DEPARTMENT OF ELECTRICAL ENGINEERING

UNIVERSITY OF WASHINGTON

COLLEGE OF ENGINEERING

SEATTLE, WASHINGTON 98195

A LARGE SCALE INFORMATION RETRIEVAL SYSTEM

by

Karun Khanna

and

A. D. C. Holden

Department of Electrical Engineering

Technical Report #156

August 1972

This work was supported by the Washington Sea Grant Program, which is a part of the National Sea Grant Program under Grants NG-1-72 and 1-35320. The Sea Grant Program is supported by the National Oceanic and Atmospheric Administration.

CONTENTS

1. INTRODUCTION

1.1 RESEARCH OBJECTIVES

This research has been directed towards the analysis and evaluation of a generalized data base management system, and the design of improved data base creation and updating methods.

One of the problems confronting potential users of generalized systems is the selection of one of these systems from a wide variety of existing systems.

In an attempt to evaluate a system, the feature analysis proposed by the Codasyl Systems Committee [8] provide helpful guidelines.

Though these systems are generalized, in the sense that they provide the necessary file processing capabilities, most of them are biased towards the direction of providing rapid responses to information requests.

Unfortunately this bias has resulted in providing cumbersome techniques for actually creating the data base to hold this large mass of information.

It has been felt by designers of generalized systems that the creation of a data base is a one time procedure and so the facilitation of this procedure has not received adequate attention.

Another limitation of generalized systems dedicated to rapid response is the inability to handle massive changes to the data base, i.e. dynamic updating, in a rapid manner.

The Multi-Access Retrieval System-MARS VI [17] has served as a focal point in this research effort.

The prime reason for considering the MARS VI system is that it is typical of the more recent generalized data management systems. An added reason for considering it has been its availability for experimental use on the existing computer facilities at the U.W. Computer Center.

This system is typical of the more recent generalized data management systems. The design philosophy has been directed towards providing:

(a) a reasonably complete set of general purpose data management capabilities,

(b) a command language which is simple, easy to learn, not prone to syntax errors and oriented towards non-programmer users,

(c) hierarchical data handling capabilities with multi-level nesting,

(d) internal structures for rapid response to queries and

(e) facilities allowing multi-access in a real-time environment

However, the system has severe input data restrictions and these restrictions hamper the creation of the actual data base though they in no way inhibit the user from easily specifying any complex hierarchical structure to relate the various elements of his data base.

The input format, used to create the data base, requires complex character manipulations which make the cost of creating the data base almost prohibitive, as FORTRAN and COMPASS on the CDC 6000 series are hardly oriented towards character manipulation in data strings. Hence, one of the major objectives of this research has been to modify the data base creation procedure, with a view to facilitating and increasing the flexibility of this procedure, as well as reducing the creation costs substantially.

The following facilities have been incorporated to meet this objective:

1. The syntax of the user language has been expanded to allow the user to create tables, which relate the key fields designated by the user to the actual physical location on the record. [Appendix A]

2. Provisions have been designed to allow the user to create tables which contain the relevant hierarchical association between the designated

key fields. [Appendix A].

The new MARS VI system employs these user created tables along with the definition tables which have been created by specifying the data base structure to validate and load actual data values into the data base.

In the original system the retrieved subsets of the data base were subject only to limited processing capabilities and no provisions had been made to input the retrieved subsets for other more powerful data analysis packages such as SPSS or BMD. [19, 2].

This capability has been provided in the new MARS system and the user is now able to conduct complex analyses of his retrieved data. This provision is important since the use of statistics is a major tool for the management of data resources.

In order to incorporate the facilities described above, additions have been made to syntax of the LOADER and RETRIEVAL modules [3.3, 3.4].

1.2 SURVEY OF THE STATE OF ART IN GENERALIZED DATA MANAGEMENT SYSTEMS -

Current trends in data management techniques are in the direction of development of generalized data management systems, i.e., systems of generalized programs which provide the common data management functions of storing, retrieving and updating data for a wide variety of applications.

There are two classes in generalized data base management systems.

1. Host-language capabilities
2. Self-contained capabilities

The host-language capabilities are merely new tools for the applications programmer to facilitate the handling of complex data structures. Generally host-language systems offer the capability to define hierarchical structures. The host-language capabilities are merely built upon the facilities of a procedural language such as COBOL, PL/1 or FORTRAN.

As host-language capabilities are primarily within the realm of an applications programmer, the emphasis has been more towards the development of self-contained systems.

In using a generalized data management system with self-contained capabilities the user is actively involved. The user is required to explicitly define a desired data structure. A pre-programmed or built-in processing algorithm automatically allocates space, establishes directories and affords a degree of independence between the generalized programs and the data with which the program interacts. Depending on the degree of independence, programs can accommodate varying degrees of change in the definition and structure of the data without the necessity of modifying the program itself.

Self-contained generalized data base management systems are geared primarily towards the non-programmer. A whole new language is offered, which is non-procedural, i.e., free from the restraints of procedural languages like PL/1 or FORTRAN. The syntax of the language is oriented towards the natural languages and is rather simple to master.

Generally, user interaction is on-line from a remote terminal, but most systems are accessible through the batch mode. Typically, the user will not be involved with the actual creation of the data base but will be formulating and directing his own queries at the data base. Updating of the data base may be yet another user interaction.

The essential features of a generalized data base management system are listed below [8].

1. Data Definition - This permits a user to define a data structure in any arbitrary hierarchy of data elements within logical entries. The

defined data structure is generally different from the storage structure, which is the particular file organization used internally by the system.

2. Data Retrieval - The retrieval function involves making selective searches within a data base. Retrieval requests typically contain a list of data items or logical entries to be outputted depending on certain selected criteria. Various operations such as computations, sorting and reformating may be performed on the extracted data.

3. Data Base Creation - This involves either a built-in processing algorithm for constructing the data base, operating entirely on the user's data input or programming in a conventional sense to create a data input acceptable to the built-in function, provided by the system.

4. Updating the Data Base - This encompasses changing, adding to or deleting values of certain data items or certain logical entries within the data base. However, the data definition or the data structure of the logical entries may not be restructured as this would imply modifying the pre-stored data definition. Update is a process somewhat analogous to retrieval in that some part of the data base must first be selected. In most systems the selection facilities are modelled on those used in the interrogation function. However, once a certain part is selected or updated, it is changed in some defined way rather than displayed in a report.

Review of some current data management systems -

1. Generalized Information System (GIS) has been developed by IBM for the System 360 and is available both in the batch and interactive environment. The language is procedural with high level operators which are used to specify the processes of file definition, file creation, file updating, file sorting and file interrogation. Hierarchical structuring

of data is permitted. The system requires a fairly large machine and a fair amount of storage, with the batch processing version available on a Model 40 with 128 kilo-bytes of core and the interactive version available on a Model 50 with 312 kilo-bytes of core. Details of the system can be found in the applications manual [10].

2. ISL-1 - Data base definition, organization, creation, maintenance and interrogation are handled by generalized procedures invoked by the user in a non-procedural command type language. The interesting feature of this system is that the definition language for specifying the data structure desired is primarily in a tabular form, whereas the other processes are specified in a string language. Simple or complex hierarchical data structures are allowed. The system is available on an IBM 360/30, 40. Further specifications of the system are available in the Codasyl Systems Committee's report on Generalized Data Base Management Systems [1].

3. Mark IV - This system arose out of batch oriented data processing and still has some features left over from essentially tabulating or card system environment. Its input is by fixed preprinted forms. It does not have good facilities for easily generating new files and extracting data excerpts for report generation. It is available on a fairly wide variety of machines. The language is non-procedural and hierarchical data structures up to nine levels of nesting are allowed. It is suited for commercial information systems and is used fairly widely.

4. Time-shared Data Management System is a general-purpose system designed to permit users to define, generate, correct, maintain and retrieve information from large data bases. The language is non-procedural and is free-form, beginning with command words followed by command objects and

modifiers. The system is oriented towards rapid response to queries and operates in both the batch and the interactive mode. Hierarchical data structures in the form of trees are provided and certain computations can be performed on the retrieval data subsets. The system is operational on an IBM 360/50H, but can also be implemented on a larger configuration under the control of a suitable time-sharing executive. Specifications of the system are available from the Systems Development Corporation [3,4,22].

5. User Language/1 System - is available as part of the standard software package for the Spectra 70. The philosophy of the system is to provide easy usage by non-programmers, including those interrogating or updating a data base from a remote terminal.

The batch processing version is available on 70 Model 45, 55 or 60, requiring 131K of memory. The time-sharing environment for both the time-sharing and batch processing mode is available on a Spectra 70 model 46.

An interesting feature of this system is the capability provided for storing frequently used and therefore predictable interrogations and updates to the data base, so that these may be invoked by the lowest level of user who need know essentially nothing about the command language. Two technical papers have been published on UL/1 [21].

Although the systems discussed are similar in basic design approach, they vary widely in processing capabilities, language used, user orientation and file structure.

Generalized data base management systems which have built-in capabilities, offer considerably reduced set-up time and a vast reduction of the time required to prepare a new interrogation or an update to the data base. In this respect, such systems make major economies in the use of people's time and can also give a more rapid satisfaction of ad hoc

information requirements [8].

A recent trend in the development of data base management systems is the effort to produce a universal language for the description and manipulation of data stored within a data base. The intent of such a language is to describe a data base independently of any programming language [25].

The use of more than one programming language to access the same data, normally requires data conversion from one format to another. The cost of handling ever increasing volumes of data combined with increasingly complex processing demands have created the urgent need for a method to create and manipulate a data base which is common to all applications but independent of any particular programming language [24].

A new CODASYL Data Description Language Committee (DDLC) has been formed for the development of a common data description language to be developed on the efforts of Codasyl Data Base Task Group. The efforts of this group will probably lead to a comprehensive data manipulation language designed to further improve execution performance and system integrity.

1.3 DEFINITIONS OF TERMINOLOGY -

Definitions of terminology used in describing the MARS VI system are given below to aid in the understanding of the system features.

a. A user-defined data base consists of sets of logical entries.

b. A logical entry is a file of information built on the definition structure. The definition structure establishes the structural relationship between data items. Data items are entered in accordance with this definition in the logical entries.

c. A component is an element or node of the definition structure.

1) Element components are tree definition nodes which are assigned data values in the data tree.

2) A repeating group component serves as a parent to components specified as members of that repeating group set, linking all its descendants and thereby allowing the establishment of a hierarchical structure.

3) A repeating group set consists of all components, repeating groups and elements attached to the same repeating group component. The repeating group set is created at the repeating group level plus 1, i.e., if the repeating group component is at level 1, the repeating group set is created at level 2.

d. Level numbers are used to identify the level of hierarchy in the tree structure. The higher the level number, the lower is the level in the tree structure.

e. A multi-record data input file is one which contains different data records linked to each other through one or more components.

f. A data file is the collection of data records into logical units.

g. A data record is a collection of elementary data items.

h. A data item is the basic unit of information that the system stores, retrieves and processes.

i. A data base management system is one in which a controlled body of data is made available in machine-based form for retrieval, processing, and reporting.

2. MAJOR COMPONENTS OF THE SYSTEM

2.1 GENERAL ORGANIZATION -

The MARS VI system uses the modularity concept in its system architecture.

Software has been developed in small, independent and easily controlled modules each with a particular task.

Each of the modules is treated as an overlay. An overlay is a portion of a program written on a file in absolute form and loaded at execution time without relocation, to ensure the fastest possible loading speed as well as reduce the size of the resident loader.

When an overlay is loaded from a file, it is identified by its level number. The level number is a pair of two-digit octal numbers ($0\text{-}77_8$) giving the primary and secondary overlay relationship. The first number is the primary level, the second is the secondary level. An overlay with a non-zero primary level and a zero secondary level (6,0) is a primary overlay. Any overlay with the same primary level and a non-zero secondary level (6,1) is associated with and subordinate to the corresponding primary and is called a secondary overlay. This difference is significant when overlays are loaded. Level (0,0) is reserved for the main overlay which remains in core memory during overlay execution.

The main overlay (0,0) is loaded first. All primary overlays are loaded at the same point immediately following the main overlay. Secondary overlays are loaded immediately following the primary overlay. Loading the next primary overlay destroys the first loaded primary overlay and any associated overlays. Similarly, the loading of a secondary overlay destroys a previously loaded secondary overlay [9]. There are a total of 33 overlays which comprise the MARS system in its entirety.

The overlay organization is

1. MAIN OVERLAY (0,0) - CONTROL MODULE
2. PRIMARY OVERLAY (1,0) - DEFINE MODULE

3. PRIMARY OVERLAY (2,0) - LOADER MODULE

4. PRIMARY OVERLAY (3,0) - UPDATE MODULE

5. PRIMARY OVERLAY (4,0) - RETRIEVAL MODULE

6. PRIMARY OVERLAY (6,0) - REPORTER MODULE

The above overlays are typically the ones the user interacts with, while utilizing the system. (See Fig. 2.1.)

2.2 PRIMARY FUNCTIONS OF SYSTEM ELEMENTS -

A. CONTROL MODULE - Communication between the user and the MARS VI system is handled by the control module. Basically the control module has two tasks. One is the supervising of user file operations, while the second entails the sequencing and execution of the other modules.

The system-user communication is handled through the following files

1. COMMAND FILE

2. REPORT FILE

3. MESSAGE FILE

4. DATA FILE

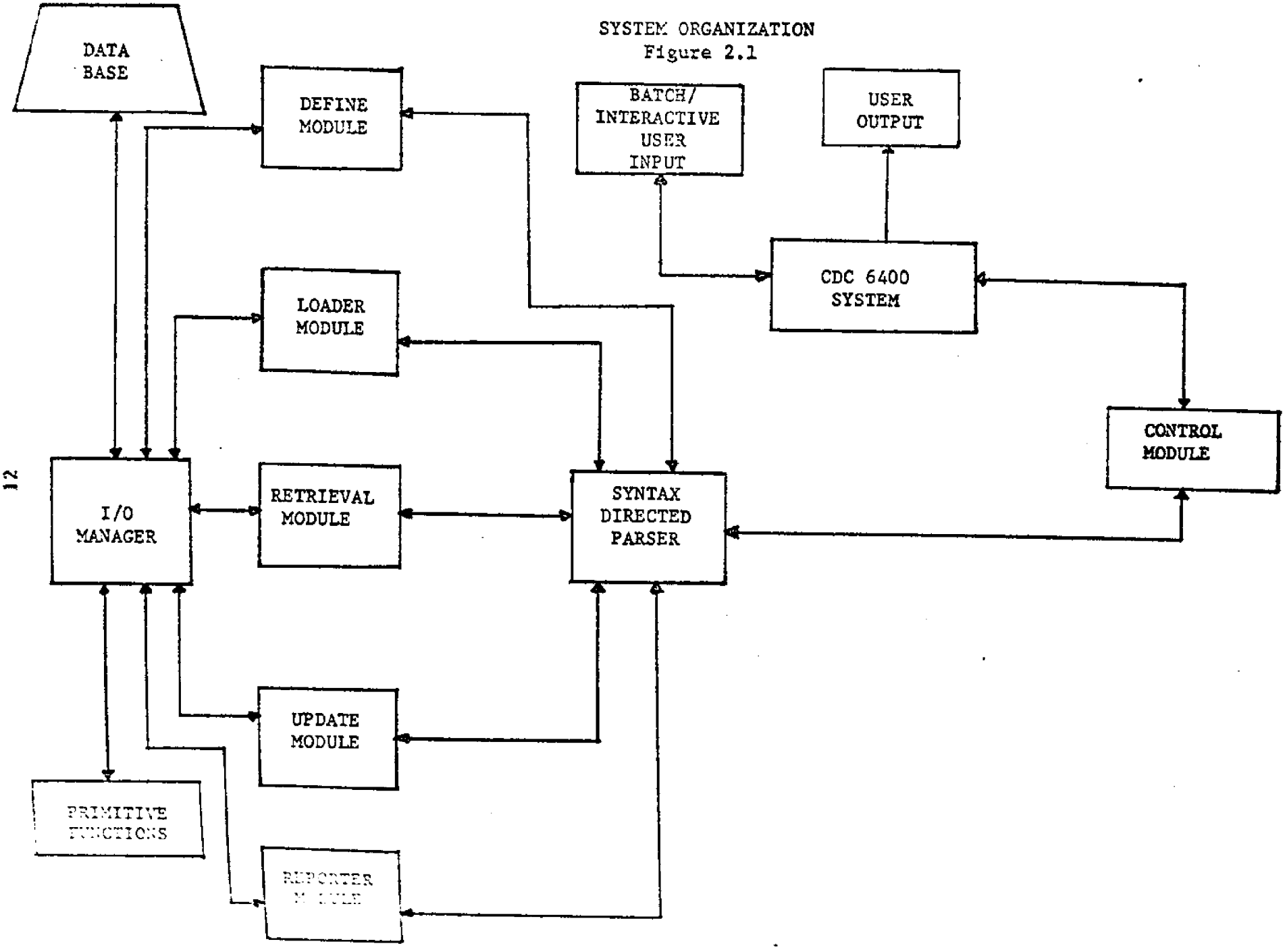
The COMMAND file contains user directives which specify the step-wise processing of the user's data.

The MESSAGE file contains an echo of each user directive as it is encountered on the COMMAND file, as well as error messages due to errors incurred while executing the directives.

The REPORT file contains results obtained by the execution of directives on the COMMAND file.

The DATA file contains the user's data, which can be either in the inverter string format or in a user desired format.

SYSTEM ORGANIZATION
Figure 2.1



The control module also interacts with the CDC 6400's SCOPE 3.3 operating system.

B. I/O MANAGER - This serves as a housekeeping system designed to relieve other modules from the task of locating and moving data in storage.

The I/O manager interacts with the data base and utilizes the PRIMITIVE FUNCTIONS.

The data base exists as a set of tables with each table being logically divided into partitions. Partition size is variable, but has been set for 256 central memory words at this installation.

A table contains entries with various items and the I/O manager uses the PRIMITIVE FUNCTIONS to store/retrieve an item or several items from a particular table entry in the table.

In addition the PRIMITIVE FUNCTIONS contain routines for converting relative addresses to absolute addresses as well as allocating partition space to the various tables in the Input/Output buffer area of the system.

The I/O manager handles all transfers of tables from external storage to disk and movement of the relevant partitions from disk storage to central memory.

C. SYNTAX-DIRECTED PARSER - The parser uses the syntax tables for parsing user directives.

The DEFINE, LOADER, RETRIEVAL, UPDATE and REPORTER modules each have their own syntax tables. Control module directives can be given in any of the other modules and hence the syntactical structure for all control directives is contained in each of the other modules.

The parser uses a scanning routine for obtaining syntactical units. A syntactical unit is recognized as a string of alphanumeric characters from A through Z and 0 through 9. The syntactical unit is delimited by any CDC character which has a display code greater than 44 octal, i.e., semicolon, + etc. Each of the delimiters, serves as a syntactical unit by itself, except 55 octal which is a blank. Hence, the parser handles user directives with leading and trailing blanks, the blanks only being considered when appearing as delimiters between syntactical units.

D. DEFINE MODULE - This permits a user to describe the contents of the entries he desires in his data base, as well as define the organization in any arbitrary hierarchy of data elements within entries.

The DEFINE module must be used to create a data base before any other module can be called. Creation of a data base here only defines the structure of the data base. Data values are not loaded by the DEFINE module.

The defining feature is divided into two phases, the declaration and the finalizing phase.

During the declaration phase the user issues commands and definition entries, which he can modify at his own discretion.

The finalizing phase makes a data base definition permanent, by creating the definition tables which contain the user-defined component number and names, the type of data associated with each component, as well as the hierarchical relationship between the user-defined components. The data structure of entries in the data base cannot be restructured during the course of this session. A session is defined as the execution of user directives on the command file till an EXIT; directive is encountered, which

terminates the execution of directives and transfers control out of the MARS VI system and back to the SCOPE supervisory system.

The user is permitted only one data base definition during a single MARS VI session.

E. LOADER MODULE - Actual data values to the data base are assigned by this module, in accordance with the finalized data base definition specified in the DEFINE module.

The LOADER module scans and analyses the input string before accepting it for loading into the data base. Various checkpoint reports at different stages of the loading as well as errors encountered in the input string can be displayed on the MESSAGE file.

The LOADER module can be called several times to add new logical entries to an already existing valued data base.

F. RETRIEVAL MODULE - The RETRIEVAL module permits to qualify, retrieve and output data at any level of data hierarchy from the data base. Relational Operators and Boolean Conditions are applied to data items in order to qualify data sets for retrieval.

Simple processing of data items is also permitted. The following arithmetic computations are performed on data base values

1. Average
2. Count
3. Maximum
4. Minimum
5. Sum
6. Standard deviation

In order to submit subsets of data retrieved to more complex processing, the user can generate an output file which can be then input to any FORTRAN program or statistical package such as SPSS or BMD [19,2].

G. UPDATE MODULE - This provides the capability of editing an existing data base through the use of directives incorporating addition, deletion, insertion and changing of data values.

Data value manipulation can involve entire logical entries, portions of logical entries, data sets or isolated data items.

The UPDATE module can in no way alter the definition structure specified in the DEFINE module.

The UPDATE module allows the user full use of retrieval specifications to qualify data sets and data items for modification.

H. REPORTER MODULE - It accepts a report format description and generates a report based on this description using the subsets of data retrieved by the RETRIEVAL module.

Typically the report format description consists of 1) the report title which is printed at the top of the first page only, 2) column headings which are printed after the title on the first page, and at the top of all subsequent pages, and 3) user-defined vertical and horizontal page spacing.

The MARS system can be activated in a batch processing mode, which includes access from remote teletypes through use of the U.W. TANDEM VERSION 1.1 system. TANDEM is a "conversational job entry" system. The TANDEM user may create and correct jobs at his terminal while enjoying almost instant turnaround. Once ready to run, the job is submitted from the terminal into the normal batch queue of the CDC 6400, and the results will be available

at the terminal (or on a printer) when it has completed execution [28].

Facilities for interactive processing will also be available with the implementation of the SCOPE 3.4 operating system, using its Intercom Package. In fact the system design is oriented towards the use of conversational operations from remote teletypes which greatly enhances the use of the simple command language.

3. DESCRIPTION OF COMMAND LANGUAGE

A syntax directed parsing system is used by the CONTROL, DEFINE, LOADER, RETRIEVAL, UPDATE, and REPORTER modules to interpret commands being issued by the user.

A metalanguage is used to describe the syntax, while the semantics are described in English, since no formal semantic language has yet been developed. The metalanguage used is BNF-like (Backus Normal Form or Backus-Naur Form). The notation does not adhere strictly to the formal definition of the metalanguage but is intended to indicate the purpose of each command [18,27].

The syntax rules as defined by the metalanguage specify which sequence of symbols are permissible; the semantic rules specify the meaning in terms of an (interpretive) execution of this string of symbols, i.e., what happens if the command is executed. In addition due to the restrictive expressive power of formal syntax rules, the semantic rules often give a further syntactic description.

The following metalanguage symbols are used in the BNF-like notation adopted:

<....>	denotes a metalinguistic variable. Symbols not enclosed between < and > denote

	themselves. Concatenation is indicated by juxtaposition.
: : =	is defined as
/	or
...	repeats the previous syntactic unit zero or more times.
[and]	denotes choice of symbols. The set of alternatives is usually stacked vertically.
;	denotes end of command, or of a user-directive.

The intent of using a metalanguage to describe the syntax rules is to ensure conciseness, which apparently tends to be incompatible with readability and clarity. Hence a description of the semantics in English, coupled with the syntax description in a metalanguage provides a definition of the command language which is most meaningful and useful to the greatest number of users.

The following description of the commend language attempts to achieve the desired objectives.

3.1 CONTROL COMMANDS -

COMMAND FILE IS <filename>;

The command file contains user directives which describe the processing, in a step-wise fashion to be accomplished with the user's data.

REPORT FILE IS <filename>;

The report file contains results obtained from the execution of user directives on the command file.

MESSAGE FILE IS <filename>;

This contains errors incurred during the execution of the command file.

DATA FILE IS <filename>;

This contains the user's data in the loader input string format, or in a user-specified format.

The default for the command file in INPUT.

The default for the message and report file is OUTPUT.

<filename>::= 1-7 alphanumeric chararters

EXIT;

This command is used to terminate the session and control passes over to the next Scope control card.

3.2 DEFINE COMMANDS -

DEFINE;

The DEFINE command is used to call the DEFINE module which is overlay (1,0) into the computer.

NEW DATA BASE <data base name>;

<data base name>::= 1 thru 20 alphanumeric characters

The user assigned data base name serves as an identification for a new data base and the DEFINE module initiates the definition version number to 1, and the data version number to zero.

The first component description in a new declaration must be a level 0 description, where the component is either an element type or a repeating group. The higher the level number, the lower is the level in the hierarchical structure.

<level 0 component description>::= <component description> (<type 0 description>)

<component description>::= <component number> <separator symbol>
<component name>

<component number>::= any 1-4 digit user defined number

<separator symbol>::= <system defined separator>/<user-defined separator>

<system defined separator>::=)

<user-defined separator>::= any legal non-alphanumeric display code character, not occurring in the data string and excluding the blank, the "," and the ";"

<component name>::= any string of display code characters up to a maximum of 150, excluding the user-defined separator symbol and the "," and the ";"

<type 0 description>::= <repeating group type>/<element type>/
<repeating group type> <nulls option>/
< element type> <padding option>

<repeating group type>::= RG/REPEATING GROUP

A repeating group component type serves as a link for subordinate components and hence has no actual data associated with it. Repeating groups may be nested to a depth of 63 levels below the 0 level.

<element type>::= <name>/<date>/<integer number>/<decimal number>/
<exponential number>

<name>::= any string of display code characters up to a maximum of 255

<date>::= mm/dd/yy / mm/dd/yyyy

If the year is in the 20th century then only the last two digits of the year code are needed.

<integer number>::= any string of numbers up to 15 characters including the sign

<decimal number>:: any string of numbers up to 15 characters including the sign and the decimal point

<exponential number>::= a string of 20 characters of the form ±n.nE±m, where n is an integer number, E is the base 10 exponent and m is an integer number limited to a maximum of three digits

<nulls option>::= WITH NULLS

<padding option>::= WITH <mm >PERCENT PADDING/WITH <amount> FUTURE ADDITIONS

<mm>::= any integer number less than 60

<amount>::= NO/FEW/SOME/MANY where,

NO = 0%

FEW = 15%

SOME = 30%

MANY = 50%

<level n component description>::= <component description>(<type n description);

$0 \leq n \leq 63$

<type n description>::= <repeating group type> IN <parent component number>/ <element type> IN <parent component number>/ <repeating group type> <nulls option>/<element type> <padding option>

<parent component number>::= any previously declared repeating group component number

<logical entry>::= <level 0 data set>/<dummy level 0 data set>/<logical entry> <repeating group data set>

<level 0 data set>::= data values that were entered for the defined level 0 elements, if any, for a specific logical entry

<repeating group data set>::= <level n data set>/<dummy repeating group data set>

<level n data set>::= a single set of actual data values for a repeating group

<dummy level 0 data set>::= data set at level 0 with no associated data values

<dummy repeating group data set>::= repeating group data set with no data values

DESCRIBE;

This allows a user to output his data base definition to the REPORT file.

CHANGE DATA BASE NAME TO <data base name>;

The user can alter the name of his data base by issuing the above command.

CHANGE NAME OF <component number> TO <component name>;

The component name associated with any component number can be changed.

CHANGE NUMBER OF <component number$_1$> TO <component number$_2$> ;

RENUMBER;

This command causes all user-defined component numbers to be renumbered in their current input string sequence order, with the first component number being set to 1, and each succeeding component number being incremented by 1.

Other variations of this command are,

RENUMBER STARTING WITH <integer number> AND INCREMENTING BY <integer number>;

RENUMBER STARTING WITH <integer number>;

Each succeeding component number is incremented by 1.

RENUMBER INCREMENTING BY <integer number>;

The starting component number is set to 1 and the succeeding ones are incremented by the specified integer number.

CHANGE TYPE OF <component number> TO <type 0 description>/<type n description>;

This command may be used to change an element component type to another element component type or to a repeating group component, but it cannot be used to change a repeating group component to an element type, because the repeating group has descendants in the hierarchical structure, associated with it.

[ADD / REMOVE] NULL OPTION [TO / FROM] [RG / REPEATING GROUP] <component number>;

The above commands allow the user to add or remove nulls from any repeating group component.

DELETE <component number>;

DELETE <component number$_1$> THRU <component number$_2$>;

DELETE <component number> TO END;

The DELETE commands delete single definition input string entries or inclusive group of entries. "TO END" signifies the end of the definition input string entries.

INSERT [BEFORE / AFTER] <component number>;

Definition input string entry or entries may be inserted before or after a desired component number.

ADD TO BOTTOM;

This command informs the module that the input string which follows is to be added to the end of the DEFINE input string.

The REORDER command is issued before the data base definition is finalized and enables the user to dictate the display order, which otherwise parallels the DEFINE input string order of component occurrence.

The reordering options are:

REORDER BY NUMBER;

The data base declaration is reordered in an ascending order of the component number occurrences.

REORDER BY LEVEL;

The data base declaration is reordered by the level numbers in an ascending order i.e. all members belonging to the level 0 appear before all members belonging to the level 1 set.

REORDER BY [RG | REPEATING GROUP];

The data base declaration is reordered by grouping together all members of the same repeating group. All level 0 components are grouped at the start of the input string.

MAP;

This finalizes the definition of the data base. The definition tables are created and the data definition number is set to 1. Once the data base definition has been finalized by issuing the MAP command, the tree structure and the type descriptions associated with the various component names cannot be altered.

3.3 LOADER COMMANDS

LOADER;

The LOADER module which is OVERLAY (2,0) is invoked by the above command.

A finalized data base definition must be referenced before data values can be loaded. If the user begins the job session using the loader module, the LOAD DATA BASE COMMAND FROM <filename>; must be issued prior to calling the LOADER module. If the call to the LOADER module follows use of another module, then the finalized data base definition is already in random access storage, and the load data base command is not required.

ENTRY TERMINATOR IS <entry terminator>;

<entry terminator>::= 1-10 alphanumeric characters except ";"

The entry terminator is used to signal the end of a logical entry.

SEPARATOR IS <separator>;

<separator>::= any CDC character except A thru Z, 0 thru 9, blank or the ";"

The default value is ")"

NOTIFY MESSAGE FILE IF ANY ERRORS OCCUR,

DISPLAY [ENTIRE ENTRY | LEVEL 0 ONLY | ERRORS ONLY] ;

The ENTIRE ENTRY option lists all accepted and rejected data values for the entire logical entry on the message file.

The LEVEL 0 ONLY option lists all level 0 data values which are accepted and all rejected data values on the message file.

The ERRORS ONLY option produces a list of rejected data values only.

Even one error in the data for a logical entry causes all remaining data for that logical entry to be rejected although the remainder of the data string will also be scanned for further errors.

In general, each data value with its associated component number is printed on a single line; and the appropriate error message is given on the print line following an incorrect data value. All accepted data values and error-free data values which have been rejected are displayed with no error messages. A total count of the number of accepted and rejected data values is given for each logical entry displayed because of error conditions in that logical entry.

ISSUE REPORT WHEN ALL LEGALITY CHECKED;

The user's data input string is scanned to the end and a checkpoint report consisting of the number of acceptable entries, number of rejected entries, number of values rejected, number of data sets accepted and the number of non-valued data sets accepted is given.

ISSUE REPORT WHEN SELECTION TABLES ARE COMPLETE;

This indicates that a report is to be outputted on to the report file when all data values for all the elements have been merged into the selection tables.

ISSUE REPORT WHEN FINAL SORT IS COMPLETE;

This indicates that a report is to be outputted on to the report file when the data sets have been sorted into data set order and the construction of the retrieval tables can begin.

ISSUE REPORT WHEN LOADING IS COMPLETE;

This indicates that a report is to be outputted onto the report file when the retrieval tables have been constructed and the LOADER module has

successfully loaded the input data string.

ISSUE REPORT WHEN ALL CHECKPOINTS OCCUR;

All four checkpoint reports mentioned above are to be outputted at the proper time on the report file.

STOP AFTER [<integer number> ERRORS | SCAN];

This directs the scanning program in the LOADER module to stop scanning the user's input data after the specified integer number of errors or after all the data has been scanned.

SUPPRESS COMMENTS;

The above command will suppress all user comments encountered in the data input string.

DATA FILE is <filename>;

The user's data file is identified by this command.

THE COMMANDS WHICH FOLLOW BELOW HAVE BEEN INTRODUCED IN THE SYNTAX FOR THE NEW MARS VI SYSTEM.

DIRECT FILE INPUT;

This command informs the system that a record description of the user's file follows, which will be used to create the tables which hold this description.

RECORD <number>;

<number>::= any integer number from 1 to 10. The record number is mandatory only if the user has a multi-record data file.

In the case where this command is not included in the user-directives, the system assumes that the user's data file contains only one type of

record and the record number is assigned a value of 1.

```
RECORD [ ID          ] IS   <record identifier name> ;
       [ IDENTIFIER  ]

FROM [ <column number>      ] [ TO   ] [ <column number>      ] ;
     [ COL <column number>  ] [ THRU ] [ COL <column number>  ]

RECORD [ ID          ] IS <record identifier name>
       [ IDENTIFIER  ]

IN [ <column number>      ] ;
   [ COL <column number>  ]
```

<record identifier name>::= any 1-6 alphanumeric characters

<column number>::= any integer number from 1 to 640

The record identification is only required for a multi-record file, so that the system is able to distinguish between different kinds of records. Variable length user records are allowed and the maximum limit to the length of the user record is 64 words or 1 physical record [5].

KEY COMMON FIELDS FOR THIS LOGICAL ENTRY ARE

```
[ C  <component number> ]
[ <compare keys>        ] ;
```

<compare keys>::= C <component number>,/<compare keys >

This command is mandatory only if the data base is hierarchically structured. The key fields to be compared in order to determine the various records belonging to a logical entry are made available to the system.

```
<component number> <separator> FROM [ COL <column number> ]
                                    [ <column number>     ]
```

[TO / THRU] [<column number> / COL <column number>] [;]

<component number> <separator> IN COL <number> [;]

The location of the fields on the physical record of the user's data file, related to a particular component number description is made available by issuing these commands. If the component number described is the last component number description in the user's record, then a semicolon must terminate that command. If other component number descriptions follow, then this particular component number description must not be terminated with a semicolon. The semicolon at the end of a component number description informs the system that the component number description for the user's record is complete and the necessary component description tables can be now created.

If the user has a multi-record data file then the following sequence of steps is necessary to create the various tables:

1) A unique record number has to be assigned to each new record.
2) A unique record identification is to be associated with each new record.
3) The various components occurring in each of the records have to be described.

SCAN;

Once a complete description of the user's data file has been submitted, this command proceeds to check the validity of data values, as well as loads actual data values into the data base, according to the definition structure specified. Hence a valued data base is only created after issuing this command.

SAVE DATA BASE ON <filename>;

Unless this command is issued a permanent copy of the data base is not [illegible]. This will save the data base on a magnetic tape. The data base is resident on three files on this particular magnetic tape. The files contain

1. The header information, which includes the data base name, the definition version number and the data version number.
2. The actual data base which consists of,
   a. The Definition tables which have been constructed by the DEFINE module.
   b. The Selection tables which have been constructed by the LOADER module and which can be altered by the UPDATE module.
   c. The Retrieval tables which have been constructed by the LOADER module and can be altered by the UPDATE module.
3. This contains trailer records which signal the end of the data base.

The SAVE DATA BASE command can be issued at any time after the LOADER module has constructed the data base. However, it should be issued prior to issuing the EXIT; command which terminates the particular session.

3.4 RETRIEVAL COMMANDS -

RETRIEVAL;

This command is used to call the RETRIEVAL module, which is OVERLAY (4,0) into central memory. If the session starts with the creation of a data base, use of the DEFINE and LOADER modules must precede the use of the RETRIEVAL module. When the session starts with the use of the RETRIEVAL module, the appropriate data base must be loaded. This is accomplished by issuing the following command:

LOAD DATA BASE FROM <filename>;

Error messages from the RETRIEVAL module are sent to the MESSAGE file, whereas the retrieved output is sent to the REPORT file. These files may be specified at any time and if not specifically given are set by default to the on-line printer.

Retrieval requests can have any one of the possible syntactic structures -

PRINT <output collection> WHERE <retrieval specifications>;

PRINT <output collection>;

UNLOAD;

UNLOAD WHERE <retrieval specification>;

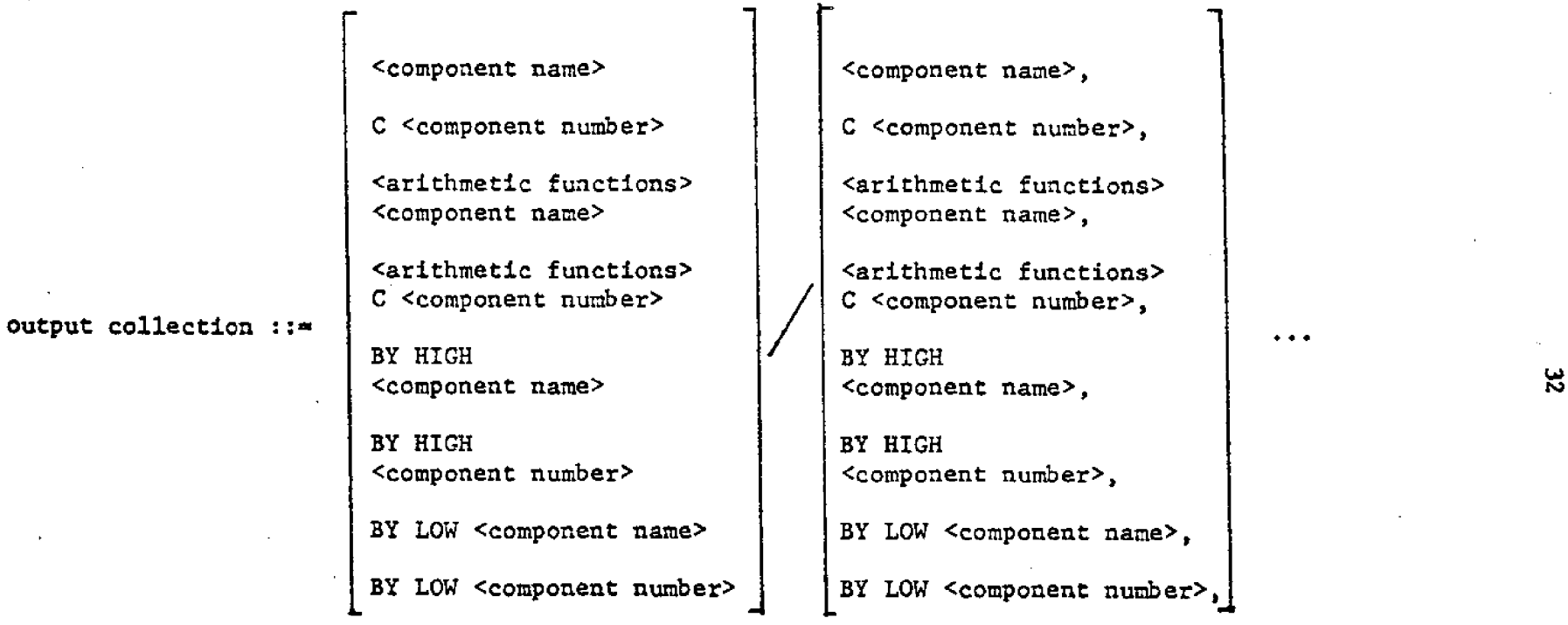
The PRINT request sends output result as desired by the user to the REPORT file, whereas the UNLOAD request produces output in the LOADER string input format.

QUALIFY ON FILE <filename> <output collection> WHERE <retrieval specifications>;

QUALIFY ON FILE < filename> < output collection>;

THE QUALIFY COMMAND IS A FEATURE OF THE NEW MARS VI SYSTEM.

The qualify command is used to retrieve subsets of data and store them on either a disk file or a magnetic tape identified by the filename specified in the command. This retrieved subset of data can then be processed by any processing program, using this file as its data file.


output collection ::=
<component name>
C <component number>
<arithmetic functions> <component name>
<arithmetic functions> C <component number>
BY HIGH <component name>
BY HIGH <component number>
BY LOW <component name>
BY LOW <component number>
<component name>,
C <component number>,
<arithmetic functions> <component name>,
<arithmetic functions> C <component number>,
BY HIGH <component name>,
BY HIGH <component number>,
BY LOW <component name>,
BY LOW <component number>,
...

<arithmetic functions>::= COUNT/SUM/AVG/MAX/MIN/SIGMA

The scope of the output is defined by the component type. Components of element type which appear in the output list cause those elements respective values to be output. Components of repeating group type cause the values of all elements in those repeating groups and their subordinate repeating groups to be output.

The arithmetic functions apply only to the element type components, except the COUNT function.

The arithmetic functions perform the following tasks -

1. COUNT - Totals the number of occurrences of an element value, repeating groups, data sets or logical entries.
2. MAX - Selects the highest data value assigned to an element. This function can be used with alphabetic or numeric data values. If used with alphabetic values, MAX is equal to the last alphabetic entry in a alphabetic listing.
3. MIN - Selects the lowest data value assigned to an element. This function can be used with alphabetic or numeric data values. If used with alphabetic values, MIN is equal to the first alphabetic entry in an alphabetic listing.
4. SUM - Totals the data values assigned to an element. This function can be used with numeric data values only.
5. AVG - Divides the total of the data values for an element by the total number of occurrences of data values for an element, i.e., AVG = SUM/COUNT. This function can only be used with numeric data values.
6. SIGMA - This computes the standard deviation of data values for an element. SIGMA can only be used with numeric data values.

The BY HIGH and the BY LOW refer to the sorting criteria to be performed. The "BY LOW" sort instruction will generate a normal 6400 display code ordering, except that a blank is always treated as the lowest character value. The "BY HIGH" sort instruction will generate an inverted display code ordering.

The sorting criteria is applicable only to the element type components.

It should be observed, that the term "ENTRY" is used to represent the level zero repeating group.

<retrieval specifications>::= <condition> [AND / OR / NOT] <condition>/

<repeating group> HAS <condition>/

<retrieval specifications>

<condition>::= <component name> <relational operator> <user supplied value>/

C <component number> <relational operator> <user-supplied value>/

<component name> EXISTS/<component name> FAILS/

C <component number> EXISTS/C<component number>

FAILS

<relational operator>::= EQ/NE/LT/LE/GT/GE

<user-supplied value>::= data string supplied by the user

The components must be of type element in the "conditions" used in the "retrieval specifications"

The "HAS" clause permits designating a level of hierarchy for retrieval different than that used in the qualifying boolean expression.

.....WHERE SAME;

This expression specifies that the WHERE clause of the immediately preceeding request is to be used as the WHERE clause of this retrieval request.

DITTO WHERE.....;

This expression specifies that the output collection of the immediately preceeding request is to be used as the output collection of this retrieval request.

3.5 UPDATE COMMANDS -

The above command invokes the UPDATA module, which is OVERLAY (3,0).

Let us first consider commands which perform single level modification i.e., modifications which affect only one level of data sets in the data base.

ADD <component name>/C <component number>/ENTRY/CO

CO <data string> WHERE <retrieval specifications>;

Within each selected data set, the ADD operation looks at the status of each element specified in the update request. If the element already has a value, no action is taken, otherwise the new user-supplied data string is added to the contents of the data set.

ENTRY or CO specifies the level 0 data set. The components may be of either type element or repeating group.

The "retrieval specifications" are the same as in the retrieval requests.

<data string>::= <data values> <separator> <separator> <entry terminator>

If the component type is repeating group then the data values are in the logical entry format.

<logical entry format>::= <data set> <separator> <separator> <entry terminator>

<data set>::= <repeating group component number> <separator> <data items>/ <data set>

<data items>::= <component number> <separator> <data value>/<data items>

<repeating group component number>::= any user assigned 1-4 digit number to a repeating group type component.

<data value>::= actual user assigned values to the components in the data base

CHANGE <component name>/C <component number>/ENTRY/CO EQ <data string> WHERE <retrieval specifications>;

For each selected data set if the element has a value, it is changed to the new value supplied in the data string. If the element has no value, no action is taken.

The other requirements are similar to the ADD operation.

REMOVE <component name>/C <component number>/ENTRY/CO WHERE <retrieval specifications>;

If the specified component is of element type, then for each selected data set, if that element has a value, the value is removed, otherwise no action occurs.

If the specified component is of type repeating group or ENTRY, then every value is removed from every selected data set.

ASSIGN <component name>/C <component number>/ENTRY/CO EQ <data string> WHERE <retrieval specifications>;

The ASSIGN operation assigns unconditionally new values for each element specified in the update request, for each of the selected data set.

The other requirements are similar to the ADD and CHANGE operations.

Multi-level or tree modifications affect more than one level of data sets, i.e., descendent data sets that lie below the selected data sets, are also modified.

The components can only be of type repeating group in all TREE operations.

REMOVE TREE ENTRY/CO/ <component name>/C <component number> WHERE <retrieval specifications>;

This operation removes each selected data set and all of its descendent data set. Each selected data-set is the top-most node of a data tree. After each data tree has been removed, the remaining data trees in the data base are relinked to close the gap at the topmost level created by the removal.

ASSIGN TREE ENTRY/CO/ <component name>/C <component number> EQ <data string> WHERE <retrieval specifications>;

This operation performs, for each selected data set, a REMOVE TREE operation and then assigns an entirely new data tree containing the data string for the request. In the data string all level 0 values must be grouped together at the beginning of the data string. The level of assignment in the data string must never return to the first specified repeating group level, but descendents and their descendents can be assigned as many values as desired.

INSERT TREE ENTRY/CO/<component names>/C <component number> EQ <data string> $\left[\begin{matrix}\text{BEFORE}\\ \text{AFTER}\end{matrix}\right]$ <retrieval specifications>;

This operation uses the selected data sets as a guide, never changing the contents of any selected data sets. The data string of new values is unlimited except that only one data set can be inserted at the level of the specified repeating group; the number of descendent data sets from the new topmost parent data set is unlimited.

If the connector word is BEFORE, the new data tree is inserted before each selected data set. If the connector word is AFTER, the new data tree is inserted after each selected data set.

.....PREVIOUS WHERE <retrieval specifications>;

The PREVIOUS option is similar to the DITTO option encountered in the retrieval request, except that it pertains to the data string. When this option is used the previous data string is simply used in this particular update request.

In order to make permanent the update actions executed, the updated data base must be saved on a magnetic tape, by issuing the save data base command.

3.6 REPORTER COMMANDS -

REPORTER;

This invokes the retrieval module which is overlay (6,0).

A report can only be generated if subsets of data have been retrieved from the data base by use of the RETRIEVAL module. However, if the retrieved data is to be used by the REPORTER module, all retrieval requests should contain the word QUALIFY, instead of the word PRINT. This causes the retrieved output to be structured for input to the RETRIEVAL module and assigns the extracted data to a SCOPE scratch file.

FORMAT;

This informs the module that a report format description is to follow.

The report format descriptions of interest to the typical user are:

a. Report Title for Report

b. Column Heading for report

c. Components to be included in the report

[TITLE IS / T] <literal string>;

<literal string>::= any alphanumeric string of display code characters except the "," or the ";"/,<literal string>;

This placed the literal string values on the first page of the report. Titles are left-justified.

[HEADING IS / H] <literal string>;

This places the literal string values beneath the title lines on the first page and at the top of each subsequent report page. Headings are left justified.

[DETAIL IS / D] <component name> / C <component number>
[<component name>, / C <component number>,] ...;

FINIS;

This indicates the end of the record format description.

GENERATE;

This causes the report to be printed on the REPORT file under the record format description.

4. FILE ORGANIZATION OF THE SYSTEM

Since the basic purpose of this system is to manage large and complex files of data effectively, the problem of file organization is of paramount importance. The system's file organization is critical in determining its speed of response, use of storage, maintenance of structural relationships among data, ability to manipulate data, and usefulness of output.

Users of information systems whose needs are predefinable rarely systems with complex file structuring capabilities. Serial-access ...es on tape or random access storage devices with dictionaries containing a few known key words are adequate enough to meet the demands of such ...ential users.

However, most growing organizations are already handling their predefinable needs to their satisfaction and are anxious to give a more ...ely response to their nonpredefinable information needs, which usually originate from the higher management echelons.

The inverted file structure used as the basis for the file organization in this system is directed towards such ad hoc queries.

4.1 FILE STRUCTURE OF THE SYSTEM -

In the earlier days of file processing, a file was taken to be a collection of records, each of which was assigned a unique address. These unique addresses were collected into a set called the directory. In order to locate a set of records, having some common key word, all the addresses in the directory were used for bringing in the corresponding records for examination to see if it contained the desired key word. This, of course, constituted uneconomical file processing [11].

The inverted file structure was developed to increase file processing efficiency. In this type of structure, the directory contains a copy of its key words and each of the key words is associated with the addresses of all those records which contain that key word. Hence in order to find a set of records containing a common key word, one has only to scan the directory, and on finding the desired key word all the associated records are available.

In this system, the inverted file structure is totally organized and ...rted by component value and occurrences of each value. Any value of any ...ponent can be used as a key for retrieving information. All potential ...ys are grouped into blocked tables in a manner designed to minimize ...rage transfer requirements.

The definition tables define only the structure of the data base. After ...a values have been entered into the data base the data base now consists ... the definition tables, the selection tables and the retrieval tables. ...e tables are partitioned into blocks of 256 central memory words and these partitions are transferred from disk to central memory as required by the system.

4.2 DESCRIPTION OF TABLES -

The table organization is schematically illustrated in Figure 4.1. A summarization of table contents and an explanation of the pointers is appended to clarify the schematic representation.

The definition tables are:

1. CDEFINA AND CDEFINB TABLES - Each entry in these tables corresponds to a component in the logical entry definition as declared by the user. Items in the tables contain properties and attributes for each component and the relationship among components as described by the user. The maximum number of components allowed in a data base is 1023 and the nesting of repeating groups is 63 below the level 0 components.

Hence in essence the CDEFINA table contains the logical entry structure ordered by user definition, whereas the CDEFINB table contains the user-

ned component number and the type of value, associated with that
ponent.

CELS TABLE - Each entry in the CELS table corresponds to a component in the logical entry definition. A maximum of 150 characters is allowed for user-defined component names. This table is not ordered as it is accessed by a relative address pointer from CDEFINB. This relative address pointer is the system-assigned component number, which is used throughout the selection and retrieval tables, rather than the user defined component number.

The selection tables are -

1. CVALDIR TABLE - Each entry in the CVALDIR table corresponds to a single partition in the CVALUES table. For each element in the logical entry definition for which data values have been entered, there are as many CVALDIR entries as the number of CVALUES partitions created for that element. CVALDIR entries need not be ordered, as the set of entries pertaining to a single item are linked. CVALDIR entries indicate the highest value in a CVALUES partition and, thus are used to point to the CVALUES partition where a data value may be found if it exists in the data base for the given element, i.e., it is a pointer to the associated block.

Thus CVALDIR is a page directory to values for each component having a value in the data base; and is accessed by item CPVDIR of the CDEFINE table.

2. CVALUES TABLE - Each unique data value assigned to a given element is represented by one and only one entry in the CVALUES table. Each CVALUES partition contains entries pertaining to only one element. Entries within

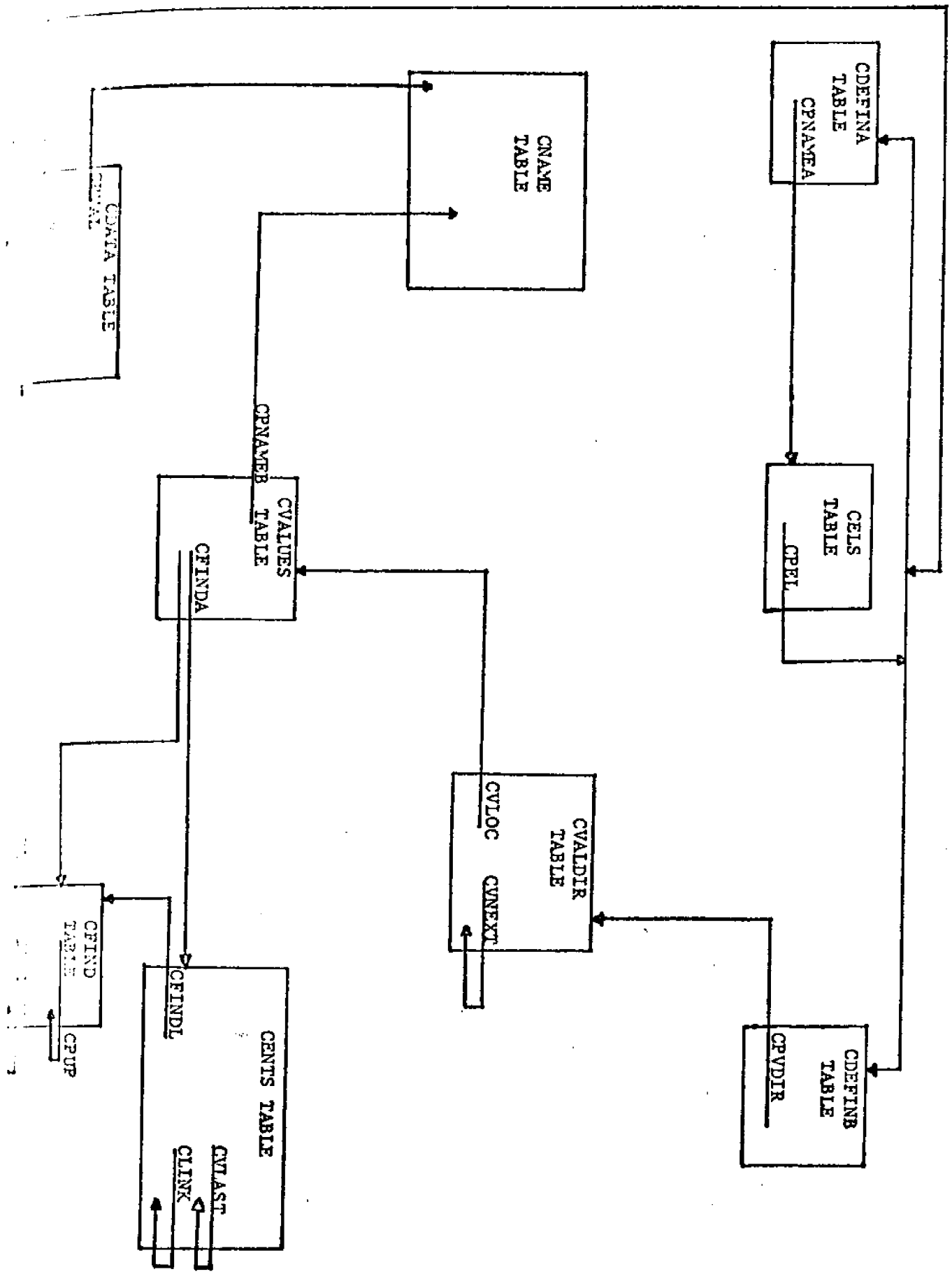


Figure 4.1 TABLE ORGANIZATION WITH RELEVANT POINTER ASSOCIATIONS

partition are ordered by low value first with respect to the total ... in table CNAME, but the set of partitions for any given element ... not be ordered or consecutive in the CVALUES table. Only the first ... characters of each unique value are stored, and there is a pointer to ... full literal string in the CNAME table. If one were to attempt a ...ary search in such a table, the disc transfer time would be excessive. ...e the CVALDIR TABLE is used so that within one disc transfer the ...propriate block of value list can be located and consequently searched.

3. CENTS TABLE - Each value in the CVALUES is cross-indexed to the CENTS table where the occurrences of that value may be found. These occurrences are called "data sets." The first word of each list contains the count of occurrences. The list of set numbers is ordered from low to high. If the value is unique, instead of pointing to the CENTS table, CVALUES points directly to the CFIND table. There is an item in the values list indicating whether or not any value has occurred only once.

4. CNAME TABLE - A CNAME entry exists for each unique data value assigned to an element. If the value was assigned to a specific element in more than one data set, one and only one copy of the value is retained in the CNAME table, and it is pointed to by pointers in the appropriate CDATA table entires. Entries in the CNAME table are used in matching when uniqueness is not assured by the corresponding CVALUES entry and are used in generating report output.

The retrieval tables are:

1. CFIND TABLE - CFIND Table is indexed by the CENTS TABLE. It resolves any hierarchical retrieval problems and locates the entries in the data table.

CFIND contains an entry for each data set that exists in CDATA. If no items were entered for a data set but other data sets exist at a deeper logical level for the same family of data sets, then a dummy CFIND entry is created for the null parent set. There are four items in CFIND that are used to accomplish retrieval of hierarchical data (1) a repeating group identifier (RGID), (2) a up pointer (3) right-pointer and (4) a down-pointer. They have the following functions: the RGID identifies the group with which each CFIND entry (data set) is associated (the identifier at level 0 in each logical entry is always zero). The up- and down-pointers are relative pointers that contain positive values. The up-pointer contains the relative location of each data set's senior in the hierarchy tree (lower data set number). Except at level zero, the up-pointer always points up. At level zero, this points to the relative location of the next logical entry. The right-pointer contains the relative location of the next data set-number at the same level (higher data set number). If there are no more data sets in the logical entry at the same level, the right-pointer equals zero. The right-pointer links all data sets at the same level across family boundaries. The down-pointer points to the junior of each data set in the hierarchy tree (higher data set number). If a data set has a junior, it is guaranteed to be in the next higher data set.

The last piece of information in CFIND is a pointer to the location of the associated data set in CDATA. Hence, the CFIND table preserves the vertical and horizontal hierarchical structures within a logical entry and between logical entries.

CDATA TABLE - Each CDATA entry serves as a directory to access actual data values entered for specific elements in a single data set. Every data set in the data base that contains at least one data value has a CDATA entry (accessed by item CPLOC of the corresponding CFIND entry for that data set). If the data set is a dummy entry in the CFIND table, then no CDATA entry exists and item CPLOC=0. The CDATA TABLE contains a component number which is a direct index back to the dictionary. It also contains a pointer into the CNAME table.

CDATA is constructed at load time in the order in which data arrives. If there is no hierarchical information in the logical entry, then there is only one data set in each entry. That data set is defined as level 0. A data set in CDATA is created for every occurrence of repeating group information at one level. A data set may be thought of as entry within an entry. There is an item in CDATA that indicates when the end of a data set has occurred.

4.2 (a) SUMMARIZATION OF TABLE CONTENTS --

1. CDEFINA TABLE - Logical entry structure declared by user definition.
2. CELS TABLE - String of display code characters comprising the user defined component name.
3. CDEFINB TABLE - User-defined component number and type of value associated with the component.
4. CVALDIR TABLE - Directory table to values for each component.
5. CVALUES TABLE - Unique values of components in an ordered and paged fashion, containing truncated or binary value.
6. CNAME TABLE - Display code string containing full display value; only one unique value for each component. The table is unordered.

CENTS TABLE - Pointers to the different data sets containing the same value for a component. The table is unordered.

CFIND TABLE - Hierarchical structure of data sets and logical entries in the data base. The table is unordered.

CDATA TABLE - Grouped by data sets and containing pointers to the component number and the value associated with it.

.2 (b) EXPLANATION OF POINTERS -

1. CPNAMEA - Pointer to the header word of the CELS table entry which contains the component name for this component.
2. CPEL - Pointer to the CDEFINA and CDEFINB table entries associated with this component name.
3. CPVDIR - Pointer to the first CVALDIR entry for an element.
4. CVNEXT - Pointer to the next CVALDIR entry for the same element.
5. CVLOC - Pointer to the first word of the relevant CVALUES partition.
6. CPNAMEB - Pointer to the first word of the CNAME table entry which contains the entire value string in display code associated with this representation in the CVALUES entry.
7. CPFINDA - Pointer to the first word of the CENTS table if this value occurred more than once for this element; otherwise this is a pointer to the single CFIND table entry in which data set this value occurred.

12. CVLAST - Pointer to the last item in the entire CENTS entry for this unique data value.

13. CLINK - Pointer linking all blocks pertaining to multiple occurrences of the same unique data value.

14. CFINDL - Pointer to the CFIND table entry in which data set this value occurred.

15. CPUP - Pointer to the CFIND entry for the data set at level n-1 with which this level n data set is associated as a repeating group.

16. CPRIGHT - Pointer to the next data set on the horizontal chain of data sets at level n, which have the same parent data set at level n-1.

17. CPDOWN - Pointer to the CFIND entry for the data set at level n+1 associated with this level n data set.

14. CPLOC - Pointer to the beginning of the CDATA table entry for this CFIND entry if any data values have been entered for this data set.

15. CPVAL - Pointer to the relevant CNAME table entry which in turn contains the actual data value associated with this element.

16. CPVAR - Pointer to the CDEFINA and CDEFINB table entry for the element in this data set.

17. CHOOK - Pointer which links a sequence of blocks that comprise one CDATA entry.

4.3 ANALYSIS OF FILE STRUCTURE -

As the data base is comprised of various tables, with each table being

ically partitioned into blocks of 256 central memory words, the size of the data base does not necessarily determine the speed of response to a query, since only a very small portion of the data base needs to be examined in order to determine the entries which qualify for retrieval.

The important factors in determining response time are:

1. Number of Boolean expressions in the WHERE clause

2. Number of unique values occurring per component in each Boolean expression involved in the WHERE clause.

3. Number of occurrences of the desired value in each Boolean expression, i.e., number of occurrences of the desired value in the various data sets.

In order to achieve this retrieval speed certain concessions have to be granted. The inverted file structure employed requires three times more storage space than that required by the original data, if all components in the data base are level 0 elements. Hence, in order to minimize storage requirements the concept of structuring data in a hierarchical structure must be employed.

The advantages of employing a hierarchical structure are:

1. Hierarchical structure is used to force physical proximity of the various child segments related to the parent segment.

2. A hierarchical structure is used to eliminate the storage of redundant data fields associated with the child structures, i.e., if these are a group of children records all with the same component value, the component value is relegated to the parent repeating group.

However, the use of hierarchical data structures at best allows the data base to occupy the same storage as the original data. But this is not a limiting factor as storage is becoming inexpensive, whereas ...

time people want to expend in order to get information is becoming more and more important.

Another important consideration of this file structure is its extreme sensitivity to ordering. Most of the tables which comprised the data were ordered by component value or set number depending on the information they contained. Hence, dynamic updating of the data base becomes a difficult problem. In the case, where only small changes are expected, a small amount of space allocated in the selection and retrieval tables, by the use of the nulls and padding feature, provides for rapid updating. However, if the changes are massive, dynamic updating in this system cannot be performed very rapidly. In fact massive doses of updating would be better carried out in the background mode (batch mode) rather than in the foreground or interactive mode.

Although this particular file organization requires more disk storage than other file organizations, it is particularly suited for rapid retrieval of information for ad hoc queries.

5. AN EXAMPLE OF DATA BASE CREATION AND USAGE

In an attempt to illustrate the potential of the new MARS VI system, the implementation of a fisheries information system for the analysis of catch and effort data, will now be given.

5.1 DATA COLLECTION - The Washington State Department of Fisheries has been collecting catch and effort information regarding 40 odd species of groundfish off the coast of Washington State, British Columbia and Alaska, since 1953. There are approximately 150,000 records of data collected in the span 1953-1970. Each of these records is an 80 character card image which

s been coded onto magnetic tapes in binary coded decimal (BCD) on scope format with a density of 555 characters per inch (cpi). The various fields of information existing in a record are illustrated in Table 5.1 [13].

Record Structure of Washington State Groundfish Data

	Columns	
Year	(1-2)	Last two digits of current year.
Month/Date	(3-6)	Number of month (1-12) and date of month of landing i.e. when the interview is taken.
Port	(7-10)	Col. 7: Primary area i.e. Seattle and Bellingham, which represent two port clusters. Col. 8-10: Secondary area i.e. port of landing within the port cluster.
Days Out/Fished	(11-14)	Col. 11-12: Total number of days from time vessel leaves home port to day of return. Col. 13-14: Total number of days fished.
Boat/Class	(15-16)	Col. 15-17: Vessel code number. Col. 18: Vessel class number.
Fishing Aids	(19-21)	Col. 19-20: Blank: these columns are not presently being used. Col. 21: Vessel fishing aid code for horsepower.
Species	(22-24)	Code number used to designate a particular species.
Area	(25-30)	Pacific Marine Fisheries Commission - Col. (25-26). Washington statistical areas - Col. (27-28). Ground areas - Col. (29-30).

TABLE 5.1

;;l Hours	(31-33)	The total number of hours (to the nearest tenth) fished in each area and at a particular depth.
.··	(34)	Code number which best defines fishing efficiency e.g. effective fishing with adequate catch per effort is designated the code number 1.
;3	(35-36)	Number of drags in each area.
;th	(37-45)	Col. 37-30: Minimum depth fished in each area. Col. 40-42: Maximum depth fished in each area. Col. 43-45: Average depth fished in each area.
itch	(46-59)	Col. 46-52: Total catch of all species at a particular depth. Col. 53-59: Estimated catch by species for a particular depth.
ercentage	(60)	Percentage of species in total catch.
Price	(61-63)	Price of species in cents per lb.
Net Type	(66-67)	Net type code number.
Mesh	(68-69)	Mesh size to nearest tenth inch without decimal place.
Limits	(70-71)	Limit restriction on species.
Search Hours	(72-73)	Actual hours spent on the grounds, running with fish finding gear operating, to the nearest hour.
	(74-80)	Not presently being used.

Comments - In essence each record contains the estimated catch in lbs. with reference to a particular area, a particular species, a particular depth, and a particular date. It should be noted that the date for a particular trip is the date of landing and is hence common to all records created for that trip.

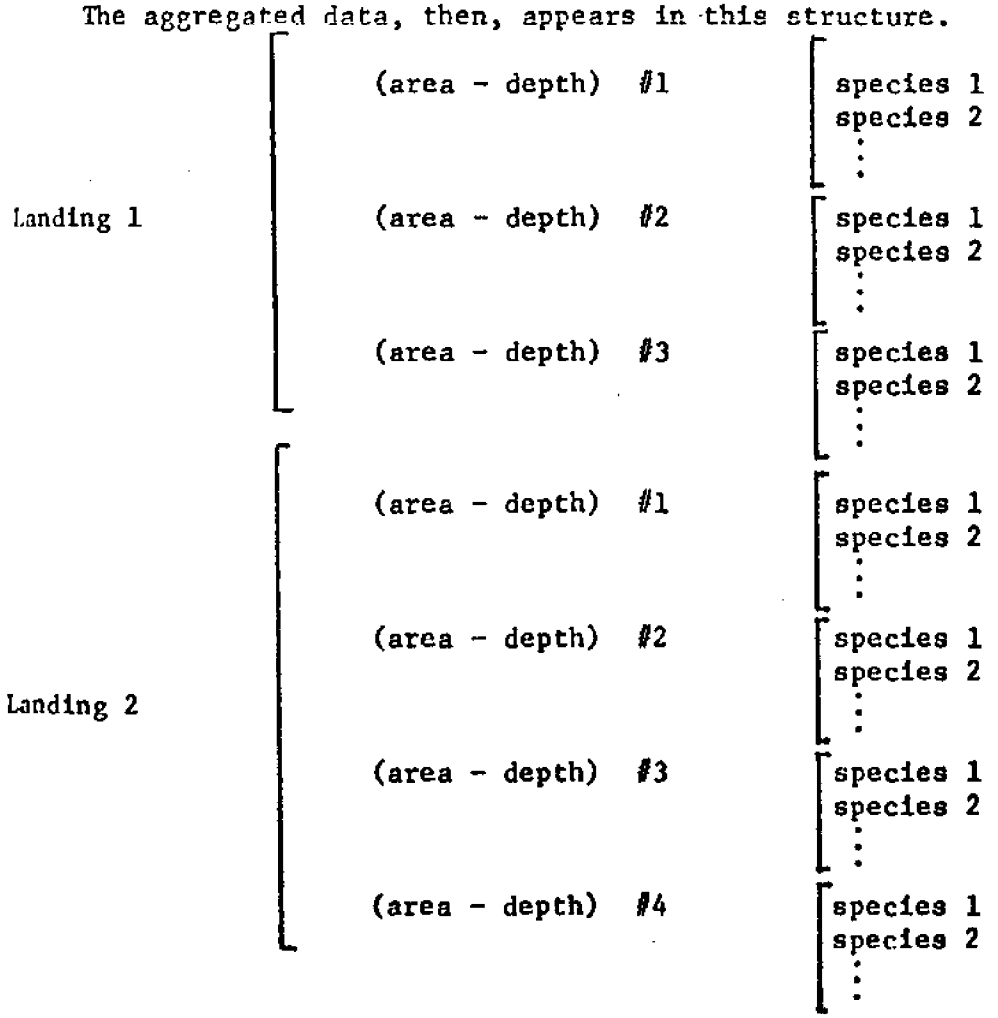
TABLE 5.1 (Contd)

The data records are divided into year files and within each year the data is sorted in the ascending order according to the following sort order,

1. Date of interview
2. Boat number
3. Washington State area
4. Depth of fishing

5.2 DATA STRUCTURE - In the groundfish interview data one card image represents "a series of drags by a particular boat, for a particular trip, catching any one species at any one depth in any one area." "Depth" is approximately by ten-fathom intervals.

The aggregated data, then, appears in this structure.



Landing-specific information includes date of landing, i.e., date when interview is conducted, port of landing, boat number, boat class, fishing aids (horsepower of boat), days fished and days at sea.

Area-depth specific information includes area, depth, hours fished, number of drags, total catch and hours searched.

Species-specific information includes species code, species catch, percent level of species within total catch, price per pound, net type, mesh size and limits.

In essence each record contains the estimated catch in pounds with reference to a particular area, a aprticular species, a particular depth, and a particular trip (identified by date of landing).

5.3 CREATION OF DATA BASE - Having understood the mode of collection of data as well as the record structure of records in the data file, one can now proceed to define the structure of the data base. In order to illustrate the new MARS VI system, two structures for the data base will be given. The definition structure for the first data base will be constituted of linearly structured entries, whereas the definition structure for the second data base will be constituted of hierarchically structured entries.

The DEFINE section for Data Base I should be: [Appendix D].

```
DEFINE;
NEW DATA BASE   DATA BASE 1;
1)  DATE OF LANDING (INTEGER NUMBER)
4)  PORT OF LANDING (INTEGER NUMBER)
6)  TOTAL DATA SPENT ON TRIP (INTEGER NUMBER)
7)  DAYS SPENT FISHING  (INTEGER NUMBER)
```

8) BOAT NUMBER (INTEGER NUMBER)

9) BOAT CLASS (INTEGER NUMBER)

10) WASHINGTON STATISTICAL AREA (INTEGER NUMBER)

15) TOTAL HOURS FISHED (INTEGER NUMBER)

17) NUMBER OF DRAGS (INTEGER NUMBER)

20) AVERAGE DEPTH OF FISHING (INTEGER NUMBER)

21) TOTAL CATCH OF ALL SPECIES (INTEGER NUMBER)

11) SPECIES CODE (INTEGER NUMBER)

22) TOTAL CATCH OF MAJOR SPECIES (INTEGER NUMBER)

MAP;

The linear structure formed by the above definition of the data base is illustrated in TABLE 5.2.

The actual loading of the data base according to the definition structure constructed above in order to create a valued data base, is accomplished by executing the following command statements:

Table 5.2

Linear Structure of Data Base

Level 0
Date of Landing
Port of Landing
Total Days Spent on Trip
Days Spent Fishing
Boat Number
Catch Information
Boat Class
Wash. Statistical Area
Total Hours Fished
No. of Drags
Ave. Depth of Fishing
Total Catch of all Species
Species Code
Total Catch of Major Species

```
    LOADER;
    DATA FILE IS GFISH;
    DIRECT FILE INPUT;
1)  FROM COL 1 TO COL 6
4)  IN COL 7
6)  COL 11-COL 12
8)  15-17
9)  IN COL 18
13) FROM COL 27 THRU COL 28
15) 31 THRU 33
17) COL 35 THRU COL 36
20) 43-45
21) COL 46-COL 52
11) COL 22-COL 24
22) FROM COL 53 to COL 59;
ISSUE REPORT WHEN ALL LEGALITY CHECKED;
SCAN;
```

The DEFINE section for Data Base II should be: [Appendix G]:

```
DEFINE;
NEW DATA BASE  DATA BASE II;
1)  DATE OF LANDING (INTEGER NUMBER)
4)  PORT OF LANDING (INTEGER NUMBER)
6)  TOTAL DAYS SPENT ON TRIP (INTEGER NUMBER)
7)  DAYS SPENT FISHING (INTEGER NUMBER)
8)  BOAT NUMBER (INTEGER NUMBER)
9)  BOAT CLASS (INTEGER NUMBER)
```

WASHINGTON STATISTICAL AREA (INTEGER NUMBER)

) TOTAL HOURS FISHED (INTEGER NUMBER)

) NUMBER OF DRAGS (INTEGER NUMBER)

) AVERAGE DEPTH OF FISHING (INTEGER NUMBER)

) TOTAL CATCH OF ALL SPECIES (INTEGER NUMBER)

) CATCH INFORMATION (REPEATING GROUP)

1) SPECIES CODE (INTEGER NUMBER IN 30)

2) TOTAL CATCH OF MAJOR SPECIES (INTEGER NUMBER IN 30)

MAP;

The hierarchical structure formed by the above definition of the data base is illustrated in Table 5.3.

Once the data structure of the data base has been defined one proceeds to actually loading the data values into the data base in order to create a valued data base. This is accomplished below:

```
LOADER;
DATA FILE IS GFISH;
DIRECT FILE INPUT;
KEY COMMON FIELDS FOR THIS LOGICAL ENTRY ARE C1, C8, C13, C20;
1) FROM COL 1 TO COL 6
4) IN COL 7
6) COL 11-COL 12
7) COL 13 THRU COL 14
8) 15-17
9) IN COL 18
13) FROM COL 27 THRU COL 28
15) 31 THRU 33
```

HIERARCHICAL STRUCTURE OF DATA BASE
Table 5.3

) COL 35 THRU COL 36

) 43-45

) COL 46-COL 52

) COL 22-COL 24

) FROM COL 53 TO COL 59;

ISSUE REPORT WHEN ALL LEGALITY CHECKED;

SCAN;

The key common fields are used to inform the system as to which fields are to be compared in order to identify records belonging to the same logical entry. In the case of this particular data structure,

C1 = Date of landing

C8 = Boat number

C13 = Washington Statistical Area

C20 = Average depth of fishing, are to be compared to achieve hierarchical association.

5.4 USAGE OF DATA BASE - Potential users of the groundfish data base could be fisheries biologist, as well as management personnel from state agencies responsible for regulating and imposing effective controls on the groundfish industry.

The functions of retrieving subsets of data from the data base, conducting statistical analyses on the retrieved data, updating the data base, as well as generating reports would be available to potential users of the system. [Appendix H].

Retrieval requests which might be of interest to users of Data Base I and Data Base II are listed below with suitable comments to enhance the

larity of the requests:

PRINT C1, C4, C8, C9, C13, C11, C22 WHERE C1 LT 540101;

The output collection contains the date of landing, port of landing, boat number, boat class, Washington Statistical Area, species code and the total catch of major species for all data of landings which were reported for the year 1953.

PRINT BOAT NUMBER, BOAT CLASS WHERE DATE OF LANDING EXISTS;

The output collection would contain all the boat numbers and their corresponding boat classes which reported even a single landing.

PRINT SUM C22, C11 WHERE C13 EQ 02 AND BOAT CLASS EQ 6;

The output collection would contain the sum of all major species and their species code for all species caught in Washington statistical area 02 by boats belonging to class 6.

PRINT ENTRY WHERE C9 GT 2 OR C13 LE 80 AND SPECIES CODE EQ 254;

The output collection would be a list of all data set(s) in the data base which have boat class greater than 2 or Washington Statistical area less than or equal to 80 and species code equal to 254.

For the hierarchically structured Data Base II it is further possible to issue the following request,

PRINT C1, C11, C22 WHERE CATCH INFORMATION HAS SPECIES CODE EQ 251 OR TOTAL CATCH OF MAJOR SPECIES LT 30000;

The output list would contain the data of landing, species code and total catch of major species for all data set(s) which satisfy the boolean expression in the WHERE clause. The interesting feature of this request is that if a certain data set in a particular repeating group family qualifies for retrieval, then all data sets belonging to that repeating group family also qualify for retrieval.

Other retrieval requests which conform to the syntactical structure specified in the description of the retrieval commands may be specified by the user. Diagnostic messages will occur if the syntactical structure is incorrect or if the WHERE clause is not satisfied [17].

If the user desires to manipulate data values in the data base involving logical entries, portions of logical entries, data sets or isolated data items, the update requests permit him this capability.

Some possible update request for Data Base I and Data Base II are listed below:

ADD BOAT CLASS EQ 3)) END WHERE BOAT NUMBER EQ 126;

The boat number 126 is assigned boat class 3 if that particular data item (i.e. boat class) does not already have a value. If it has a value, no action is taken.

CHANGE C4 EQ 1)) END WHERE C1 EQ 530506 and C8 EQ 199;

The port of landing is assigned the value 1 for the selected data set (s) if that data item already has a particular value. If the data item does not have a value, no action is taken.

REMOVE ENTRY WHERE C8 EQ 99 AND C13 EQ 10;

The entire level 0 data set(s) which satisfy the boolean expression are removed from the data base.

ASSIGN C13 EQ 02)) END WHERE C8 EQ 260;

All selected data sets are altered, whether the particular data item i.e. Washington Statistical Area has a value or not. Hence this command is unlike the ADD and CHANGE operations which are conditional.

The update requests below may be issued only for Data Base II:

ASSIGN CATCH INFORMATION EQ 11)254 22)25000)) END WHERE C11 EQ 254;

As the request specifies catch information which is a repeating group each selected data set is entirely empyied of all values and filled entirely with the values specified in this update request.

REMOVE TREE ENTRY WHERE C9 EQ 9;

This would remove all logical entries which satisfy the boolean expression. In this sense, tree operations affect more than one level of data sets, i.e., they are multi-level operations.

ASSIGN TREE CATCH INFORMATION EQ 11)251 22)15000)) END WHERE C1 LE 530501;

For each selected data set, a REMOVE TREE operation is performed and then an entirely new data tree containing the data string specified in the request is inserted in its place.

INSERT TREE ENTRY EQ 1) 530413 4) 2 6) 10 7)06 8) 151 9) 1 13) 08 15) 010 17) 25 20) 080 21) 55000 11) 256 22) 40000)) END BEFORE C1 EQ 530416;

The user specified logical entry is inserted in the data base before the logical entry (s) with date of landing 16th April 1953. The AFTER clause may be used instead of the BEFORE clause depending on where the insertion of the new logical entry is desired.

In case new logical entries are to be appended to the end of an already existing data base it is more efficient and economical to use the LOADER module. As an example, consider that catch and effort data for 1954 is to be integrated with the 1953 data base in order to develop a comprehensive data base. In order to develop a comprehensive data base which is linearly structured, the procedure outlined below is required:

LOAD DATA BASE FROM L1953;

```
    LOADER;
    DATA FILE IS DFISH;
    DIRECT FILE INPUT;
1)  FROM COL 1 TO COL 6
4)  IN COL 7
6)  COL 11-COL 12
7)  COL 13 THRU COL 14
8)  15-17
9)  IN COL 18
13) FROM COL 27 THRU COL 28
15) 31 THRU 33
17) COL 35 THRU COL 36
20) 43-45
21) COL 46-COL 52
11) COL 22-COL 24
22) FROM COL 53 TO COL 59;
ISSUE REPORT WHEN ALL LEGALITY CHECKED;
SCAN;
SAVE DATA BASE ON L5354;
EXIT;
```

It was assumed that the 1953 linearly structured groundfish data base i.e. Data Base I was available on filename L1953; whereas the 1954 groundfish catch and effort data is available on filename DFISH.

The comprehensive linearly structured data base for 1953 and 1954 groundfish data is now saved on a magnetic tape, identified by its filename, L5354.

For the purpose of creating a comprehensive hierarchically structured data base, it is assumed that the 1953 hierarchically structured groundfish data base is available on filename H1953, whereas the 1954 groundfish catch and effort data is available on filename DFISH.

The procedure outlined below is required to achieve the desired objective:

```
LOAD DATA BASE FROM H1953;
LOADER;
DATA FILE IS DFISH;
DIRECT FILE INPUT;
KEY COMMON FIELDS FOR THIS LOGICAL ENTRY ARE C1, C8, C13, C20;
1)  FROM COL 1 to COL 6
4)  IN COL 7
6)  COL 11-COL 12
7)  COL 13 THRU COL 14
8)  15-17
9)  IN COL 18
13) FROM COL 27 THRU COL 28
15) 31 THRU 33
17) COL 35 THRU COL 36
20) 43-45
21) COL 46-COL 52
11) COL 22-COL 24
22) FROM COL 53 TO COL 59;
ISSUE REPORT WHEN ALL LEGALITY CHECKED;
SCAN;
```

SAVE DATA BASE ON H5354;

EXIT;

The comprehensive data base for the 1953-1954 groundfish catch and information data would now be available on filename H5354. It is mandatory that the filename be a magnetic tape, otherwise the data base will not be saved.

In order to generate reports of the retrieved subsets of data, the following procedure should be applied for data retrieved from either Data Base I or Data Base II:

```
RETRIEVAL;
QUALIFY C1, C9, C13, C21 WHERE C1 EXISTS;
QUALIFY C8, C9 WHERE C1 EXISTS;
REPORTER;
FORMAT;
LINES PER PAGE = 55;
TITLE IS GROUNDFISH 1953 CATCH REPORT;
HEADING IS DATA OF LANDING, BOAT CLASS, WASHINGTON STATISTICAL AREA, TOTAL
CATCH OF ALL SPECIES;
DETAIL IS C1, C9, C13, C21;
FINIS;
FORMAT;
LINES PER PAGE = 40;
TITLE IS GROUNDFISH 1953 BOAT INFORMATION:
HEADING IS BOAT NUMBER, BOAT CLASS;
DETAIL IS C8, C9;
SP 13 IN D1;
FINIS;
```

EXPORTER;

GENERATE;

Statistical analyses of the retrieved subsets of data from Data Base I and Data Base II may be performed either in the same session, or saved on a magnetic tape for input to a statistical package or program at a later time. However, the same retrieval request is required in either case.

QUALIFY ON FILE SRETV C1, C13, C21, C22, C11 WHERE C11 EQ 251;

This would retrieve all data set(s) which satisfy the boolean expression and the output collection would be available on filename SRETV, which may be either a disk scratch file or a magnetic tape. The format specifications for the output collection would be in accordance with the format specified by the user's input data file.

6. CONCLUSIONS

6.1 CONCLUSIONS OF THE RESEARCH

In order to compare the performance of the new MARS VI system with the old MARS VI system, two data bases were created using both systems.

Both the data bases used 2101 records from the 1953 groundfish catch and information data. Data Base I was linearly structured and hence was composed of 2101 logical entries, each logical entry having 13 different attributes. Data Base II was hierarchically structured and this hierarchical structuring, was composed of 648 logical entries having 14 different attributes.

The cost of creating the two data bases in the old and the new MARS VI system is summarized in Table 6.1.

The cost of creating a data base using the new MARS VI system can be

TABLE 6.1

COST OF CREATING DATA BASE

	OLD MARS VI SYSTEM	NEW MARS VI SYSTEM
DATA BASE I	CP = 351.346 SEC	CP = 256.409 SEC
	PP = 065.042 SEC	PP = 115.019 SEC
	IO = 079.393 SEC	IO = 068.915 SEC
	COST = $41.14	COST = $35.54
DATA BASE II	CP = 161.324 SEC	CP = 106.816 SEC
	PP = 037.373 SEC	PP = 062.779 SEC
	IO = 033.278 SEC	IO = 025.925 SEC
	COST = $19.33	COST = $14.64

further reduced by decreasing the field length requirements. Presently the new MARS VI system requires a field length of 75000 (octal) central memory words whereas the old MARS VI system requires a field length of 60000 (octal) central memory words. This increase in the field length of the new MARS VI system came about as a result of the new programs added to enhance the old system. However by causing these new programs to be treated as a separate secondary overlay, the field length may be reduced back to 60K (octal) central memory words.

It is expected that the reduction in the field length requirements of the new MARS VI system will cut the costs indicated in Table 6.1 as the job cost formula is,

$$C_i = P * [CP_i + FL_i (PP_i + 4 * IO_i)]$$

where

P = job priority (1-4)

C_i = the cost of the i^{th} step

CP_i = the central processer time used in the i^{th} step

FL_i = fraction of the central memory used in the i^{th} step

(200,000 octal = 100% of central memory)

PP_i = peripheral processor time used during the i^{th} step

IO_i = disk channel time used during the i^{th} step.

The actual % reductions expected because of the field length reduction from 75K (octal) to 60K (octal) will be:

Data Base I -

% reduction in cost = 12.075

Data Base II -

% reduction in cost = 12.183

This research effort has successfully achieved the following objectives:

(a) The severe input data restrictions which caused enormous problems to create the data base have been removed. The user of the new MARS VI system does not have to concern himself with the peculiar input required by the old MARS VI system, but merely describes the entries of his file in the new syntax of the LOADER module. The new syntax is easy to understand. Consequently the creation of a data base is simplified.

(b) The introduction of the QUALIFY command in the RETRIEVAL module of the new MARS VI system has given the user a powerful tool to conduct complex analyses of his retrieved data, a feature which was not available in the old MARS VI system.

This feature is significant, as only a few of the current generalized data management systems offer this capability [1].

An area which could not be considered in this research was the quantitative analysis of retrieval and update efficiency.

The reason for this was the unavailability of interactive facilities on the CDC 6400 system. This made it impossible to determine on-line time response which could have served as a useful measure of retrieval and update efficiency.

In order to successfully achieve the research objectives, major changes were necessary to the CONTROL module and the LOADER module, while minor modifications were required in the RETRIEVAL module.

In making these changes it was strongly felt that even with the modularity architecture employed in the MARS VI system, it is a difficult

task to incorporate new features. However, as has been demonstrated by this research effort, the original system can be enhanced to provide greater flexibility and ease of usage.

6.2 RECOMMENDATIONS FOR FURTHER RESEARCH -

During the past decade there has been a significant amount of research devoted to the design and construction of natural language question-answering systems. Simmons' article [26] provides an excellent survey of work in that field. Paralleling the development of natural language question-answering systems has been the development of generalized data management systems.

Unfortunately there has been distinct differences in the scope and orientation of the activities in question-answering and data base management systems. Researchers in the question-answering area have stressed the development techniques to interpret the meaning of questions posed in English. Their main concern has been on translating natural language questions into a form that can be answered by direct retrieval from a file, and small-scale experimental computer models have been built to test these techniques. In a striking contrast, work on data management systems has been aimed at early operational use and has emphasized techniques for efficiently storing and rapidly accessing large quantities of formatted data, while relying on a relatively simple artificial query language for data retrieval purposes.

An interesting system could evolve, if these two developing technologies could be suitable combined to achieve a capability of interrogating large formatted data bases in a subset of English with reasonable on-line response times.

Kellog, in his article [12] presents some evidence for the thesis that, for some kinds of structured data bases, an English subset may be usefully realizable within the not-too-distant future.

The approach which has been suggested is the formal specification of a query language, a similar specification of the grammar for an English subset, and the description of a semantic interpretation procedure that allows questions to be translated into queries that will fetch or compute relevant facts.

Hence, in order to design a system where users may converse with large computer data files in an interactive mode using subsets of ordinary English, one would require the following basic units in the system.

1. A question-processing sub-system to translate English questions into query-language form

2. A generalized data management system like the MARS VI system which would serve as the answering subsystem.

Systems which link together the convenience of phrasing questions in English with the data storage and retrieval system appear to be the developing trend in this area.

A meaningful continuation of this research would be the development of the system proposed above. However, in order to widen the class of translatable questions it is suggested that a significant attempt be made to extend the expressiveness of the query language.

Research directed towards this goal would certainly lead to improved man-machine communication and make it possible for a computer system not only to allow economical representation of and rapid access to facts stored in the system but <u>also</u> assist in the logical analysis of the facts stored within the system.

7. BIBLIOGRAPHY

1. A Survey of Generalized Data Base Management Systems. CODASYL Systems Committee, May 1969.

2. Biomedical Computer Programs - BMD - University of California, 1971.

3. Bleier, R. E. Treating Hierarchical Data Structures in the SDC Time-Shared Data Management System. Proceedings of the 22nd National Conference, 1967, p. 67, 41-49.

4. Bleier, R.E. and Vorhaus, A.H. File Organization in the SDC Time-Shared Data Management System, IFIP, F92 - F97, 1968.

5. CDC 6400 Users Guide. University of Washington Computer Center, 1971.

6. Dodd, George G. Elements of Data Management Systems. Computing Surveys, 117-133, June 1969.

7. Everett, Gerald D. Remote File Management System. University of Texas Computation Center, Austin, Reports TSD 0-6, 1969.

8. Feature Analysis of Generalized Data Base Management Systems. CODASYL Systems Committee. Communications of the ACM, 1971 14(5), 309-318.

9. Fortran Reference Manual. Control Data 6400/6500/6600 Computer System, 1969.

10. GIS Application Description Manual. IBM Document, H20-0574.

11. Hsiao, D. and Harary F. A Formal System for Information Retrieval from Files. Communications of the ACM, 1970, 13(2), 67-73.

12. Kellogg, C.H. On-Line Translation of Natural Language Questions into Artificial Language Queries. SDC Document, SP-2827, 1967.

13. Khanna, Karun. Information Storage and Retrieval Mini-system for Washington State Groundfish Data. Information Systems NW02, Norfish, University of Washington, August 1971.

14. Khanna, Karun. Report on MARS VI System. Information System, NI05, Norfish, University of Washington, August 1971.

15. Khanna, Karun. A Multi-Access Retrieval System for the CDC 6400. Information Systems, NI17, Norfish, University of Washington, April 1972.

16. Khanna K., Buss, J., Reeves, J. Upgrading of the 1953-1966 Washington Groundfish Data. Information System, NW10, Norfish, University of Washington, September 1971.

17. MARS VI - Multi-Access Retrieval System - For Full Inversion. Control Data 6400/6500/6600 Computer System. Reference Manual, 1971.

19. Naur, P. Revised Report on the Algorithmic Language ALGOL 60. Communications of the ACM, Jan. 1963, 6(1), 1-17.

20. Nie, N.H, Bent, D.H., and Hull, C.H. SPSS - Statistical Package for the Social Sciences, 1970.

21. Olle, T.W. MIS: Data Bases. Datamation, Nov. 1970, 47-50.

22. Olle, T.W. UL/1: A User Language for Document Applications. Proceedings of the ASIS Conference, Vol. 5, 1968, 151-153.

23. Raucher, V.L. and Schwimmer, H.S. The Basic Language Specifications for the Time-Shared Data Management System. SDC Document TM-3370, 1967.

24. Rangel, R.G. A Design Approach to User Customized Information Systems. FJCC, 1968, 171-177.

25. Report of the CODASYL Data Base Task Group. April 1971.

26. Schubert, R.F. Basic Concepts in Data Base Management Systems. Datamation, July 1972, 42-47.

27. Simmons, R.F. Answering English Questions by Computer: A Survey. Communications of the SCM, 1965, 8(1), 53-70.

28. Smith, L. and Golde, H. Programming Language Descriptions. Technical Report No. 69-3-5, Computer Science Groups, University of Washington, March 1969.

29. TANDEM Users Guide. University of Washington Computer Center, Seattle, Washington, July 1971.

Appendix A. User defined component description tables for Data Base II

CT1 TABLE

Start position of component	System assigned component #	Length of component	Pointer to the beginning of a data set	User assigned component #	Multi-level flag
1	1	6	0	1	1
7	2	4	0	4	0
11	3	2	0	6	0
13	4	2	0	1	0
15	5	3	1	8	1
18	6	1	0	9	0
27	7	2	1	13	1
31	8	3	0	15	0
35	9	2	0	17	0
43	10	3	1	20	1
46	11	7	0	21	0
22	12	0	0	30	2
22	13	3	0	11	0
53	14	7	0	22	0

CT2 TABLE

Level # of component	Type of component	System assigned component # of parent repeating group	Pointer to the data string table for comparison of multi-level entries	Pointer to the next multi-level entry	End of component table flag
0	4	0	1	5	0
0	4	0	0	0	0
0	4	0	0	0	0
0	4	0	0	0	0
0	4	0	2	7	0
0	4	0	0	0	0
0	4	0	3	10	0
0	4	0	0	0	0
0	4	0	0	0	0
0	4	0	4	12	0
0	4	0	0	0	0
1	0	12	0	0	0
1	4	12	0	0	0
1	4	12	0	0	1

Appendix B. Program to convert 19[illegible]3 groundfish data into the "inverter" string [illegible]

```
//////////////////////////////////////////////////////////////////////////////////////////
15.10.40. 03/02/72 G401SFY  /// START OF LIST /// EQ 20
                                                                                  PAGE 001
RUN  VERSION JUN 72 C    18:15 08/02/72

                   PROGRAM NORFSH(DIN,DOUT,OUTPUT,TAPE1=DIN,TAPE2=DOUT,INPUT,
                  1TAPE5=INPUT,TAPE6=OUTPUT)
 000033            INTEGER N(96),EED
           * 1   DATE          01-06     01-11      04-11      04-05,07-08,10-11
 000033            DATA N/ 1H ,1H1,1H), 2*1H ,1H/,2*1H ,1H/,2*1H ,
           * 04  GROUP         07---     12-15      15
                 4        1H ,1H4,1H),              1H ,
           * 06  DAY OUT       11-12     16-20      19-20
                 6        1H ,1H5,   1H),           2*1H ,
           * 07  DAYS FISHED   13-14     21-25      24-25
                 7        1H ,1H7,   1H),           2*1H ,
           * 05  BOAT          15-17     26-31      29-31
                 8        1H ,1H8,   1H),           3*1H ,
                 91H ,1H9,1H),1H ,
           * 11  SPECIES       22-24     32-38      36-38
                 1        1H ,1H1,1H1,1H),          3*1H ,
           * 13  WASH          27-28     39-44      43-44
                 3        1H ,1H1,1H3,1H),          2*1H ,
           * 15  HOURS FISHED 31-33      45-51      49-51
                 5        1H ,1H1,1H5,1H),          3*1H ,
           * 17  DRAGS         35-36     52-57      56-57
                 7        1H ,1H1,1H7,1H),          2*1H ,
           * 20  AVE DEPTH     43-45     58-64      62-64
                 1        1H ,1H2,1H0,1H),          3*1H ,
                 11H ,1H2,1H1,1H),7*1H ,
           * 22  SPECIES CATCH 52-59     65-75      69-75
                 2        1H ,1H2,1H2,1H),          7*1H ,
                 *1H ,2*1H),1H5,1HN,1HD/
 000033            DATA EED/6H ))END/
 000033            READ(5,10) IA
 000011         10 FORMAT(I5)
 000011            DO 50 IEOF=1,IA
           *                               DATE + GROUP
 000013            READ(1,20) N(10),N(11),N(4),N(5),N(7),N(8),N(15),
           *                               DAYS OUT + DAYS FISHED
                  1N(19),N(20),N(24),N(25),
           *                               BOAT + SPECIES
                  2N(29),N(30),N(31),N(35),
                  3N(40),N(41),N(42),
           *                               WASH + HOURS FISHED
                  4N(47),N(48),N(53),N(54),N(55),
           *                               DRAGS + AVE DEPTH
                  5N(60),N(61),N(66),N(67),N(68),
                  6N(73),N(74),N(75),N(76),N(77),N(78),N(79),
           *                               SPECIES + CATCH
                  7N(84),N(85),N(86),N(87),N(88),N(89),N(90)
 000142     20     FORMAT(7A1,3X,8A1,3X,3A1,2X,2A1,2X,3A1,1X,2A1,6X,3A1,14A1)
 000142            IF(EOF,1) 100,30
 000145     30     WRITE(2,21)N
 000153     21     FORMAT(96A1)
 000153          50 CONTINUE
 000156         100 WRITE(2,101) EED
 000154         101 FORMAT(A10)
 000154            END FILE2
 000156            END
```

1)04/13/53	4)1	5)109	7)03	8)120	9)5	11)254	13)07	15)045	17)15	20)175	21)	46100	22)	15000))END
1)04/13/53	4)1	5)109	7)03	8)120	9)5	11)235	13)07	15)045	17)15	20)175	21)	46100	22)	1100))END
1)04/13/53	4)1	5)107	7)03	8)120	9)5	11)237	13)07	15)045	17)15	20)175	21)	46100	22)	30000))END
1)04/13/53	4)1	5)109	7)05	8)141	9)4	11)241	13)02	15)022	17)11	20)070	21)	13600	22)	11500))END
1)04/13/53	4)1	5)109	7)05	8)141	9)4	11)207	13)02	15)022	17)11	20)070	21)	13600	22)	1000))END
1)04/13/53	4)1	5)109	7)05	8)141	9)4	11)251	13)02	15)022	17)11	20)070	21)	13600	22)	1100))END
1)04/13/53	4)1	5)109	7)05	8)141	9)4	11)207	13)03	15)020	17)10	20)075	21)	43500	22)	4500))END
1)04/13/53	4)1	5)109	7)05	8)141	9)4	11)241	13)03	15)020	17)10	20)075	21)	43500	22)	14000))END
1)04/13/53	4)1	5)109	7)05	8)141	9)4	11)231	13)03	15)020	17)10	20)075	21)	43500	22)	25000))END
1)04/14/53	4)1	5)107	7)05	8)999	9)0	11)207	13)07	15)034	17)17	20)170	21)	56500	22)	45000))END
1)04/14/53	4)1	5)107	7)05	8)999	9)0	11)205	13)07	15)034	17)17	20)170	21)	56500	22)	2500))END
1)04/14/53	4)1	5)107	7)05	8)999	9)0	11)234	13)07	15)034	17)17	20)170	21)	56500	22)	9000))END
1)04/15/53	4)1	5)109	7)06	8) 24	9)3	11)212	13)12	15)010	17)05	20)060	21)	3500	22)	1000))END
1)04/15/53	4)1	5)109	7)06	8) 24	9)3	11)241	13)12	15)010	17)05	20)060	21)	3500	22)	2500))END
1)04/15/53	4)1	5)109	7)06	8) 24	9)3	11)251	13)15	15)020	17)10	20)062	21)	7500	22)	6000))END
1)04/15/53	4)1	5)109	7)06	8) 24	9)3	11)241	13)15	15)020	17)10	20)062	21)	7500	22)	1000))END
1)04/15/53	4)1	5)109	7)06	8) 24	9)3	11)202	13)15	15)020	17)10	20)062	21)	7500	22)	500))END
1)04/15/53	4)1	5)109	7)06	8) 24	9)3	11)251	13)17	15)018	17)09	20)032	21)	8300	22)	4000))END
1)04/15/53	4)1	5)109	7)06	8) 24	9)3	11)236	13)17	15)018	17)09	20)032	21)	8300	22)	1500))END
1)04/15/53	4)1	5)109	7)06	8) 24	9)3	11)231	13)17	15)018	17)09	20)032	21)	8300	22)	1800))END
1)04/15/53	4)1	5)109	7)06	8) 24	9)3	11)207	13)17	15)018	17)09	20)032	21)	8300	22)	1000))END
1)04/15/53	4)1	5)109	7)06	8) 24	9)3	11)254	13)17	15)002	17)01	20)120	21)	3000	22)	3000))END
1)04/16/53	4)1	5)113	7)10	8) 6	9)4	11)253	13)04	15)058	17)29	20)081	21)	35935	22)	11230))END
1)04/16/53	4)1	5)113	7)10	8) 6	9)4	11)207	13)04	15)058	17)29	20)081	21)	35935	22)	16000))END
1)04/16/53	4)1	5)113	7)10	8) 6	9)4	11)221	13)04	15)058	17)29	20)081	21)	35935	22)	335))END
1)04/16/53	4)1	5)113	7)10	8) 6	9)4	11)232	13)04	15)058	17)29	20)081	21)	35935	22)	4000))END
1)04/16/53	4)1	5)113	7)10	8) 6	9)4	11)241	13)04	15)058	17)29	20)081	21)	35935	22)	3600))END
1)04/16/53	4)1	5)113	7)10	8) 6	9)4	11)231	13)04	15)058	17)29	20)081	21)	35935	22)	850))END
1)04/16/53	4)1	5)113	7)10	8) 6	9)4	11)252	13)05	15)006	17)03	20)073	21)	1000	22)	1000))END
1)04/16/53	4)1	5)113	7)10	8) 6	9)4	11)252	13)07	15)010	17)05	20)075	21)	1000	22)	1000))END
1)04/17/53	4)1	5)109	7)06	8)132	9)3	11)207	13)07	15)043	17)17	20)172	21)	34000	22)	21000))END
1)04/17/53	4)1	5)109	7)06	8)132	9)3	11)254	13)07	15)043	17)17	20)172	21)	34000	22)	13000))END
1)04/17/53	4)1	5)109	7)05	8)132	9)3	11)207	13)08	15)006	17)03	20)050	21)	1520	22)	200))END
1)04/17/53	4)1	5)109	7)06	8)132	9)3	11)231	13)08	15)006	17)03	20)050	21)	1520	22)	320))END
1)04/17/53	4)1	5)109	7)06	8)132	9)3	11)241	13)08	15)006	17)03	20)050	21)	1520	22)	1000))END
1)04/17/53	4)1	5)109	7)06	8)132	9)3	11)275	13)12	15)006	17)03	20)060	21)	2700	22)	1200))END
1)04/17/53	4)1	5)109	7)06	8)132	9)3	11)241	13)12	15)006	17)03	20)060	21)	2700	22)	1000))END
1)04/17/53	4)1	5)109	7)06	8)132	9)3	11)251	13)12	15)006	17)03	20)060	21)	2700	22)	500))END
1)04/20/53	4)1	5)112	7)06	8) 2	9)3	11)231	13)03	15)030	17)15	20)095	21)	29000	22)	1000))END
1)04/20/53	4)1	5)112	7)05	8) 2	9)3	11)251	13)03	15)030	17)15	20)095	21)	29000	22)	5000))END
1)04/20/53	4)1	5)112	7)06	8) 2	9)3	11)207	13)03	15)030	17)15	20)095	21)	29000	22)	4500))END
1)04/20/53	4)1	5)112	7)05	8) 2	9)3	11)206	13)03	15)030	17)15	20)095	21)	29000	22)	3500))END
1)04/20/53	4)1	5)112	7)05	8) 2	9)3	11)241	13)03	15)030	17)15	20)095	21)	29000	22)	15000))END
1)04/20/53	4)1	5)112	7)06	8) 2	9)3	11)237	13)05	15)030	17)15	20)074	21)	15600	22)	11500))END
1)04/20/53	4)1	5)112	7)05	8) 2	9)3	11)241	13)05	15)030	17)15	20)074	21)	15600	22)	100))END
1)04/20/53	4)1	5)112	7)06	8) 2	9)3	11)251	13)05	15)030	17)15	20)074	21)	15600	22)	4900))END
1)04/20/53	4)1	5)109	7)06	8) 56	9)3	11)000	13)12	15)002	17)01	20)050	21)	0	22)	0))END
1)04/20/53	4)1	5)109	7)06	8) 56	9)3	11)207	13)12	15)020	17)10	20)063	21)	15500	22)	1000))END
1)04/20/53	4)1	5)109	7)06	8) 56	9)3	11)231	13)12	15)020	17)10	20)063	21)	15500	22)	900))END
1)04/20/53	4)1	5)109	7)06	8) 56	9)3	11)241	13)12	15)020	17)10	20)063	21)	15500	22)	12000))END
1)04/20/53	4)1	5)109	7)05	8) 56	9)3	11)205	13)12	15)020	17)10	20)063	21)	15500	22)	2000))END
1)04/20/53	4)1	5)109	7)06	8) 56	9)3	11)251	13)15	15)010	17)05	20)080	21)	8000	22)	8000))END
1)04/20/53	4)1	5)109	7)06	8) 56	9)3	11)251	13)16	15)024	17)12	20)043	21)	9500	22)	5500))END
1)04/20/53	4)1	5)109	7)05	8) 56	9)3	11)231	13)16	15)024	17)12	20)043	21)	9500	22)	1500))END
1)04/20/53	4)1	5)109	7)06	8) 56	9)3	11)205	13)16	15)024	17)12	20)043	21)	9500	22)	2000))END
1)04/20/53	4)1	5)109	7)06	8) 56	9)3	11)207	13)16	15)024	17)12	20)043	21)	9500	22)	1000))END
1)04/20/53	4)1	5)109	7)03	8)154	9)3	11)251	13)04	15)005	17)04	20)085	21)	15000	22)	10000))END
1)04/20/53	4)1	5)109	7)03	8)154	9)3	11)207	13)04	15)008	17)04	20)085	21)	15000	22)	5000))END
1)04/20/53	4)1	5)109	7)03	8)154	9)3	11)207	13)05	15)006	17)03	20)071	21)	5500	22)	3000))END
1)04/20/53	4)1	5)109	7)03	8)154	9)3	11)241	13)05	15)006	17)03	20)071	21)	5500	22)	2000))END
1)04/20/53	4)1	5)108	7)03	8)154	9)3	11)231	13)05	15)006	17)03	20)071	21)	5500	22)	500))END
1)04/20/53	4)1	5)108	7)03	8)154	9)3	11)205	13)07	15)008	17)02	20)172	21)	500	22)	500))END
1)04/21/53	4)1	5)107	7)04	8)148	9)3	11)207	13)07	15)039	17)17	20)179	21)	42000	22)	40000))END

```
//////////////////////////////////////////////////////////////////////////////
22.21.43. 31/02/72 G401SKT /// START OF LIST /// EO ZG

VERSION 3.5F 12/31/70
MARS VI IS READY -
DEFINE:
NEW DATA BASE DESCRIPTION 1983ZLCV:
 1)DATE OF INTERVIEW(DATE)
 4)PORT OF INTERVIEW(INTEGER NUMBER)
 6)TOTAL DAYS SPENT ON TRIP(INTEGER NUMBER)
 7)DAYS SPENT FISHING(INTEGER NUMBER)
 8)BOAT NUMBER(INTEGER NUMBER)
 9)BOAT CLASS(INTEGER NUMBER)
 14)STATISTICAL AREA(INTEGER NUMBER)
 15)TOTAL HOURS FISHED(INTEGER NUMBER)
 17)NUMBER OF DRAGS(INTEGER NUMBER)
 20)AVERAGE DEPTH OF FISHING(INTEGER NUMBER)
 21)TOTAL CATCH OF ALL SPECIES(INTEGER NUMBER)
111)SPECIES CODE(INTEGER NUMBER)
22)TOTAL CATCH OF MAJOR SPECIES(INTEGER NUMBER)
MAP:
LOCKED:
DATA FILE IS GOFFN:
ISSUE REPORT WHEN ALL CHECKPOINTS OCCUR:
SCAN:


     21.44.---
     30/02/72

         ALL LEGALITY IS CHECKED FOR THIS LOADER CALL
              NUMBER OF ACCEPTABLE ENTRIES =  2101
              NUMBER OF VALUES ACCEPTED =  27313
              NUMBER OF ENTRIES WITH REJECTED VALUES =     0
              NUMBER OF VALUES REJECTED =     0
              NUMBER OF VALUES EXCLUDED =     0
              NUMBER OF DATA SETS ACCEPTED =  2101
              NUMBER OF NON-VALUED DATA SETS ACCEPTED =     0

 21.33.41.    MERGE COMPLETED FOR ELEMENT C   1
TOTAL NUMBER OF UNIQUE VALUES FOR THIS ELEMENT =      145

 21.33.52.    MERGE COMPLETED FOR ELEMENT C   4
TOTAL NUMBER OF UNIQUE VALUES FOR THIS ELEMENT =        2

 21.34.06.    MERGE COMPLETED FOR ELEMENT C   6
TOTAL NUMBER OF UNIQUE VALUES FOR THIS ELEMENT =       24

 21.34.16.    MERGE COMPLETED FOR ELEMENT C   7
TOTAL NUMBER OF UNIQUE VALUES FOR THIS ELEMENT =       10

 21.34.27.    MERGE COMPLETED FOR ELEMENT C   8
TOTAL NUMBER OF UNIQUE VALUES FOR THIS ELEMENT =       72

 21.34.37.    MERGE COMPLETED FOR ELEMENT C   9
TOTAL NUMBER OF UNIQUE VALUES FOR THIS ELEMENT =        5

 21.34.47.    MERGE COMPLETED FOR ELEMENT C  13
TOTAL NUMBER OF UNIQUE VALUES FOR THIS ELEMENT =       15

 21.34.44.    MERGE COMPLETED FOR ELEMENT C  15
TOTAL NUMBER OF UNIQUE VALUES FOR THIS ELEMENT =       46

 21.34.56.    MERGE COMPLETED FOR ELEMENT C  17
TOTAL NUMBER OF UNIQUE VALUES FOR THIS ELEMENT =       29
```

```
  21.55.00.     MERGE COMPLETED FOR ELEMENT C  20
 TOTAL NUMBER OF UNIQUE VALUES FOR THIS ELEMENT =      122

  21.55.09.     MERGE COMPLETED FOR ELEMENT C  21
 TOTAL NUMBER OF UNIQUE VALUES FOR THIS ELEMENT =      410

  21.55.20.     MERGE COMPLETED FOR ELEMENT C  11
 TOTAL NUMBER OF UNIQUE VALUES FOR THIS ELEMENT =       24

  21.55.30.     MERGE COMPLETED FOR ELEMENT C  22
 TOTAL NUMBER OF UNIQUE VALUES FOR THIS ELEMENT =      246

      21.55.32.
          SELECTION TABLES ARE COMPLETE

      21.58.28.
          FINAL SORT IS COMPLETE

      21.59.17.
          LOADING COMPLETED
                NUMBER OF ACCEPTABLE ENTRIES =  2101
                NUMBER OF VALUES ACCEPTED =  27313
                NUMBER OF ENTRIES WITH REJECTED VALUES =     0
                NUMBER OF VALUES REJECTED =     0
                NUMBER OF VALUES EXCLUDED =     0
                NUMBER OF DATA SETS ACCEPTED =  2101
                NUMBER OF NON-VALUED DATA SETS ACCEPTED =     0
                NUMBER OF NULLS CREATED IN THIS JOB =     0
                TOTAL NUMBER OF UNIQUE VALUES IN DATA BASE =  1151
                 TOTAL SIZE OF CURRENT DATA BASE =   216 PARTITIONS  OR    552960 CHARACTERS

 GROUNDFISH 1953LEV    DEFINITION VERSION    1   DATA VERSION    1

      21.59.18.     08/02/72
 SAVE DATA BASE ON SFILE1
- SAVE/LOAD COMPLETED
 EXIT!
```

Appendix D. Creation of Data Base I using the new MARS VI system.

```
VERSION 3.3F 12/01/70
MARS VI IS READY -
COMMAND FILE IS TTY;
DEFINE;
NEW DATA BASE GROUNDFISH 932LEV;
1)DATE OF INTERVIEW(INTEGER NUMBER)
2)PORT OF INTERVIEW(INTEGER NUMBER)
6)TOTAL DAYS SPENT ON TRIP(INTEGER NUMBER)
7)DAYS SPENT FISHING(INTEGER NUMBER)
8)BOAT NUMBER(INTEGER NUMBER)
9)BOAT CLASS(INTEGER NUMBER)
13)WASHINGTON STATISTICAL AREA(INTEGER NUMBER)
14)TOTAL HOURS FISHED(INTEGER NUMBER)
17)NUMBER OF GEARS(INTEGER NUMBER)
20)AVERAGE DEPTH OF FISHING(INTEGER NUMBER)
21)TOTAL CATCH OF ALL SPECIES(INTEGER NUMBER)
11)SPECIES CODE(INTEGER NUMBER)
22)TOTAL CATCH OF MAJOR SPECIES(INTEGER NUMBER)
MAP;
LOADEM:
DATA FILE IS GFISH;
OBJECT FILE INPUT;
1)FROM COL 1 TO COL 6
4)IN COL 7
6)COL 11-COL 12
7)COL 13 THRU COL 14
9)15-17
9)IN COL 18
13)FROM COL 27 THRU COL 28
15)31 THRU 33
17)COL 35 THRU COL 36
27)43-45
21)COL 46-COL 52
11)COL 22-COL 24
22)FROM COL 53 TO COL 59;
ISSUE REPORT WHEN ALL LEGALITY CHECKED;
SCAN;

     15.27.31.
     08/02/72

         ALL LEGALITY IS CHECKED FOR THIS LOADER CALL
              NUMBER OF ACCEPTABLE ENTRIES =  2101
              NUMBER OF VALUES ACCEPTED =   27313
              NUMBER OF ENTRIES WITH REJECTED VALUES =     0
              NUMBER OF VALUES REJECTED =        0
              NUMBER OF VALUES EXCLUDED =        0
              NUMBER OF DATA SETS ACCEPTED =   2101
              NUMBER OF NON-VALUED DATA SETS ACCEPTED =      0

GROUNDFISH 932LEV      DEFINITION VERSION     1   DATA VERSION     1

     15.27.06.      08/02/72
SAVE DATA BASE ON DD1993;
SAVE/LOAD COMPLETED
EXIT;
```

Data Base 11.

```
//////////////////////////////////////////////////////////////////////////////////////
     17.50.30. 08/02/72 G4010EN  /// START OF LIST /// EQ 21

     RUN  VERSION JUN 72 C    17148 08/02/72                                                        PAGE 001

                       PROGRAM NORFSH(DIN,DOUT,OUTPUT,TAPE1=DIN,TAPE2=DOUT,INPUT,
                      1TAPES=INPUT,TAPE6=OUTPUT)
     000003            INTEGER N(101),EEO
     000003            INTEGER OYEAR,OBOAT,OAREA,ODEPTH
                 * 1  DATE           01-06    01-11     04-11     04-05,07-08,10-11
     000003            DATA N/ 1H ,1H1,1H), 2*1H ,1H/,2*1H ,1H/,2*1H ,
                 * 04 GROUP          07---    12-15     15
                      4        1H ,1H4,1H),             1H ,
                 * 06 DAY OUT        11-12    16-20     19-20      -
                      5        1H ,1H6,    1H),         2*1H ,
                 * 07 DAYS FISHED    13-14    21-25     24-25      -
                      7        1H ,1H7,    1H),         2*1H ,
                 * 08 BOAT           15-17    26-31     29-31      -
                      8        1H ,1H8,    1H),         3*1H ,
                      91H ,1H9,1H),1H ,
                 * 11 SPECIES        22-24    32-38     36-38      -
                      1        1H ,1H1,1H1,1H),         3*1H ,
                 * 13 WASH           27-28    39-44     43-44      -
                      3        1H ,1H1,1H3,1H),         2*1H ,
                 * 15  HOURS FISHED 31-33     45-51     49-51      -
                      5        1H ,1H1,1H5,1H),         3*1H ,
                 * 17 DRAGS          35-36    52-57     56-57      -
                      7        1H ,1H1,1H7,1H),         2*1H ,
                 * 20 AVE DEPTH      43-45    58-64     62-64      -
                      1        1H ,1H2,1H0,1H),         3*1H ,
                      11H ,1H2,1H1,1H),7*1H ,
                 * 22 SPECIES CATCH 52-59     65-75     69-75      -
                      2        1H ,1H2,1H2,1H),         7*1H ,
                      *1H ,2*1H),1HE,1HN,1HD,
                      *1H ,1H3,1H0,1H),1H /
     000003            DATA EEO/6H ))END/
     000003            READ(5,10) IA
     000011         10 FORMAT(I5)
     000011            DO 50 IEOF=1,IA
                 *                               DATE + GROUP
     000013            READ(1,20) N(10),N(11),N(4),N(5),N(7),N(8),N(15),
                 *                               DAYS OUT + DAYS FISHED
                      1N(19),N(20),N(24),N(25),
                 *                               BOAT + SPECIES
                      2N(29),N(30),N(31),N(35),
                      3N(40),N(41),N(42),
                 *                               WASH + HOURS FISHED
                      4N(47),N(48),N(53),N(54),N(55),
                 *                               DRAGS + AVE DEPTH
                      5N(60),N(61),N(66),N(67),N(68),
                      6N(73),N(74),N(75),N(76),N(77),N(78),N(79),
                 *                               SPECIES + CATCH
                      7N(84),N(85),N(86),N(87),N(88),N(89),N(90)
     000142        20  FORMAT(7A1,3X,8A1,3X,3A1,2X,2A1,2X,3A1,1X,2A1,6X,3A1,14A1)
     000142            IF(EOF,1) 100,30
     000145        30  IF(IEOF .EQ. 1)GO TO 250
     000147            GO TO 251
     000150       250  CALL SUB1(N(4),N(5),N(7),N(8),OYEAR)
     000154            CALL SUB2(N(29),N(30),N(31),OBOAT)
     000157            CALL SUB3(N(47),N(48),OAREA)
```

```
000162          CALL SUB2(N(66),N(67),N(68),ODEPTH)
000165          GO TO 354
000166     251  CALL SUB1(N(4),N(5),N(7),N(8),NYEAR)
000172          CALL SUB2(N(29),N(30),N(31),NBOAT)
000175          CALL SUB3(N(47),N(48),NAREA)
000200          CALL SUB2(N(66),N(67),N(68),NDEPTH)
000203     252  IF(NYEAR .EQ. OYEAR)GO TO 350
000205          GO TO 400
000206     350  IF(NBOAT .EQ. OBOAT)GO TO 351
000210          GO TO 400
000211     351  IF(NAREA .EQ. OAREA)GO TO 352
000213          GO TO 400
000214     352  IF(NDEPTH .EQ. ODEPTH)GO TO 353
000216     400  OYEAR=NYEAR
000220          OBOAT=NBOAT
000221          OAREA=NAREA
000223          ODEPTH=NDEPTH
000224          WRITE(2,23) (N(IK),IK=91,96)
000232      23  FORMAT(6A1)
000232     354  WRITE(2,21)((N(IK),IK=1,35),(N(IK),IK=43,79))
000242      21  FORMAT(72A1)
000242     353  WRITE(2,22)((N(IK),IK=97,101),(N(IK),IK=36,42),(N(IK),IK=80,90))
000254      22  FORMAT(23A1)
000254       50 CONTINUE
000257      100 WRITE(2,101) EEO
000265      101 FORMAT(A10)
000265          END FILE2
000267          END
```

```
                                        IDENT   SUBR
                                        ENTRY   SUB1,SUB2,SUB3
      0                         SUB1    BSSZ    1
      1  56110                          SA1     B1
             56220                      SA2     B2
                    56330               SA3     B3
                           56440        SA4     B4
      2  43006                          MX0     6
             11110                      BX1     X1*X0
                    11220               BX2     X2*X0
                           11330        BX3     X3*X1
      3  11440                          BX4     X4*X3
             20266                      LX2     54
                    12612               BX6     X1+X2
                           20360        LX3     48
      4  12563                          BX6     X6+X3
             20452                      LX4     42
                    12664               BX6     X6+X4
                           56650        SA6     B5
      5  0200000000 +                   JP      SUB1
      6                         SUB2    BSSZ    1
      7  56110                          SA1     B1
             56220                      SA2     B2
                    56330               SA3     B3
                           43006        MX0     6
     10  11110                          BX1     X1*X0
             11220                      BX2     X2*X0
                    11330               BX3     X3*X0
                           20266        LX2     54
     11  12612                          BX6     X1+X2
             20360                      LX3     48
                    12663               BX6     X6+X3
                           56640        SA6     B4
     12  0200000006 +                   JP      SUB2
     13                         SUB3    BSSZ    1
     14  56110                          SA1     B1
             56220                      SA2     B2
                    43006               MX0     6
                           11110        BX1     X1*X0
     15  11220                          BX2     X2*X0
             20266                      LX2     54
                    12612               BX6     X1+X2
                           56630        SA6     B3
     16  0200000013 +                   JP      SUB3
     17                                 END

          34100  STORAGE USED                45 STATEMENTS      3 SYMBOLS
                 6400 ASSEMBLY            0.275 SECONDS        9 REFERENCES
```

```
1)04/13/53 4)1 6)09 7)03 8)120 9)5 13)07 15)045 17)15 20)175 21)  46100
30)  11)254 22)  15000
30)  11)205 22)   1100
30)  11)207 22)  30000
))END
1)04/13/53 4)1 6)09 7)05 8)141 9)4 13)02 15)022 17)11 20)070 21)  13600
30)  11)241 22)  11500
30)  11)207 22)   1000
30)  11)251 22)   1100
))END
1)04/13/53 4)1 6)09 7)05 8)141 9)4 13)03 15)020 17)10 20)075 21)  43500
30)  11)209 22)   4500
30)  11)241 22)  14000
30)  11)251 22)  25000
))END
1)04/14/53 4)1 6)07 7)05 8)999 9)0 13)07 15)034 17)17 20)170 21)  56500
30)  11)207 22)  45000
30)  11)205 22)   2500
30)  11)254 22)   9000
))END
1)04/15/53 4)1 6)09 7)06 8) 24 9)3 13)12 15)010 17)05 20)060 21)   3500
30)  11)202 22)   1000
30)  11)241 22)   2500
))END
1)04/15/53 4)1 6)09 7)06 8) 24 9)3 13)15 15)020 17)10 20)062 21)   7500
30)  11)251 22)   6000
30)  11)241 22)   1000
30)  11)202 22)    500
))END
1)04/15/53 4)1 6)09 7)06 8) 24 9)3 13)17 15)018 17)09 20)032 21)   6300
30)  11)251 22)   4000
30)  11)206 22)   1500
30)  11)231 22)   1500
30)  11)207 22)   1000
))END
1)04/15/53 4)1 6)09 7)06 8) 24 9)3 13)17 15)002 17)01 20)120 21)   3000
30)  11)254 22)   3000
))END
1)04/16/53 4)1 6)13 7)10 8)  6 9)4 13)04 15)056 17)29 20)081 21)  35935
30)  11)253 22)  11200
30)  11)207 22)  16000
30)  11)221 22)    335
30)  11)252 22)   4000
30)  11)241 22)   3600
30)  11)231 22)    800
))END
1)04/16/53 4)1 6)13 7)10 8)  6 9)4 13)05 15)006 17)03 20)073 21)   1000
30)  11)252 22)   1000
))END
1)04/16/53 4)1 6)13 7)10 8)  6 9)4 13)07 15)010 17)05 20)076 21)   1000
30)  11)252 22)   1000
))END
1)04/17/53 4)1 6)08 7)06 8)132 9)3 13)07 15)043 17)17 20)172 21)  34000
30)  11)207 22)  21000
30)  11)254 22)  13000
))END
1)04/17/53 4)1 6)08 7)06 8)132 9)3 13)08 15)006 17)03 20)050 21)   1520
30)  11)207 22)    200
30)  11)231 22)    320
30)  11)241 22)   1000
))END
1)04/17/53 4)1 6)08 7)06 8)132 9)3 13)12 15)006 17)03 20)060 21)   2700
30)  11)206 22)   1200
```

```
 VERSION 3.3F 12/01/70
 MAPS VI IS READY -
* DEFINE:
 NEW DATA BASE GROUNDFISH 1953N:
  1)DATE OF INTERVIEW(DATE)
  4)PORT OF INTERVIEW(INTEGER NUMBER)
  6)TOTAL DAYS SPENT ON TRIP(INTEGER NUMBER)
  7)DAYS SPENT FISHING(INTEGER NUMBER)
  8)BOAT NUMBER(INTEGER NUMBER)
  9)BOAT CLASS(INTEGER NUMBER)
  13)WASHINGTON STATISTICAL AREA(INTEGER NUMBER)
  15)TOTAL HOURS FISHED(INTEGER NUMBER)
  17)NUMBER OF DRAGS(INTEGER NUMBER)
  23)AVERAGE DEPTH OF FISHING(INTEGER NUMBER)
  21)TOTAL CATCH OF ALL SPECIES(INTEGER NUMBER)
  30)CATCH STATISTICS(REPEATING GROUP)
  11)SPECIES CODE(INTEGER NUMBER IN 30)
  22)TOTAL CATCH OF MAJOR SPECIES(INTEGER NUMBER IN 30)
 MAP:
 LOADER:
- DATA FILE IS GOS3N:
 ISSUE REPORT WHEN ALL CHECKPOINTS OCCUR:
 SCAN:
 ONLY ONE ENTRY TERMINATOR BEFORE EOF        REJECTIONS =        0

      21.24.06.
     08/02/72

          ALL LEGALITY IS CHECKED FOR THIS LOADER CALL
                NUMBER OF ACCEPTABLE ENTRIES =   649
                NUMBER OF VALUES ACCEPTED =   11330
                NUMBER OF ENTRIES WITH REJECTED VALUES =      0
                NUMBER OF VALUES REJECTED =        0
                NUMBER OF VALUES EXCLUDED =        0
                NUMBER OF DATA SETS ACCEPTED =   2749
                NUMBER OF NON-VALUED DATA SETS ACCEPTED =      0

  21.28.00.    MERGE COMPLETED FOR ELEMENT C   1
 TOTAL NUMBER OF UNIQUE VALUES FOR THIS ELEMENT =     145

  21.28.02.    MERGE COMPLETED FOR ELEMENT C   4
 TOTAL NUMBER OF UNIQUE VALUES FOR THIS ELEMENT =       2

  21.28.03.    MERGE COMPLETED FOR ELEMENT C   6
 TOTAL NUMBER OF UNIQUE VALUES FOR THIS ELEMENT =      24

  21.28.05.    MERGE COMPLETED FOR ELEMENT C   7
 TOTAL NUMBER OF UNIQUE VALUES FOR THIS ELEMENT =      10

  21.28.07.    MERGE COMPLETED FOR ELEMENT C   8
 TOTAL NUMBER OF UNIQUE VALUES FOR THIS ELEMENT =      72

  21.28.09.    MERGE COMPLETED FOR ELEMENT C   9
 TOTAL NUMBER OF UNIQUE VALUES FOR THIS ELEMENT =       5

  21.28.11.    MERGE COMPLETED FOR ELEMENT C  13
 TOTAL NUMBER OF UNIQUE VALUES FOR THIS ELEMENT =      16

  21.28.13.    MERGE COMPLETED FOR ELEMENT C  15
 TOTAL NUMBER OF UNIQUE VALUES FOR THIS ELEMENT =      44
```

```
   21.28.14.    MERGE COMPLETED FOR ELEMENT C  17
--TOTAL NUMBER OF UNIQUE VALUES FOR THIS ELEMENT =      29

   21.28.15.    MERGE COMPLETED FOR ELEMENT C  20
--TOTAL NUMBER OF UNIQUE VALUES FOR THIS ELEMENT =     122

   21.28.19.    MERGE COMPLETED FOR ELEMENT C  21
- TOTAL NUMBER OF UNIQUE VALUES FOR THIS ELEMENT =     410

   21.28.23.    MERGE COMPLETED FOR ELEMENT C  11
- TOTAL NUMBER OF UNIQUE VALUES FOR THIS ELEMENT =      24

   21.28.28.    MERGE COMPLETED FOR ELEMENT C  22
- TOTAL NUMBER OF UNIQUE VALUES FOR THIS ELEMENT =     246


--   21.29.29.

         SELECTION TABLES ARE COMPLETE


      21.29.03.

         FINAL SORT IS COMPLETE


      21.29.23.

--   --LOADING COMPLETED
               NUMBER OF ACCEPTABLE ENTRIES =    648
               NUMBER OF VALUES ACCEPTED =    11730
               NUMBER OF ENTRIES WITH REJECTED VALUES =      0
               NUMBER OF VALUES REJECTED =      0
               NUMBER OF VALUES EXCLUDED =      0
               NUMBER OF DATA SETS ACCEPTED =    2749
               NUMBER OF NON-VALUED DATA SETS ACCEPTED =      0
               NUMBER OF NULLS CREATED IN THIS JOB =      0
               TOTAL NUMBER OF UNIQUE VALUES IN DATA BASE =    1151
                TOTAL SIZE OF CURRENT DATA BASE =    124 PARTITIONS  OR     317440 CHARACTERS

- GROUNDFISH 1953N       DEFINITION VERSION     1   DATA VERSION     1

      21.29.28.      04/02/72
--SAVE DATA BASE ON SFILE:
 SAVE/LOAD COMPLETED
 EXIT:
```

Appendix G. Creation of Data Base II using the new MARS VI system.

```
VERSION 3.3F 12/01/70
MARS VI IS READY -
COMMAND FILE IS TTY;
DEFINE;
NEW DATA BASE GROUNDFISH $3MLEV;
1)DATE OF INTERVIEW(INTEGER NUMBER)
4)PORT OF INTERVIEW(INTEGER NUMBER)
6)TOTAL DAYS SPENT ON TRIP(INTEGER NUMBER)
7)DAYS SPENT FISHING(INTEGER NUMBER)
8)BOAT NUMBER(INTEGER NUMBER)
9)BOAT CLASS(INTEGER NUMBER)
13)WASHINGTON STATISTICAL AREA(INTEGER NUMBER)
15)TOTAL HOURS FISHED(INTEGER NUMBER)
17)NUMBER OF DRAGS(INTEGER NUMBER)
20)AVERAGE DEPTH OF FISHING(INTEGER NUMBER)
21)TOTAL CATCH OF ALL SPECIES(INTEGER NUMBER)
30)CATCH INFORMATION(REPEATING GROUP)
11)SPECIES CODE(INTEGER NUMBER IN 30)
22)TOTAL CATCH OF MAJOR SPECIES(INTEGER NUMBER IN 30)
MAP;
LOADER;
DATA FILE IS GFISH;
DIRECT FILE INPUT;
KEY COMMON FIELDS FOR THIS LOGICAL ENTRY ARE C1,C6,C13,C20;
1)FROM COL 1 TO COL 6
4)IN COL 7
6)COL 11-COL 12
7)COL 13 THRU COL 14
8)15-17
9)IN COL 18
13)FROM COL 27 THRU COL 28
15)31 THRU 33
17)COL 35 THRU COL 36
20)43-45
21)COL 46-COL 52
11)COL 22-COL 24
22)FROM COL 53 TO COL 59;
ISSUE REPORT WHEN ALL LEGALITY CHECKED;
SCAN;


     19.23.14.
     08/02/72

          ALL LEGALITY IS CHECKED FOR THIS LOADER CALL
               NUMBER OF ACCEPTABLE ENTRIES =    648
               NUMBER OF VALUES ACCEPTED =    11330
               NUMBER OF ENTRIES WITH REJECTED VALUES =      0
               NUMBER OF VALUES REJECTED =      0
               NUMBER OF VALUES EXCLUDED =      0
               NUMBER OF DATA SETS ACCEPTED =   2749
               NUMBER OF NON-VALUED DATA SETS ACCEPTED =      0

GROUNDFISH $3MLEV      DEFINITION VERSION     1  DATA VERSION     1

     19.26.35.      08/02/72
SAVE DATA BASE ON DB1953M;
SAVE/LOAD COMPLETED
EXIT;
```

Appendix H. Retrieval, update and report generation from [illegible]

```
////////////////////////////////////////////////////////////////////////////////
 08.14.36. 08/03/72 G-01SRJ  /// START OF LIST /// EQ 21

 VERSION 3.3F 12/01/70
 MARS VI IS READY -
 LOAD DATA BASE FROM DB1953M;
 SAVE/LOAD COMPLETED
 RETRIEVAL:
 PRINT C1,C4,C6,C9,C13,C11,C22 WHERE C1 LT 530417;
  1)530413
  4)1
  8)141
  9)4
  13)03
  11)209
  11)241
  11)251
  22)   4500
  22)  14000
  22)  25000
  1)530413
  4)1
  8)120
  9)5
  13)07
  11)254
  11)205
  11)207
  22)  15000
  22)   1100
  22)  30000
  1)530413
  4)1
  8)141
  9)4
  13)02
  11)241
  11)207
  11)251
  22)  11500
  22)   1000
  22)   1100
  1)530414
  4)1
  8)999
  9)0
  13)07
  11)207
  11)205
  11)254
  22)  45000
  22)   2500
  22)   9000
  1)530415
  4)1
  8) 24
  9)3
  13)17
  11)251
  11)206
  11)231
  11)207
  22)   4000
  22)   1500
  22)   1800
  22)   4800
```

```
1)530415
4)1
8) 24
9)3
13)17
11)254
22)   3000
1)530415
4)1
8) 24
9)3
13)12
11)202
11)241
22)   1000
22)   2500
1)530415
4)1
8) 24
9)3
13)15
11)251
11)241
11)202
22)   6000
22)   1000
22)    500
1)530416
4)1
8)  6
9)4
13)07
11)252
22)   1000
1)530416
4)1
8)  6
9)4
13)04
11)253
11)207
11)221
11)252
11)241
11)231
22)  11200
22)  16000
22)    335
22)   4000
22)   3600
22)    800
1)530415
4)1
8)  6
9)4
13)05
11)252
22)   1000
--- END OF RETRIEVAL ---
PRINT BOAT NUMBER,BOAT CLASS WHERE BOAT CLASS GE 51
8) 52
9)5
8) 52
9)5
81120
```

```
    8)120
    9)5
    8)120
    9)5
    8)120
    9)5
    8)120
    9)5
    8)110
    9)5
    8)  8
    9)5
    8) 22
    9)5
    8) 22
    9)5
    8) 22
    9)5
    8)120
    9)5
    8) 52
    9)5
    8)120
    9)5
    8)166
    9)5
--- END OF RETRIEVAL ---
PRINT SUM C22,C11 WHERE C13 EQ 02 AND BOAT CLASS EQ 3
AND SPECIES CODE EQ 251;
SUM 22)  2.750005000000E+04
--- END OF RETRIEVAL ---
 11)251
 11)251
 11)251
 11)251;
 11)251
--- END OF RETRIEVAL ---
PRINT ENTRY WHERE C22 GT 55000 AND C20 EQ 175 AND C8 EQ 120;
 DIAGNOSTIC- AND IS UNSATISFIED -NONFATAL
 DIAGNOSTIC- AND IS UNSATISFIED -NONFATAL
 DIAGNOSTIC-WHERE CLAUSE NOT SATISFIED -NONFATAL
UPDATE;
REQUEST....
ADD BOAT CLASS EQ 0 )) END WHERE BOAT NUMBER EQ 999;
      58 SELECTED DATA SETS
DATA BASE UNALTERED
REQUEST....
CHANGE C4 EQ 1 )) END WHERE C1 EQ 531122 AND C8 EQ 48;
      2 SELECTED DATA SETS
DATA BASE UNALTERED
REQUEST....
REMOVE ENTRY WHERE C1 EQ 531213 AND C15 EQ 016;
 DIAGNOSTIC-C   1 EQ 531213     UNSATISFIED-NONFATAL
 DIAGNOSTIC- AND IS UNSATISFIED -NONFATAL
 DIAGNOSTIC-WHERE CLAUSE NOT SATISFIED -NONFATAL
      0 SELECTED DATA SETS
DATA BASE UNALTERED
REQUEST....
ASSIGN CATCH INFORMATION EQ 11)251 22)25000 )) END WHERE C30 HAS
C11 EQ 251;
SYSTEM OR MACHINE ERROR ENCOUNTERED IN SUBROUTINE RRR.
    833 SELECTED DATA SETS
REQUEST....
INSERT TREE ENTRY EQ 1)530413 4)2 6)10 7)96 8)056 9)3 13)06
15)016 17)08 20)041 21)0026000 30) 11)241 22)0013000 )) END
RECORD CT 60 6306161
```

```
5 SELECTED DATA SETS
REQUEST....
TERMINATE:
  VERSION      1    2 TERMINATED
  REQUEST....
  RETRIEVAL:
  QUALIFY C8.C9 WHERE C1 EXISTS;
  --- END OF RETRIEVAL ---
  REPORTER:
  FORMAT OR GENERATE =
  LINES PER PAGE  IS =
  40;
  READY FOR FORMAT -
  TITLE IS GROUNDFISH 1953 BOAT INFORMATION;
  HEADING IS BOAT NO,BOAT CLASS;
  DETAIL IS C8,C9;
  SP 13 IN D1;
  FINIS;
  REPORTER;
  FORMAT OR GENERATE =
```

GROUNDFISH 1953 BOAT INFORMATION

BOAT NO	BOAT CLASS
120	5
141	4
141	4
56	3
56	3
56	3
56	3
56	3
999	0
24	3
24	3
24	3
24	3
6	4
6	4
6	4
132	3
132	3
132	3
154	3
2	3
2	3
56	3
56	3
56	3
56	3
154	3
154	3
148	3