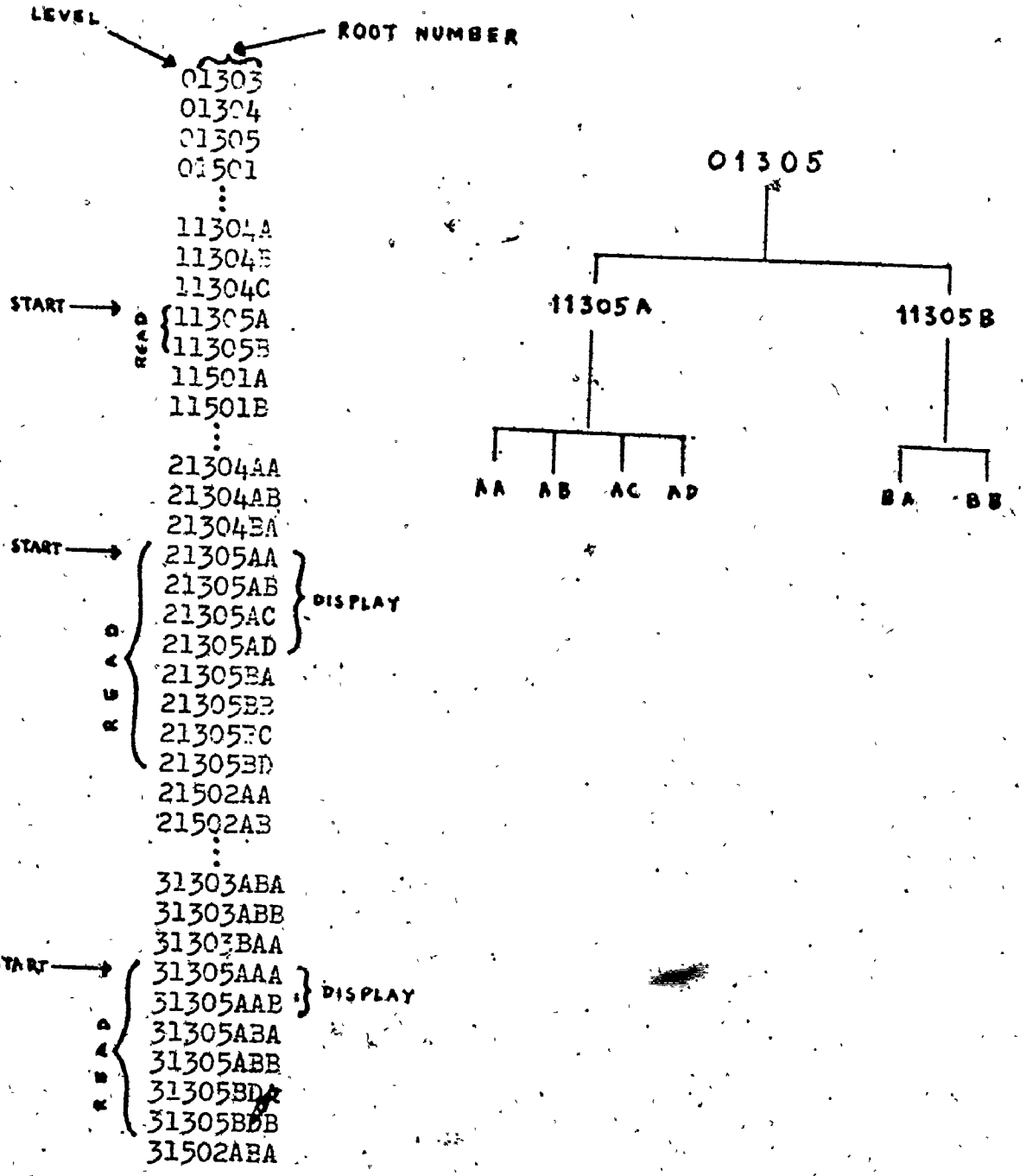


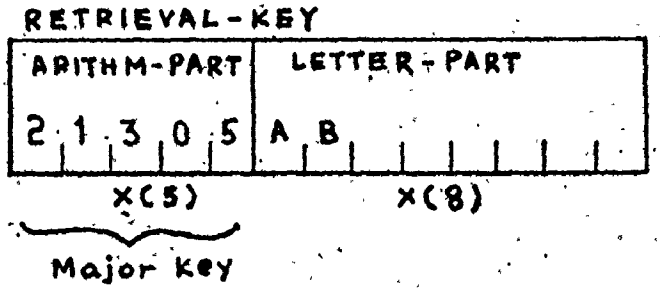
FIG. 20

Classif-Code: is ordered at any level in ascending order of the root number part.

been eliminated by the previous level choices, it will be useless to display them.

The primary key of search, RETRIEVAL-KEY is constructed step-by-step by the program according to the user's request and varies in length, becoming longer and longer as we proceed to higher levels (fig. 20).

The part of the key variable in length is the LETTER-PART which increases by one character at every level.



Definition of the primary key group-item.

FIG. 21

The ARITHM-PART remains constant in length since the 5-digit part identifies the particular tree which has been chosen from the very begining. We therefore define the 5-digit leading part of the key as a Major Key. So to read only records (nodes) of the selected tree, out of the same level of the other trees, we shall start our access arm from the first record which will have arithm-part = major key with the statement (fig. 21):

START NODE FILE KEY = ARITHM-PART in START-PAR (fig.20).

Then we will perform the READ statement until the above condition is not satisfied. In this way the system will read all the nodes (records) of the same level (same major key) of the particular

tree only fig. 20). But when we read the whole level, say Level-2, we do not want to have all the records of this Level displayed, but only the nodes of the branch (children) of which the father node has been previously selected by the user.

If say, the user has selected 11305A, then only the records 21305AA, 21305AB, 21305AC qualify to be displayed out of all read records. Here is where the "chopping technique" is used.

When the user chooses a node, the key is moved to NOMINAL-KEY item and its level (first digit) is increased by 1 and so becomes the major key of search for the next level. The chosen 11305A, becomes 21305A (fig. 22). The access arm is located at 21305AA and reads until encounters 21502AA.

At every successive reading, the RETRIEVAL-KEY which accomodates the key of the most recently read record, is moved to a data item which has length one character less than the number of characters different from space character (Δ) in the RETRIEVAL-KEY. The number of characters different from the space (Δ) character at a particular level-n (n=0-8) is 5+ n. Therefore the item used to chop the last character should have lenght 5+ n-1 = 4+ n at a particular level.

When the Level-2 is read, RETRIEVAL-KEY accomodates 5+ 2 = 7 chs other than space.

The key 21305AB moved to X-6 which has length 4+ 2 = 6 chrs becomes after choping 21305A which when compared with the NOMINAL-KEY containing its father node, qualifies to be displayed

CHOPPING TECHNIQUE

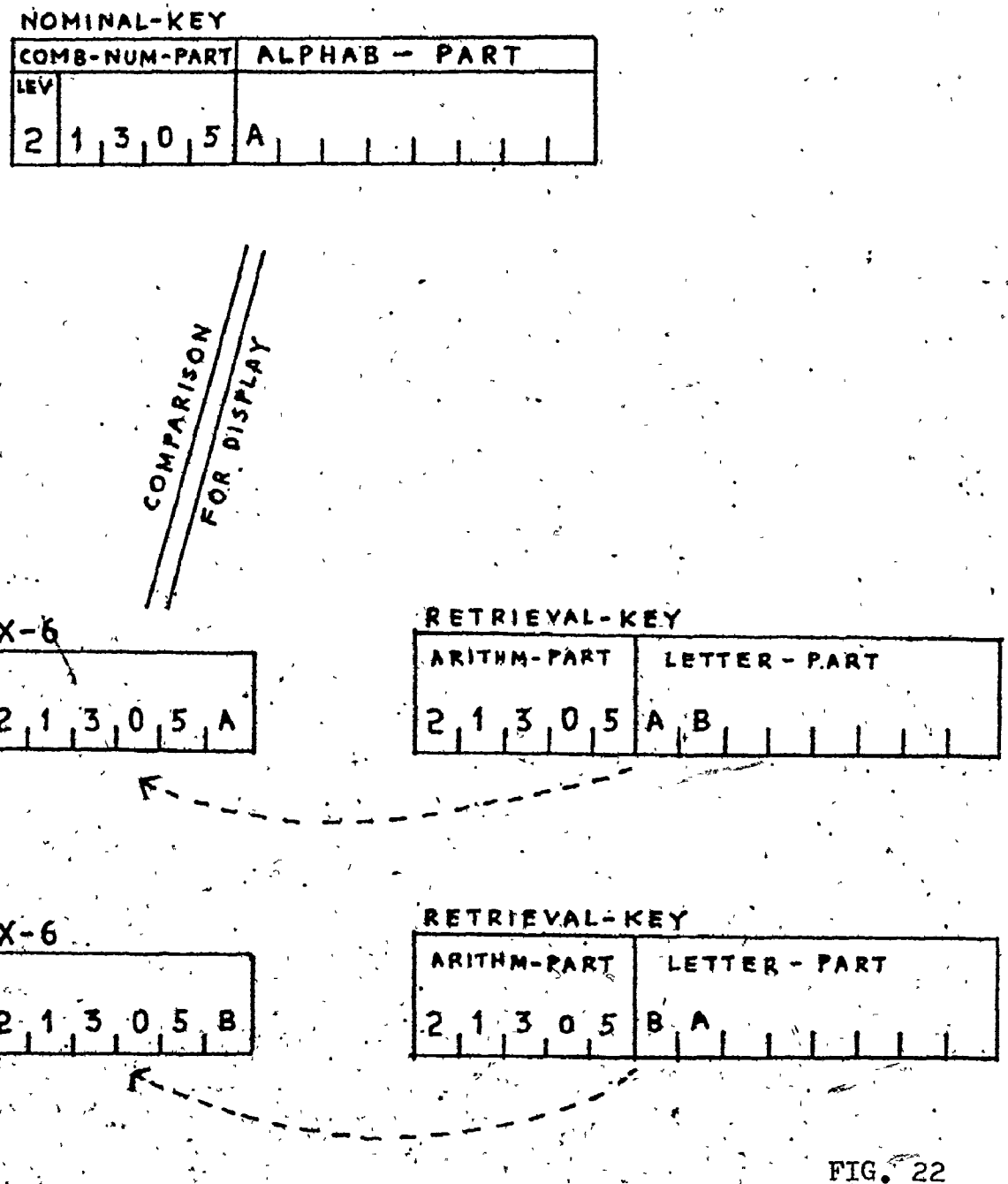


FIG. 22

(fig. 22).

Key 21305BA being moved to X-6, it becomes 21305B after choping and is different from NOMINAL-KEY; so it belongs to another branch and is not displayed.

If the user chooses 21305AA from Level-2, then NOMINAL-KEY will contain 31305AA, and after coping of the Level-3 at X-7 item having length 4+3=7, only records 31305AAA, 31305AAB will qualify for display (fig. 20).

3.4 ASCII-DISPLAY CODE Conversion:

While the information stored on the fixed part of every record is in CDC-display code, that is 10 chrs per word (6-bit per chr), the variable part is in ASCII code.

The input structure of one text word is:

4 bit of zero
8 x 7-bit characters } total 60 bits

and so each item in the Data Base consists of N-words of 8 chrs per word (fig. 23).

For sorting manipulation purposes CDC provides a routine XMOVE (xy, s,d,l) which converts a string of 12-bit chrs in ASCII to EBCDIC or 6-bit display code.

Here:

x,y = A string of 12-bit chrs in ASCII
= C string of 12-bit chrs in EBCDIC
= X string of 6-bit chrs in CDC-Display.

and

xy-parameter is written in 2Hxy form.

s-parameter denotes the start location of the source character string to be moved.

d-parameter denotes the start location of the destination character string.

l-parameter is an integer designating the number of characters in the input string to be moved.

To take advantage of this facility we must first modify the ASCII word format of 8 chrs x 7 bit per word to be in the format of 12-bit characters, that is 5 characters per word in ASCII. So each 7-bit character goes into a 12-bit character in the form:

5 bits of zero
7 bits of character } x 5 60 bits

and fig. 23 shows the mapping.

The FORTRAN subroutine ASCII performs this task using bit manipulation functions.

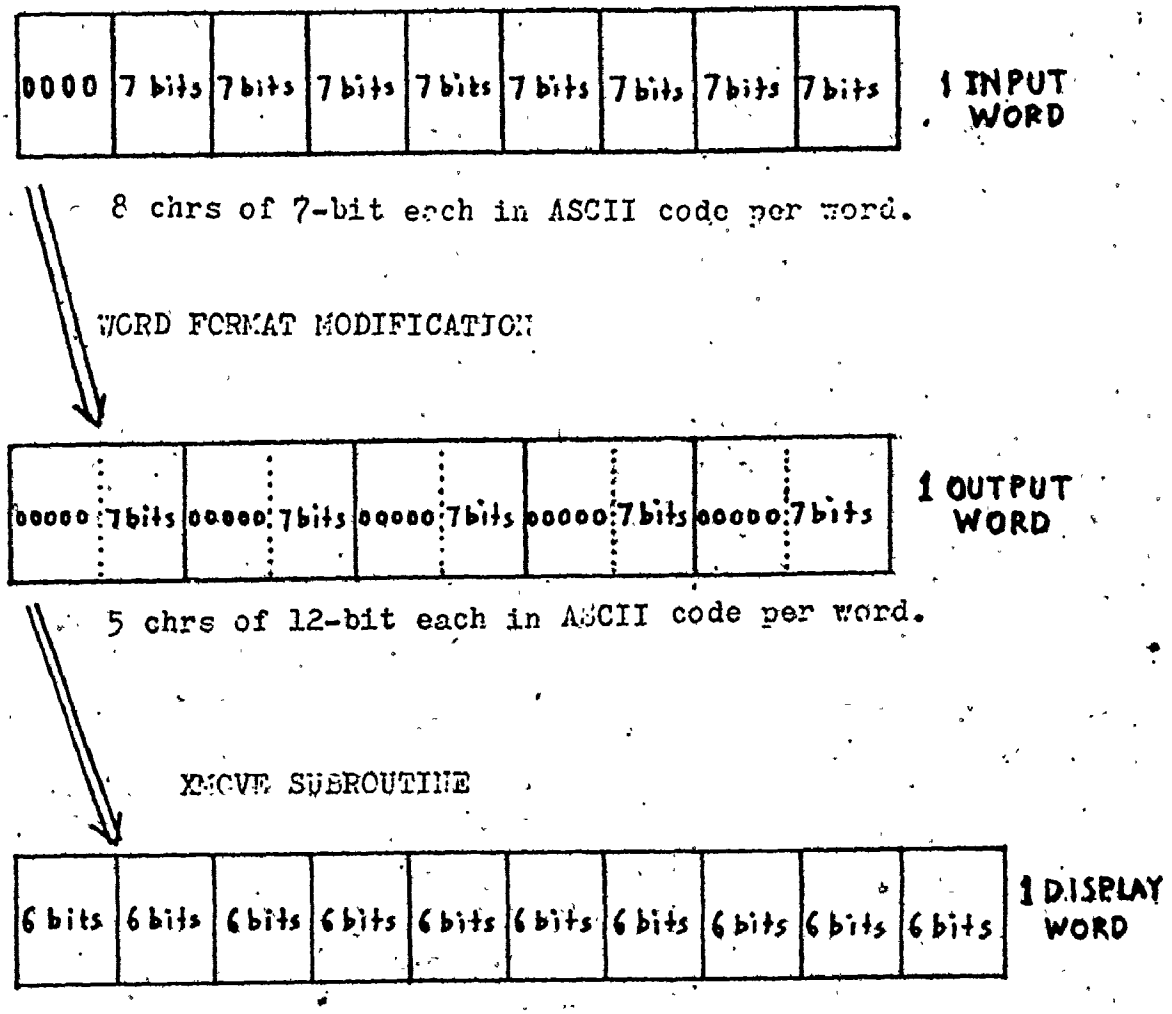
The last step is to convert the 12-bit characters of ASCII to 6-bit characters of CDC-Display code using the subroutine XMOVE.

The Usage is:

CALL XMOVE(2HAX,ISCR,ICUT,OUTCNT)

where

2HAX -denotes the conversion of 12-bit ASCII code to 6-bit CDC-Display code.



WORD FORMAT OF THE VARIABLE ASCII TEXT

FIG. 25

ISCR -is the source array containing the ASCII code.

ICUT -is the destination array which will accomodate the output display code.

OUTCNT -is the integer which counts the number of ASCII characters which will be moved from ISCR array to ICUT array.

The ASCII conversion subroutine is entered through the COZOL program fig. 24.

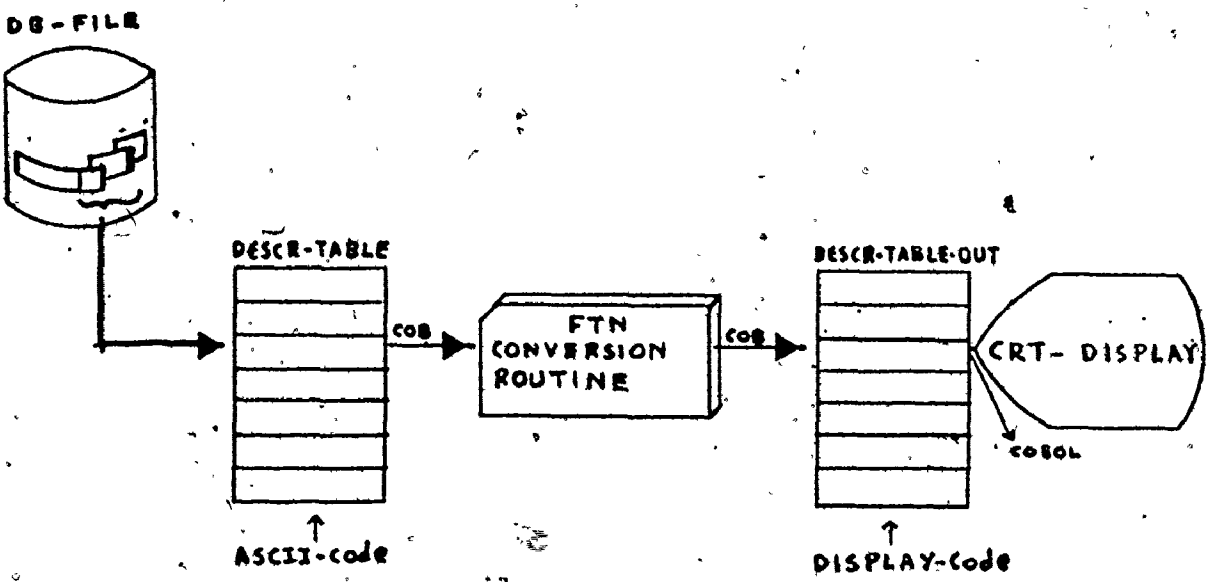


FIG. 24

More details about the functions of the different paragraphs of the Retrieval program are contained in the comments accompanying every paragraph of the program listing in the Appendix A.

C H A P T E R IV

C O N C L U D I N G R E M A R K S

We have mentioned previously that this project will implement an interactive query-answer system, so the retrieval programs were supposed to be used interactively through the different terminals.

Under the NOS1 Operating System the maximum memory allocated to every user at the terminals does not exceed the 67700B words, while the COBOL retrieval program together with all the necessary packages and attached Libraries (Schema, Sub-schema) requires at least 114400B storage words. Because of that, the immediate use of the designed system interactively has been abandoned for the time being and a simulation of the query procedure has been decided.

4.1 Output Samples:

Instead of the terminals, I used the Line-printer, and the retrieval query-program has been run in Batch.

Taking advantage of the fact, that we know in advance the nodes (classif-number) where the routines are attached; I entered the different Key-numbers (different classif-codes in sequence), Key-words (BACKUP etc.) and the Routine-numbers using data cards in the proper order in which the actual user was supposed to do.

In APPENDIX B, one can see seven outputs, where the user tries all the available possibilities of the designed system.

The performance was satisfactory. Here the BACKUP facility is used at the different stages, working properly and all the error checking facilities give the required results.

4.2 Recommendations:

In order to run interactively our retrieval application programs, for the time being the RETRO1, a minimum allocated memory of 114400B words is necessary to the terminals.

The present physical limitation of 124000B words creates a lot of problems in doing so. A future expansion in Central Memory will give the possibility of the interactive use of the system. In that case two slight modifications of the existing program will convert the RETRO1 from BATCH to INTERACTIVE.

One extra statement will be added in the SPECIAL-NAMES paragraph:

'TERMINAL' IS CRT-SCREEN

and the SPECIAL NAME paragraph of the interactive RETRO1 program will become

```
SPECIAL-NAMES.
    SUB-SCHEMA IS RETRIEVALSUB5
    'TERMINAL' IS CRT-SCREEN.
```

The second change will be the addition of the new nmemonic CRT-SCREEN to all DISPLAY and ACCEPT verbs throughout the program.-

DISPLAY $\left\{\begin{matrix}\text{literal}\\\text{data-name}\end{matrix}\right\}$ UPON CRT-SCREEN.

ACCEPT {literal / data-name} FROM CRT-SCREEN.

The RETRO1 retrieval application program accesses the two existing files, the NODE-FILE and the ROUT-FILE which are linked together through the ROUTRETRIEVE relation.

When the new files (ARGUMENT and PARAMETER) will be added, there will be of course the necessity to access them. Some extra paragraphs included in RETRO1 program will perform this task.

An issue of READ ROUTPARAM relation will retrieve all the record instances containing the arguments of the selected routine one by one and for each argument CDCS will retrieve its description from the PARAM-FILE. In the retrieval application programs will be added another one: THE RETRO2.

This program will access directly the ROUTINE-FILE, ARGUMENT-FILE, PARAMETER-FILE related through ROUTPARAM relation and it may be called when the user knows in advance the routine and wants information about it and its arguments.

In this program the user is going to supply the Routine-name (which is known to him) and the program will use it as alternate key to access ROUT-FILE and retrieve the required routine or routines.

A good idea would be to use these two retrieval programs RETRO1,RETRO2 as subroutines.

A very small primitive program coded in COBOL may be used as interface between the retrieval programs and the user. This small program will interrogate the user to find out what retrieval

program he prefers, according to his intention and communication between these separately compiled programs will be attained through the Inter-program Communication facility. This facility allows a COBOL main program (in our case the primitive interface program) to transfer control to a sub program, written in COBOL, or any high level language (in our case the retrieval sub programs).

A CALL statement is used in the main program to transfer the control to the retrieval programs.

The decisions of the user, to what program he prefers, are accessed by both programs, and are passed either through the COMMON-STORAGE SECTION or through parameters (data-items) in the CALL statement (then the USING phrase is employed in the CALL statement of the calling program and a LINKAGE SECTION in the called program describes the data that is passed between).

There will be, finally, the last retrieval program which is trivial, this program accesses the PRIVATE-FILE through the ROUTINE-file and will be used only by the Program Librarian.

Of course the Data Base facility will allow any programer who may have access to it, to write any number of application programs for any purpose, depending upon his requests and intentions.

REFERENCES

Martin James. Computer Data-Base organization. Prentice-Hall, Inc. (1975).

Atwood, A. System Analysis. Hayden Publishing Co., N. Y. (1976).

Redish K. A. Tree Structures for a program Library Index. Dept. of Appl. Mathematics, McMaster University, Hamilton, Ont..

Redish K. A. Proposed Format for A Program Library. Dept. of Appl. Mathematics, McMaster University, Hamilton, Ont..

Control Data Corporation. DMS-170 DDL Version 2 Reference Manual Vol. 1, SCHEMA Definition. CDC # 60498400(1976).

Control Data Corporation. DMS-170 DDL Version 2 Reference Manual Vol. 2, COBOL SUB-SCHEMA Defintion. CDC # 60498500 (1976).

Control Data Corporation. NOS 1 Reference Manual, Vol. 1. CDC # 60435400 (1976).

Control Data Corporation. DMS-170 Data Base Administator User's Guide. CDC # 60499100 (1978).

Control Data Corporation. DMS-170 CDCS Version 1 Reference Manual. CDC # 60498700 (1976).

Control Data Corporation. COBOL Version 5 Reference Manual. CDC # 60497100 (1977).

Control Data Corporation. COBOL Version 5 User's Guide. CDC # 60497200 (1977).

Control Data Corporation. DMS-170 CYBER Record Manager Version 1 Reference Manual, NOS 1 Operating System. CDC # 60495700 (1977).

APPENDIX A

PROGRAM LISTINGS

1. UTILITY PROGRAMS

```
PROJ.                                                    PAP AS.
STUDENT(7117149,PAPPAS9)
CHARGE(RKARED)
FILE(TREEPAP,BT=C,RT=W,MRL=500,MBL=5120,CM=NO)
FILE(PROGPAP,BT=C,RT=W,MRL=500,MBL=5120,CM=NO)
FETCH(TREEPAP,PROGPAP/UN=1013714)
COBOL5.
LDSET(FILES=TREEPAP/PROGPAP)
LGO.
)END OF RECORD
```

```
       IDENTIFICATION DIVISION.
      *
       PROGRAM-ID.     CNT-TST.
      *-----------------------
      *
      *****************************************************************
      *                                                               *
      *     THIS PROGRAM ACCESSES THE PROGRAM LIBRARIAN'S SUPPLIED    *
      *     FILES--TREEPAP AND PROGPAP--AND EXTRACTS THE MAXIMUM      *
      *     NUMBER OF 10 CHRS TRAILERS USED ON THE VARIABLE TEXT.     *
      *     THIS IS THE MAXIMUM VECTOR SIZE.                          *
      *     THIS IS AN INFORMATION FOR THE SCHEMA DESIGN,IN THE       *
      *     DECISION OF THE MAXIMUM VECTOR SIZE.                      *
      *                                                               *
      *****************************************************************
      *
       AUTHOR. C. PAPPAS.
       INSTALLATION. MACMASTER.
       DATE-COMPILED.
      *
      *
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER.  CDC6400.
       OBJECT-COMPUTER.  CDC6400.
      *
       INPUT-OUTPUT SECTION.
       FILE-CONTROL.
      *
           SELECT DISK-NODE ASSIGN TO TREEPAP.
           SELECT DISK-ROUT ASSIGN TO PROGPAP.
           SELECT TEMP-OUT ASSIGN TO OUTPUT.
      *
      *
       DATA DIVISION.
```

```
PROCEDURE DIVISION.
A.
    OPEN INPUT DISK-NODE
         OUTPUT TEMP-OUT.
    PERFORM P-R.

B.
*---------------------------------------------------------------
*
*   THIS PARAGRAPH EXTRACTS THE MAXIMUM VECTOR SIZE OF THE
*   NODE-REC IN DISK-NODE FILEANC OUTPUTS THE 50 FIRST RECORDS
*   OF THE FILE FOR SAMPLE
*
*---------------------------------------------------------------

    READ DISK-NODE AT END GO TO C.
    IF N-OF-W IS GREATER THAN COUNT-W MOVE N-OF-W TO COUNT-W.
    IF COUNTER IS GREATER THAN 50 GO TO BA.
    MOVE DET-AIL TO DET-INF.
    MOVE N-OF-W TO N-OF.
    MOVE SPACES TO TEMP-REC.
    WRITE TEMP-REC FROM LINE-1.
    ADD 1 TO COUNTER.

BA.
    GO TO B.

C.
*---------------------------------------------------------------
*
*   THIS PARAGRAPH PRINTS THE MAXIMUM VECTOR SIZE OF THE
*   DISK-NODE FILE,TOGETHER WITH THE ACCOMPANYING MESSAGES
*
*---------------------------------------------------------------

    MOVE COUNT-W TO NUM-OUT.
    MOVE SPACES TO TEMP-REC.
    WRITE TEMP-REC FROM HEDER.
    CLOSE DISK-NODE.

D.
    OPEN INPUT DISK-ROUT.

E.  PERFORM P-R.
```

```
F.
*---------------------------------------------------------------
*
*    THIS PARAGRAPH EXTRACTS THE MAXIMUM VECTOR SIZE OF THE
*    ROUT-REC IN DISK-ROUT FILEANC OUTPUTS THE 50 FIRST RECORDS
*    OF THE FILE FOR SAMPLE
*
*---------------------------------------------------------------
*
     READ DISK-ROUT AT END GO TO G.
     IF NUM-W IS GREATER THAN COUNT-W MOVE NUM-W TO COUNT-W.
     IF COUNTER IS GREATER THAN 50 GO TO FA.
     MOVE OTH-INF TO DET-INF.
     MOVE NUM-W TO N-OF.
     MOVE SPACES TO TEMP-REC.
     WRITE TEMP-REC FROM LINE-1.
     ADD 1 TO COUNTER.
*
FA.
     GO TO F.
*
G.
*---------------------------------------------------------------
*
*    THIS PARAGRAPH PRINTS THE MAXIMUM VECTOR SIZE OF THE
*    DISK-ROUT FILE,TOGETHER WITH THE ACCOMPANYING MESSAGES
*
*---------------------------------------------------------------
*
     MOVE COUNT-W TO NUM-OUT.
     MOVE SPACES TO TEMP-REC.
     WRITE TEMP-REC FROM HEDER.
     GO TO FINISH.
*
P-R.
*---------------------------------------------------------------
*
*    THIS PARAGRAPH INITIALIZES DATA ITEMS AND RECORDS INVOLVED
*    IN THE COUNTING AND PRINTING.
*
*---------------------------------------------------------------
*
     MOVE ZERO TO COUNTER.
     MOVE ZEROS TO COUNT-W.
     MOVE '1' TO TEMP-REC.
```

```
*
 F.
*---------------------------------------------------------------
*
*    THIS PARAGRAPH EXTRACTS THE MAXIMUM VECTOR SIZE OF THE
*    ROUT-REC IN DISK-ROUT FILEANC OUTPUTS THE 50 FIRST RECORDS
*    OF THE FILE FOR SAMPLE
*
*---------------------------------------------------------------
*
     READ DISK-ROUT AT END GO TO G.
     IF NUM-W IS GREATER THAN COUNT-W MOVE NUM-W TO COUNT-W.
     IF COUNTER IS GREATER THAN 50 GO TO FA.
     MOVE OTH-INF TO DET-INF.
     MOVE NUM-W TO N-OF.
     MOVE SPACES TO TEMP-REC.
     WRITE TEMP-REC FROM LINE-1.
     ADD 1 TO COUNTER.
*
 FA.
     GO TO F.
*
 G.
*---------------------------------------------------------------
*
*    THIS PARAGRAPH PRINTS THE MAXIMUM VECTOR SIZE OF THE
*    DISK-ROUT FILE,TOGETHER WITH THE ACCOMPANYING MESSAGES
*
*---------------------------------------------------------------
*
     MOVE COUNT-W TO NUM-OUT.
     MOVE SPACES TO TEMP-REC.
     WRITE TEMP-REC FROM HEDER.
     GO TO FINISH.
*
 P-R.
*---------------------------------------------------------------
*
*    THIS PARAGRAPH INITIALIZES DATA ITEMS AND RECORDS INVOLVED
*    IN THE COUNTING AND PRINTING.
*
*---------------------------------------------------------------
*
     MOVE ZERO TO COUNTER.
     MOVE ZEROS TO COUNT-W.
     MOVE '1' TO TEMP-REC.
```

```
PROJ.                                                          PAPPAS.
STUDENT(7117149,PAPPAS9)
CHARGE(RKAREO)
FILE(TREEPAP,BT=C,RT=W,MRL=500,MBL=5120,CM=NO)
FETCH(TREEPAP/UN=1013714)
LOSET(FILES=TREEPAP)
LGO.
SAVE(NODEFI=TEMP1)
END OF RECORD
```

```
       IDENTIFICATION DIVISION.
      *
       PROGRAM-ID. CREATE-NODEFI-TEMP1.
      *-------------------------------
      *
      *****************************************************************
      *                                                               *
      *  THIS PROGRAM ACCESS THE ALREADY EXISTING SEQUENTIAL FILE     *
      *  CONTAINING THE SUBJECTS WHICH ARE AVAILABLE IN THE PROGRAM   *
      *  LIBRARY, IN SEQUENCE OF CLASSIFICATION NUMBER.               *
      *                                                               *
      *  MODIFIES THE RECORD STRUCTURE AND CREATES ANOTHER SEQUENTIAL *
      *  DATA FILE NODEFI SAVED ON A DISK WITH PERMANENT NAME--TEMP1, *
      *  IN A RECORD FORMAT THAT FITS THE RECORD DESCRIPTION OF THE   *
      *  DATA-BASE.                                                   *
      *                                                               *
      *****************************************************************
      *
       AUTHOR. C. PAPPAS.
       INSTALLATION. MACMASTER.
       DATE-COMPILED.
      *
      *
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER.  CDC6400.
       OBJECT-COMPUTER.  CDC6400.
       INPUT-OUTPUT SECTION.
       FILE-CONTROL.
      *
      *---------------------------------------------------------------
      *
      *  TREEPAP--IS THE EXISTING SEQUTIAL INPUT FILE TO BE MODIFIED.
      *
      *  NODEFI--IS THE OUPUT SEQUTIAL DATA FILE WHICH WILL BE CREATED.
      *
```

```
*  SORTFLE--IS THE SORTING FILE USED TO SORT OUTPUT FILE IN
*  SEQUENCE OF MODIFIED CLASSIFICATION NUMBER.
*
*----------------------------------------------------------------
*
     SELECT DISKSQ-FILE     ASSIGN TO TREEPAP.
     SELECT NODE-FILE ASSIGN TO NODEFI.
     SELECT SORT-FILE ASSIGN TO SORTFLE.
*
*
 DATA DIVISION.
 FILE SECTION.
*
 FD   DISKSQ-FILE
      LABEL RECORDS ARE OMITTED.
 01   DISKSQ-REC.
*
*  THIS IS THE FORMAT OF THE RECORD OF THE INPUT-FILE.
*  TREE-KEY--STANDS FOR CLASSIFIC NO. BEFORE MODIFICATION.
*
      02  TREE-KEY        PIC X(12).
      02  LIB-CO          PIC A.
      02  FILLER          PIC X.
      02  N-OF-W          PIC 999.
      02  FILLER          PIC X(3).
      02  N-DESC          PIC X(10)
                          OCCURS 1 TO 20 TIMES
                          DEPENDING ON N-OF-W.
*
 SD   SORT-FILE
      DATA RECORD IS SORT-REC.
 01   SORT-REC.
*
*  S-KEY--STANDS FOR THE MODIFIED CLASSIFICATION NUMBER
*  SORT-REC--IS GOING TO BE SORTED IN ASCENDING ORDER OF S-KEY
*
      02  S-KEY.
          03 PRE-FIX      PIC X.
          03 AFTR-FIXE    PIC X(12).
      02  S-CO            PIC A.
      02  S-WORDS         PIC 999.
      02  S-DESC          PIC X(10)
                          OCCURS 1 TO 20 TIMES
                          DEPENDING ON S-WORDS.
*
 FD   NODE-FILE
```

```
     LABEL RECORDS ARE OMITTED.
01   NODE-REC.
*
*    NODE-REC--IS THE RECORD OF THE OUTPUT FILE WHICH IS SORTED
*    SEQUENTIALLY IN ASCENDING ORDER OF CLASSIF-NO(DUMMY-KEY).
*
*    DUMMY-KEY--THIS ITEM IS USED TO PROVIDE THE SPACE (L-N)
*    TO ACCOMODATE THE LEVEL NUMBER USED TO MODIFY CLASS-CODE
*    ITEM AND CREATE ITEM WITH PIC X(13) TO CONFORM WITH THE
*    DATA-BASE CLASSIF-NO KEY USED.
*
       02 CLASSIF-NO        PIC X(13).
       02 DUMMY-KEY REDEFINES CLASSIF-NO.
          03 L-N            PIC X.
          03 CLAS-CODE      PIC X(12).
       02 LIBRARY-INDEX     PIC A.
       02 NODE              PIC AAAA.
       02 NO-OF-WORDS       PIC 999.
       02 NODE-DESCR        PIC X(10)
                            OCCURS 1 TO  20 TIMES
                            DEPENDING ON NO-OF-WORDS.
*
 WORKING-STORAGE SECTION.
 77   SUBSCRIPT             PIC 999.
 01   TALLY                 PIC 9.
*    TALLY ITEM ACCOMODATES THE LEVEL PREFIX WHICH WILL MODIFY
*    CLASSIFICATION NUMBER.
*
*
 PROCEDURE DIVISION.
 MAIN-LINE SECTION.
 INITIALIZATION.
      OPEN INPUT DISKSQ-FILE
           OUTPUT NODE-FILE.
      MOVE 0 TO TALLY.
 SORT-CARD SECTION.
 SORTING.
      SORT SORT-FILE ON ASCENDING KEY S-KEY
           INPUT PROCEDURE IS INP-PR
           OUTPUT PROCEDURE IS OUT-PR.
      STOP RUN.
*
 INP-PR SECTION.
 KEY-TRANSFER.
*
*-----------------------------------------------------------------
```

```
*
*   WE TRANSFER THE ITEMS OF INPUT RECORD TO SORT RECORD TO BE
*   SORTED.
*
*---------------------------------------------------------------
*
     READ DISKSQ-FILE AT END GO TO END-KEY-MODIF.
     MOVE TREE-KEY TO AFTR-FIXE.
     MOVE LIB-CO TO S-CO.
     MOVE N-OF-W TO S-WORDS.
     PERFORM INP-TRANS-PROC VARYING SUBSCRIPT FROM 1 BY 1
          UNTIL SUBSCRIPT IS GREATER THAN S-WORDS.
*
 KEY-MODIFICATION.
*
*---------------------------------------------------------------
*
*   MAXIMUM TREE LEVEL(DEPTH) IS 8,SO SUBTRACTING FROM THAT, THE NO
*   OF SPACES OF CLASSIF- ITEM, RESULTS THE LEVEL NO WHICH IS
*   STORED IN TALLY, THEN TALLY IS MOVED AHEAD OF THE CLASSIF-NO
*   MODIFYING IT.
*
*LEVEL NO. IS GOING TO BE ADDED IN THE KEY.
*
*---------------------------------------------------------------
*
     INSPECT AFTR-FIXE TALLYING TALLY FOR ALL SPACES.
     SUBTRACT 8 FROM TALLY.
     MOVE TALLY TO PRE-FIX.
*KEY IS NOW MODIFIED.
*
     MOVE 0 TO TALLY.
     RELEASE SORT-REC.
*  RECORD AFTER MODIFICATION IS RELEASED TO SORT FILE FOR SORTING
*
     GO TO KEY-TRANSFER.
*
 INP-TRANS-PROC.
*
*---------------------------------------------------------------
*
*   VARIABLE TEXT IS TRANSFERED FROM INPUT RECORD TO SORT RECORD.
*
*---------------------------------------------------------------
*
     MOVE N-DESC(SUBSCRIPT) TO S-DESC (SUBSCRIPT).
```

```
*
 END-KEY-MODIF.
     CLOSE DISKSQ-FILE.
*
 CUT-PR SECTION.
 TEMPDEV-TRANSFER.
*
*-----------------------------------------------------------------
*
*   SORTED RECORDS ARE RETURNED AND ALL ITEMS OF SORT-REC
*   TRANSFERED TO OUTPUT FILE--NODE-REC WHICH IS WRITTEN OUT OF
*   THE NEW MODIFIED SEQUENTIAL FILE NODEFI WHICH IS SAVED UNDER
*   THE PERMANENT NAME--TEMP1.
*
*-----------------------------------------------------------------
*
     RETURN SORT-FILE AT END GO TO END-TRANSFER.
     MOVE S-KEY TO DUMMY-KEY.
     MOVE S-CO TO LIBRARY-INDEX.
     MOVE S-WORDS TO NO-OF-WORDS.
     PERFORM OUT-TRANS-PROC VARYING SUBSCRIPT FROM 1 BY 1
          UNTIL SUBSCRIPT IS GREATER THAN NO-OF-WORDS.
     WRITE NODE-REC.
     GO TO TEMPDEV-TRANSFER.
*
 OUT-TRANS-PROC.
*
*-----------------------------------------------------------------
*
*  VARIABLE TEXT IS TRANSFERED FROM SORT RECORD TO OUTPUT RECORD.
*
*-----------------------------------------------------------------
*
     MOVE S-DESC (SUBSCRIPT) TO NODE-DESC (SUBSCRIPT).
*
 END-TRANSFER.
     CLOSE NODE-FILE.
*
```

```
PROJ.                                                          PAPPAS.
STUDENT(7117149,PAPPAS9)
CHARGE(RKARED,BATCH)
COBOL5.
FILE(PROGPAP,BT=C,RT=W,MRL=500,MBL=5120,CM=NO)
FETCH(PROGPAP/UN=1013714)
LOSET(FILES=PROGPAP)
LGO.
SAVE(ROUTFI=TEMP2)
END OF RECORD
      IDENTIFICATION DIVISION.
      *
      PROGRAM-ID. CREATE-ROUTFI-TEMP2.
      * ------------------------------
      *
      *****************************************************************
      *                                                               *
      *   THIS PROGRAM ACCESS THE ALREADY EXISTING SEQUENTIAL FILE    *
      *   CONTAINING THE INFORMATION FOR EACH ROUTINE, IN SEQUENCE OF *
      *   ROUTINE-NAME(ALPHABETIC).                                   *
      *                                                               *
      *   CREATES NEW KEY (ROUTINE-NO), RESTRUCTURES THE RECORD       *
      *   FORMAT TO FIT DATA-BASE AND FINALLY STORES IN ASCENDING ORDER *
      *   OF ROUTINE-NO AND ALPHABETIC OF ROUTINE-NAME IN A SEQUENTIAL *
      *   FILE--ROUTFI, SAVED WITH PERMANENT NAME--TEMP2.             *
      *                                                               *
      *****************************************************************
      *
      AUTHOR. C. PAPPAS.
      INSTALLATION. MACMASTER.
      DATE-COMPILED.
      *
      *
      ENVIRONMENT DIVISION.
      CONFIGURATION SECTION.
      SOURCE-COMPUTER.  CDC6400.
      OBJECT-COMPUTER.  CDC6400.
      INPUT-OUTPUT SECTION.
      FILE-CONTROL.
      *
      *---------------------------------------------------------------
      *
      *   PROGPAP--IS THE EXISTING SEQUENTIAL INPUT FILE TO BE MODIFIED.
      *
      *   ROUTFI--IS THE OUTPUT SEQUENTIAL FILE WHICH WILL BE CREATED.
```

```
*
*-----------------------------------------------------------------
*
     SELECT DISKSQ-FILE     ASSIGN TO PROGPAR.
     SELECT ROUT-FILE       ASSIGN TO ROUTFI.
*
*
 DATA DIVISION.
 FILE SECTION.
*
 FD   DISKSQ-FILE
      LABEL RECORDS ARE OMITTED.
 01   DISKSQ-REC.
*
*   THIS IS THE FORMAT OF RECORD IN THE INPUT FILE, WHICH WILL BE
*   REARANGED AND MODIFIED BY THE PROGRAM.
*
      02  TREE-KEY          PIC X(12).
      02  NAME-OF-ROUT      PIC X(15).
      02  LIB-NA            PIC A(7).
      02  ROUT-DETAILS.
        03  ACCESS-LEV      PIC 9.
        03  SUPORT-LEV      PIC A.
        03  DOC-NO          PIC X(8).
        03  SOUP-LANG       PIC A.
        03  PROG-T          PIC A.
        03  CHAV            PIC 9.
      02  NO-OF-W           PIC 999.
      02  TEXT-OF           PIC X(10)
                            OCCURS 1 TO 30 TIMES
                            DEPENDING ON NO-OF-W.
*
      FD ROUT-FILE
      LABEL RECORDS ARE OMITTED.
 01   ROUT-REC.
*
*   THIS IS THE RESULTING RECORD AGREEING WITH THE FORMAT OF
*   DATA-BASE RECORD WHICH WILL BE STORED SEQUENTIALLY ON
*   TEMPORARY FILE ROUTFI, IN ASCENDING ORDER OF ROUTINE-NO. AND
*   SAVED WITH PERMANENT NAME--TEMP2.
*
      02  ROUTINE-NO        PIC 9(4).
      02  ROUT-NAME         PIC X(15).
      02  LIBR-NAME         PIC A(7).
      02  CLASS-NO          PIC X(13).
      02  ARTIF-KEY REDEFINES CLASS-NO.
```

```
           03  L-NO            PIC X.
           03  CLASS           PIC X(12).
       02  ROUT-DETAILS.
           03  AVAIL-LEVEL     PIC 9.
           03  SUP-LEV         PIC A.
           03  DOCUM-NO        PIC X(8).
           03  S-L-C           PIC A.
           03  PROGR-TYPE      PIC A.
           03  C-M-AV          PIC 9.
       02  L-OF-O              PIC 999.
       02  TASK-OF-ROUT        PIC X(10)
                               OCCURS 1 TO 90 TIMES
                               DEPENDING ON L-OF-O.
*
 WORKING-STORAGE SECTION.
 77   SUBSCRIPT                PIC 999.
 01   TALLY                    PIC 9.
*   IT ACCOMODATES THE LEVEL PREFIX WHICH WILL MODIFY
*    CLASSIFICATION NUMBER.
*
 01   TAPE-RECORDS-LEFT PIC X(3).
 01   NUMB-OF-ROUTINES  PIC 9(4).
*
*
 PROCEDURE DIVISION.
 MAIN-LINE.
      PERFORM INITIALIZATION.
      PERFORM ITEMS-TRANSFER THRU ROUT-ENUMERATION
              UNTIL TAPE-RECORDS-LEFT = "NO ".
      PERFORM END-OF-JOB.
      STOP RUN.
*
 INITIALIZATION.
      OPEN INPUT DISKSQ-FILE
           OUTPUT ROUT-FILE.
      MOVE "YES" TO TAPE-RECORDS-LEFT.
      MOVE 0 TO TALLY.
      MOVE 0 TO NUMB-OF-ROUTINES.
      READ DISKSQ-FILE AT END MOVE "NO " TO TAPE-RECORDS-LEFT.
*FIRST RECORD NOW IS IN BUFFER.
*
 ITEMS-TRANSFER.
*
*---------------------------------------------------------------
*
*   ITEMS OF INPUT RECORD(DISKSQ-REC) ARE TRANSFERED TO OUTPUT
```

```
*   RECORD(ROUT-REC).
*
*---------------------------------------------------------------
*
      MOVE SPACES TO ROUT-REC.
      MOVE NAME-OF-ROUT TO ROUT-NAME.
      MOVE LIB-NA TO LIB-NAME.
      MOVE ROUT-DETAILS OF DISKSQ-REC TO ROUT-DETAILS OF ROUT-REC.
      MOVE TREE-KEY TO CLASS.
      MOVE NO-OF-W TO L-OF-D.
      PERFORM TEXT-TRANSFER VARYING SUBSCRIPT FROM 1 BY 1
          UNTIL SUBSCRIPT IS GREATER THAN L-OF-D.
*
 ALTERN-KEY-MODIFICATION.
*
*---------------------------------------------------------------
*
*   WE MODIFY CLASSIFICATION NO., ACCORDINGLY(TO BE USED AS
*   ALTERNATE KEY) TO AGREE WITH THE CLASSIF-NO OF NODE-FILE.
*
*---------------------------------------------------------------
*
      INSPECT CLASS TALLYING TALLY FOR ALL SPACES.
      SUBTRACT 8 FROM TALLY.
      MOVE TALLY TO L-NO.
      MOVE 0 TO TALLY.
*
 ROUT-ENUMERATION.
*
*---------------------------------------------------------------
*
*   ROUTINES ARE NUMBERED IN STEPS OF 10, SO CREATING A KEY FOR
*   THE ROUT-REC AND TAKING CARE OF POSSIBLE RECORD INSERTION IN
*   THE FUTURE.
*
*---------------------------------------------------------------
*
      ADD 10 TO NUMB-OF-ROUTINES.
      MOVE NUMB-OF-ROUTINES TO ROUTINE-NO.
      WRITE ROUT-REC.
      READ DISKSQ-FILE AT END MOVE 'NO' TO TAPE-RECORDS-LEFT.
*
 TEXT-TRANSFER.
*
*---------------------------------------------------------------
*
```

```
*  THE VARIABLE TEXT IS TRANSFERED FROM INPUT REC(DISKS)-REC) TO
*  OUTPUT REC(ROUT-REC).
*-----------------------------------------------------------------
*     MOVE TEXT-OF (SUBSCRIPT) TO TASK-OF-ROUT (SUBSCRIPT).
 END-OF-JOB.
      CLOSE DISK30-FILE ROUT-FILE.
```

2. THE SCHEMA AND SUB-SCHEMAS CODING

```
PROJ.                                                          PAPPAS.
STUDENT(7117149,PAPAS1)
CHARGE(RKARED,BATCH)
FILE(TREEARE,FO=IS,RT=T)
FILE(ROUTARE,FO=IS,RT=T)
FILE(ARGUMEN,FO=IS,RT=T)
FILE(PARAMET,FO=IS,RT=T)
FILE(PRIVATE,FO=IS,RT=F)
DDL(OS,SC=HELPLIB)
SAVE(HELPLIB=CYL1SCH/CT=PU,M=P)
END OF RECORD
/*****************************************************************/
/*                                                               */
/* THE FOLLOWING CODE WRITTEN IN DDL-VERSION 2 WITHIN DMS-170    */
/* UNDER NOS 1 OPERATING SYSTEM, DESCRIBES THE SCHEMA FOR THE    */
/* HELPLIBRARY-DB DATA BASE.                                     */
/*                                                               */
/* THE DATA BASE CONSISTS OF 5 AREAS AND EACH AREA CONTAINS 1    */
/* RECORD DESCRIPTION.                                           */
/* THREE RELATIONS ARE DEFINED OVER THE SCHEMA.                  */
/*                                                               */
/*****************************************************************/
/*                                                               */
/*                                                               */
/* SCHEMA IDENTIFICATION ENTRY.                                  */
/* ----------------------------                                  */
/*                                                               */
     SCHEMA NAME IS HELPLIBRARY-DB.
/*                                                               */
/*                                                               */
/* AREA DESCRIPTION ENTRY.                                       */
/* -----------------------                                       */
/*                                                               */
     AREA NAME IS TREEAREA.
     AREA NAME IS ROUTAREA.
     AREA NAME IS ARGUMENTAREA.
     AREA NAME IS PARAMETAREA.
     AREA NAME IS PRIVATEAREA.
/*                                                               */
/*                                                               */
/* RECORD DESCRIPTION ENTRY.                                     */
/* -------------------------                                     */
/*                                                               */
     RECORD NAME IS NODE-REC
          WITHIN TREEAREA.
```

```
/*                                                                    */
/* THIS RECORD CONTAINS THE DESCRIPTION OF A PARTICULAR SUBJECT       */
/* OF THE LIBRARY TREE .                                              */
/* WE MAINTAIN ONE RECORD FOR EACH NODE                               */
/*                                                                    */
01   CLASSIF-NO           PIC 'X(13)'.
01   LIBRARY-INDEX        PIC 'A'.
01   NODE                 PIC 'AAAA'.
01   LENGTH-IN-WORDS      PIC '999'
                          CHECK IS VALUE 1 THRU 25.
01   NODE-DESCRIPTION     PIC 'X(10)'
                          OCCURS LENGTH-IN-WORDS TIMES.
/*                                                                    */
/*                                                                    */
     RECORD NAME IS ROUT-REC
            WITHIN ROUTAREA.
/*                                                                    */
/* THIS RECORD STORES ALL THE INFORMATION CONCERNING A PARTI          */
/* CULAR ROUTINE                                                      */
/* WE MAINTAIN ONE RECORD FOR EACH ROUTINE                            */
/*                                                                    */
01   ROUTINE-NO           PIC '9(4)'.
01   ROUT-NAME            PIC 'X(15)'.
01   LIBR-NAME            PIC 'A(7)'.
01   CLASS-NO             PIC 'X(13)'.
01   ROUT-DETAILS         OCCURS 1 TIMES.
     02 AVAIL-LEVEL       PIC '9'
                          CHECK IS VALUE 3 THRU 4.
     02 SUP-LEV           PIC 'A'.
     02 DOCUM-NO          PIC 'X(8)'.
     02 SOUR-LANG-CODE    PIC 'A'.
     02 PROGR-TYPE        PIC 'A'.
     02 COMP-MACH-AV      PIC '9'.
01   LENG-OF-DESCR        PIC '999'
                          CHECK IS VALUE 1 THRU 100.
01   LINE-TEXT            PIC 'X(10)'
                          OCCURS LENG-OF-DESCR TIMES.
/*                                                                    */
/*                                                                    */
     RECORD NAME IS PARAM-REC
            WITHIN ARGUMENTAREA.
/*                                                                    */
/* THIS RECORD STORES ALL THE INFORMATION CONCERNING A PARTI          */
/* CULAR PARAMETER                                                    */
/* WE MAINTAIN ONE RECORD FOR EACH PARAMETER                          */
/*                                                                    */
```

```
01    ARG-NO                PIC ≠9(4)≠.
01    ROUT-NO               PIC ≠9(4)≠.
01    ARGUM-DETAILS         OCCURS 1 TIMES.
      02 ARGUM-NAME         PIC ≠X(7)≠.
      02 TYPE-OF            PIC ≠A≠.
      02 INP-OUT            PIC ≠AA≠.
01    NO-OF-POINT           PIC ≠99≠
                            CHECK IS VALUE 0 THRU 25.
01    POINT-R               PIC ≠9(4)≠
                            OCCURS NO-OF-POINT TIMES.
/*                                                              */
/*                                                              */
      RECORD NAME IS PARAM-DESCR-REC
             WITHIN PARAMETAREA.
/*                                                              */
/* THIS RECORD CONTAINS THE DESCRIPTION OF THE PARTICULAR       */
/* PARAMETER                                                    */
/* WE MAINTAIN ONE RECORD FOR EACH PARAMETER                    */
/*                                                              */
01    ARG-NUM               PIC ≠9(4)≠.
01    NUM-OF-WORDS          PIC ≠999≠
                            CHECK IS VALUE 1 THRU 100.
01    PAR-DESCR             PIC ≠X(10)≠
                            OCCURS NUM-OF-WORDS TIMES.
/*                                                              */
/*                                                              */
      RECORD NAME IS PRIV-INFO-REC
             WITHIN PRIVATEAREA.
/*                                                              */
/* THIS RECORD STORES ALL THE INFORMATION CONCERNING ONLY       */
/* THE CHIEF LIBRARIAN                                          */
/* WE MAINTAIN ONE RECORD FOR EACH ROUTINE                      */
/*                                                              */
01    ROUTINE-NO            PIC ≠9(4)≠.
01    SOURCE-TAPE           PIC ≠X(7)≠.
01    FILE-POS-WS           PIC ≠X(100)≠.
01    DECK-NAME             PIC ≠X(9)≠.
01    MONIT-NAME            PIC ≠X(7)≠.
01    DOC-SOUR              PIC ≠A≠.
01    MAINT-BY              PIC ≠X(4)≠
                            OCCURS 4 TIMES.
/*                                                              */
/*                                                              */
/* DATA CONTROL ENTRY.                                          */
/* ------------------                                           */
/*                                                              */
```

```
     DATA CONTROL.
/*                                                                    */
     AREA NAME IS TREEAREA
     KEY IS CLASSIF-NO OF NODE-REC.
/*                                                                    */
     AREA NAME IS ROUTAREA
     KEY IS ROUTINE-NO OF ROUT-REC
     KEY IS ALTERNATE ROUT-NAME OF ROUT-REC
          DUPLICATES ARE INDEXED.
/*                                                                    */
     AREA NAME IS ARGUMENTAREA
     KEY IS ARG-NO OF PARAM-REC
     KEY IS ALTERNATE ROUT-NO OF PARAM-REC
          DUPLICATES ARE INDEXED.
/*                                                                    */
     AREA NAME IS PARAMETAREA
     KEY IS ARG-NUM.
/*                                                                    */
     AREA NAME IS PRIVATEAREA
     KEY IS ROUTINE-NO OF PRIV-INFO-REC.
/*                                                                    */
/*                                                                    */
/* RELATION ENTRY.                                                    */
/* ---------------                                                    */
/*                                                                    */
/*--------------------------------------------------------------------*/
/*                                                                    */
/* THE RELATION--ROUTRETRIEVE--RELATES(JOINS) THE AREAS(FILES)        */
/* TREEAREA-ROUTAREA.                                                 */
/*                                                                    */
/*--------------------------------------------------------------------*/
/*                                                                    */
     RELATION NAME IS ROUTRETRIEVE
          JOIN WHERE CLASSIF-NO EQ CLASS-NO.
/*                                                                    */
/*                                                                    */
/*--------------------------------------------------------------------*/
/*                                                                    */
/* THE RELATION--ROUTPARAM--RELATES(JOINS) THE AREAS(FILES)           */
/* ROUTAREA-ARGUMENTAREA-PARAMETAREA.                                 */
/*                                                                    */
/*--------------------------------------------------------------------*/
/*                                                                    */
     RELATION NAME IS ROUTPARAM
          JOIN WHERE ROUTINE-NO OF ROUT-REC EQ ROUT-NO
            OF PARAM-REC ARG-NO OF PARAM-REC EQ
```

```
                ARG-NUM OF PARAM-DESCR-REC.

/*-------------------------------------------------------------------*/
/*                                                                   */
/* THE RELATION--ROUTPRIV--RELATES(JOINS) THE AREAS(FILES)           */
/* ROUTAREA-PRIVATAREA.                                              */
/*                                                                   */
/*-------------------------------------------------------------------*/

     RELATION NAME IS ROUTPRIV
            JOIN WHERE ROUTINE-NO OF ROUT-REC EQ
              ROUTINE-NO OF PRIV-INFO-REC.
```

```
PROJ.
STUDENT(7117149,PAPAS1)                                          PAPPAS.
CHARGE(RKAREA0,BATCH)
FETCH(HELPLIB=CYL1SCH)
DDL(CS,SB=SUBSLIB,SC=HELPLIB)
SAVE(SUBSLIB=CYL2CRS/PW=PAP,CT=S,M=W)
END OF RECORD
      ****************************************************
      *                                                  *
      *     THIS IS THE CREATION SUB-SCHEMA FOR THE AREA *
      *     TREEAREA WHICH IS RENAMED--NODE-FILE--FOR USE*
      *     BY THE COBOL CREATION APPLICATION PROGRAM.   *
      *                                                  *
      *     PRIMARY KEY IS--CLASSIF-NO--FOR NODE-FILE.   *
      *                                                  *
      ****************************************************
      *
      *
       TITLE DIVISION.
       ---------------
      *
       SS TREEFILECREATIONSUB1 WITHIN HELPLIBRARY-DB.
      *
      *
       ALIAS DIVISION.
       ---------------
      *
       AD REALM TREEAREA BECOMES NODE-FILE.
       AD DATA LENGTH-IN-WORDS BECOMES NO-OF-WORDS.
       AD DATA NODE-DESCRIPTION BECOMES NODE-DESCR.
      *
      *
       REALM DIVISION.
       ---------------
      *
       RD NODE-FILE.
      *
      *
       RECORD DIVISION.
       ----------------
      *
      *
       01  NODE-REC.
      *
            02 CLASSIF-NO      PIC X(13).
```

```
02 LIBRARY-INDEX   PIC A.
02 NODE            PIC AAAA.
02 NO-OF-WORDS     PIC 999.
02 NODE-DESCR      PIC X(10)
                   OCCURS 1 TO 20 TIMES
                   DEPENDING ON NO-OF-WORDS.
```

```
PROJ.                                                          PAPPAS.
STUDENT(7117149,PAPAS1)
CHARGE(RKAREA,BATCH)
FETCH(HELPLIB=CYL1SCH)
FETCH(SUBSLIB=CYL2CRS)
DDL(C5,SB=SUBSLIB,SC=HELPLIB)
REPLACE(SUBSLIB=CYL2CRS)
END OF RECORD
      ****************************************************
      *                                                  *
      *     THIS IS THE CREATION SUB-SCHEMA FOR THE AREA *
      *     ROUTAREA WHICH IS RENAMED--ROUT-FILE--FOR USE*
      *     BY THE COBOL CREATION APPLICATION PROGRAM.   *
      *                                                  *
      *     PRIMARY KEY IS--ROUTINE-NO--FOR ROUT-FILE.   *
      *     ALTERNATE KEY IS--ROUT-NAME--FOR ROUT-FILE.  *
      *                                                  *
      ****************************************************
      *
      *
       TITLE DIVISION.
      * --------------
      *
       SS ROUTFILECREATIONSUB2 WITHIN HELPLIBRARY-08.
      *
      *
       ALIAS DIVISION.
      * --------------
      *
       AD REALM ROUTAREA BECOMES ROUT-FILE.
       AD DATA SOUR-LANG-CODE BECOMES S-L-C.
       AD DATA COMP-MACH-AV BECOMES C-M-AV.
       AD DATA LENG-OF-DESCR BECOMES L-OF-D.
       AD DATA LINE-TEXT BECOMES TASK-OF-ROUT.
      *
      *
       REALM DIVISION.
      * --------------
      *
       RD ROUT-FILE.
      *
      *
       RECORD DIVISION.
      * ---------------
      *
```

```
01  ROUT-REC.
*
    02 ROUTINE-NO       PIC 9(4).
    02 ROUT-NAME        PIC X(15).
    02 LIBR-NAME        PIC A(7).
    02 CLASS-NO         PIC X(13).
    02 ROUT-DETAILS       OCCURS 1 TIMES.
       03 AVAIL-LEVEL   PIC 9.
       03 SUP-LEV       PIC A.
       03 DOCUM-NO      PIC X(8).
       03 S-L-C         PIC A.
       03 PROGR-TYPE    PIC A.
       03 C-M-AV        PIC 9.
    02 L-OF-O           PIC 999.
    02 TASK-OF-ROUT     PIC X(10)
                        OCCURS 1 TO 80 TIMES
                        DEPENDING ON L-OF-O.
*
*
```

```
PROJ.                                                          PAPPAS.
STUDENT(7117149,PAPAS1)
CHARGE(RKAREA,BATCH)
FETCH(HELPLIB=CYL1SCH)
FETCH(SUBSLIB=CYL2CRS)
CCL(C5,SB=SUBSLIB,SC=HELPLIB)
REPLACE(SUBSLIB=CYL2CRS)
END OF RECORD
```

```
*****************************************************
*                                                   *
*     THIS IS THE CREATION SUB-SCHEMA FOR THE AREAS *
*     ARGUMENTAREA, PARAMETAREA WHICH ARE RENAMED   *
*     ARGUM-FILE, PARAM-FILE  RESPECTIVELY FOR USE  *
*     BY THE COBOL CREATION APPLICATION PROGRAM.    *
*                                                   *
*     PRIMARY KEY IS--ARG-NO--FOR ARGUM-FILE.       *
*     ALTERNATE KEY IS--ROUT-NO--FOR ARGUM-FILE.    *
*     PRIMARY KEY IS--ARG-NUM--FOR PARAM-FILE.      *
*                                                   *
*****************************************************
*
*
 TITLE DIVISION.
* --------------
*
 SS PARAMFILECREATIONSUB3 WITHIN HELPLIBRARY-DB.
*
*
 ALIAS DIVISION.
* --------------
*
 AD REALM ARGUMENTAREA BECOMES ARGUM-FILE.
 AD REALM PARAMETAREA BECOMES PARAM-FILE.
*
*
 REALM DIVISION.
* --------------
*
 RD ARGUM-FILE,PARAM-FILE.
*
*
 RECORD DIVISION.
* ---------------
*
*-----------------------------------------------------
```

```
*
*     DESCRIPTION OF PARAM-REC OF ARGUM-FILE.          *
*
*-----------------------------------------------------
*
 01   PARAM-REC.
*
      02 ARG-NO              PIC 9(4).
      02 ROUT-NO             PIC 9(4).
      02 ARGUM-DETAILS       OCCURS 1 TIMES.
         03 ARGUM-NAME            PIC X(7).
         03 TYPE-OF           PIC A.
         03 INP-OUT           PIC AA.
      02 NO-OF-POINT         PIC 99.
      02 POINT-R             PIC 9(4)
                             OCCURS 0 TO 25 TIMES
                             DEPENDING ON NO-OF-POINT.
*
*-----------------------------------------------------
*
*     DESCRIPTION OF PARAM-DESCR-REC OF PARAM-FILE.    *
*
*-----------------------------------------------------
*
 01   PARAM-DESCR-REC.
*
      02 ARG-NUM             PIC 9(4).
      02 NUM-OF-WORDS        PIC 999.
      02 PAR-DESCR           PIC X(10)
                             OCCURS 1 TO 80 TIMES
                             DEPENDING ON NUM-OF-WORDS.
*
*
```

```
PROJ.                                                          PAPPAS.
STUDENT(7117149,PAPAS1)
CHARGE(RKAREO,BATCH)
FETCH(HELPLIB=CYL1SCH)
ODL(C5,SB=PRIVLIB,SC=HELPLIB)
SAVE(PRIVLIB=CYL3PRS/PW=COS,CT=P,M=W)
END OF RECORD
      ***********************************************************
      *                                                         *
      *     THIS IS THE CREATION SUB-SCHEMA FOR THE AREA        *
      *     PRIVATEAREA RENAMED--PRIVE-FILE--FOR USE            *
      *     BY THE COBOL CREATION APPLICATION PROGRAM.          *
      *                                                         *
      *     PRIMARY KEY IS--ROUTINE-NO--FOR PRIVE-FILE.         *
      *                                                         *
      ***********************************************************
      *
      *
       TITLE DIVISION.
      * ---------------
      *
       SS PRIVATEFILECREATICNSUB4 WITHIN HELPLIBRARY-DB.
      *
      *
       ALIAS DIVISION.
      * ---------------
      *
       AD REALM PRIVATEAREA BECOMES PRIVE-FILE.
       AD DATA FILE-POS-WS BECOMES F-P-W.
      *
      *
       REALM DIVISION.
      * ---------------
      *
       RD PRIVE-FILE.
      *
      *
       RECORD DIVISION.
      * ----------------
      *
      *
       01   PRIV-INFO-REC.
      *
            02 ROUTINE-NO      PIC 9(4).
            02 SOURCE-TAPE     PIC X(7).
```

```
02 F-P-W          PIC X(100).
02 DECK-NAME      PIC X(9).
02 MONIT-NAME     PIC X(7).
02 DOC-SOUR       PIC A.
02 MAINT-BY       PIC X(4)
                  OCCURS 4 TIMES.
```

```
PROJ.                                                          PAPPAS.
STUDENT(7117149,PAPAS1)
CHARGE(RKAREO,BATCH)
FETCH(HELPLIB=CYL1SCH)
COL(C5,SB=RETRLIB,SC=HELPLIB)
SAVE(RETRLIB=CYL4RTS/PW=PAP,CT=PU,M=R)
END OF RECORD
      ***********************************************************
      *                                                         *
      *     THIS IS THE SUB-SCHEMA FOR THE RETRIEVAL OF         *
      *     INFORMATION CONTAINED IN THE FOUR(4) FILES          *
      *     NODE-FILE, ROUT-FILE, ARGUM-FILE, PARAM-FILE.       *
      *                                                         *
      *     THESE FILES ARE LINKED THRU TWO(2) RELATIONS        *
      *     NAMELY ROUTRETRIEVE AND ROUTPARAM.                  *
      *                                                         *
      *     PRIMARY KEY IS--CLASSIF-NO--FOR NODE-FILE.          *
      *     PRIMARY KEY IS--ROUTINE-NO--FOR ROUT-FILE.          *
      *     ALTERNATE KEY IS--ROUT-NAME--FOR ROUT-FILE.         *
      *     PRIMARY KEY IS--ARG-NO--FOR ARGUM-FILE.             *
      *     ALTERNATE KEY IS--ROUT-NO--FOR ARGUM-FILE.          *
      *     PRIMARY KEY IS--ARG-NUM--FOR PARAM-FILE.            *
      *                                                         *
      ***********************************************************
      *
       TITLE DIVISION.
      * --------------
      *
       SS RETRIEVALSUB5 WITHIN HELPLIBRARY-DB.
      *
      *
       ALIAS DIVISION.
      * --------------
      *
       AD REALM TREEAREA BECOMES NODE-FILE.
       AD REALM ROUTAREA BECOMES ROUT-FILE.
       AD REALM ARGUMENTAREA BECOMES ARGUM-FILE.
       AD REALM PARAMETAREA BECOMES PARAM-FILE.
      *
       AD DATA LENGTH-IN-WORDS BECOMES NO-OF-WORDS.
       AD DATA NODE BECOMES NOTE.
       AD DATA NODE-DESCRIPTION BECOMES NODE-DESCR.
       AD DATA SOUR-LANG-CODE BECOMES S-L-C.
       AD DATA COMP-MACH-AV BECOMES C-M-AV.
```

```
 AD DATA LENG-OF-DESCR BECOMES L-OF-D.
*
*
 REALM DIVISION.
* ---------------
*
 RD NODE-FILE ROUT-FILE ARGUM-FILE PARAM-FILE.
*
*
 RECORD DIVISION.
* ----------------
*
*----------------------------------------------------------
*
*    DESCRIPTION OF NODE-REC OF NODE-FILE                  *
*
*----------------------------------------------------------
*
 01  NODE-REC.
*
     02 CLASSIF-NO      PIC X(13).
     02 RETRIEVAL-KEY REDEFINES CLASSIF-NO.
        03 ARITHM-PART PIC X(5).
        03 LETTER-PART PIC X(8).
     02 LIBRARY-INDEX   PIC A.
     02 NOTE            PIC AAAA.
     02 NO-OF-WORDS     PIC 999.
     02 NODE-DESCR      PIC X(10)
                        OCCURS 1 TO  20 TIMES
                        DEPENDING ON NO-OF-WORDS.
*
*----------------------------------------------------------
*
*    DESCRIPTION OF ROUT-REC OF ROUT-FILE                  *
*
*----------------------------------------------------------
*
 01  ROUT-REC.
*
     02 ROUTINE-NO      PIC 9(4).
     02 ROUT-NAME       PIC X(15).
     02 LIBR-NAME       PIC A(7).
     02 CLASS-NO        PIC X(13).
     02 ROUT-DETAILS      OCCURS 1 TIMES.
        03 AVAIL-LEVEL  PIC 9.
        03 SUP-LEV      PIC A.
```

```
           03 DOCUM-NO     PIC X(8).
           03 S-L-C        PIC A.
           03 PROGR-TYPE   PIC A.
           03 C-M-AV       PIC 9.
        02 L-OF-D          PIC 999.
        02 LINE-TEXT       PIC X(10)
                           OCCURS 1 TO  80 TIMES
                           DEPENDING ON L-OF-D.
*
*---------------------------------------------------
*
*     DESCRIPTION OF PARAM-REC OF ARGUM-FILE       *
*
*---------------------------------------------------
*
 01   PARAM-REC.

        02 ARG-NO          PIC 9(4).
        02 ROUT-NO         PIC 9(4).
        02 ARGUM-DETAILS   OCCURS 1 TIMES.
           03 ARGUM-NAME        PIC X(7).
           03 TYPE-OF      PIC A.
           03 INP-OUT      PIC AA.
        02 NO-OF-POINT     PIC 99.
        02 POINT-R         PIC 9(4)
                           OCCURS 0 TO 25 TIMES
                           DEPENDING ON NO-OF-POINT.
*
*---------------------------------------------------
*
*     DESCRIPTION OF PARAM-DESCR-REC OF PARAM-FILE *
*
*---------------------------------------------------
*
 01   PARAM-DESCR-REC.

        02 ARG-NUM         PIC 9(4).
        02 NUM-OF-WORDS    PIC 999.
        02 PAR-DESCR       PIC X(10)
                           OCCURS 1 TO  80 TIMES
                           DEPENDING ON NUM-OF-WORDS.
*
*
 RELATION DIVISION.
*
```

```
*-------------------------------------------------------------*
*                                                             *
*    RELATION-ROUTRETRIEVE  JOINS THE FILES                   *
*    NODE-FILE--ROUT-FILE                                     *
*                                                             *
*    RELATION-ROUTPARAM  JOINS THE FILES                      *
*    ROUT-FILE--ARGUM-FILE--PARAM-FILE                        *
*                                                             *
*-------------------------------------------------------------*
*
*    RN IS ROUTRETRIEVE.
*    RN IS ROUTPARAM.
*
```

3. APPLICATION PROGRAMS

```
PROJ.
STUDENT(7117149,PAPPAS9)                                        PAPPAS.
CHARGE(FKARED,BATCH)
FETCH(NODEFI=TEMP1,HELPLIB=CYL1SCH,SUBSLIB=CYL2CRS)
COBOL5(O=SUBSLIB)
LGO.
SAVE(TREEARE=PF1)
END OF RECORD
```

```
       IDENTIFICATION DIVISION.
      *
       PROGRAM-ID. DB-NODEFILE-CREATION.
      *-------------------------------
      *
      *****************************************************************
      *                                                               *
      *   THIS PROGRAM ACCESS THE SEQ/AL TEMPORARY FILE--NODEFI=TEMP1 *
      *   AND USING THE SUB-SCHEMA--TREEFILECREATIONSUB1--CREATES     *
      *   AN INDEX SEQ/AL FILE--TREEARE--WHICH IS SAVED UNDER         *
      *   PERMANENT NAME--PF1.                                        *
      *                                                               *
      *****************************************************************
      *
       AUTHOR. C. PAPPAS.
       INSTALLATION. MACMASTER.
       DATE-COMPILED.
      *
      *
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER.  CDC6400.
       OBJECT-COMPUTER.  CDC6400.
      *
       SPECIAL-NAMES.
           SUB-SCHEMA IS TREEFILECREATIONSUB1.
      *    ----------------------------------
      *
       INPUT-OUTPUT SECTION.
       FILE-CONTROL.
      *
      *---------------------------------------------------------------
      *
      *   NODEFI--IS THE INPUT TEMPORARY SEQ/AL FILE
      *   TREEARE--IS THE OUTPUT INDEX SEQ/AL FILE DECLARED IN THE
      *   SCHEMA.
      *
```

```
*-----------------------------------------------------------------
*
     SELECT TRANSFER-FILE          ASSIGN TO NODEFI.
     SELECT NODE-FILE              ASSIGN TO TREEARE.
*
*
 DATA DIVISION.
 FILE SECTION.
 FD   TRANSFER-FILE
      LABEL RECORDS ARE OMITTED.
 01   TRANSFER-REC.
*
*   THIS IS THE FORMAT OF THE INPUT RECORD
*   THE FILE DESRIPTION OF THE INDEX SEQ/AL OUTPUT FILE AND
*   OUTPUT RECORD ARE IN THE SUB-SCHEMA .
*
      02  TR-KEY              PIC X(13).
      02  TR-IND              PIC A.
      02  TR-NOTE             PIC A(4)..
      02  TR-WORDS            PIC 999.
      02  TR-TEXT             PIC X(10)
                              OCCURS 1 TO 20 TIMES
                              DEPENDING ON TR-WORDS.
*
 WORKING-STORAGE SECTION.
 77   SUBSCRIPT               PIC 999.
 01   CARDS-LEFT              PIC X(3).
*
*
 PROCEDURE DIVISION.
 MAIN-LINE.
      OPEN INPUT TRANSFER-FILE
           OUTPUT NODE-FILE.
      MOVE "YES" TO CARDS-LEFT.
      READ TRANSFER-FILE AT END MOVE "NO" TO CARDS-LEFT.
      PERFORM RECORDS-TRANSFER UNTIL CARDS-LEFT = "NO".
      PERFORM END-OF-JOB.
      STOP RUN.
*
 RECORDS-TRANSFER.
*
*-----------------------------------------------------------------
*
*   HERE WE TRANSFER THE ITEMS OF THE INPUT RECORD TO THE
*   OUTPUT RECORD
*   THE ITEMS ON THE RIGHT OF--TO--OF THE MOVE STATEMENT ARE
```

```
*  FOUND IN THE SUB-SCHEMA.
*-----------------------------------------------------------------
*
     MOVE TR-KEY TO CLASSIF-NO.
*  THE PRIMARY KEY IN THIS FILE IS--CLASSIF-NO.
     MOVE TR-IND TO LIBRARY-INDEX.
     MOVE TR-WORDS TO NO-OF-WORDS.
     PERFORM TEXT-TRANSFER VARYING SUBSCRIPT FROM 1 BY 1
        UNTIL SUBSCRIPT IS GREATER THAN NO-OF-WORDS.
     WRITE NODE-REC INVALID KEY PERFORM BAD-RECORD.
     READ TRANSFER-FILE AT END MOVE "NO" TO CARDS-LEFT.
TEXT-TRANSFER.
*-----------------------------------------------------------------
*
*  VARIABLE TEXT IS TRANSFERRED FROM THE INPUT RECORD TO
*  OUTPUT RECORD.
*-----------------------------------------------------------------
*
     MOVE TR-TEXT (SUBSCRIPT) TO NODE-DESCR (SUBSCRIPT).
BAD-RECORD.
*-----------------------------------------------------------------
*
*  IT DISPLAYS ANY RECORD WITH DOUBLE PRIMARY KEY.
*  THE WRONG(INVALID) RECORD IS NOT WRITTEN IN THE INDEX SEQ/AL
*  OUTPUT FILE.
*-----------------------------------------------------------------
*
     DISPLAY "INVALID RECORD.-" .
     DISPLAY NODE-REC.
END-OF-JOB.
     CLOSE TRANSFER-FILE NODE-FILE.
```

```
PROJ.                                                          PAP AS.
STUDENT(7117149,PAPP**59)
CHARGE(RKAREO,BATCH)
FETCH(ROUTFI=TEMP2,HELPLIB=CYL1SCH,SUBSLIB=CYL2CRS)
COBOL5(O=SUBSLIB)
LGO.
SAVE(ROUTARE=PF2)
END OF RECORD
```

```
       IDENTIFICATION DIVISION.
      *
       PROGRAM-ID. DB-ROUTFILE-CREATION.
      *-------------------------------
      *
      ************************************************************************
      *                                                                      *
      *   THIS PROGRAM ACCESS THE SEQ/AL TEMPORARY FILE--ROUTFI=TEMP2        *
      *   AND USING THE SUB-SCHEMA--ROUTFILECREATIONSUB2--CREATES            *
      *   AN INDEX SEQ/AL FILE--ROUTARE--WHICH IS SAVED UNDER                *
      *   PERMANENT NAME--PF2.                                               *
      *                                                                      *
      ************************************************************************
      *
       AUTHOR. C. PAPPAS.
       INSTALLATION. MACMASTER.
       DATE-COMPILED.
      *
      *
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER.  CDC6400.
       OBJECT-COMPUTER.  CDC6400.
      *
       SPECIAL-NAMES.
           SUB-SCHEMA IS ROUTFILECREATIONSUB2.
      *    ----------------------------------
      *
       INPUT-OUTPUT SECTION.
       FILE-CONTROL.
      *
      *-----------------------------------------------------------------
      *
      *   ROUTFI--IS THE INPUT TEMPORARY SEQ/AL FILE
      *   ROUTARE--IS THE OUTPUT INDEX SEQ/AL FILE DECLARED IN THE
      *   SCHEMA.
      *   ROUTIND--ANY INDEX SEQ/AL FILE HAVING ALTERNATE KEYS
```

```
*   DECLARED,AS HERE ROUT-NAME,HAS A FILE TO ACCOMODATE THEM
*   HERE IS -ROUTINO.
*
*   THE PRIMARY KEY IN ROUT-FILE IS--ROUTINE-NO
*   THE ALTERNATE KEY IN ROUT-FILE IS--ROUT-NAME
*
*---------------------------------------------------------------
*
     SELECT TRANSFER-FILE ASSIGN TO ROUTFI.
     SELECT ROUT-FILE ASSIGN TO ROUTARE, ROUTINO.
*
*
 DATA DIVISION.
 FILE SECTION.
 FD   TRANSFER-FILE
      LABEL RECORDS ARE OMITTED.
 01   TRANSFER-REC.
*
*   THIS IS THE FORMAT OF THE INPUT RECORD
*   THE FILE DESRIPTION OF THE INDEX SEQ/AL OUTPUT FILE AND
*   OUTPUT RECORD ARE IN THE SUB-SCHEMA .
*
      02  ROUTINE-NO     PIC 9(4).
      02  ROUT-NAME      PIC X(15).
      02  LIBR-NAME      PIC A(7).
      02  CLASS-NO       PIC X(13).
      02  ROUT-DETAILS.
        03  AVAIL-LEVEL  PIC 9.
        03  SUP-LEV      PIC A.
        03  DOCUM-NO     PIC X(8).
        03  S-L-C        PIC A.
        03  PROGR-TYPE   PIC A.
        03  C-M-AV       PIC 9.
      02  L-OF-D         PIC 999.
      02  TASK-OF        PIC X(10)
                         OCCURS 1 TO 80 TIMES
                         DEPENDING ON L-OF-D
                         OF TRANSFER-REC.
*
 WORKING-STORAGE SECTION.
 77   SUBSCRIPT          PIC 999.
 01   CARDS-LEFT           PIC X(3).
*
*
 PROCEDURE DIVISION.
 MAIN-LINE.
```

```
      OPEN INPUT TRANSFER-FILE
           OUTPUT ROUT-FILE.
      MOVE ≠YES≠ TO CARDS-LEFT.
      READ TRANSFER-FILE AT END MOVE ≠NO≠ TO CARDS-LEFT.
      PERFORM RECORDS-TRANSFER UNTIL CARDS-LEFT = ≠NO≠.
      PERFORM END-OF-JOB.
      STOP RUN.
*
 RECORDS-TRANSFER.
*
*---------------------------------------------------------------
*
*   INPUT RECORD --TRANSFER-REC--IS TRANFERED TO THE OUTPUT
*   RECORD--ROUT-REC.
*
*---------------------------------------------------------------
*
      MOVE CORR TRANSFER-REC TO ROUT-REC.
      PERFORM TEXT-TRANSFER VARYING SUBSCRIPT FROM 1 BY 1
           UNTIL SUBSCRIPT IS GREATER THAN L-OF-D OF ROUT-REC.
      WRITE ROUT-REC INVALID KEY PERFORM BAD-RECORD.
      READ TRANSFER-FILE AT END MOVE ≠NO≠ TO CARDS-LEFT.
*
 TEXT-TRANSFER.
*
*---------------------------------------------------------------
*
*   VARIABLE TEXT IS TRANSFERED FROM TRANSFER-REC TO ROUT-REC.
*
*---------------------------------------------------------------
*
     MOVE TASK-OF (SUBSCRIPT) TO TASK-OF-ROUT (SUBSCRIPT).
*
 BAD-RECORD.
*
*---------------------------------------------------------------
*
*   IT DISPLAYS ANY RECORD WITH DOUBLE PRIMARY KEY.
*   THE WRONG(INVALID) RECORD IS NOT WRITTEN IN THE INDEX SEQ/AL
*   OUTPUT FILE.
*
*---------------------------------------------------------------
*
      DISPLAY ≠INVALID RECORD.-≠ .
      DISPLAY ROUT-REC.
```

```
END-OF-JOB.
    CLOSE TRANSFER-FILE ROUT-FILE.
```

```
PROJ.                                                          PAPPAS.
STUDENT(7117149,PAPAS1)
CHARGE(FKARED,BATCH)
FETCH(HELPLIB=CYL1SCH,RETRLIB=CYL4R1S)
COBOL5(O=RETRLIB)
FTN.
SAVE(LGO=APLPPG1)
END OF RECORD

       IDENTIFICATION DIVISION.
      *
       PROGRAM-ID. RETRO1.
      *--------------------
      *
      ************************************************************
      *                                                          *
      *   THIS IS THE MOST IMPORTANT PROGRAM OF THE DATA-BASE    *
      *   IT IS AN INTERACTIVE PROGRAM, ALLOWING USER-MACHINE    *
      *   DIALOGUE.                                              *
      *                                                          *
      *   BY SUCCESSIVE INTERROGATION THE PROGRAM, BUILDS BY     *
      *   ITSELF STEP-BY-STEP THE SEARCH KEY TO ACCESS THE FILES.*
      *   IT ALSO PROVIDES, ERROR RECOVERY, BACK-UP, AND AT ANY  *
      *   TIME TERMINATION FACILITIES TO THE USER.               *
      *                                                          *
      *   THE OBJECTIVE OF THIS PROGRAM, IS TO SEARCH UNDER USER'S*
      *   CONTROL THE LIBRARY TREE TO RETRIEVE THE ROUTINES      *
      *   ASSOCIATED WITH THE USER'S SPECIFIC PROBLEM.           *
      *   IT GIVES ANY INFORMATION CONCERNING THE AVAILABLE      *
      *   ROUTINES AND THE CONTROL CARDS TO ACCESS THEM IN THE   *
      *   SYSTEM.                                                *
      *   THIS PROGRAM HANDLES THE TWO AREAS OF THE DATA-BASE    *
      *   WHICH STORE THE INFORMATION ABOUT FIELDS OF INTEREST AND*
      *   ROUTINES ASSOCIATED WITH THEM.                         *
      *                                                          *
      ************************************************************
      *
       AUTHOR. C. PAPPAS.
       INSTALLATION. MACMASTER.
       DATE-COMPILED.
      *
      *
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER.  CDC6400.
       OBJECT-COMPUTER.  CDC6400.
```

```
*
 SPECIAL-NAMES.
     SUB-SCHEMA IS RETRIEVALSUB5.
*    ----------------------------
*
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
*
*-----------------------------------------------------------------
*
*    NODE-FILE--IS THE INPUT INDEX SEQ/AL FILE STORING
*    THE DATA CONCERNING THE SUBJECTS OF INTEREST
*    ROUT-FILE--IS THE INPUT INDEX SEQ/AL FILE STORING
*    THE DATA CONCERNING THE ROUTINES
*    ROUTIND--IS THE INDEX FILE TO STORE THE ALTERNATE KEYS
*
*-----------------------------------------------------------------
*
     SELECT NODE-FILE ASSIGN TO TREEARE.
     SELECT ROUT-FILE ASSIGN TO ROUTARE, ROUTIND.
*
*
 DATA DIVISION.
 FILE SECTION.
*
 WORKING-STORAGE SECTION.
*-----------------------------------------------------------------
*
*    THE FOLLOWING DATA-ITEMS STORE THE KEYS OF THE FIRST AND
*    LAST RECORDS OF EACH OF THE 5 AVAILABLE GROUPS.
*    ACCORDING TO USER'S CHOICE THE PROPER 2 KEYS ARE BECOMING
*    CURRENT KEYS OF SEARCH.
*
*-----------------------------------------------------------------
*
 77   END-A              PIC X(13)          VALUE '01806'.
 77   GROUP-B            PIC X(13)          VALUE '01301'.
 77   END-B              PIC X(13)          VALUE '01502'.
 77   GROUP-C            PIC X(13)          VALUE '01602'.
 77   END-C              PIC X(13)          VALUE '01708'.
 77   GROUP-D            PIC X(13)          VALUE '04102'.
 77   END-D              PIC X(13)          VALUE '04404'.
 77   GROUP-E            PIC X(13)          VALUE '04501'.
 77   END-E              PIC X(13)          VALUE '04506'.
*
 77   GROUP-NO           PIC X.
```

```
*     IT ACCOMODATES (ACCEPTS) THE USER≠S CHOICE FOR GROUP
*     SELECTION.
*
 77   END-TEST              PIC X(13).
*     IT ACCOMODATES THE LAST KEY OF EACH GROUP, SO BECOMES THE
*     CRITERION OF SEARCH COMPLETION FOR THE PARTICULAR GROUP.
*
 77   HEAD-CNTR             PIC 9                VALUE ZERO.
*     IT TAKES CARE OF THE HEADER ≠SELECT ONE≠ TO BE DISPLAYED
*     ONLY ONCE AT THE TOP OF EACH RECORD DISPLAY.
*
 77   ROUT-CNTR             PIC 9                VALUE ZERO.
*     IT TAKES CARE OF THE HEADER ≠WE HAVE THE FOLL. ROUTINES≠
*     TO BE DISPLAYED ONLY ONCE AT THE TOP OF THE LIST OF ROUTINES
*     DISPLAYED.
*
 77   I                     PIC 99.
 77   COUNTER               PIC 99.
*     I--USED AS SUBSCRIPT TO SAVE THE DISPLAYED KEYS IN A TABLE,
*     FOR USE BY THE KEY-VALIDITY PAR.
*     COUNTER--FOR THE SAME REASON AT CHECKING TIME.
*
 77   SAVE-KEY              PIC X(13).
*     IT STORES ALWAYS THE PREVIOUS LEVEL CURRENT KEY, TO SERVE
*     THE BACK-UP FACILITY.
*
 77   X-6                   PIC X(6).
 77   X-7                   PIC X(7).
 77   X-8                   PIC X(8).
 77   X-9                   PIC X(9).
 77   X-10                  PIC X(10).
 77   X-11                  PIC X(11).
 77   X-12                  PIC X(12).
 77   DISPLAY-CHOICE        PIC X(13).
*     THESE ITEMS OF PROGRESSIVELY LARGER LENGTH, ARE USED AS
*     PART OF THE MECHANISM, TO CHOOSE ONLY THE RECORDS WHICH
*     ARE TO BE DISPLAYED.
*
 77   CUR-REC-WDS           PIC 999.
*     THIS ITEM STORES THE NUMB-OF-WORDS IN DISPLAY CODE OF CURRENT
*     RECORD TO BE DISPLAYED, AFTER CONVERSION HAS TAKEN PLACE.
*
 77   SUBSCRIPT             PIC 99.
 77   J                     PIC 9.
*     THESE SUBSCRIPTS ARE AUXILIARY AND USED IN THE HANDLING OF
*     ARRAYS, USED FOR THE TRANSFER OF VARIABLE LENGTH TEXT.
```

```
*
 77   OUTPUT-CRIT          PIC 9.
*     THIS ITEM TAKES CARE FOR THE DISPLAY OF THE OUTPUT LINE, AS
*     SOON AS IT IS FILLED WITH 72 CHARACTERS, TO FIT THE SCREEN
*     SIZE.
*
 77   WORDS-IN             USAGE IS COMPUTATIONAL-2.
 77   WORDS-OUT            USAGE IS COMPUTATIONAL-2.
*     THESE COMPUT-2 ITEMS ARE USED TO COMMUNICATE THE INPUT AND
*     OUTPUT OF NUM-OF-WORDS ITEMS TO THE FORTRAN ROUTINE.
*
 77   REC-ITEM             PIC X(4).
*     IT IS USED TO DENOTE TO THE VARIABLE TEXT TRANSFERRING
*     ROUTINES FROM  WHAT FILE (NODE OR ROUT) TEXT IS GOING TO BE
*     TRANSFERRED.
*
 77   WORD-S               PIC 999.
*     THIS ITEM STORES THE NUME-OF-WORDS IN ASCII TO BE TRANSFERRED
*     TO FTN CONVERSION ROUTINE TO BE CONVERTED TO DISPLAY CODE.
*
 77   RELN-CNTR            PIC 99.
*     COUNTS THE NUMBER OF RETRIEVED ROUTINES.
*
 77   LINE-CNTR            PIC 99.
*     IT STORES THE NUMBER OF LINES DISPLAYED AT EVERY DISPLAY OF
*     THE ROUTINES.
*
 01   TRIAL-LINE           PIC X(72).
*     THE SCREEN-OUTPUT LINE.
*
 01   NOMINAL-KEY.
*     IT STORES THE CURRENT SEARCH PRIMARY KEY TO BE PASSED TO
*     SUBSCHEMA KEY.
*     IT IS BUILT HERE DURING INTERACTIVE QUERY WITH THE USER.

      02  COMB-NUM-PART.
          03 LEVEL          PIC 9.
          03 NUMERIC-PART   PIC X(4).
      02 ALPHABETIC-PART    PIC X(8).
*
 01   KEY-ENTER-ITEM.
*     THIS ITEM ACCEPTS THE USER'S KEY REQUEST FROM THE TERMINAL
*
      02  NUM-IT            PIC X(4).
      02  ALPH-IT           PIC X(8).
*
```

```
 01   BACKUP-KEY.
*     IT STORES THE KEY FROM WHICH PREVIOUS LEVEL HAS STARTED
*     DISPLAYED.
*
      02 ITEM-1          PIC X.
      02 ITEM-2          PIC X(12).
*
 01   NODE-LINE-REC.
*     THIS VARIABLE LENGTH RECORD OUTPUTS THE INFORMATION IN THE
*     REQ'D FORMAT FROM NODE-FILE.
*
      02  GROUP-HEADER.
       03 CARR           PIC X              VALUE SPACES.
       03 DUMMY-KEY.
          04 L-N         PIC X.
          04 CLAS-CODE   PIC X(12).
       03 FILLER         PIC X.
      02 GR-N-TEXT.
         03 NODE-TEXT    PIC X(10)   OCCURS 5 TIMES.
*
 01   ROUT-LINE-REC.
*     THIS VARIABLE LENGTH RECORD OUTPUTS THE INFORMATION IN THE
*     REQ'D FORMAT FROM ROUT-FILE.
*     IN OTHER WORDS, CONVEYS THE ROUTINE INFORMATION.
*
      02 ROUT-HEADER.
        03 CAR           PIC X              VALUE SPACES.
        03 R-NO          PIC 9(4).
        03 FILL          PIC X(2).
        03 R-NA          PIC X(15).
      02 GROUP-TEXT.
        03 ROUT-TEXT     PIC X(10)  OCCURS 5 TIMES.
*
 01   CHECK-TABLE.
*     THIS TABLE IS USED TO STORE THE DISPLAYED KEYS, AT EVERY
*     DISPLAY, TO BE USED FOR ERROR CHECKING.
*
      02  CHECK-ITEM    PIC X(12)
          OCCURS 20 TIMES.
*
 01   DESCR-TABLE.
      02 TEMP-ITEM      PIC X(10)
         OCCURS 50 TIMES.
 01   DESCR-TABLE-OUT.
      02 TEMP-OUT       PIC X(10)
         OCCURS 100 TIMES.
```

```
*     THESE TWO TABLES COMMUNICATE WITH THE FTN CONVERSION
*     ROUTINE, THE VARIABLE TEXT TO BE CONVERTED.
*     FIRST IS USED FOR STORING THE ASCII TEXT BEFORE CONVERSION,
*     THE OTHER TO RECEIVE THE DISPLAY CODE AFTER CONVERSION FROM
*     FTN ROUTINE.
*
*
 PROCEDURE DIVISION.
 INITIALIZATION.
      OPEN INPUT NODE-FILE ROUT-FILE.
      MOVE SPACES TO KEY-ENTER-ITEM.
*
 INTRODUCTION-PAR.
*-----------------------------------------------------------------
*
*     THIS PARAGRAPH INCLUDES THE INSTRUCTIONS TO THE INTERACTIVE
*     USER, DISPLAYED AS SOON AS THE PROGRAM IS CALLED, IT ALSO
*     COMMUNICATES TO THE USER THE PURPOSE OF THE PROGRAM.
*
*-----------------------------------------------------------------
*
      DISPLAY ≠THIS PROGRAM DISPLAYS THE AREAS OF INTEREST≠.
      DISPLAY ≠IN THE LIBRARY AND YOU CHOOSE ONE AT A TIME≠.
      DISPLAY SPACES.
      DISPLAY ≠AFTER THE SYSTEM QUESTION-PLEASE,CHOOSE ONE≠.
      DISPLAY ≠ENTER A KEY NUMBER PRECEDING THE DESCRIPTION≠.
      DISPLAY ≠YOU HAVE SELECTED≠.
      DISPLAY SPACES.
      DISPLAY ≠ENTER THE KEYWORD-BACKUP-IF YOU WISH TO RETURN≠.
      DISPLAY ≠TO A PREVIOUS DISPLAY STAGE≠.
      DISPLAY SPACES.
      DISPLAY ≠ENTER THE KEYWORD-GIVEUP-IF YOU WISH TO INTERRUPT≠.
      DISPLAY ≠THE PROGRAM AT ANY STAGE≠.
      DISPLAY SPACES.
*
 DIALOGUE-PAR.
*-----------------------------------------------------------------
*
*     THIS PARAGRAPH DISPLAYS TO THE USER THE AREAS OF INTEREST,
*     FOR WHICH THERE ARE ROUTINES AVAILABLE, IN THE SYSTEM.
*     FROM THESE AREAS, THE USER MUST CHOOSE ONE.
*
*-----------------------------------------------------------------
*
      DISPLAY SPACES.
      DISPLAY ≠THERE ARE AVAILABLE THE FOLLOWING AREAS≠.
```

```
      DISPLAY ≠IN THE LIBRARY≠.
      DISPLAY SPACES.
      DISPLAY ≠      1. FACILITIES, SORTING, PLOTTING≠.
      DISPLAY ≠      2. STATISTICS≠.
      DISPLAY ≠      3. ENGINEERING≠.
      DISPLAY ≠      4. NUMERICAL ANALYSIS≠
      DISPLAY ≠         NOT INCLUDING VECTORS,EIGENVALUES≠
      DISPLAY ≠         AND MATRICES≠.
      DISPLAY ≠      5. NUMERICAL ANALYSIS,INCLUDING VECTORS,EIGENV≠
      DISPLAY ≠         AND MATRICES≠.
      DISPLAY SPACES.
      DISPLAY ≠PLEASE, CHOOSE ONE≠.
*
* GROUP-CHOICE-PAR.
*-----------------------------------------------------------------
*
*     HERE, THE PROGRAM BECOMES AWARE OF THE USER≠S CHOICE.
*
*-----------------------------------------------------------------
*
*    ACCEPT GROUP-NO.
*
* ZERO-LEVEL-PAR.
*-----------------------------------------------------------------
*
*     HERE, THE PROGRAM ANALYZES USER≠S CHOICE, AND BRANCHES TO
*     PERFORM HIS REQUEST.
*
*     IT ALSO TAKES CARE OF POSSIBLE USER≠S ERROR.
*
*     ACCESS ARM MOVES TO THE TOP OF THE AREA SELECTED AND
*     RETRIEVES ALL THE RECORDS UP TO THE END OF THE GROUP CHOSEN.
*
*-----------------------------------------------------------------
*
      IF GROUP-NO = 1 AND KEY-ENTER-ITEM = ≠BACKUP≠
                      MOVE ≠00003≠ TO RETRIEVAL-KEY
                      START NODE-FILE.
      IF GROUP-NO = 1 PERFORM READ-GROUP-PAR
        UNTIL RETRIEVAL-KEY = END-A
          ELSE IF GROUP-NO = 2 MOVE GROUP-B TO RETRIEVAL-KEY
                               MOVE END-B TO END-TEST
                               PERFORM START-GROUP-PAR
      ELSE IF GROUP-NO = 3 MOVE GROUP-C TO RETRIEVAL-KEY
                           MOVE END-C TO END-TEST
                           PERFORM START-GROUP-PAR
```

```
           ELSE IF GROUP-NO = 4 MOVE GROUP-D TO RETRIEVAL-KEY
                                MOVE END-C TO END-TEST
                                PERFORM START-GROUP-PAR
           ELSE IF GROUP-NO = 5 MOVE GROUP-E TO RETRIEVAL-KEY
                                MOVE END-E TO END-TEST
                                PERFORM START-GROUP-PAR
           ELSE IF GROUP-NO = ≠6≠ GC TO END-OF-JOB
           ELSE IF GROUP-NO = ≠8≠
           DISPLAY SPACES
           DISPLAY ≠WRONG USE OF THE FACILITY,YOU ARE JUST IN THE≠
           DISPLAY ≠BEGINING OF THE PROGRAM,SO ENTER A NUMBER 1-5≠
           DISPLAY ≠OR GIVEUP≠
           GO TO GROUP-CHOICE-PAR
           ELSE
           DISPLAY SPACES
                DISPLAY ≠WRONG INPUT NUMBER, ENTER AGAIN≠
           GO TO GROUP-CHOICE-PAR.
           DISPLAY SPACES.
           MOVE 0 TO LEVEL.
           MOVE 0 TO HEAD-CNTR.
*
 NOCE-CHOICE-PAR.
*----------------------------------------------------------------
*
*     HERE, USER CHOOSES A PARTICULAR, SPECIFIC FIELD FROM THE
*     GROUP OF THE RELATED FIELDS.
*     SYSTEM, CHECKS THE VALIDITY OF ITS CHOICE, TAKING CARE OF
*     POSSIBLE MISPUNCH AND ISSUING THE PROPER MESSAGE IN CASE OF
*     ERROR.
*
*----------------------------------------------------------------
*
           ACCEPT KEY-ENTER-ITEM.
                IF KEY-ENTER-ITEM = ≠GIVEUP≠ GO TO END-OF-JOB.
           IF KEY-ENTER-ITEM = ≠BACKUP≠ PERFORM DECISION-PAR
           ELSE
           MOVE 1 TO COUNTER
           MOVE LEVEL TO ITEM-1
           MOVE CHECK-ITEM (1) TO ITEM-2
           GO TO KEY-VALIDITY-PAR.
*
 PART-FOUND.
*----------------------------------------------------------------
*
*     PROGRAM BRANCHES HERE, IF USER HAS PUNCHED AN EXISTING FIELD
*     AND PREPARES TO EXECUTE THE USER≠S REQUEST.
```

```
*
*---------------------------------------------------------------
*
     MOVE NUM-IT TO NUMERIC-PART.
     MOVE ALPH-IT TO ALPHABETIC-PART.
     MOVE NOMINAL-KEY TO SAVE-KEY.
     ADD 1 TO LEVEL.
*
 EXTRA-PAR.
*---------------------------------------------------------------
*
*    HERE THE PROGRAM TRIES TO FIND, THE USER'S REQUEST.
*    IN CASE IT IS FOUND, IT WILL BE RETRIEVED, ELSE THE PROGRAM
*    WILL ISSUE PROPER MESSAGE AND WILL TERMINATE.
*
*---------------------------------------------------------------
*
     MOVE COMB-NUM-PART TO RETRIEVAL-KEY.
     PERFORM START-PAR.
     IF HEAD-CNTR = 0 PERFORM BACK-UP-PAR
     PERFORM END-OF-JOB
     ELSE
     MOVE 0 TO HEAD-CNTR.
     GO TO NOTE-CHOICE-PAR.
*
 START-GROUP-PAR.
*---------------------------------------------------------------
*
*    HERE, PROGRAM DIRECTS ACCESS ARM TO LOCATE THE USER'S
*    SELECTED AREA, ON THE PERIPHERAL, ARM LOCKS AT THE BEGINING
*    OF THE GROUP AND CALLS READ-GROUP-PAR TO READ SEQ/ALLY THE
*    RECORDS OF THE GROUP.
*
*---------------------------------------------------------------
*
     START NOTE-FILE.
     PERFORM READ-GROUP-PAR UNTIL
         RETRIEVAL-KEY = END-TEST.
*
 START-PAR.
*---------------------------------------------------------------
*
*    HERE, PROGRAM DIRECTS ACCESS ARM TO LOCATE THE USER'S
*    REQUESTED FIELD OF INTEREST WITHIN A GROUP AND START
*    READING SEQ/ALLY FURTHER ON.
*
```

```
*-----------------------------------------------------------------
*
     START NODE-FILE KEY = ARITHM-PART
     INVALID KEY PERFORM BACK-UP-PAR
     PERFORM END-OF-JOB.
     PERFORM READ-PAR UNTIL
       ARITHM-PART EXCEEDS COMB-NUM-PART.
*
 READ-PAR.
*-----------------------------------------------------------------
*
*    HERE, PROGRAM ORDERS THE SEQ/AL READING OF ALL RECORDS
*    WHICH ARE RELATED WITH THE USER'S REQUEST.
*
*-----------------------------------------------------------------
*
     READ NODE-FILE NEXT RECORD.
     IF ARITHM-PART = COMB-NUM-PART
       IF LEVEL = 1 PERFORM CRT-DISPLAY-PAR
         ELSE PERFORM REC-DISPL-PAR.
*
 REC-DISPL-PAR.
*-----------------------------------------------------------------
*
*    THIS IS THE SO CALLED 'FILTER' PAR, IT CHOOSES THE RECORDS
*    WHICH WILL BE DISPLAYED TO THE USER, OUT OF THE RECORDS
*    READ BY THE SYSTEM.
*
*-----------------------------------------------------------------
*
     IF LEVEL = 2 MOVE RETRIEVAL-KEY TO X-6
                  MOVE X-6 TO DISPLAY-CHOICE
     ELSE IF LEVEL = 3 MOVE RETRIEVAL-KEY TO X-7
                       MOVE X-7 TO DISPLAY-CHOICE
     ELSE IF LEVEL = 4 MOVE RETRIEVAL-KEY TO X-8
                       MOVE X-8 TO DISPLAY-CHOICE
     ELSE IF LEVEL = 5 MOVE RETRIEVAL-KEY TO X-9
                       MOVE X-9 TO DISPLAY-CHOICE
     ELSE IF LEVEL = 6 MOVE RETRIEVAL-KEY TO X-10
                       MOVE X-10 TO DISPLAY-CHOICE
     ELSE IF LEVEL = 7 MOVE RETRIEVAL-KEY TO X-11
                       MOVE X-11 TO DISPLAY-CHOICE
     ELSE IF LEVEL = 8 MOVE RETRIEVAL-KEY TO X-12
                       MOVE X-12 TO DISPLAY-CHOICE
       ELSE GO TO EXIT-PAR.
*    IF NOMINAL-KEY = DISPLAY-CHOICE
```

```
           PERFORM CRT-DISPLAY-PAR.
*
 EXIT-PAR.
     EXIT.
*
 BACK-UP-PAR.
*-----------------------------------------------------------------
*
*    THIS PARAGRAPH HAS A VERY IMPORTANT ROLE, IS CALLED IN THE
*    CASE WHERE NEXT LEVEL KEY IS NOT FOUND, IT MEANS, THAT A
*    LEAF OF THE TREE HAS BEEN REACHED.
*    IN THAT CASE, THE KEY OF THE PREVIOUS LEVEL, SAVED
*    IN--SAVE-KEY--ITEM BECOMES CUPRENT KEY AND SYSTEM SEARCHES
*    THE PREVIOUS LEVEL, WHICH IS THE LAST.
*
*-----------------------------------------------------------------
*
     MOVE SAVE-KEY TO RETRIEVAL-KEY.
     MOVE 0 TO RELN-CNTR.
     START NODE-FILE.
     PERFORM READ-RELATION-PAR.
*
 READ-RELATION-PAR.
*-----------------------------------------------------------------
*
*    HERE, SINCE THE USER'S RECORD HAS BEEN FOUND, THE SYSTEM
*    READS THE RELATION, SO RETRIEVING THE ROUTINES RELATED TO
*    THE REQUIRED NODE (FIELD OF INTEREST).
*
*    HERE, WE HAVE THE FOLL. CASES.
*    IF THERE ARE RECORDS IN ROUT-FILE HAVING ITEM--CLASS-NO
*    EQUAL SAVE-KEY, THESE RECORDS ARE RETRIEVED AND DISPLAYED,
*    AS BEEING THE AVAILABLE ROUTINES.
*
*    IF '1' SYMBOLS ARE ENCOUNTERED, IT MEANS THE END OF LEVEL
*    HAS BEEN ENCOUNTERED.
*
*    IF '1' SYMBOLS ARE ENCOUNTERED AND RELN-CNTR=0, THEN NOT
*    ROUTINES ARE AVAILABLE, FOR THE PARTICULAR FIELD.
*
*-----------------------------------------------------------------
*
     READ ROUTRETRIEVE NEXT RECORD AT END
          GO TO END-OF-JOB.
     IF CLASS-NO OF POUT-REC = SAVE-KEY
     ADD 1 TO RELN-CNTR
```

```
      PERFORM CRT-ROUT-DISPLAY THRU CRT-ROUT-LABEL
      GO TO READ-RELATION-PAR
      ELSE IF ROUTINE-NO OF ROUT-REC = ALL '1'
                        AND RELN-CNTR = 0
          OR ROUT-NAME OF ROUT-REC = ALL '1'
                        AND RELN-CNTR = 0
              PERFORM MESSAGE-1
      ELSE DISPLAY SPACES
      DISPLAY 'HERE ARE ALL THE AVAILABLE ROUTINES'
      DISPLAY 'FOR THIS PURPOSE'
      PERFORM FURTHER-QUESTION-PAR.
      MOVE 0 TO ROUT-CNTR.
*
 CRT-DISPLAY-PAR.
*---------------------------------------------------------------
*
*     THIS PARAGRAPH TAKES CARE FOR THE DISPLAY OF THE INFORMATION
*     STORED IN THE PARTICULAR RECORD OF THE NODE-FILE REQUIRED
*     BY THE USER.
*
*---------------------------------------------------------------
*
      IF HEAD-CNTR = 0 MOVE 1 TO HEAD-CNTR  MOVE 0 TO I
      DISPLAY SPACES
          DISPLAY 'SELECT ONE'.
          DISPLAY SPACES.
      ADD 1 TO I.
      MOVE NO-OF-WORDS OF NODE-REC TO WORDS-IN, WORD-S.
      MOVE 'NODE' TO REC-ITEM.
      PERFORM DISPL-CONV-PAR.
      MOVE WORDS-OUT TO CUR-REC-WDS.
      MOVE SPACES TO GR-N-TEXT.
      MOVE RETRIEVAL-KEY OF NODE-REC TO DUMMY-KEY
                                OF NODE-LINE-REC.
      MOVE SPACE TO L-N.
      MOVE CLAS-CODE OF NODE-LINE-REC TO CHECK-ITEM(I).
      PERFORM TEXT-MOVING-PAR.
*
 MESSAGE-1.
*---------------------------------------------------------------
*
*     THIS MESSAGE IS DISPLAYED WHEN THE SYSTEM DOES NOT FIND
*     ROUTINE FOR THE PARTICULAR PURPOSE.
*
*---------------------------------------------------------------
*
```

```
     RERFORM SPACE-PAR 3 TIMES.
     DISPLAY ≠NOT ANY AVAILABLE ROUTINE, FOR THIS PURPOSE≠.
*
 CRT-ROUT-DISPLAY.
*-----------------------------------------------------------------
*
*    THIS PARAGRAPH TAKES CARE FOR THE DISPLAY OF THE HEADERS
*    OF THE RETRIEVED ROUTINES.
*    IT ALSO TAKES CARE TO HANDLE THE SCREEN VERTICAL SHIFTING
*    IN CASE THE TEXT EXCEEDS THE MAXIMUM OF 24 LINES PER SCREEN
*
*-----------------------------------------------------------------
*
     IF ROUT-CNTR = 0 MOVE 1 TO ROUT-CNTR
     PERFORM SPACE-PAR 2 TIMES
     MOVE 3 TO LINE-CNTR
     DISPLAY ≠WE HAVE THE FOLL.ROUTINES≠.
     DISPLAY SPACES.
     ADD 1 TO LINE-CNTR.
     IF LINE-CNTR = 21 PERFORM PASS-PAR
     DISPLAY SPACES.
     MOVE ROUTINE-NO TO R-NO.
*
 CRT-ROUT-LABEL.
*-----------------------------------------------------------------
*
*    THIS PARAGRAPH TAKES CARE FOR THE DISPLAY OF THE INFORMATION
*    STORED IN THE PARTICULAR RECORD OF THE ROUT-FILE REQUIRED
*    BY THE USER.
*
*-----------------------------------------------------------------
*
     MOVE ROUT-NAME TO R-NA.
     MOVE ≠ROUT≠ TO REC-ITEM.
     MOVE SPACES TO FILL, GROUP-TEXT.
     MOVE L-OF-O TO WORDS-IN, WORD-S.
     PERFORM DISPL-CONV-PAR.
     MOVE WORDS-OUT TO CUR-REC-WDS.
     PERFORM TEXT-MOVING-PAR.
*
 END-OF-JOB.
     CLOSE MODE-FILE ROUT-FILE.
     STOP RUN
*
 READ-GROUP-PAR.
*-----------------------------------------------------------------
```

```
*
*     HERE, PROGRAM ORDERS THE SEQ/AL RETRIEVAL OF ALL THE FIELDS
*     OF THE USER*S SELECTED GROUP.
*
*-----------------------------------------------------------------
*
      READ NODE-FILE NEXT RECORD.
      PERFORM CRT-DISPLAY-PAR.
*
 KEY-VALIDITY-PAR.
*-----------------------------------------------------------------
*
*     THIS PARAGRAPH CHECKS FOR THE VALIDITY OF THE KEYS, ENTERED
*     BY THE USER.
*
*     THIS IS DONE AS FOLLOWS.-
*     WHEN CRT-DISPLAY-PAR DISPLAYS THE RECORDS IT SAVES THE KEYS
*     IN A TABLE--CHECK-TABLE IN WORKING STORAGE THRU VARIABLE--I
*     WHEN THE USER ENTERS A KEY, THIS PARAGRAPH SEARCHES THE
*     TABLE TO FIND THE KEY, IF NOT DISPLAYS MESSAGE, ELSE
*     PROCEEDS TO RETRIEVE.
*
*-----------------------------------------------------------------
*
      IF CHECK-ITEM (COUNTER) = KEY-ENTER-ITEM GO TO PART-FOUND.
      IF COUNTER IS GREATER THAN I
      DISPLAY SPACES
         DISPLAY *YOU HAVE JUST ENTERED WRONG KEY*
         DISPLAY *REENTER CAREFULLY,ONE OF THE DISPLAYED KEYS.*
      GO TO NODE-CHOICE-PAR.
      ADD 1 TO COUNTER.
      GO TO KEY-VALIDITY-PAR.
*
 DISPL-CONV-PAR.
*-----------------------------------------------------------------
*
*     THIS PARAGRAPH CONTROLS THE TRANSFER OF THE ASCII VARIABLE
*     TEXT FROM THE FILE RECORDS FOR CONVERSION TO DISPLAY CODE
*     AND INTERFACES THE FORTRAN SUBROUTINE FOR THAT.
*
*-----------------------------------------------------------------
*
      MOVE SPACES TO DESCR-TABLE.
      PERFORM TRANSFER-PROCEDURE VARYING SUBSCRIPT FROM 1 BY 1
            UNTIL SUBSCRIPT IS GREATER THAN WORD-S.
      MOVE SPACES TO DESCR-TABLE-OUT.
```

```
     ENTER FORTRAN-X ASCII USING DESCR-TABLE, DESCR-TABLE-OUT
                                 WORDS-IN, WORDS-OUT.
*
 TRANSFER-PROCEDURE.
*-----------------------------------------------------------------
*
*    THIS PARAGRAPH ACTUALLY TRANSFERS THE VARIABLE ASCII TEXT
*    FROM EITHER FILE TO TEMP-ITEM TABLE IN WORKING STORAGE
*    BEFORE PASSES TO CONVERSION FTN SUBROUTINE.
*
*-----------------------------------------------------------------
*
     IF REC-ITEM = ≠NODE≠
     MOVE NODE-DESCR (SUBSCRIPT) TO TEMP-ITEM (SUBSCRIPT)
     ELSE
     MOVE LINE-TEXT (SUBSCRIPT) TO TEMP-ITEM (SUBSCRIPT).
*
 TEXT-MOVING-PAR.
*-----------------------------------------------------------------
*
*    THIS PARAGRAPH CONTROLS THE TRANSFER OF THE CONVERTED
*    VARIABLE DISPLAY CODE TO THE OUTPUT RECORDS.
*
*-----------------------------------------------------------------
*
     MOVE 1 TO SUBSCRIPT, J
     PERFORM TEXT-TRANSFER-PAR UNTIL SUBSCRIPT IS GREATER
                                     THAN CURREC-WDS.
     IF OUTPUT-CRIT IS NOT EQUAL TO 0 PERFORM TEXT-DISPLAY-PAR.
*
 TEXT-TRANSFER-PAR.
*-----------------------------------------------------------------
*
*    THIS PARAGRAPH TRANSFERS THE CONVERTED DISPLAY CODE OF THE
*    VARIABLE TEXT FROM TEMP-OUT TABLE OF WORKING STORAGE TO THE
*    OUTPUT RECORD WHICH WILL BE DISPALYED ON THE SCREEN.
*    IT ALSO CONTROLS THE NUMBER OF CHARACTERS PER LINE NOT TO
*    EXCEED THE MAXIMUM OF 72 PER LINE..
*
*-----------------------------------------------------------------
*
     IF REC-ITEM = ≠NODE≠
     MOVE TEMP-OUT (SUBSCRIPT) TO NODE-TEXT (J)
     ELSE
     MOVE TEMP-OUT (SUBSCRIPT) TO ROUT-TEXT (J).
     SUBTRACT J FROM 5 GIVING OUTPUT-CRIT.
```

```
     ADD 1 TO SUBSCRIPT, J.
     IF OUTPUT-CRIT = 0 PERFORM TEXT-DISPLAY-PAR
                        MOVE 1 TO J.
*
 TEXT-DISPLAY-PAR.
*----------------------------------------------------------------
*
*    THIS PARAGRAPH TRANSFERS THE OUTPUT RECORDS FROM EITHER
*    FILE, CONTAINING THE INFORMATION TO BE DISPLAYED ON THE
*    SCREEN, TAKING CARE THE NUMBER OF LINES NOT TO EXCEED THE
*    MAXIMUM OF 24 PER SCREEN AT A TIME.
*
*----------------------------------------------------------------
*
     MOVE SPACES TO TRIAL-LINE.
     IF REC-ITEM = ≠NODE≠
     MOVE NODE-LINE-REC TO TRIAL-LINE
     DISPLAY TRIAL-LINE
     MOVE SPACES TO NODE-LINE-REC
     ELSE
     MOVE ROUT-LINE-REC TO TRIAL-LINE
     DISPLAY TRIAL-LINE
     ADD 1 TO LINE-CNTR
     MOVE SPACES TO ROUT-LINE-REC
     IF LINE-CNTR = 21 PERFORM PASS-PAR.
*
 FURTHER-QUESTION-PAR.
*----------------------------------------------------------------
*
*    THIS PARAGRAPH GIVES SUPPLEMENTARY INFORMATION CONCERNING
*    ANY PARTICULAR ROUTINE, FROM THE ONES DISPLAYED TO THE USER
*    UPON USER≠S REQUEST.
*
*    THIS INFORMATION INCLUDES;
*    AVAILABILITY LEVEL, SUPPORT LEVEL, CONTROL CARDS TO CALL
*    AND USE THE ROUTINES.
*
*----------------------------------------------------------------
*
     PERFORM SPACE-PAR 3 TIMES.
     DISPLAY ≠IF YOU WANT MORE INFORMATION ABOUT A PARTICULAR≠
     DISPLAY ≠ROUTINE,ENTER THE PRECEEDING NUMBER≠
     DISPLAY ≠ELSE ENTER 0000≠.
     ACCEPT R-NO.
     IF R-NO = 0000 GO TO END-OF-JOB
     ELSE
```

```
MOVE R-NO TO ROUTINE-NO.
READ ROUT-FILE INVALID KEY GO TO FURTHER-QUESTION-PAR.
PERFORM SPACE-PAR 2 TIMES.
DISPLAY ≠IF YOU WISH TO SEE AGAIN THE ROUTINE DISPLAY≠
DISPLAY ≠ENTER THE KEYWORD- YES  -ELSE-NO≠.
ACCEPT KEY-ENTER-ITEM.
IF KEY-ENTER-ITEM = ≠YES≠ PERFORM CRT-POLI-LABEL
                          MOVE 0 TO LINE-CNTR.
DISPLAY SPACES.
MOVE ROUT-NAME TO R-NA.
IF AVAIL-LEVEL = 1
DISPLAY ≠THIS ROUTINE IS AVAILABLE ON LINE TO ALL JOBS≠
DISPLAY ≠USING A SINGLE CONTROL CARD OR FTN CARD≠
ELSE
IF AVAIL-LEVEL = 2
DISPLAY ≠THIS ROUTINE IS AVAILABLE ON LINE, BUT AN≠
DISPLAY ≠ADDITIONAL CONTROL CARD IS REQUIRED TO MAKE IT≠
DISPLAY ≠AVAILABLE TO YOUR JOB≠
DISPLAY SPACES
DISPLAY ≠ THE ROUTINE--≠ R-NA ≠--CAN BE ACCESSED FOR USE≠
DISPLAY ≠WITH YOUR PROGRAM BY THE FOLLOWING CONTROL CARDS≠
DISPLAY SPACES
DISPLAY ≠          GRAB(≠ LIBR-NAME ≠)≠
DISPLAY ≠          LOSET(LIB=≠ LIBR-NAME ≠)≠
DISPLAY SPACES
ELSE
IF AVAIL-LEVEL = 3
DISPLAY ≠THIS ROUTINE IS NOT AVAILABLE ON LINE. ASK FOR≠
DISPLAY ≠THE SOURCE DECK FROM THE APPLICATION LIBRARIAN≠
DISPLAY ≠AT ROOM SS-270≠
ELSE
DISPLAY SPACES.
PERFORM SPACE-PAR 3 TIMES.
IF SUP-LEV = ≠A≠
DISPLAY ≠THIS ROUTINE IS FULLY SUPPORTED BY THE COMPUTER≠
DISPLAY ≠CENTER AND FULL PRIORITY WILL BE GIVEN IN ITS≠
DISPLAY ≠DEBUGGING, BY THE PROGRAMMING ASSISTANT≠
ELSE
IF SUP-LEV = ≠B≠
DISPLAY ≠IT IS SUPPORTED BY COMPUTER CENTER IN THE SENSE≠
DISPLAY ≠THAT ERRORS WILL BE CORRECTED WHEN RESOURCES≠
DISPLAY ≠AVAILABLE PERMIT. AVOID USING IN NEW PROGRAMS≠
ELSE
IF SUP-LEV = ≠X≠
DISPLAY ≠ROUTINE FULLY SUPPORTED BY EXTERNAL SOURCES≠
ELSE
```

```
      IF SUP-LEV = ≠U≠
      DISPLAY ≠ROUTINE UNSUPPORTED, CORRECTION AT THE≠
      DISPLAY ≠RESPONSIBILITY OF THE USER≠
      ELSE
      DISPLAY SPACES.
      PERFORM SPACE-PAR 3 TIMES.
      MOVE SPACES TO KEY-ENTER-ITEM.
      IF RELN-CNTR IS GREATER THAN 1
      DISPLAY ≠DO YOU WANT INFORMATION ABOUT ANOTHER ROUTINE≠
      DISPLAY ≠ENTER YES OR NO≠
      ACCEPT KEY-ENTER-ITEM
      IF KEY-ENTER-ITEM = ≠YES≠
      GO TO FURTHER-QUESTION-PAR.
*
 SPACE-PAR.
*-----------------------------------------------------------------
*
*     THIS PARAGRAPH IS CALLED FROM DIFFERENT PARTS OF THE
*     PROGRAM, TO TAKE CARE OF THE SPACE CONTROL BETWEEN THE
*     MESSAGES.
*
*-----------------------------------------------------------------
*
      DISPLAY SPACES.
*
 DECISION-PAR.
*-----------------------------------------------------------------
*
*     THIS PARAGRAPH PROVIDES THE FACILITY TO THE USER TO RECALL
*     THE PREVIOUS LEVEL OF DISPLAY IN CASE IS NOT SATISFIED WITH
*     THE FIELD HE HAS CHOSEN.
*     SO HE CAN MAKE ANOTHER CHOICE.
*
*-----------------------------------------------------------------
*
      IF LEVEL = 0 GO TO DIALOGUE-PAR
      ELSE IF LEVEL = 1 GO TO ZERO-LEVEL-PAR
      ELSE MOVE BACKUP-KEY TO NOMINAL-KEY
                                   PERFORM EXTRA-PAR.
*
 PASS-PAR.
*-----------------------------------------------------------------
*
*     THIS PARAGRAPH IS CALLED WHEN THE TEXT DISPLAYED, FINALLY
*     OCCUPIES THE ENTIRE SCREEN, AND THERE IS MORE TEXT COMING.
*     SCREEN SHIFTING PASSES TO THE CONTROL OF THE USER.
```

```
*    USER SO HOLDS SCREEN MOTIONLESS TO GIVE TIME TO READ IT,
*    AND THEN RELEASES IT.
*
*-----------------------------------------------------------------
*
     DISPLAY ≠READ ROUTINE DESCRIPTION,HOLD ANY PROBABLY WANTED≠
     DISPLAY ≠ROUTINE NUMBERS,THEN ENTER THE KEYWORD-PASS-TO≠
     DISPLAY ≠CONTINUE THE ROUTINE DISPLAY≠.
     ACCEPT KEY-ENTER-ITEM.
     IF KEY-ENTER-ITEM = ≠PASS≠ MOVE 0 TO LINE-CNTR
     ELSE GO TO PASS-PAR.
*
```

```
      SUBROUTINE ASCII(IN,IOUT,TEM1,TEM2)
C
CCC   ***************************************************************
CCC   *                                                             *
CCC     THIS ROUTINE CALLS THREE OTHER ROUTINES AND ALL TOGETHER CONVERT
CCC     THE INPUT ASCII CODE (IN-ARRAY) TO DISPLAY CODE (IOUT-ARRAY).
CCC   *                                                             *
CCC     IN-ARRAY TO STORE THE INPUT ASCII CODE.
CCC     IOUT-ARRAY TO RETURN THE DISPLAY CODE.
CCC   *                                                             *
CCC   ***************************************************************
C
        DIMENSION IN(50),IOUT(100),ISCR(100)
        INTEGER OUTCNT,CH,FILLER
        NOFWIN=TEM1
C
        DO 5 I=1,100
      5 IOUT(I)=0
C       INITIALIZE TO 0 ALL THE -IOUT- ARRAY, BEFORE APPLYING THE .OR.
CC      OPERATION.
C
        INCNT=0
C       IT COUNTS THE NUMBER OF CHARACTERS PER WORD IN THE--IN--ARRAY,
CC      THAT IS, THE NUMBER OF 7-BIT GROUPS.
C
        OUTCNT=0
C       IT COUNTS THE NUMBER OF CHARACTERS PER WORD IN THE--IOUT--ARRAY,
CC      THAT IS, THE NUMBER OF 12-BIT GROUPS.
C
        LIMIT=8*NOFWIN
C       LIMIT--IT COUNTS THE NUMBER OF CHARACTERS TO BE CONVERTED.
C
        DO 10 I=1,LIMIT
        CALL GETCH(CH,IN,INCNT,OUTCNT)
        CALL PUTCH(CH,IOUT,INCNT,OUTCNT)
     10 CONTINUE
C
        FILLER=10-MOD(LIMIT,10)
C       FILLER--IT COUNTS THE NUMBER OF LEFT CHARACTER SPACES AT THE LAST
CC      WORD OF OUTPUT (CONVERTED) ARRAY.
C
        IF(FILLER.GT.0) CALL FILL(FILLER,IOUT,INCNT,OUTCNT)
        NOFWOT=OUTCNT/10
        K=OUTCNT/5
        DO 20 I=1,K
     20 ISCR(I)=IOUT(I)
```

```
      CALL XMOVE(2HAX,ISCR,IOUT,OUTCNT)
      TEM2=.ICFWOT
      RETURN
      END
      SUBROUTINE GETCH(CH,IN,INCNT,CUTCNT)
C     ************************************************************************
C     *                                                                      *
C     *  THIS ROUTINE IS CALLED TO REMOVE A CHARACTER FROM AN ARRAY.         *
C     *                                                                      *
C     ************************************************************************
C
      DIMENSION IN(50)
      INTEGER WORD,OUTCNT,CHAR,CH
      INCNT=INCNT+1
C
C     WORD=(INCNT-1)/8+1
C     WORD-VARIABLE TO STORE A FULL WORD OF ASCII CHARACTERS.
C
C     CHAR=MOD(INCNT-1,8)+1
C     AFTER EVERY WORD IS STRIPPED, STARTS FROM 1 AGAIN.
C
      CH=SHIFT( IN(WORD),-(8-CHAR)*7).AND.177B
C     A CHARACTER IN 7-BIT FORM IS SENT OUTSIDE FROM LEFT TO RIGHT,
C     STORED IN THE VARIABLE-CH.
C
      RETURN
      END
      SUBROUTINE PUTCH(CH,IOUT,INCNT,OUTCNT)
C     ************************************************************************
C     *                                                                      *
C     *  THIS SUBROUTINE IS CALLED TO PUT A CHARACTER INTO AN ARRAY, AT      *
C     *  A TIME.                                                             *
C     *  TO FILL AN ARRAY, IS CALLED SEVERAL TIMES.                          *
C     *                                                                      *
C     ************************************************************************
C
      DIMENSION IOUT(100)
      INTEGER WORD,OUTCNT,CHAR,CH
      OUTCNT=OUTCNT+1
      WORD=(OUTCNT-1)/5+1
      CHAR=MOD(OUTCNT,5)
      IOUT(WORD)=IOUT(WORD).OR.SHIFT(CH,(5-CHAR)*12)
      RETURN
      END
```

```
      SUBROUTINE FILL(FCNT,IOUT,INCNT,OUTCNT)
C     ***************************************************************
C     *                                                             *
C     *  THIS ROUTINE PERFORMS THE TASK OF FILLING WITH BLANK FILLER THE
C     *  RIGTH PART OF THE LAST WORD OF THE DISPLAY TEXT.           *
C     *                                                             *
C     ***************************************************************
C
      DIMENSION IOUT(100)
      INTEGER FCNT,OUTCNT
C
C     FCNT-IT COUNTS THE NUMBER OF UNUSED BYTES IN THE LAST WORD TO BE
C     FILLED WITH BLANKS.
C
      DO 10 I=1,FCNT
      CALL PUTCH(40B,IOUT,INCNT,OUTCNT)
   10 CONTINUE
      RETURN
      END
```

APPENDIX B

QUERY-ANSWER SAMPLE OUTPUTS

```
 THIS PROGRAM DISPLAYS THE AREAS OF INTEREST
 IN THE LIBRARY AND YOU CHOOSE ONE AT A TIME                SAMPLE 1

 AFTER THE SYSTEM QUESTION-PLEASE, CHOOSE ONE
 ENTER A KEY NUMBER PRECEDING THE DESCRIPTION
 YOU HAVE SELECTED

 ENTER THE KEYWORD-BACKUP-IF YOU WISH TO RETURN
 TO A PREVIOUS DISPLAY STAGE

 ENTER THE KEYWORD-GIVEUP-IF YOU WISH TO INTERRUPT
 THE PROGRAM AT ANY STAGE


 THERE ARE AVAILABLE THE FOLLOWING AREAS
 IN THE LIBRARY

      1. FACILITIES, SORTING, PLOTTING
      2. STATISTICS
      3. ENGINEERING
      4. NUMERICAL ANALYSIS
         NOT INCLUDING VECTORS,EIGENVALUES
         AND MATRICES
      5. NUMERICAL ANALYSIS,INCLUDING VECTORS,EIGENV
         AND MATRICES

 PLEASE, CHOOSE ONE

   ? 1

SELECT ONE

  0003          TAPE HANDLING.
  0100          UTILITY (INTERNAL) PROGRAMS
  0106          FILE ORGANIZATION.
  0109          TIMING
  0302          COMPILERS
  0306          PREPROCESSING AND EDITING
  0601          SORTING
```

```
   0704          INPUT
   0804          OUTPUT - B C D
   0806          PLOTTING
      ? 0109
WE HAVE THE FOLL.ROUTINES
  8450  TOCK          TO TIME PARTS OF A PROGRAM IN SECONDS. THREE ENTRY
                      POINTS ARE TOCKP, TOCKS, TOCKSP

HERE ARE ALL THE AVAILABLE ROUTINES
FOR THIS PURPOSE


IF YOU WANT MORE INFORMATION ABOUT A PARTICULAR
ROUTINE,ENTER THE PRECEEDING NUMBER
ELSE ENTER 0
      ? 8450

IF YOU WISH TO SEE AGAIN THE ROUTINE DISPLAY
ENTER THE KEYWORD- YES  -ELSE-NO
      ? NO
THIS ROUTINE IS AVAILABLE ON LINE TO ALL JOBS
USING A SINGLE CONTROL CARD OR FTN CARD


THIS ROUTINE IS FULLY SUPPORTED BY THE COMPUTER
CENTER AND FULL PRIORITY WILL BE GIVEN IN ITS
DEBUGGING, BY THE PROGRAMMING ASSISTANT
```

SAMPLE 2.

```
THIS PROGRAM DISPLAYS THE AREAS OF INTEREST
IN THE LIBRARY AND YOU CHOOSE ONE AT A TIME

AFTER THE SYSTEM QUESTION-PLEASE, CHOOSE ONE
ENTER A KEY NUMBER PRECEDING THE DESCRIPTION
YOU HAVE SELECTED

ENTER THE KEYWORD-BACKUP-IF YOU WISH TO RETURN
TO A PREVIOUS DISPLAY STAGE

ENTER THE KEYWORD-GIVEUP-IF YOU WISH TO INTERRUPT
THE PROGRAM AT ANY STAGE

THERE ARE AVAILABLE THE FOLLOWING AREAS
IN THE LIBRARY

     1. FACILITIES, SORTING, PLOTTING
     2. STATISTICS
     3. ENGINEERING
     4. NUMERICAL ANALYSIS
        NOT INCLUDING VECTORS,EIGENVALUES
        AND MATRICES
     5. NUMERICAL ANALYSIS,INCLUDING VECTORS,EIGENV
        AND MATRICES

PLEASE, CHOOSE ONE

      ?1

    0003        TAPE HANDLING.
    0100        UTILITY (INTERNAL) PROGRAMS
    0106        FILE ORGANIZATION.
    0109        TIMING
    0302        COMPILERS
    0306        PREPROCESSING AND EDITING
    0601        SORTING
    0606        CHARACTER AND SYMBOL MANIPULATION
```

0704 INPUT

0804 OUTPUT - B C D

0806 PLOTTING

? 0601

SELECT ONE

0601A RECORDS DO NOT EXCEED STORAGE LIMITATIONS OF COMPUTER.

0601B RECORDS EXCEED STORAGE LIMITATIONS OF COMPUTER

? 0601A

SELECT ONE

0601AA MULTI-WORD RECORDS

0601AB SINGLE-WORD RECORDS

? 0601AA

SELECT ONE

0601AAA SORTING ON ONE COMPLETE WORD OF A RECORD OM

0601AAB OTHER

? 0601AAB

NOT ANY AVAILABLE ROUTINE, FOR THIS PURPOSE

SAMPLE 3

```
THIS PROGRAM DISPLAYS THE AREAS OF INTEREST
IN THE LIBRARY AND YOU CHOOSE ONE AT A TIME

AFTER THE SYSTEM QUESTION-PLEASE,CHOOSE ONE
ENTER A KEY NUMBER PRECEDING THE DESCRIPTION
YOU HAVE SELECTED

ENTER THE KEYWORD-BACKUP-IF YOU WISH TO RETURN
TO A PREVIOUS DISPLAY STAGE

ENTER THE KEYWORD-GIVEUP-IF YOU WISH TO INTERRUPT
THE PROGRAM AT ANY STAGE

THERE ARE AVAILABLE THE FOLLOWING AREAS
IN THE LIBRARY

     1. FACILITIES, SORTING, PLOTTING
     2. STATISTICS
     3. ENGINEERING
     4. NUMERICAL ANALYSIS
        NOT INCLUDING VECTORS,EIGENVALUES
        AND MATRICES
     5. NUMERICAL ANALYSIS,INCLUDING VECTORS,EIGENV
        AND MATRICES

PLEASE, CHOOSE ONE
     ?1

     0003        TAPE HANDLING.
     0100        UTILITY (INTERNAL) PROGRAMS
     0106        FILE ORGANIZATION.
     0109        TIMING
     0302        COMPILERS
     0306        PREPROCESSING AND EDITING
     0601        SORTING
     0606        CHARACTER AND SYMBOL MANIPULATION
```

```
0804        OUTPUT - B C D
0806        PLOTTING

   ? 0003

SELECT ONE
  0003A     VOLUME.

   ? 0003A

SELECT ONE
  0003AB    COPY

   ? 0003AB

WE HAVE THE FOLL. ROUTINES
 2070  COPYST    UTILITY PROGRAM TO CONVERT A CODED SCOPE FILE TO A
                 STRANGER TAPE - USEFUL IN TRANSFERRING DATA FILES
                 TO OTHER INSTALLATIONS
 8460  TPCON     UTILITY PROGRAM TO CONVERT A 7-TRACK STRANGER TAPE
                 TO A FILE IN SCOPE FORMAT

HERE ARE ALL THE AVAILABLE ROUTINES
FOR THIS PURPOSE

IF YOU WANT MORE INFORMATION ABOUT A PARTICULAR
ROUTINE, ENTER THE PRECEEDING NUMBER
ELSE ENTER 0

   ? 8460
```

IF YOU WISH TO SEE AGAIN THE ROUTINE DISPLAY
ENTER THE KEYWORD- YES -ELSE-NO

? NO

THIS ROUTINE IS AVAILABLE ON LINE, BUT AN
ADDITIONAL CONTROL CARD IS REQUIRED TO MAKE IT
AVAILABLE TO YOUR JOB

THE ROUTINE--TPCON --CAN BE ACCESSED FOR USE
WITH YOUR PROGRAM BY THE FOLLOWING CONTROL CARDS

```
GRAB(MACLIB )
LDSET(LIB=MACLIB )
```

IT IS SUPPORTED BY COMPUTER CENTER IN THE SENSE
THAT ERRORS WILL BE CORRECTED WHEN RESOURCES
AVAILABLE PERMIT. AVOID USING IN NEW PROGRAMS

DO YOU WANT INFORMATION ABOUT ANOTHER ROUTINE
ENTER YES OR NO

? NO

SAMPLE 4

```
THIS PROGRAM DISPLAYS THE AREAS OF INTEREST
IN THE LIBRARY AND YOU CHOOSE ONE AT A TIME

AFTER THE SYSTEM QUESTION-PLEASE, CHOOSE ONE
ENTER A KEY NUMBER PRECEDING THE DESCRIPTION
YOU HAVE SELECTED

ENTER THE KEYWORD-BACKUP-IF YOU WISH TO RETURN
TO A PREVIOUS DISPLAY STAGE

ENTER THE KEYWORD-GIVEUP-IF YOU WISH TO INTERRUPT
THE PROGRAM AT ANY STAGE

THERE ARE AVAILABLE THE FOLLOWING AREAS
IN THE LIBRARY

     1. FACILITIES, SORTING, PLOTTING
     2. STATISTICS
     3. ENGINEERING
     4. NUMERICAL ANALYSIS
        NOT INCLUDING VECTORS,EIGENVALUES
        AND MATRICES
     5. NUMERICAL ANALYSIS,INCLUDING VECTORS,EIGENV
        AND MATRICES

PLEASE, CHOOSE ONE

     ?2

     1301          STATISTICAL DESIGN
     1302          DATA PRELIMINARIES
     1303          STATISTICAL INFERENCE
     1304          EVALUATION OF STATISTICAL FUNCTIONS
     1501          SIMULATION
     1502          LINEAR PROGRAMMING

     ? BACKUP
```

```
THERE ARE AVAILABLE THE FOLLOWING AREAS
IN THE LIBRARY

     1. FACILITIES, SORTING, PLOTTING
     2. STATISTICS
     3. ENGINEERING
     4. NUMERICAL ANALYSIS
        NOT INCLUDING VECTORS,EIGENVALUES
        AND MATRICES
     5. NUMERICAL ANALYSIS,INCLUDING VECTORS,EIGENV
        AND MATRICES

PLEASE, CHOOSE ONE

     ?5

SELECT ONE

   4501          OPERATIONS ON MATRICES
   4502          EIGENVALUES AND EIGENVECTORS
   4503          EVALUATING DETERMINANTS
   4504          SIMULTANEOUS LINEAR EQUATIONS
   4505          VECTOR ANALYSIS
   4506          MATRIX INVERSION

     ?4504

SELECT ONE

   4504A         INCONSISTENT SYSTEM
   4504B         CONSISTENT SYSTEM

     ?4504B
```

SELECT ONE

4504BA COMPLEX EQUATIONS

4504BB REAL EQUATIONS

?4504BB

SELECT ONE

4504BBA SYMMETRIC SYSTEM

4504BBB TRIANGULAR SYSTEM

4504BBD GENERAL SYSTEM

?4504BBD

NOT ANY AVAILABLE ROUTINE, FOR THIS PURPOSE

SAMPLE 5.

```
THIS PROGRAM DISPLAYS THE AREAS OF INTEREST
IN THE LIBRARY AND YOU CHOOSE ONE AT A TIME
AFTER THE SYSTEM QUESTION-PLEASE, CHOOSE ONE
ENTER A KEY NUMBER PRECEDING THE DESCRIPTION
YOU HAVE SELECTED

ENTER THE KEYWORD-BACKUP-IF YOU WISH TO RETURN
TO A PREVIOUS DISPLAY STAGE

ENTER THE KEYWORD-GIVEUP-IF YOU WISH TO INTERRUPT
THE PROGRAM AT ANY STAGE


THERE ARE AVAILABLE THE FOLLOWING AREAS
IN THE LIBRARY

     1. FACILITIES, SORTING, PLOTTING
     2. STATISTICS
     3. ENGINEERING
     4. NUMERICAL ANALYSIS
        NOT INCLUDING VECTORS,EIGENVALUES
        AND MATRICES
     5. NUMERICAL ANALYSIS, INCLUDING VECTORS,EIGENV
        AND MATRICES

PLEASE, CHOOSE ONE

     ?1

0003          TAPE HANDLING.
0100          UTILITY (INTERNAL) PROGRAMS
0106          FILE ORGANIZATION..
0109          TIMING
0302          COMPILERS
0306          PREPROCESSING AND EDITING
```

0601	SORTING
0606	CHARACTER AND SYMBOL MANIPULATION
0607	INFORMATION CLASSIFICATION, STORAGE, AND RETRIEVAL
0704	INPUT
0804	OUTPUT - B C D
0806	PLOTTING

?0601

SELECT ONE

0601A	RECORDS DO NOT EXCEED STORAGE LIMITATIONS OF COMPUTER.
0601B	RECORDS EXCEED STORAGE LIMITATIONS OF COMPUTER

?0601B

NOT ANY AVAILABLE ROUTINE, FOR THIS PURPOSE

SAMPLE 6

```
THIS PROGRAM DISPLAYS THE AREAS OF INTEREST
IN THE LIBRARY AND YOU CHOOSE ONE AT A TIME

AFTER THE SYSTEM QUESTION-PLEASE, CHOOSE ONE
ENTER A KEY NUMBER PRECEDING THE DESCRIPTION
YOU HAVE SELECTED

ENTER THE KEYWORD-BACKUP-IF YOU WISH TO RETURN
TO A PREVIOUS DISPLAY STAGE

ENTER THE KEYWORD-GIVEUP-IF YOU WISH TO INTERRUPT
THE PROGRAM AT ANY STAGE

THERE ARE AVAILABLE THE FOLLOWING AREAS
IN THE LIBRARY

      1. FACILITIES, SORTING, PLOTTING
      2. STATISTICS
      3. ENGINEERING
      4. NUMERICAL ANALYSIS
         NOT INCLUDING VECTORS,EIGENVALUES
         AND MATRICES
      5. NUMERICAL ANALYSIS,INCLUDING VECTORS,EIGENV
         AND MATRICES

PLEASE, CHOOSE ONE

? 3

SELECT ONE

  1602          ENGINEERING - CIVIL
  1708          COMBINATORIAL ARITHMETIC

? 1602
```

```
WE HAVE THE FOLL. ROUTINES

 8220   STPESS          A PROBLEM-ORIENTED COMPUTER LANGUAGE FOR THE ANALY
                        -- SIS OF LINEAR TWO OR THREE DIMENSIONAL STRUCTURES

HERE ARE ALL THE AVAILABLE ROUTINES
FOR THIS PURPOSE


IF YOU WANT MORE INFORMATION ABOUT A PARTICULAR
ROUTINE, ENTER THE PRECEEDING NUMBER
ELSE ENTER 0000
    ? 0000
```

SAMPLE 7

```
THIS PROGRAM DISPLAYS THE AREAS OF INTEREST
IN THE LIBRARY AND YOU CHOOSE ONE AT A TIME

AFTER THE SYSTEM QUESTION-PLEASE, CHOOSE ONE
ENTER A KEY NUMBER PRECEDING THE DESCRIPTION
YOU HAVE SELECTED

ENTER THE KEYWORD-BACKUP-IF YOU WISH TO RETURN
TO A PREVIOUS DISPLAY STAGE

ENTER THE KEYWORD-GIVEUP-IF YOU WISH TO INTERRUPT
THE PROGRAM AT ANY STAGE


THERE ARE AVAILABLE THE FOLLOWING AREAS
IN THE LIBRARY

     1. FACILITIES, SORTING, PLOTTING
     2. STATISTICS
     3. ENGINEERING
     4. NUMERICAL ANALYSIS
        NOT INCLUDING VECTORS,EIGENVALUES
        AND MATRICES
     5. NUMERICAL ANALYSIS, INCLUDING VECTORS,EIGENV
        AND MATRICES

PLEASE, CHOOSE ONE

     ? 8

WRONG INPUT NUMBER, ENTER AGAIN

     ? BACKUP

WRONG USE OF THE FACILITY,YOU ARE JUST IN THE
BEGINING OF THE PROGRAM,SO ENTER A NUMBER 1-5
OR GIVEUP

     ? 4
```

SELECT ONE

4102	HYPERBOLIC FUNCTIONS
4104	ROOTS AND POWERS OF NUMBERS
4201	POLYNOMIAL OPERATIONS
4202	ROOTS OF POLYNOMIALS
4203	EVALUATION OF SPECIAL FUNCTIONS
4204	SIMULTANEOUS NON-LINEAR ALGEBRAIC EQUATIONS
4205	ROOTS OF TRANSCENDENTAL EQUATIONS
4206	SUMMATION OF SERIES AND CONVERGENCE ACCELERATION
4301	NUMERICAL INTEGRATION
4302	SOLUTION OF ORDINARY DIFFERENTIAL EQUATIONS
4304	NUMERICAL DIFFERENTIATION
4401	TABLE LOOK-UP AND INTERPOLATION
4402	CURVE-FITTING
4403	SMOOTHING A TABULATED FUNCTION
4404	OPTIMIZATION

? BACKUP

```
THERE ARE AVAILABLE THE FOLLOWING AREAS
IN THE LIBRARY

     1. FACILITIES, SORTING, PLOTTING
     2. STATISTICS
     3. ENGINEERING
     4. NUMERICAL ANALYSIS
        NOT INCLUDING VECTORS,EIGENVALUES
        AND MATRICES
     5. NUMERICAL ANALYSIS,INCLUDING VECTORS,EIGENV
        AND MATRICES

PLEASE, CHOOSE ONE

     ?1

SELECT ONE

  0003          TAPE HANDLING.
  0100          UTILITY (INTERNAL) PROGRAMS
  0106          FILE ORGANIZATION.
  0109          TIMING.

  0302          COMPILERS
  0306          PREPROCESSING AND EDITING
  0601          SORTING
  0606          CHARACTER AND SYMBOL MANIPULATION
  0607          INFORMATION CLASSIFICATION, STORAGE, AND RETRIEVAL

  0704          INPUT
  0804          OUTPUT - B C D
  0806          PLOTTING

     ? BACKUP
```

```
THERE ARE AVAILABLE THE FOLLOWING AREAS
IN THE LIBRARY

     1. FACILITIES, SORTING, PLOTTING
     2. STATISTICS
     3. ENGINEERING
     4. NUMERICAL ANALYSIS
        NOT INCLUDING VECTORS,EIGENVALUES
        AND MATRICES
     5. NUMERICAL ANALYSIS,INCLUDING VECTORS,EIGENV
        AND MATRICES

PLEASE, CHOOSE ONE

     ? 2

SELECT ONE

  1301      STATISTICAL DESIGN
  1302      DATA PRELIMINARIES
  1303      STATISTICAL INFERENCE
  1304      EVALUATION OF STATISTICAL FUNCTIONS
  1305      GENERATION AND TESTING OF " RANDOM" NUMBERS

  1501      SIMULATION
  1502      LINEAR PROGRAMMING

     ? 1303

SELECT ONE

  1303A     CHARACTERIZING DISTRIBUTIONS
  1303B     CHARACTERIZING RELATIONSHIPS
  1303C     TIME SERIES

     ? 1303B
```

SELECT ONE

1303BA	PICTORIAL AND DESCRIPTIVE
1303BB	CURVE- FITTING
1303BD	FACTOR ANALYSIS
1303BE	CLUSTER ANALYSIS
1303BF	PRINCIPAL COMPONENT ANALYSIS
1303BG	CANONICAL VARIATES
1303BH	DISCRIMINANT ANALYSIS
1303BI	OTHER

? BACKUP

SELECT ONE

1303A	CHARACTERIZING DISTRIBUTIONS
1303B	CHARACTERIZING RELATIONSHIPS
1303C	TIME SERIES

? 1303A

SELECT ONE

1303AA	FINITE POPULATIONS
1303AB	LARGE POPULATIONS

? BACKUP

SELECT ONE

1303A	CHARACTERIZING DISTRIBUTIONS
1303B	CHARACTERIZING RELATIONSHIPS
1303C	TIME SERIES

? 1303B

SELECT ONE

1303BA	PICTORIAL AND DESCRIPTIVE
1303BB	CURVE- FITTING
1303BC	CONTINGENCY TABLES
1303BD	FACTOR ANALYSIS
1303BE	CLUSTER ANALYSIS
1303BF	PRINCIPAL COMPONENT ANALYSIS
1303BG	CANONICAL VARIATES
1303BH	DISCRIMINANT ANALYSIS
1303BI	OTHER

? 1303BB

SELECT ONE

1303BBA	LINEAR IN ALL VARIABLES
1303BBB	CURVE- FITTING WITH SPLINES
1303BBC	NON-LINEAR REGRESSION

? 1308 BBA

```
REENTER CAREFULLY, ONE OF THE DISPLAYED KEYS.

      ? 1303BBA

SELECT ONE
   1303BBAA     REGRESSION ON ALL VARIABLES
   1303BBAB     REGRESSION ON A SUBSET OF VARIABLES
   1303BBAC     ANALYSIS OF VARIANCE AND OF CONVARIANCE
   1303BBAD     POLYNOMIAL REGRESSION
   1303BBAE     PERIODIC REGRESSION
   1303BBAF     REGRESSION ON PRINCIPAL COMPONENTS
   1303BBAG     GENERAL

      ? 1303BBAA

WE HAVE THE FOLL. ROUTINES
 0690  BEMIRI     CALCULATE MEANS, STANDARD DEVIATIONS AND SIMPLE LI
                  NEAR REGRESSION PARAMETER ESTIMATES FOR ARRAYS WHI
                  CH CONTAIN MISSING VALUES (IN CORE VERSION)

 0700  BEMIRO     CALCULATE MEANS, STANDARD DEVIATIONS AND SIMPLE LI
                  NEAR REGRESSION PARAMETER ESTIMATES FOR ARRAYS WHI
                  CH CONTAIN MISSING VALUES (OUT OF CORE VERSION)

 0860  BMDP1R     MULTIPLE LINEAR REGRESSION ON ALL DATA AND ON GROU
                  PS OR SUBSETS OF THE DATA. EQUATIONS WITH OR WITHO
                  UT AN INTERCEPT CAN BE CHOSEN. IF GROUPS ARE DEFIN
                  ED, HOMOGENEITY OF REGRESSION COEFFICIENTS ACROSS
                  GROUPS IS TESTED. CASES CAN BE WEIGHTED OR EXCLUDE
                  D

READ ROUTINE DESCRIPTION, HOLD ANY PROBABLY WANTED
ROUTINE NUMBER, THEN ENTER THE KEYWORD-PASS-TO
CONTINUE THE ROUTINE DISPLAY

      ? PASS
```

1060 BMDP6R PARTIAL CORRELATION AND MULTIVARIATE REGRESSION. C
OMPUTES PARTIAL CORRELATIONS OF A SET OF VARIABLES
REMOVING THE LINEAR EFFECTS OF A SECOND SET. CAN
BE USED FOR REGRESSION, ESPECIALLY WHEN THERE ARE
MULTIPLE DEPENDENT VARIABLES

1280 BMD01R SIMPLE LINEAR REGRESSION (ONE-WAY ANALYSIS OF COVA
RIANCE). PERFORMS SIMPLE LINEAR REGRESSION ANALYSI
S ON SINGLE OR COMBINED CATEGORIES WITH UNEQUAL SA
MPLE SIZES. ALSO PROVIDES ANALYSIS-OF-COVARIANCE I
NFORMATION

1410 BMD03R MULTIPLE REGRESSION WITH CASE COMBINATIONS. PERFOR
MS MULTIPLE REGRESSION AND CORRELATION DATA ANALYS
IS WITHIN EACH SELECTION OF SUBSAMPLES. SELECTION
MAY BE ANY SPECIFIED SET OF SUBSAMPLES

7200 RLINCF RESPONSE CONTROL USING A FITTED SIMPLE LINEAR REGR
ESSION MODEL

READ ROUTINE DESCRIPTION, HOLD ANY PROBABLY WANTED
ROUTINE NUMBER, THEN ENTER THE KEYWORD-PASS-TO
CONTINUE THE ROUTINE DISPLAY

? PAS

READ ROUTINE DESCRIPTION, HOLD ANY PROBABLY WANTED
ROUTINE NUMBER, THEN ENTER THE KEYWORD-PASS-TO
CONTINUE THE ROUTINE DISPLAY

? PASS

7210 RLINPF INVERSE PREDICTION USING A FITTED SIMPLE LINEAR RE
GRESSION MODEL

HERE ARE ALL THE AVAILABLE ROUTINES
FOR THIS PURPOSE

```
 IF YOU WANT MORE INFORMATION ABOUT A PARTICULAR.
 ROUTINE,ENTER THE PRECEEDING NUMBER
 ELSE ENTER 0

      ? 0690

 THIS ROUTINE IS AVAILABLE ON LINE, BUT AN
 ADDITIONAL CONTROL CARD IS REQUIRED TO MAKE IT
 AVAILABLE TO YOUR JOB

  THE ROUTINE--BEMIRI          --CAN BE ACCESSED FOR USE
 WITH YOUR PROGRAH BY THE FOLLOWING CONTROL CARDS

          GRAB(MMUIMSL)
          LDSET(LIB=MMUIMSL)


 ROUTINE FULLY SUPPORTED BY EXTERNAL SOURCES


 DO YOU WANT INFORMATION ABOUT ANOTHER ROUTINE
 ENTER YES OR NO
      ? YES

 IF YOU WANT MORE INFORMATION ABOUT A PARTICULAR
 ROUTINE,ENTER THE PRECEEDING NUMBER
 ELSE ENTER 0

      ? 1060

 THIS ROUTINE IS NOT AVAILABLE ON LINE ASK FOR
 THE SOURCE DECK FROM THE APPLICATION LIBRARIAN
 AT ROOM SS-270


 ROUTINE FULLY SUPPORTED BY EXTERNAL SOURCES
```

```
DO YOU WANT INFORMATION ABOUT ANOTHER ROUTINE
ENTER YES OR NO
     ? YES
IF YOU WANT MORE INFORMATION ABOUT A PARTICULAR
ROUTINE, ENTER THE PRECEEDING NUMBER
ELSE ENTER 0
     ? 7200

THIS ROUTINE IS AVAILABLE ON LINE, BUT AN
ADDITIONAL CONTROL CARD IS REQUIRED TO MAKE IT
AVAILABLE TO YOUR JOB

 THE ROUTINE--RLINCF          --CAN BE ACCESSED FOR USE
WITH YOUR PROGRAM BY THE FOLLOWING CONTROL CARDS

        GRAB(MMUIMSL)
        LDSET(LIB=MMUIMSL)


ROUTINE FULLY SUPPORTED BY EXTERNAL SOURCES


DO YOU WANT INFORMATION ABOUT ANOTHER ROUTINE
ENTER YES OR NO
     ? NO
```

A P P E N D I X C

U S E R ' S G U I D E

U S E R ' S G U I D E

HELPLIBRARY-DB is an interactive query-answer system of which the logical description is shown in the Schema fig. 25.

The system will allow the User who wants to solve a particular problem to retrieve the routines which are related to his problem through a designed man-machine dialogue. The Creation sub-schemas are shown in figures 26, 26a, b, c.

The whole Data Base consists of 5 areas where all the available data can be accomodated.

The RETRIEVAL-SUBSCHEMA (fig. 27) links 4(four) of the above areas (files) into meaningful relations to facilitate the man-machine dialogue.

The NODE-FILE (TREEAREA in schema) consisting of NODE-REC contains all the Library available problems (one problem per record), each problem identified by its classification number, which is the number denoting the natural position of the node in the Library tree which contains all the relevant problems to the one under consideration.

The ROUT-FILE (ROUTAREA in schema) consisting of ROUT-REC contains all the Library available sub-routines (one subroutine per record), which are linked to the relevant problems through the Data Base ROUTRETRIEVE relation. The ROUT-REC contains all the related information concerning the particular routine.

The PARAMETER-FILE (ARGUMENTAREA in schema) consisting

of PARAMETER-REC contains the arguments of the existing routines with all information about a particular argument except the description of its function. The argument records are linked to its routine records through ROUTPARAM relation.

The PARAM-DESCR-FILE (PARAMETAREA in schema) consisting of ROUT-PARAM-REC contains the description of the function of each parameter and each record is linked to its PARAMETER-record through ROUTPARAM relation.

The PRIVATE-FILE (PRIVATEAREA in schema) consisting of PRIV-INFO-REC contains data concerning only the Program Librarian and for Privacy reasons is not included in the User's Retrieval Sub-schema. It is linked to ROUTINE-FILE through ROUTPRIV relation.

The organization of the files in this Data Base is INDEX SEQUENTIAL so to be able to achieve a random and sequential access to almost all the files.

The Record type employed is RT=T that is variable length record to accomodate the variable description of the different items. The Use of the Data Base facility is based on a man-machine dialogue which follows the flowchart of fig. 28, 28a,b,c.

When the User logs in a small interface primitive program in COBOL interogates the User if he knows a routine and he wishes to get more information about its use or he has a particular problem to solve and wants to retrieve routines related to his problem. According to the answer of the User, the system loads one of the two programs RETRO1 or RETRO2, which continue

the progressive interrogation to the USER until satisfy his needs.

In APPENDIX B there is a set of samples of this man-machine dialogue together with the USER answers. The USER under the guidance of the Program which has been loaded, chooses and enters codes and commands out of the ones displayed by the program giving information to the program to advance the query until the final retrieval of the required information (routines, parameters, control cards etc.).

Error facilities are provided by the program, that is if USER enters wrong code or command the program announces wrong entrance and provides the necessary consulting messages. The logical name of the schema LIBRARY, where the schema directory has been stored, is HELPLIBRARY-DB and the Physical device name is CYL1SCH.

All the creation sub-schemas:

TREEFILECREATIONSUB1: Sub-schema for Creation of the NODE-FILE (fig. 26)

ROUTFILECREATIONSUB2: Sub-schema for Creation of the ROUT-FILE (fig. 26a)

PARAMFILECREATIONSUB3: Sub-schema for Creation of ARGUMENT-FILE and PARAMETER-FILE (fig. 26b)

have been stored in the LIBRARY having logical name SUBSLIB and permanent Physical name CYL2CRS.

The PRIVATEFILECREATIONSUB4: Sub-schema for the Creation

of the PRIVATE-FILE (fig. 26c) in the LIBRARY having logical name PRIVLIB and permanent Physical name CYL3PRS.

The Retrieval sub-schema RETRIEVALSUB5 in separate LIBRARY with logical name RETRLIB and permanent CYL4RTS.

The RETRIEVALSUB5 sub-schema is reffered by the two application retrieval COBOL5 programs RETR01 and RETR02. These application programs have also stored and saved on a permanent file under logical name APLPRG1 and APLPRG2.

HELPLIBRARY-DB SCHEMA

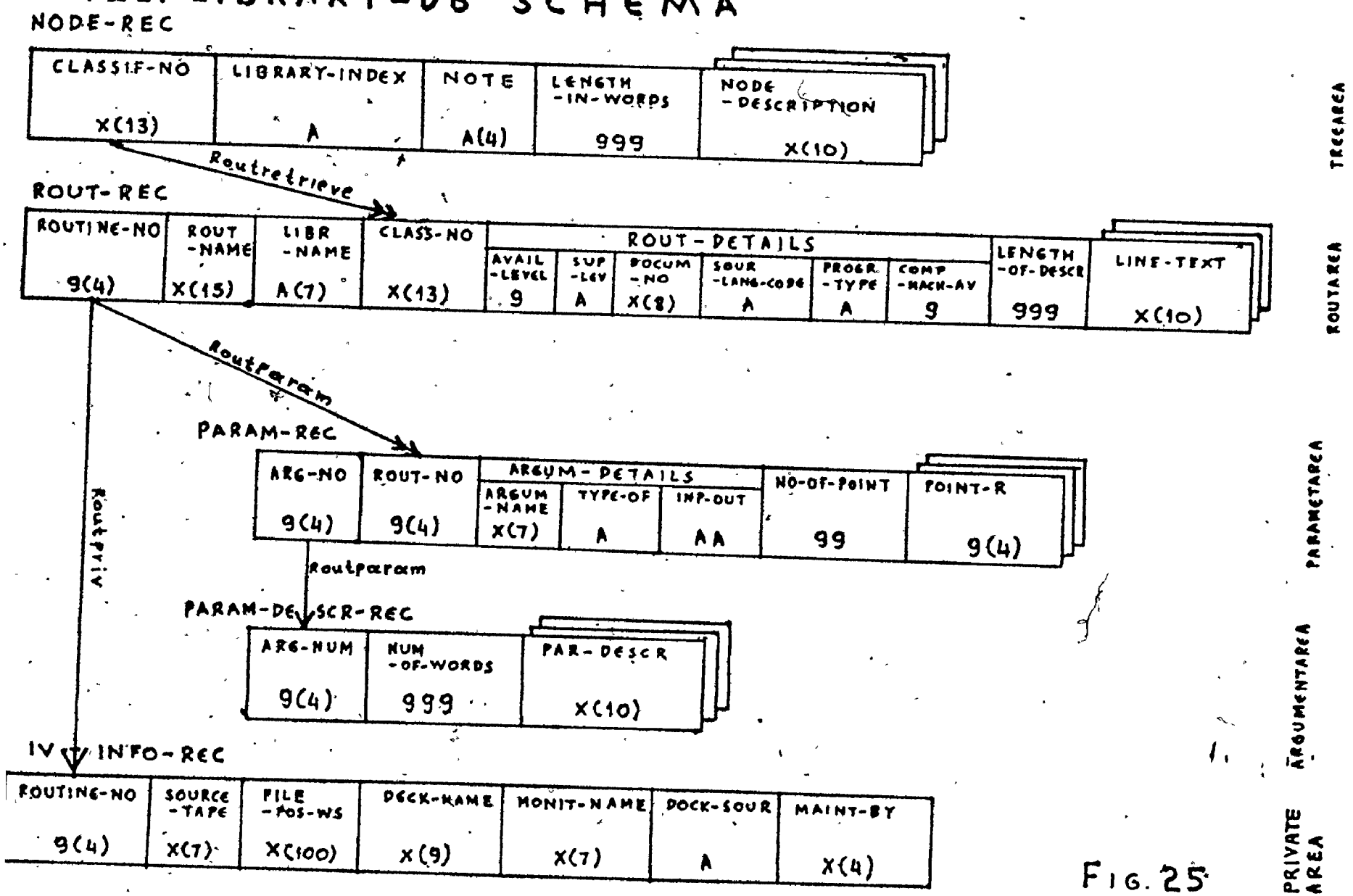


FIG. 25

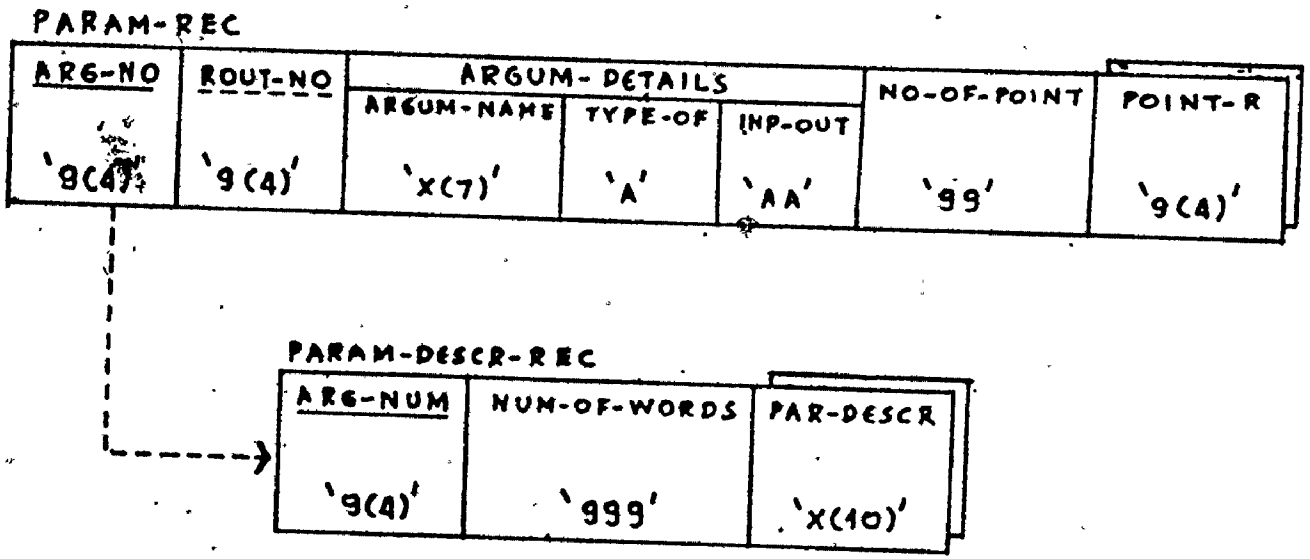
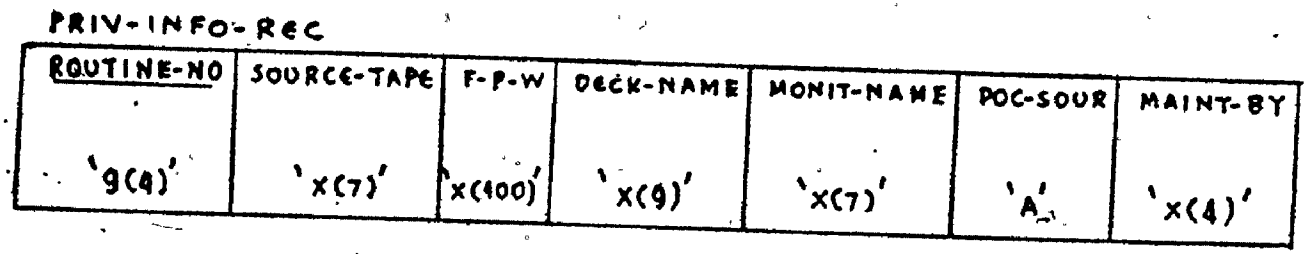


FIG. 26b

Sub-schema for the ARGUM & PARAM-FILES



Sub-schema for the PRIVE-FILE

FIG. 26c CREATION SUB-SCHEMAS

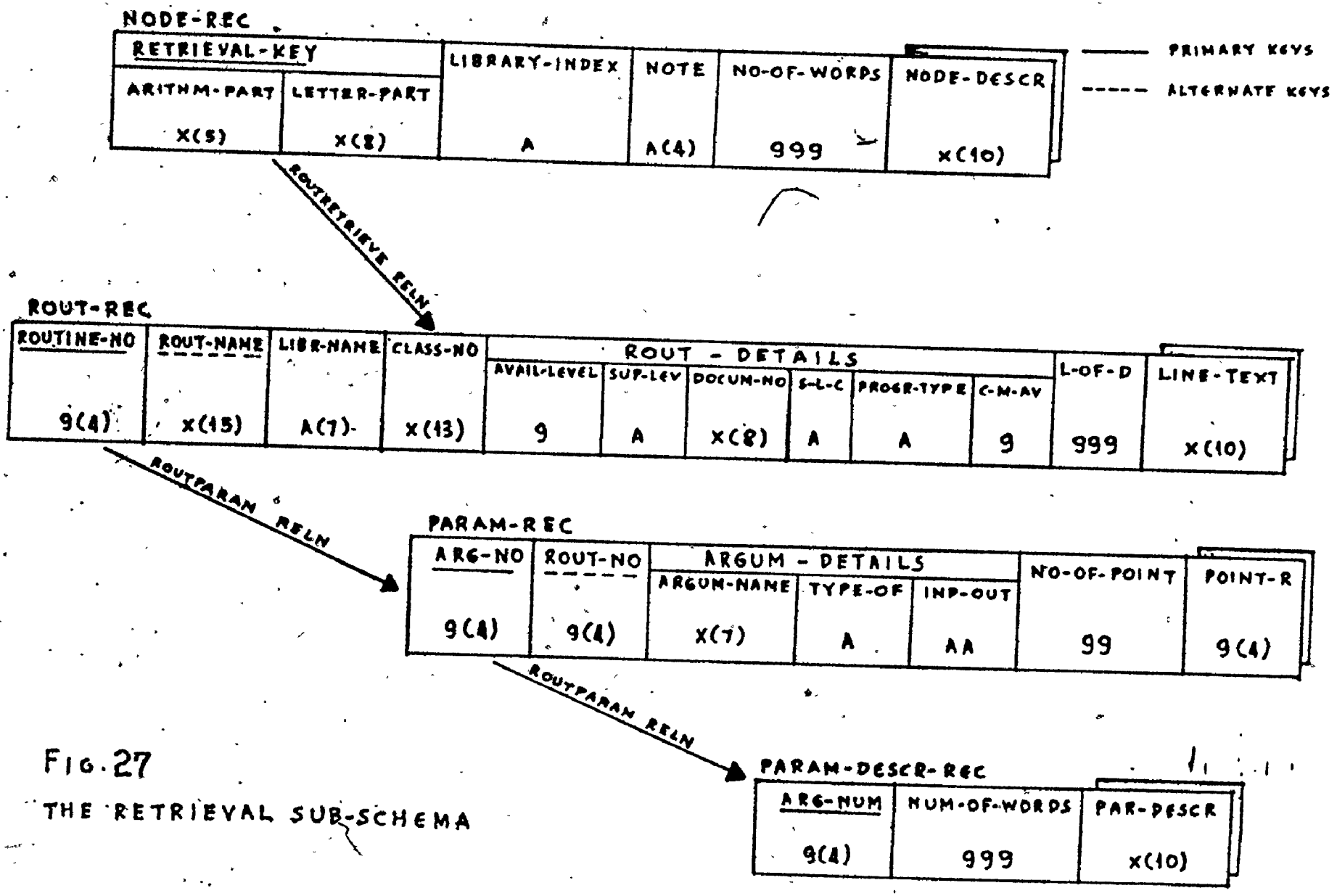


FIG. 27

THE RETRIEVAL SUB-SCHEMA

THE FLOWCHART OF THE RETRIEVAL PROGRAM

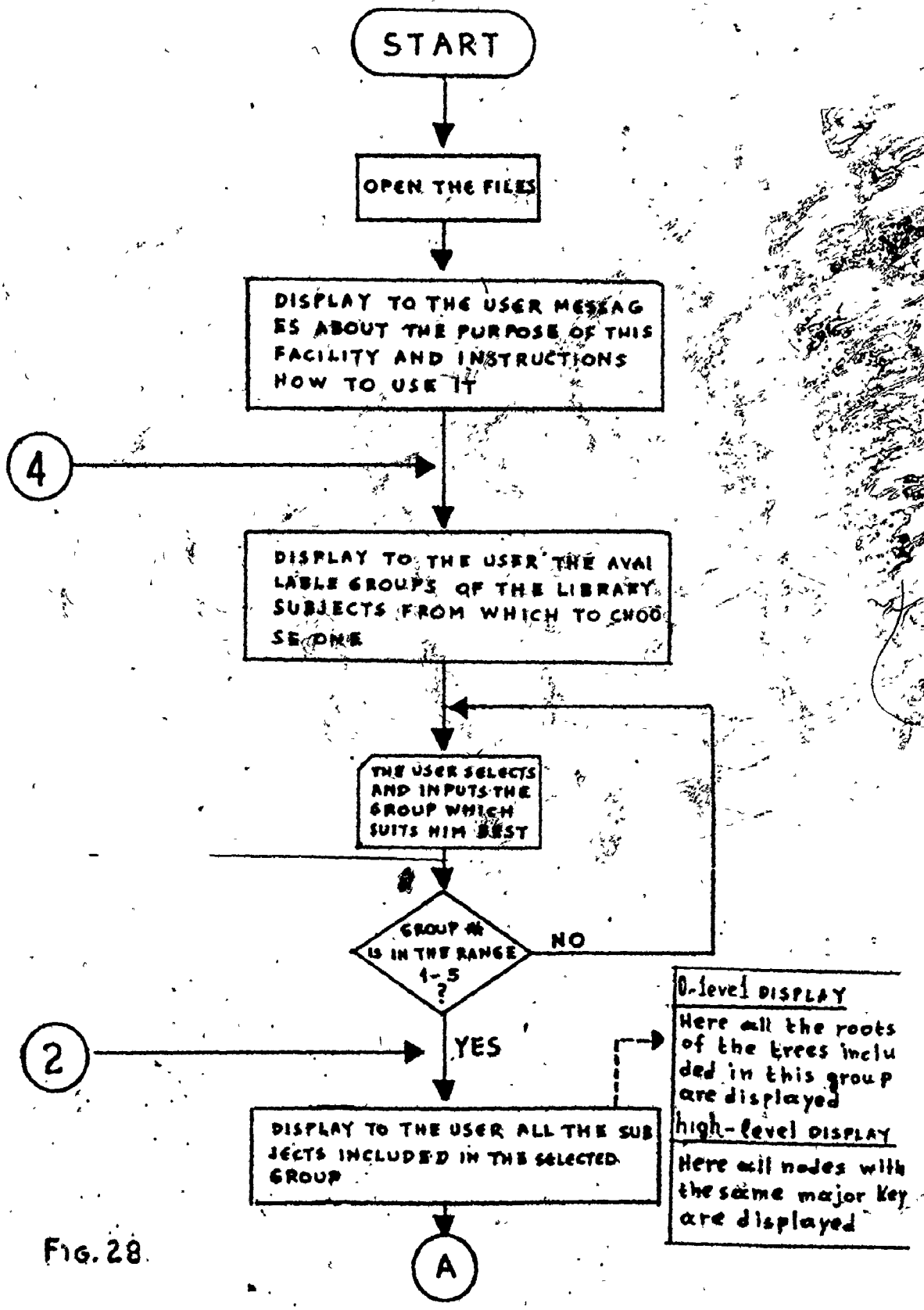


Fig. 28

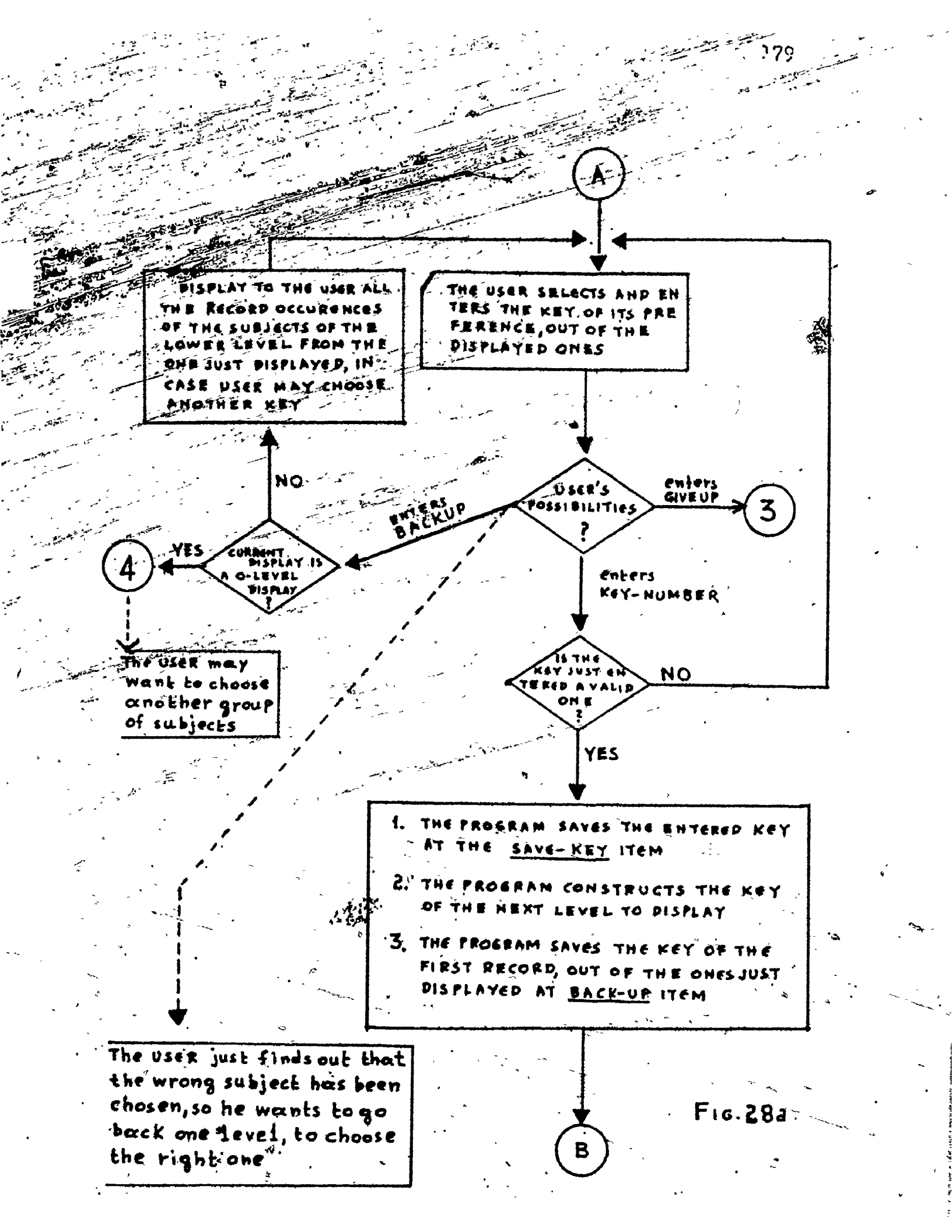
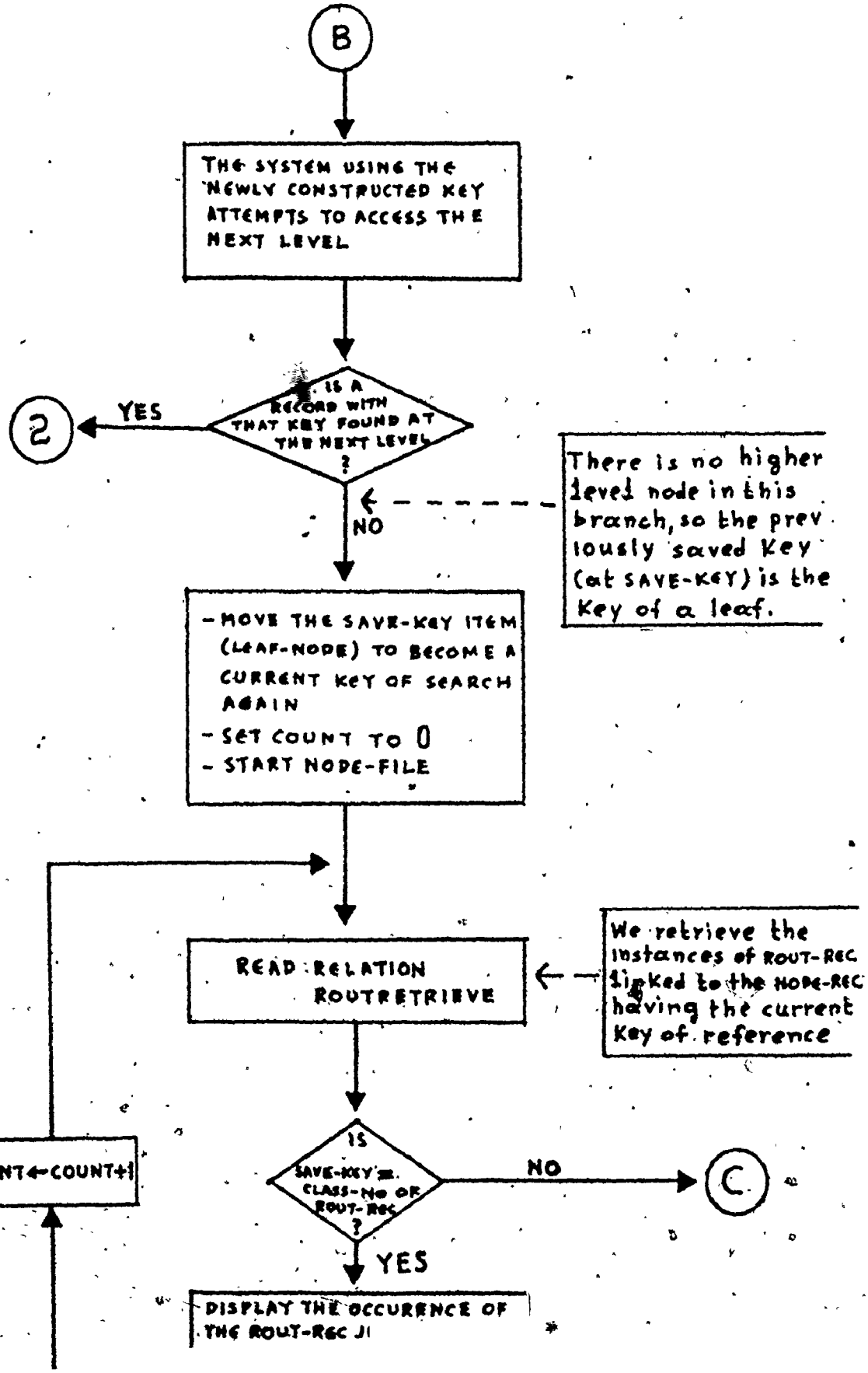

A
DISPLAY TO THE USER ALL THE RECORD OCCURENCES OF THE SUBJECTS OF THE LOWER LEVEL FROM THE ONE JUST DISPLAYED, IN CASE USER MAY CHOOSE ANOTHER KEY
THE USER SELECTS AND ENTERS THE KEY OF ITS PREFERENCE, OUT OF THE DISPLAYED ONES
NO
USER'S POSSIBILITIES ?
enters GIVEUP
3
ENTERS BACKUP
4
YES
CURRENT DISPLAY IS A 0-LEVEL DISPLAY ?
enters KEY-NUMBER
The USER may want to choose another group of subjects
IS THE KEY JUST ENTERED A VALID ONE ?
NO
YES
1. THE PROGRAM SAVES THE ENTERED KEY AT THE SAVE-KEY ITEM
2. THE PROGRAM CONSTRUCTS THE KEY OF THE NEXT LEVEL TO DISPLAY
3. THE PROGRAM SAVES THE KEY OF THE FIRST RECORD, OUT OF THE ONES JUST DISPLAYED AT BACK-UP ITEM
The USER just finds out that the wrong subject has been chosen, so he wants to go back one level, to choose the right one
B

FIG. 28a


B
THE SYSTEM USING THE NEWLY CONSTRUCTED KEY ATTEMPTS TO ACCESS THE NEXT LEVEL
IS A RECORD WITH THAT KEY FOUND AT THE NEXT LEVEL ?
YES
2
NO
There is no higher level node in this branch, so the previously saved Key (at SAVE-KEY) is the Key of a leaf.
- MOVE THE SAVE-KEY ITEM (LEAF-NODE) TO BECOME A CURRENT KEY OF SEARCH AGAIN
- SET COUNT TO 0
- START NODE-FILE
READ RELATION ROUTRETRIEVE
We retrieve the instances of ROUT-REC linked to the NODE-REC having the current Key of reference
IS SAVE-KEY = CLASS-NO OF ROUT-REC ?
NO
C
YES
COUNT←COUNT+1
DISPLAY THE OCCURENCE OF THE ROUT-REC J1

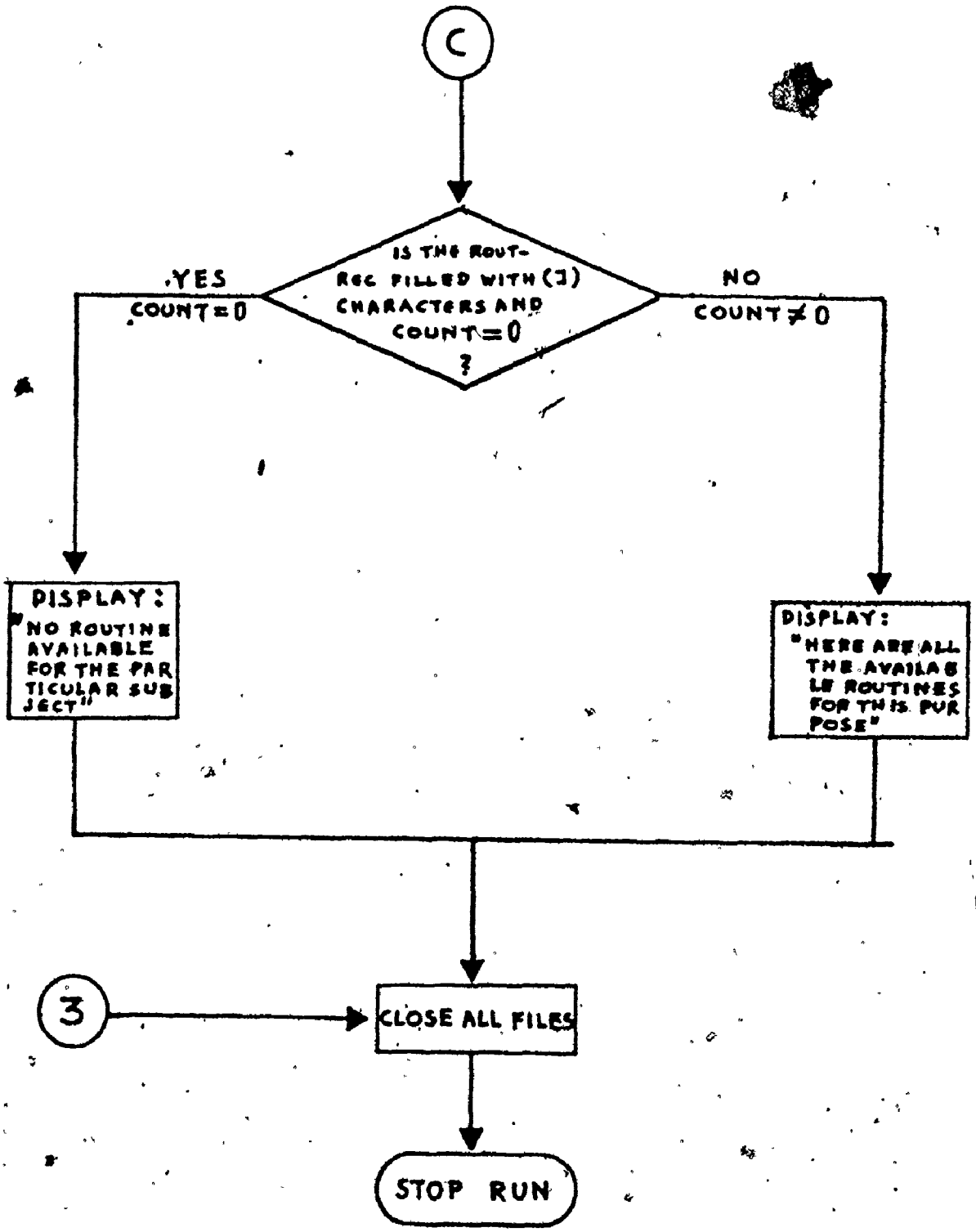

C
IS THE ROUT-
REC FILLED WITH (3)
CHARACTERS AND
COUNT=0
?
YES
COUNT=0
NO
COUNT≠0
DISPLAY:
"NO ROUTINE
AVAILABLE
FOR THE PAR
TICULAR SUB
JECT"
DISPLAY:
"HERE ARE ALL
THE AVAILAB
LE ROUTINES
FOR THIS PUR
POSE"
3
CLOSE ALL FILES
STOP RUN

FIG. 28c