NATIONAL WATER DATA EXCHANGE (NAWDEX) SYSTEM 2000 DATA RETRIEVAL MANUAL

By Owen O. Williams and William A. Knecht

U.S. GEOLOGICAL SURVEY
Open-File Report 81—419

1981

UNITED STATES DEPARTMENT OF THE INTERIOR

JAMES G. WATT, Secretary

GEOLOGICAL SURVEY

Doyle G. Frederick, Acting Director

For additional information write to:

Chief Hydrologist
U.S. Geological Survey, WRD
421 National Center
Reston, Virginia 22092

CONTENTS

FIGURES

APPENDIXES

NATIONAL WATER DATA EXCHANGE (NAWDEX)

SYSTEM 2000 DATA RETRIEVAL MANUAL

By Owen O. Williams (USGS) and William A. Knecht (CACI, Inc.)

MANAGEMENT OVERVIEW

This manual is designed to teach the reader how to retrieve data stored in a computer by using the SYSTEM 2000 Data Base Management System. SYSTEM 2000 is a registered trademark of Intel Corporation[1/]. The manual offers general instructions on the use of SYSTEM 2000 to retrieve any type of data, and specific instructions on how to retrieve data from the U. S. Geological Survey's Water Data Sources Directory (WDSD) and Master Water Data Index (MWDI) Data Bases.

The manual is divided into five major sections plus six appendixes for reference. Section I contains introductory material, including the purpose and scope of the manual, instructions on how to use the manual, basic computer concepts, and basic SYSTEM 2000 concepts. Section II contains basic instructions on retrieving and displaying data, and section III goes into more complex retrievals. Section IV explains how to make efficient use of techniques developed in sections II and III. Section V summarizes and discusses practical applications, and section IV contains all the appendixes required for reference.

1/SYSTEM 2000 is a registered trademark of Intel and its mention does not imply endorsement by the U.S. Geological Survey.

GLOSSARY

This Glossary is a compilation of data processing terms which are used in this manual. The first time the term appears in the manual it is underscored and followed by a footnote 1/. The terms are explained in both the text and the glossary.

ACTION CLAUSE - The part of a SYSTEM 2000 command that tells the system what action to take with the selected data records. All schema items to be printed are specified in the Action Clause.

ALPHANUMERIC CHARACTER - Any numeric digit (a number from 0 to 9), any letter of the alphabet from A to Z, or any special symbol, such as a plus sign, comma, percent sign, etc. A list of alphanumeric characters is in appendix D.

ANCESTOR DATA RECORD - A data record at the next higher level than another data record within the same data family.

BATCH-FATAL ERROR - Error that causes an exit from SYSTEM 2000 only during batch processing.

BATCH PROCESSING - The process of submitting punch cards through a card reader and receiving the output through a high-speed printer.

COMMAND - An English-language-like specification issued by the user to the computer instructing it to perform some function.

COMPONENT - One particular kind of item stored in a data base. In a SYSTEM 2000 data base, each component is identified by a unique name and number.

COMPUTER PROGRAM - A set of instructions to a computer directed toward the accomplishment of some goal.

CONDITIONAL CLAUSE - The part of a SYSTEM 2000 command that qualifies data records in the data base according to one or more conditions.

DATA BASE - A group of data stored together and organized according to characteristics the data have in common.

DATA BASE ADMINISTRATOR - The person or group of persons responsible for overall control of the data base system.

DATA FAMILY - Those data records vertically related as one branch of the hierarchial tree.

DATA ITEM - The value of a single item of data.

DATA RECORD - Each single occurrence of a group of data values all related to the same subject within a schema record.

DATA SUBTREE - Any data record at a given level plus all its descendant data records.

DISJOINT DATA RECORDS - Data record siblings descended from a common parent belonging to the same data sturcture but not to the same data family.

DISJOINT SCHEMA RECORDS - Schema records that do not belong to the same schema family.

EFFICIENCY - The ability to produce a desired result with a minimum of work by the computer.

FATAL ERROR - An error which causes an unplanned exit from SYSTEM 2000.

FORMAT STATEMENT - A set of options used to modify the standard format of output produced by the PRINT and LIST command.

INPUT - Data that are read into the computer.

ITEM TYPE - A designation of the way a component should be stored by the computer whether as alphanumeric characters or as numbers.

JOB CONTROL LANGUAGE (JCL) - A high-level language by which the information necessary to initiate a job and allocate machine resources is passed to the IBM Operating Systems.

KEY COMPONENT - A component for which a retrieval index is created, thereby making data set qualification more efficient when based on values for that component.

LEFT-JUSTIFIED - A string of characters, such as a title, positioned within a column in such a way that its first character falls in the leftmost position of the column.

NESTING - The process of enclosing successive combinations of conditions within parentheses.

NON-KEY COMPONENT - A component which has no retrieval index, therefore making data record qualification inefficient when based only on values for that component.

ONLINE PROCESSING - The process by which a command is typed in at a terminal, the command processed by the computer, and the results returned to the terminal prior to the issuance of another command.

OUTPUT - Data that are written out by the computer.

PARAMETRIC STRING - A predefined command containing one or more variables for which the user supplies values at the time the command is invoked.

PICTURE SPECIFICATION - An indication of the maximum size of the data items that can be stored for a component.

QUALIFIED DATA RECORDS - Those data records that satisfy an entire Conditional Clause.

RIGHT-JUSTIFIED - A string of characters, such as a title, positioned within a column in such a way that its last character falls on the right-most part of the column.

SCHEMA - The complete list of all data base component numbers and names, and their KEY or NON-KEY designation, item type, and field length. Also includes all Strings and Stored Functions.

SCHEMA ENTRY - All the data at all levels stored for one of the major items in a data base.

SCHEMA FAMILY - Those schema records vertically related as one single branch of the schema.

SCHEMA RECORD - A group of components related to the same subject that may have multiple occurrences of data.

SELF-CONTAINED LANGUAGE - A language made up of a number of words and phrases used to create SYSTEM 2000 data retrieval commands.

SIMPLE STRING - A predefined command requiring no user-supplied variables.

STORED FUNCTION - A mathematical function which is predefined by the Data Base Administrator and included as part of the schema. It is invoked as part of an Action Clause in order to perform some calculation.

STRING - A command which is predefined by the Data Base Administrator and included as part of the schema. There are two kinds of strings, parametric and simple.

SYSTEM FUNCTION - A predefined SYSTEM 2000 operator that enables the user to obtain the maximum, minimum, average, sum, count, or standard deviation of selected component values.

TIME SHARING OPTION (TSO) - A computer program that allows for interactive or online processing in IBM SYSTEM 370 computers.

SYSTEM 2000 RETRIEVAL MANUAL

SECTION I. INTRODUCTORY MATERIAL

SECTION I. INTRODUCTORY MATERIAL

I.A. INTRODUCTION

Purpose

This manual is designed to teach the reader how to retrieve data stored in a computer by using the SYSTEM 2000 Data Base Management System. It provides instruction to the person who has had little or no prior experience with data processing in how to effectively and efficiently use SYSTEM 2000. For the complete novice in data processing, Appendix A gives some general instructions in preparing and running jobs on the Geological Survey's central computer facility in Reston, Virginia.

General instructions on the use of SYSTEM 2000 without regard to the kind of data to be accessed are contained in this manual. Also included are specific instructions on how to retrieve from two of the U.S. Geological Survey's National Water Data Exchange (NAWDEX) data bases, the Water Data Sources Directory (WDSD) and the Master Water Data Index (MWDI).

Scope

SYSTEM 2000 offers a wide range of capabilities for creating data bases, updating data bases (adding, changing, or deleting data), data retrieval, data reporting, and interfacing with other computer programs. However, this manual covers only the subjects of data retrieval and data reporting, which are the processes of finding one or more elements of data among a large mass of data and displaying them in a manner that is most useful to the intended user.

It is understood that the reader may have little knowledge of data processing, and, therefore, the instructions begin with very basic concepts of data processing which must be understood before SYSTEM 2000 can be explained. Each new concept depends on the concepts developed immediately before, so there is continuity of instruction throughout the manual. Practical applications of all concepts are illustrated through numerous examples and exercises to be completed by the reader. A data base reserved for student learing purposes has been stored in the U.S. Geological Survey's computer installation in Reston, Virginia, and is available to the reader so that firsthand experience in SYSTEM 2000 can be obtained by actually submitting jobs to the computer and obtaining results.

The manual is divided into five major sections plus a set of appendices for reference. The contents of the five sections are as follows:

Section I. Introductory material including the purpose and scope of the manual, how to use the manual, basic computer concepts, and basic SYSTEM 2000 concepts.

Section II. Information on entry and exit from SYSTEM 2000, followed by basic instructions on how to begin the process of data retrieval and display.

Section III. Data retrieval and display at a more complex level than section II.

Section IV. How to make efficient use of the techniques developed in sections II and III.

Section V. Practical application of everything that has been discussed in sections I - IV. Sections I - IV teach the reader how SYSTEM 2000 works and what capabilities are available. Section V concentrates on using this knowledge to solve problems.

Upon completion of this manual, the reader should understand how to use SYSTEM 2000 to retrieve data from any SYSTEM 2000 data base. By studying the examples and completing the exercises in the manual, the reader also gains firsthand experience in retrieving data from an actual SYSTEM 2000 data base.

How to Use The Manual

This manual is intended to be a "stand-alone" document - that is, no other reference material is required. Therefore, it is important to follow the sequence of instruction without skipping any parts of the manual. In general, the sequence of instruction is (1) introduction of a new concept, (2) introduction of the SYSTEM 2000 capabilities related to that concept, (3) specific examples on how to apply those capabilities, (4) exercises for the student to complete, and (5) lead-in to the next concept.

Two kinds of problems and solutions are presented throughout the manual: (1) Example Problems illustrate the practical application of a concept or SYSTEM 2000 command in order to solve a problem. The solution to the problem and the results of applying that solution are included so that the student can readily see the entire problem-to-answer process. (2) Student Exercises are test problems for the student to work on his or her own. They cover subject matter introduced since the previous Student Exercise. The solutions and accompanying explanations are in appendix E. The main purpose of Student Exercises is for the reader to determine whether or not he or she has a grasp of the subject matter just presented and is ready to go on to the next part of the manual. If a reader has difficulty in working the Student Exercises, the material pertaining to the exercises should be reviewed. Failure to do so will almost certainly result in greater confusion further on.

Many of the Example Problems and several parts of the Student Exercises are tailored so the reader can use the computer to obtain the desired results. Readers are strongly encouraged to use the computer whenever possible to carry out the instructions in an Example Problem or to complete a Student Exercise. Any problem that can be worked on the computer will be marked by an asterisk (*). These Exercises should be run through the computer, and the results compared with the results shown in appendix E. If an error is made, the Exercise should be corrected and rerun until the correct solution is obtained.

Many of the Student Exercises have multiple-choice questions; more than one choice may be correct for those questions. All answers to marked Example Problems and the Student Exercises have been tested on the computer to verify their accuracy.

A number of terms in this manual, expecially in section I, are probably new to the person with no computer experience. These terms are underscored and followed by a footnote. The terms are defined within the text and again in a Glossary.

Throughout this manual the following documentation symbols are used:

< >	The enclosed item is mandatory and its contents are supplied by the SYSTEM 2000 user.
[ ]	The enclosed item is mandatory and restricted to the contents shown. When more than one item is contained within the brackets, the user must specify one of them.
{ }	The enclosed item is optional. When more than one item is contained within the braces, the user may specify only one of them.
Δ	Designates a single blank space.
ALL CAPS	Terms in all capital letters must appear as shown.
Lowercase	Terms in lowercase letters are supplied by the user.

These symbols do not appear in actual commands or output; they are used for documentation purposes only.

Reader Comments

The best way to improve a tutorial manual such as this is by responding to the comments of users of the manual. No one is better qualified to judge the effectiveness of a tutorial manual than a reader who starts out totally unfamiliar with the material that the manual is designed to teach.

The last page in this manual is reserved for reader comments and criticisms of the material in this manual. When you have completed the manual please fill in the last page and return it to the address below. Please return the comment page even if you have no suggestions for improvement. Your opinions of the manual's strong points will also be helpful.

National Water Data Exchange
U.S. Geological Survey
421 National Center
Reston, Virginia 22092

I.B. DATA BASE MANAGEMENT CONCEPTS

Data Storage and Retrieval

Among the most useful characteristics of a computer system are the capacity to store enormous amounts of data, the ability to search rapidly through the data, and when the needed data are located, to either display them or otherwise make them available in a useful format. The concepts of data storage and retrieval are interrelated, but they can be approached separately for the purpose of learning about each one. Basic concepts of data storage are presented first, followed by the fundamentals of data retrieval.

Data can be stored in computer-processable form in a number of ways, but one characteristic common to all is an organizational structure of some sort. Groups of data can be stored together and maintained separately from other groups of data that are organized differently. Groups of data stored together are commonly referred to as a data base[1]. For example, a data base may consist of data on sources of water data throughout the United States. This data base might be maintained separately from another data base of daily streamflow measurements. A unique name is assigned to each data base so that the computer and the user of the computer can distinguish among various data bases.

A data base contains many kinds of related data. For example, a data base on sources of water data may store for each source, its name, organization code, whether or not it is a NAWDEX member, the type of organization that it is, its orientation, and the kinds of water data that it collects. Each of these kinds of data is referred to as a component[1] of the data base. Data bases are structured according to related kinds of data, so that any component data value can be rapidly located according to its relationship to other component data values in the same data base. If someone wanted to find the types of data collected by a specific organization, it might be necessary to know the organization's name or organization code. The user must be familiar with the components within a data base and their relationships before data can be retrieved effectively.

Data retrieval is frequently accomplished by means of a computer program[1], which can be defined as a set of computer-readable instructions directed toward the accomplishment of some goal. Computer programs usually are written to perform a number of different tasks at the direction of the person using the program. Most programs utilize input and create output in successfully accomplishing their tasks. Input[1] is defined as data that are read into the computer, and output[1] is defined as data that are produced by the computer. For example, if a program instructs the computer to read a water data sources data base, the data read from the data base are input. If the program instructs the computer to print a report of the data that were read in from the data base, the report is output. It may be that the program can select data for different source organizations from the data base and can print any one of a number of different reports depending upon the specifications of the user at the time the program is run. The specifications provided by the user are also "input", because they are in a form that the computer can read.

[1]Defined in glossary.

There are two common ways that the user can communicate specifications to the computer in the form of input. One method involves punching cards and reading the cards into the computer through a card reader. A complete set of punched cards may contain many things besides input to the program. Possibly included are an announcement to the computer that a job is about to begin; the name of the data base to be used as input; the name of the program to be run; an instruction to begin running the designated program; and a final card telling the computer that for that particular job no more input cards will follow. After all the cards for one job are read in, the job is processed and the computer typically prints the output on a high-speed printer (some printers can produce 1000 lines per minute) which may be located near the card reader. This method of submitting punched cards through a card reader and receiving the output through a high-speed printer is commonly known as batch processing[1].

A second method of computer access involves the use of a computer terminal with keyboard to send input to and receive output from the computer. Some terminals have a video screen on which the input and output are displayed, and others use paper like a typewriter. Instead of punching cards and reading them into a card reader, the user types input on the terminal. Instead of returning on a high-speed printer, the output normally is printed on the terminal's paper or shown on the video screen. This method of communication is commonly known as online processing[1].

Both batch and online processing have advantages and disadvantages, and some computer programs are structured to be used most effectively in only batch or online processing. SYSTEM 2000 is not limited in this way --it has features which allow the user to use both methods of communication efficiently.

SYSTEM 2000 Language

SYSTEM 2000 is a Data Base Management System designed to offer the user extremely flexible capabilities for retrieval and reporting of data. Any data base to be accessed by SYSTEM 2000 must be structured in a special way, a subject that is discussed in section I.C. Once the data base is created, the user has available many options for selecting individual data values or sets of data values from the data base.

This flexibility is made possible because SYSTEM 2000 has its own natural language (known in SYSTEM 2000 terminology as Self-Contained Facility)[1]; this language allows the user to construct English-language-like data retrieval specifications, known as commands[1], from a dictionary of words and clauses. Commands are constructed in much the same way that a normal sentence is put together in ordinary writing. That is, a sentence has a verb, describing some action that occurs; a subject, describing who or what the verb is acting upon; one or more adjectives, describing the nature of the subject or direct object; and so on. The sentence also has a syntax in which punctuation marks are used to clarify ordered strings of words.

[1]Defined in glossary.

Similarly, in creating a SYSTEM 2000 data retrieval command, the user builds the command out of standard SYSTEM 2000 words and clauses that are structured according to rules of syntax. Part of the command describes what action is to occur; another part describes what is to be acted upon; still another part describes the conditions under which the action is to be taken. For example, a single command may instruct SYSTEM 2000 to search a data base for all organizations that collect surface water quality data in a particular state, then print a report of all the names and addresses of NAWDEX Assistance Centers belonging to those organizations that can be contacted about supplying the data. This single command has three functions --the first function is to tell SYSTEM 2000 to find data associated with organizations that collect a specific kind of data in a specific state; the second function instructs SYSTEM 2000 to act, once the data are found, only upon information about the Assistance Centers that can supply the data; the third function tells SYSTEM 2000 to print the data in a certain format.

Section II describes the words and clauses that are used and the rules that must be followed in order to combine words and clauses into effective commands. However, in order to learn how to retrieve data from a data base, it first is essential to know what data are stored in the data base and how the data are organized. The nature of a SYSTEM 2000 data base is explained in section I.C.

Data Base Administrator

The subject of this manual is data retrieval, which is the only function of a data base system ever encountered by the majority of users. There are, however, a number of other very important functions——such as design, creation, and maintenance of the data base——that someone must be responsible for to make data retrieval possible. These functions are the responsibility of the data base administrator[1] (DBA), a person or group of persons who have overall control of the data base system.

Among the tasks of the DBA are deciding which data will be stored in the data base, deciding how the data will be represented in the data base, defining editing criteria for data used to update the data base, defining security procedures to prevent access by unauthorized persons, interfacing with users of the data base, supervising intial creation of the data base, and approving all changes to the data base.

The role of the DBA is discussed here because many features of SYSTEM 2000 that affect data retrieval are controlled only by the DBA and not by the user. Features that are the responsibility of the DBA are noted so that the reader will realize that they are design features and may vary among data bases.

[1]Defined in glossary.

STUDENT EXERCISE 1-1

(1) A group of data stored together and organized according to a common feature is known as

   a. Input

   b. A Data Base

   c. A Component

   d. A Disk File

(2) Communicating with a computer by submitting punched cards through a card reader and receiving the printout on a high-speed printer is known as

   a. Online Processing

   b. Output Processing

   c. Input Processing

   d. Batch Processing

(3) English-language-like data retrieval specifications created by combining special words and clauses and used to tell SYSTEM 2000 what to do are known as

   a. Sentences

   b. Language

   c. Commands

   d. User-Defined Functions

(4) Communicating with a computer by typing input at a typewriter-like terminal and receiving output at the same terminal is known as

   a. Online Processing

   b. Output Processing

   c. Input Processing

   d. Batch Processing

I.C. THE SYSTEM 2000 DATA BASE

How Data are Stored in the Data Base

A SYSTEM 2000 data base consists of many kinds of related data, and each individual kind of data is called a component. Each SYSTEM 2000 data base has a unique name to distinguish it from all other SYSTEM 2000 data bases, and each component within the data base has a unique name and number to distinguish it from all other components in the data base. The name chosen for each component is generally one that is meaningful to users of the data base; the component number is used internally by SYSTEM 2000. For example, assume that a data base is being created to contain descriptive information on sources of water data throughout the United States, its territories, and some foreign countries, and that information on each source organization must include its name, organization code, whether or not it is a NAWDEX member, the type of organization, its orientation, and the date of the latest update of data about the organization.

To distinguish this data base from any others it is called WDSDTEST. Each component of WDSDTEST is given a number and name. The complete list of component numbers and names, plus a description of each component in the data base, is known as the schema[1]. In order to retrieve data from any SYSTEM 2000 data base, its schema must be known because the description of the various components determines how SYSTEM 2000 commands are processed. The schema is controlled by the Data Base Administrator.

Numbers used with component names generally are in ascending consecutive order, but this is not mandatory. Both of the lists below are correct.

1*	NAWDEX_AGCY	6*	NAWDEX_AGCY
2*	ORG_NAME	2*	ORG_NAME
3*	NAWDEX_MBR	100*	NAWDEX_MBR
4*	TYPE	255*	TYPE
6*	ORIENTATION1	19*	ORIENTATION1
7*	ORIENTATION2	20*	ORIENTATION2
9*	LAST_UPDATE	33*	LAST_UPDATE

NAWDEX_AGCY is the organization code assigned by NAWDEX. It is unique to each organization, so a specific code should never appear twice in the data base. ORG_NAME is the name of the organization.

Each component name must be a contiguous string of characters. In this example the underscore character (_) is used to provide the continuity and give the appearance of a two word name.

[1]Defined in Glossary.

NAWDEX_MBR is either "YES" or "NO", depending on whether or not the organization is a member of NAWDEX. TYPE is the type of organization (Federal, State, County, Private, etc.). This field will contain only a specific predefined code representing the type of organization. ORIENTATION1 and ORIENTATION2 are used to store the primary and secondary orientations of the organization (Industry, Flood Control, Navigation, etc.). LAST_UPDATE contains the date of the last update of data about the organization. The asterisk between the component number and component name is a notation used as part of the SYSTEM 2000 syntax.

A schema contains descriptive information on each component of the data base. Each component definition has five distinct pieces. The first two pieces are component number and component name, which have just been describec The remaining three items are the key/non-key indicator, data type, and size of the data field (called the picture specification). These are defined and explained later.

The schema serves to identify all components found in a data base. The WDSDTEST data base, defined above, shows that for any given source organizatic a maximum of 7 pieces of data are present, and those same 7 pieces of data can be present for any other source organization in the data base. No data are present that cannot be classified by one of the components in the schema. If any organization is chosen at random from the WDSDTEST data base the ORG_NAME or TYPE or LAST_UPDATE might be available for that organization, but the number of people employed by that organization is definitely not in the data base, because there is no component that exists to describe number of employees.

The amount of data in the WDSDTEST data base is independent of the data base schema. That is, there may be data for half a dozen organizations or for hundreds of organizations, but the data base schema does not change, because it describes the kinds of data that are stored, not how much. For any component there may be many occurrences of data, and each occurrence is called a data item[1]. A data item is a single entity, such as an organization code fc a specific organization or the number of surface water data collection station operated by that organization. The component name NAWDEX_AGCY occurs only once in the definition but NAWDEX organization codes, which are data items classified under the component name NAWDEX_AGCY, occur once for every organization present in the WDSDTEST data base. The other components (organization name, type, etc.) also may occur once for each organization stored in the data base. Inasmuch as it is possible to have 7 data items for each organization, there may be 7,000 data items if 1,000 organizations are stored in the data base.

All the data items for a single organization are related and kept separat from the data items for any other organization, and are collectively called a data record[1]. A data record is defined as "a single occurrence of a group of data items all related to the same entity." The entity in this case is an organization. Figure 1 shows three hypothetical data records (data items for three organizations) in the WDSDTEST data base.

[1]Defined in glossary.

If the user commands SYSTEM 2000 to retrieve all the data where the organization code (NAWDEX_AGCY) equals TX888, then only the data items from Data Record 1 are returned. No data items from Data Records 2 and 3 are returned because they are related to other organizations. As another example, suppose that the user requests names of organizations that are NAWDEX members. The first and third organizations are NAWDEX members, so Data Records 1 and 3 qualify and the ORG_NAME items from those two data records are extracted and printed. Data Record 2 does not qualify because its value for the component NAWDEX_MBR is N.

Components		Data Records		
		1	2	3
1*	NAWDEX_AGCY	TX888	CA555	USWB
2*	ORG_NAME	TEX RES DEPT	CAL HYDR ORG	U.S. WAT BRD
3*	NAWDEX_MBR	Y	N	Y
4*	TYPE	S	S	F
6*	ORIENTATION1	F	T	B
7*	ORIENTATION2	-	-	P
9*	LAST_UPDATE	06/19/1976	04/19/1976	01/12/1976

FIGURE 1. -- Three hypothetical data records.

How Data Are Set Up For Retrieval

It was stated earlier that the definition of the various components determines how SYSTEM 2000 commands are processed. Although SYSTEM 2000 offers the user a great amount of flexibility in creating data retrieval commands, the efficiency[1] of a command can be enhanced or greatly reduced depending on how the data base is set up initially.

To perform any task the computer must carry out a series of instructions and go through a number of steps before the task is completed; thus, an efficient command is defined as a command that can be completed with a minimum of work by the computer. For example, two different commands can be used to accomplish the same goal; the first command requires the computer to go through fewer steps to complete the task than the second command, thus the first command is more efficient.

Again consider the WDSDTEST data base in figure 1. An example was given in which the names of NAWDEX members were retrieved. To accomplish the data retrieval, SYSTEM 2000 qualified all data records that had a data item equal to Y for the component NAWDEX_MBR. But how did SYSTEM 2000 know which data

[1]Defined in glossary.

records have a value of Y for NAWDEX_MBR? There are two methods, one efficient and one inefficient, that it could have used to make this determination, and which method was chosen depends entirely upon the data base schema. Before examining how the data base schema controls the way SYSTEM 2000 retrieves data, the two methods of data retrieval are examined, along with the reasons why one method is efficient and one is inefficient.

The best way to explain methods of data retrieval is with an analogy. Assume that before the invention of the computer, all data on water data organizations were maintained on 3x5 cards in handwritten form. Each card contained an organization code, name, type, orientation, etc., for one organization. There were data on 10,000 organizations so there were 10,000 cards on file. A single card, then, was analogous to a data record and the entire file of cards was analogous to a data base. Assume that the cards were arranged in no logical order. If a request was received for the names of all State organizations (TYPE=S), the only way to find the data was to read the TYPE code on all 10,000 cards and when an S was found, to turn the card up on its end so it would stand out. After the TYPE code on every card had been searched, then all the cards standing on end (all those with TYPE S) were inspected and the ORG_NAME on each one written down to create the desired list. This process had to be repeated every time a similar request was made. If there were only 10 cards on file the process would not be too cumbersome. But with 10,000 cards in the file data retrieval is extremely time consuming.

The problem here is that even though there might be only a dozen State organizations, all 10,000 cards had to be read to ascertain this fact. A much easier way to find all the data for a certain TYPE is to create an index of the data in the data base. The first step in creating an index is to number all the cards sequentially from 1 to 10,000. New cards are added onto the end and given the next higher sequence number. Then an index of TYPE codes is created, with one index card per each unique TYPE code. If there are 50 different codes there are 50 cards in the index. On each index card is written the sequence number of every card in the data base having the unique TYPE code. There is one index card, then, containing the sequence number of every card in the data base having TYPE code S. If a request for the name of all State organizations is received, one needs only to turn to the index of TYPE codes, find the index card for TYPE code S, then select the appropriate cards in the data base according to their sequence number. This process is much more efficient than reading through all 10,000 cards.

Other indexes might be created so that cards could be located according to NAWDEX_AGCY, ORG_NAME, or some other component. If there are 10 different components in the data base, is it a good idea to create an index for each one? Not necessarily, for two reasons. If an index is created for each component, the number of index cards might be far greater than the number of cards in the data set. If storage space is limited there may not be room for all the index cards. Also, when the file is updated the indexes must also be updated. If updates are frequent, an excessive amount of time may be spent in accomplishing this task.

Generally, there is a tradeoff between the efficiency of data retrieval versus the storage and maintenance of the indexes. Careful judgement must be used when deciding whether or not to create an index for a component. For example, if 15 inquiries per week are received concerning data that must be retrieved according to TYPE code then perhaps TYPE should be indexed; however, if only one inquiry per year is received, then probably TYPE should not be indexed.

These two methods of data retrieval illustrate the basis for the concept of key[1] and non-key[1] designation of components, which is the third piece of descriptive information found in every SYSTEM 2000 schema. When a component is designated as KEY, an index is created for that component and it can be used to retrieve data in an efficient manner. If a component is designated as NON-KEY no index is created and using it to retrieve data will probably be very inefficient. The Data Base Administrator makes the choice whether to designate components as KEY or NON-KEY. Once this choice is made and the data base is defined, the person who is retrieving data from the data base may code his selection criteria such that a KEY component is treated as NON-KEY but cannot code a NON-KEY component to be treated as KEY (Refer to Section IV for discussion of the NON-KEY operand). Therefore, it is very important that the user be aware of not only which components are in the data base, but also which components are KEY and NON-KEY. Selecting data by use of NON-KEY components usually requires more computer time, and therefore the computer's response to user commands takes longer. Also, computer time costs money and so data retrieval using NON-KEY components can be much more expensive than data retrieval using KEY components. However, all components are not designated as KEY because of the storage required for the indexes and the computer time required to update the indexes each time an occurrence of KEY component is updated.

In some cases information as to which components are KEY or NON-KEY is provided in written documentation about the data base. If no documentation is available the user can command SYSTEM 2000 to print the schema. The method for doing this is discussed in section II.B.

As an example of how KEY and NON-KEY components are described, a partial data base schema showing component numbers and names, plus KEY and NON-KEY designation is illustrated:

```
1*   NAWDEX_AGCY    (KEY
2*   ORG_NAME       (NON-KEY
3*   NAWDEX_MBR     (NON-KEY
4*   TYPE     (NON-KEY
6*   ORIENTATION1     (KEY
7*   ORIENTATION2     (KEY
9*   LAST_UPDATE     (NON-KEY
```

[1]Defined in the glossary.

The schema presented here is incomplete because two or more pieces of information for each component are yet to be added. But it does illustrate how components are designated as KEY and NON-KEY. If the designation is omitted, the component is automatically a KEY component.

How the Format of Data in Each Component is Determined

Once it has been determined which components are KEY and which are NON-KEY, the next step is to define the nature of the data that is stored in each component of the data base - is it all numeric, all alphabetic, or a mixture of numeric and alphabetic? If numeric, is it integer or does the number have decimal places? How many decimal places? Is it a date? All these questions must be answered in order to gain a thorough understanding of the data base and to be able to retrieve data from the data base.

For example, refer to figure 1 on page I-12 which shows the data stored in the sample WDSDTEST data base. Notice that NAWDEX_MBR, TYPE, ORIENTATION1, and ORIENTATION2 contain a single alphabetic character. ORG_NAME contains names twelve letters long. NAWDEX_AGCY contains combinations of letters and numbers, and LAST_UPDATE contains dates in MM/DD/YYYY format. All these characteristics of the data are determined by the Data Base Administrator and are just as much a part of the data base schema as KEY or NON-KEY.

Without knowing whether a component contains data in numeric, alphabetic, or date format, comparisons cannot be made for the purpose of data retrieval. For example, if a user wants to find the names of all State organizations, the user must know that TYPE is represented by a single alphabetic character. In some other data base the actual description of the TYPE might be stored instead of a letter code.

Numeric and alphabetic codes are frequently used to represent lengthy data items in order to minimize storage space and to lessen the changes for error by users. It requires less space to store '31' than 'MASSACHUSETTS', and it is much easier to punch or type the numeric code than the full name. Of course, codes are usually meaningless without further information on what they represent. Therefore, it is important to obtain as much supporting documentation as possible about the values in a data base and about the meanings of any codes that may be used before attempting any access.

Six types of data can be stored in a SYSTEM 2000 data base: NAME, TEXT, DATE, Integer, Decimal, and Money. Each of these types is explained below.

NAME - Any string of alphanumeric characters[1] (250 characters maximum) with all leading, trailing, and extraneous (more than one) embedded blanks discarded when the data are stored. An alphanumeric character is defined as any numeric digit (a number from 0 to 9), any letter of the alphabet from A to Z, or any one of a set of special symbols, such as a plus sign, comma, percent sign, etc. A list of alphanumeric characters is in appendix D. Only a single embedded blank is retained between characters or words which are input with one or more blanks between them, and

[1]Defined in glossary.

any extraneous blanks are discarded. For example, ΔΔJOHNΔΔZ.ΔΔSMITH, ΔΔΔΔPRESIDENT is stored as JOHNΔZ.ΔSMITH,ΔPRESIDENT. The data item may be entered as shown in the first part of the example, but when it is retrieved it is printed as shown in the latter part.

TEXT - Any string of alphanumeric characters (250 characters maximum) with all blanks retained in the data. In the above example, if the component is defined as TEXT, the data item is retrieved and printed exactly as it was entered, complete with all preceding, internal, and trailing blanks.

DATE - A fixed format of MM/DD/YYYY, standing for month, day and year. "M", "D", and "Y" represent digits.

INTEGER - A string of digits (0 to 9) with an optional sign.

DECIMAL - A string of digits (0 to 9) with a preceding, embedded, or trailing decimal point, and an optional sign.

MONEY - This is stored the same way as DECIMAL but it is printed differently upon retrieval. The number is displayed with a preceding dollar sign ($); if negative, it has a CR (for credit) printed to its right. Only two decimal places can be accommodated.

Each component in the data base schema has associated with it one of the six terms defined above in order to describe the type of data stored for that component; thus, the item type[1] is the fourth piece of information found in every schema.

Presented below is the WDSDTEST data base schema from page I-14 expanded to include the data types:

```
1*   NAWDEX_AGCY    (KEY NAME
2*   ORG_NAME      (NON-KEY NAME
3*   NAWDEX_MBR    (NON-KEY NAME
4*   TYPE     (NON-KEY NAME
6*   ORIENTATION1    (KEY NAME
7*   ORIENTATION2    (KEY NAME
9*   LAST_UPDATE    (NON-KEY DATE
```

Only one more piece of information is needed to complete the data base schema: the size and format of the values for the item called a picture specification[1]. This is simply an indication of the maximum size of the data values that can be stored for a component.

[1]Defined in glossary.

From the data base schema of WDSDTEST it is evident that NAWDEX_AGCY is data type NAME but how large is it? How many letters can be used to describe ORIENTATION1? Note that the only component whose size is fixed is LAST_UPDATE, described as data type DATE; it will be in the form MM/DD/YYYY in WDSDTEST.

Two symbols are used in the picture specification; 'X' and '9'. An 'X' is used to describe the length of data types NAME and TEXT. It means that the data item may be constructed out of either alphabetic or numeric characters, special symbols, or a combination of them. A '9' is used to describe the length of data types INTEGER, DECIMAL, and MONEY, and indicates that the data item must be purely numeric— no letters of the alphabet or special symbols (commas, periods, slashes, etc.) are allowed. The length of the data field is determined by the number of X's or 9's used to describe the picture.

For example, if the picture specification is XXX, then any combination of three or fewer alphanumeric characters can be stored in the field. If the picture specification is 9999, then up to a four-digit number can be stored in the field. If the picture specification is 9999.99, then a number composed of four digits plus two decimal places can be stored in the field. An alternate way to describe the length of a data field is by using the notations X(n) or 9(n) where n is the number of characters in the field. In other words, XXXX is the same as X(4); 9(5) is the same as 99999; and 9(4).9(2) is the same as 9999.99.

It is not mandatory that the picture specification be included in the schema of every component. If it is missing, SYSTEM 2000 assumes the following lengths automatically:

Data Type	Picture Designation
NAME or TEXT	X(7)
INTEGER	9(7)
DECIMAL	9(6).9(2)
MONEY	9(6).9(2)

Therefore (KEY INTEGER) is the same as (KEY INTEGER 9(7)) or (INTEGER), since the KEY or NON-KEY specification also is optional.

The schema of the WDSDTEST data base, complete with the required descriptive information on each component, is given below.

```
1*   NAWDEX_AGCY    (KEY NAME      X(5))

2*   ORG_NAME     (NON-KEY NAME    X(23))

3*   NAWDEX_MBR     (NON-KEY NAME    X)
```

```
4*   TYPE          (NON-KEY NAME    X)

6*   ORIENTATION1  (KEY NAME     X)

7*   ORIENTATION2  (KEY NAME     X)

9*   LAST_UPDATE   (NON-KEY DATE)
```

All five pieces of descriptive information in the definition are necessary for carrying out the task of data retrieval. From the example, it can be seen that to retrieve data for a particular well by NAWDEX_AGCY, up to a 5-character code must be supplied to SYSTEM 2000. Additionally, since TYPE, ORIENTATION1, and ORIENTATION2 items are each stored in one character, it is evident that a system of codes is used in place of the full descriptions. Whenever coding systems are used to represent values in a data base, it is necessary to obtain documentation explaining the meaning of the coding schemes before accessing the data base. All coding systems used in the data bases in this manual are explained within the text of the manual.

STUDENT EXERCISE 1-2

(1) What five pieces of information are found in every data base schema?

(2) Components that are most efficiently used as a basis for selecting data from a data base are known as

a. NON-KEY Components

b. KEY Components

c. Data Records

d. Organization Codes

(3) Why aren't all components designated as KEY components?

(4) When a component is not designated as either KEY or NON-KEY in the data base schema, it automatically becomes

a. KEY

b. NON-KEY

c. Neither of the Above

d. Rejected by SYSTEM 2000 Along with an Error Message

(5) Which of the answers below is not a SYSTEM 2000 data type?

a. NAME

b. NUMBER

c. DATE

d. SPACE

e. ALPHABETIC

(6) If a component is defined as data type TEXT, and the following data item is input:

ΔΔJOHNΔΔΔA.ΔSMITH,ΔΔJR. how is it printed when retrieved?

a. JOHNΔΔΔA.ΔSMITH,ΔΔJR.

b. JOHNΔA.ΔSMITH,ΔJR.

c. ΔΔJOHNΔA.ΔSMITH,ΔJR.

d. ΔΔJOHNΔΔΔA.ΔSMITH,ΔΔJR.

(7) How is the above data item printed if it is defined as data type NAME? Choose from among the same four choices.

(8) Which data types are suitable for storing all the following values: 12685, 13423, 498, 77799, 58621, and 15?

a. INTEGER 99999

b. INTEGER 9999

c. DECIMAL 9999.9

d. NAME X(5)

e. INTEGER 9(5)

(9) Which data types are suitable for storing all the following values: 12685, 134A3, 13.67, 4%63R, ZRTMF, M/D/Y, and X9X99?

a. INTEGER 99999

b. DECIMAL 999.99

c. NAME X(5)

d. TEXT XXXXX

e. ALPHANUMERIC X(5)

(10) There are ten errors in the following data base schema. List all of them.

```
  1*  AAA (KEY INTEGER   9(6))

  5*  BBB (NAME   X(14))

  3*  CA53 (NON_KEY INTEGER   X(6))

 12*  IN_OUT (KEY DATE   XX/XX/XXXX)

104*  CCCC (KEY TEXT  X(10))

 33*  DD (NON_KEY   99.9(3))

 17*  E (NON_KEY DECIMAL   9(7))

 18*  FL4_5 (TEXT XXXXXXXXX)

 19*  MM AA (DECIMAL   XX.X)
```

```
10*  NNNN (NON-KEY DATE   9(6))

11*  RRR (DECIMAL)

12*  SSS (KEY NUMBER   9(8))

13*  TTT (KEY NAME   9999)

16*  ZZZ (TEXT  X(2000))
```

SECTION II. BASIC DATA RETRIEVAL AND REPORTING TECHNIQUES

SECTION II. BASIC DATA RETRIEVAL AND REPORTING TECHNIQUES

II.A. SAMPLE SYSTEM 2000 DATA BASES

In order to aid the reader of this manual in learning SYSTEM 2000, two sample SYSTEM 2000 data bases have been established to provide a basis for the Example Problems and Student Exercises. To provide a realistic format, the data base schema are subsets of the actual Water Data Sources Directory (WDSD) and Master Water Data Index (MWDI) data bases currently in use at the U.S. Geological Survey's National Center. The data items are fictitious but are designed to permit the demonstration of SYSTEM 2000 capabilities while still being realistic.

The sample data base that is a subset of the WDSD is called WDSDTEST and its definition has already been introduced. The other sample data base is called MWDITEST. It contains data on specific water data collection sites, each data set representing one site. A site may be a streamflow station, surface water quality monitoring station, a water well in which water levels are recorded periodically, etc. The MWDITEST data base consists of four sites. The components for each site are as follows:

1* NAWDEX_ID - The NAWDEX identification number assigned to the site.

2* LATITUDE - Latitude at the site's location.

3* LONGITUDE - Longitude at the site's location.

4* NAWDEX_AGCY - NAWDEX code for the organization operating the site. Comparable to C1* NAWDEX_AGCY in WDSDTEST.

7* STATION_NAME - Name of the station.

71* NON_US_CNTRY - FIPS alphabetic country code. Valued only if the station is located outside the U.S.

10* HYDROL_UNIT - Hydrologic unit for the station location.

12* SITE_TYPE - Two-letter code indicating type of site (stream, well, etc.). Codes are explained in appendix F.

17* NAWDEX_OFC - NAWDEX code for the organization's office responsible for operating the site. Comparable to C102* OFC_CODE in WDSDTEST.

22* STATE_COUNTY - State/County code used for retrieval. First two characters are FIPS numeric state code. Next three characters are FIPS numeric county code.

The WDSDTEST and MWDITEST data base schemas are illustrated in figure 2. The WDSDTEST and MWDITEST data items are illustrated in figure 3. Refer to these figures for problems in section II.

Notice that a few of the data items are missing. SYSTEM 2000 does not require the occurrence of a data item for every component in every data record. Therefore, if an item has not yet been recorded, it may be omitted now and added to the data base at a later date. Or the component may never be valued, such as NON_US_CNTRY for a site in the United States.

The sample data bases and their illustrated data records are stored on the U.S. Geological Survey's computer in Reston, Virginia. The data bases are set up so that readers of this manual can use them to gain experience in working with SYSTEM 2000 while learning about it. The Student Exercises in the remainder of this manual can all be performed using the two data bases, and students are encouraged to do so. Specific instructions on how to gain access to the sample data bases are given in appendix A.

Data Base Name: WDSDTEST

1* NAWDEX_AGCY (KEY NAME X(5))

2* ORG_NAME (NON-KEY NAME X(23))

3* NAWDEX_MBR (NON-KEY NAME X)

4* TYPE (KEY NAME X)

6* ORIENTATION1 (KEY NAME X)

7* ORIENTATION2 (NON-KEY NAME X)

9* LAST_UPDATE (NON-KEY DATE)

Data Base Name: MWDITEST

1* NAWDEX_ID (KEY NAME X(20))

2* LATITUDE (KEY INTEGER 9(6))

3* LONGITUDE (KEY INTEGER 9(7))

4* NAWDEX_AGCY (KEY NAME X(5))

7* STATION_NAME (NON-KEY TEXT X(15))

71* NON_US_CNTRY (NON-KEY NAME XX)

10* HYDROL_UNIT (NON-KEY NAME X(8))

12* SITE_TYPE (KEY NAME XX)

17* WDSD_OFC_CODE (KEY NAME X(4))

22* STATE_COUNTY (KEY INTEGER 9(5))

Refer to these data bases for problems in Section II

FIGURE 2.--WDSDTEST and MWDITEST
Initial Data Base Schemas.

DATA BASE NAME: WDSDTEST

Components	Data Sets		
	#1	#2	#3
1* NAWDEX_AGCY	USWCC	CA539	TX899
2* ORG_NAME	FEDERAL WATER CONS COMM	CAL-MEX BOUNDARY COMM	EAST TEXAS ENVIR ASSOC
3* NAWDEX_MBR	Y	N	Y
4* TYPE	F	I	G
6* ORIENTATION1	E	A	W
7* ORIENTATION2	P	-	B
9* LAST_UPDATE	04/12/1976	12/06/1975	06/19/1976

Refer to this data base for problems in Section II

Figure 3.--WDSDTEST data base initial structure.

Data Base Name: MWDITEST

Components		Data Set			
		#1	#2	#3	#4
1*	NAWDEX_ID	USWCC15340102	CA539M01623	TX8993714301	TX89907358000
2*	LATITUDE	330652	323021	315124	321017
3*	LONGITUDE	1170808	1145530	0940931	0940931
4*	NAWDEX_AGCY	USWCC	CA539	TX899	TX899
7*	STATION_NAME	SANTA MARGARITA R	COLORADO R	-	LAKE ROSE NR TYL
71*	NON_US_COUNTRY	-	MX	-	-
10*	HYDROL_UNIT	18100200	-	12300050	14700200
12*	SITE_TYPE	SW	SW	GW	LK
17*	WDSD_OFC_CODE	0601	01	02	01
22*	STATE_COUNTY	06073	80000	48419	48423

Refer to this data base for problems in section II

Figure 3 (cont'd).--MWDITEST data base initial structure.

II.B. HOW TO ENTER SYSTEM 2000

Beginning a Job

To prepare the U.S.G.S. computer system for access to the desired SYSTEM 2000 data base(s), the user must first determine whether batch processing or online processing is to be used. Then, before issuing the first SYSTEM 2000 command, the user enters a number of IBM Job Control Language[1] (JCL) commands if the access is in batch mode or Time Sharing Option[1] (TSO) commands if the access is in online mode. JCL and TSO commands, which are not to be confused with SYSTEM 2000 commands, are used to communicate with the computer - that is, to initiate and terminate each job, allocate facilities required for the job, and to activate the computer programs. For further explanation of JCL or TSO requirements prior to accessing the SYSTEM 2000 data base on the U.S.G.S. computer see appendix A.

SYSTEM 2000 Command Syntax

All SYSTEM 2000 commands must conform to certain syntax requirements. These requirements are few, however, and allow the user a degree of flexibility in entering commands. This section introduces basic rules of syntax that should be followed for all SYSTEM 2000 commands, regardless of their complexity. See appendix B for a comprehensive illustration of syntax requirements for each SYSTEM 2000 command.

1. Rule: Every command must be followed by a colon (:).

   Explanation: The colon tells SYSTEM 2000 that the preceding command has ended and another one is ready to begin. With batch processing (card reader) this feature permits the user to punch more than one command on a card, with a colon following each command. However, SYSTEM 2000 assumes that everything it reads until it encounters a colon is a single command, and if the colon is omitted at the end of a command, an error results. When communicating with a computer interactively (online processing) SYSTEM 2000 checks the command for syntax errors, but waits until a colon is sent before executing the command. Thus, when a command is entered without a colon at the end, the command is not executed. (See rule 3.)

2. Rule: Unnecessary blanks are ignored by SYSTEM 2000.

   Explanation: The usual way to punch a command on a card is to begin the command in column 1 of the card, punch the words making up the command with each word separated from the last by one blank space, followed by a colon at the end of the command. However, if two or more blanks are left between words, or if the first few columns on the card are left blank before beginning the command, or if blanks are left between the end of the command and the colon, SYSTEM 2000 still accepts the command. The user may even punch the first half of a command on one card and the second half of the command on the following card. An exception to this is when titles are specified in a LIST command or when qualifying on a TEXT field. These exceptions will be explained later.

[1]Defined in glossary.

3. Rule: When communicating with SYSTEM 2000 via online processing (typing in commands at a terminal) do not enter a command until SYSTEM 2000 asks for one.

   Explanation: SYSTEM 2000 asks for a command by printing three dashes (---). At this point a command should be entered. SYSTEM 2000 then processes the command, returns the appropriate response, and prints three dashes. If the command is sent without the colon at the end, three dashes are returned but the command is not executed. Since SYSTEM 2000 is still waiting for the end of the command, a colon may be sent on a separate line to initiate processing of the command.

   When communicating with SYSTEM 2000 via batch processing (reading in punched cards) this rule has no effect. (See rule 1, explanation).

Initial SYSTEM 2000 Commands

Immediately upon entry into SYSTEM 2000 two commands must be issued in order to gain access into the data base.

The first command is --

```
USER,<password>:
```

where <password> is a security password valid for the data base which is to be accessed. This password is not arbitrary - the user must know it before he can retrieve data from a data base. A single data base must have at least one password, but more are allowed if desired. Passwords are assigned by the Data Base Administrator.

As stated in section I.C, two sample data bases are used throughout this manual: WDSDTEST and MWDITEST. The password assigned to both data bases is TEST. Therefore, to retrieve data from either data base the first SYSTEM 2000 command entered should be USER, TEST:

If this command is entered correctly, SYSTEM 2000 responds by printing three dashes. If it is entered with a syntax error (USR, TEST: or USERTEST:, for example) SYSTEM 2000 rejects it and prints an appropriate error message followed by three dashes. With online processing, USER,TEST: should be reentered correctly. With batch processing, the run will terminate. If the command is entered syntactically correct but the password is incorrect, SYSTEM 2000 accepts the command but an error occurs when the DBN IS command (explained below) is entered. Appendix C has more information on errors that may occur and how to recover from them.

The second SYSTEM 2000 command required is --

```
DBN IS <data base name>:
```

where <data base name> is the name of the data base from which data are to be retrieved. For example, the proper command for accessing the WDSDTEST data base is

```
DBN IS WDSDTEST:
```

The proper command for accessing the MWDITEST data base is DBN IS MWDITEST:

The data base name must be the correct name for which the user password previously issued is valid. If it is, the following message is returned by SYSTEM 2000 --

```
ASSIGNED ...<data base name>
```

along with the date and time the data base was last updated. If the command is entered with an error in syntax, or if the data base name is incorrect, or if the password entered in the USER command was incorrect, SYSTEM 2000 rejects this command. See appendix C for more information on possible errors and methods of recovery.

Following successful entry of the USER and DBN IS command, SYSTEM 2000 is ready to accept data retrieval commands. When finished issuing data retrieval commands, the user may exit from the SYSTEM 2000 program by following the procedures outlined in section II.C. If retrieval from one data base is complete and retrieval from another SYSTEM 2000 data base is desired, it is not necessary to exit from SYSTEM 2000 to make the switch.

II.C. HOW TO EXIT FROM SYSTEM 2000

Normal Exit

The user may exit from SYSTEM 2000 by issuing the following command --

```
EXIT:
```

When this command is entered correctly the following message is returned --

```
END SYSTEM 2000
```

Following this message the user must enter either JCL (if batch) or TSO (if online) commands to terminate the run. See appendix A for details.

Exiting Under Unusual Circumstances - Online Processing

When communicating with SYSTEM 2000 via online processing (typing in commands at a terminal) it might be necessary to exit from SYSTEM 2000 at a time when the EXIT: command is not accepted; that is, when SYSTEM 2000 is not waiting for a command. For example, assume that the user mistakenly enters a data retrieval command which causes a large volume of unwanted data to be printed. Similarly, if a user enters a data retrieval command and then realizes that the command asks for the wrong data, allowing SYSTEM 2000 to process the command would waste the computer's as well as the user's time. In both of the cases, the user does not want the current command to be completed. However, the EXIT: command will not interrupt command processing because it can be entered only when SYSTEM 2000 is waiting for the next command. To terminate an unfinished command, depress the key labeled "BREAK" (or "INTERRUPT" or its equivalent). This stops processing of the current command and causes an exit from SYSTEM 2000. Refer to appendix C for instructions on how to reenter SYSTEM after this type of exit.

NOTE: Use this procedure with discretion because it may require as much computer time to exit from and reenter SYSTEM 2000 as it would have taken to complete the command that was terminated. Also, section II.E describes the LIMIT command, which should always be used to guard against printing out large volumes of unwanted data.

Unwanted Exit From SYSTEM 2000

Under certain circumstances a command may be issued which SYSTEM 2000 cannot complete and the result is an error message and an unwanted exit from SYSTEM 2000. This kind of error is called a fatal error[1]. When this occurs, all SYSTEM 2000 commands following the one which caused the problem are ignored. There are also other kinds of errors, called batch-fatal errors[1], which cause an exit from SYSTEM 2000 only during batch processing. If a batch-fatal error occurs during online processing, an error message is printed and the command is rejected but the user remains in SYSTEM 2000.

Information on errors and how to recover from them is presented in appendix C.

[1]Defined in glossary.

II.D. HOW TO PRINT THE DATA BASE SCHEMA

Printing the Complete Schema

When the data base schema was introduced in section I.C, it was stressed that the user must have adequate knowledge of the data base schema before making any attempts at data retrieval. The best way to become familiar with the data base schema is to obtain some sort of written documentation containing the schema, when possible. If there is no other alternative, the following command can be used to print the entire data base schema --

```
DESCRIBE:
```

Caution is advised when using this command to print the schema of an unknown data base, because it may be quite lengthy. If communication is via online processing it could require a long time to print the full schema on the terminal's low-speed printer. If communication is via batch, output is typically on a high-speed printer and a long schema may not pose a problem. Once the schema is printed, keep it for reference so that it will not have to be listed again.

Example 2-1

Problem: Assuming that the user has just entered SYSTEM 2000, obtain the schema for the WDSDTEST data base.

Solution: Issue the commands:

```
USER, TEST:

DBN IS WDSDTEST:

DESCRIBE:
```

The DESCRIBE command causes the entire data base schema to be printed.

Printing Selected Components in the Schema

Using options of the DESCRIBE command it is possible to print the description of a single component or a series of components in the data base schema. To list a single component use the following command --

```
DESCRIBE {component number} :
```

where {component number} is a letter C followed by an integer number. For example, in the MWDITEST data base (see figs. 2 and 3), the component number for NAWDEX_ID is represented as C1; the component number for LATITUDE is represented as C2; and so on.

Example 2-2*

(*indicates that problem can be worked in the computer as explained on page I-4.)

Problem: What commands are issued to print the schema of two components, NAWDEX_AGCY and HYDROL_UNIT, from the MWDITEST data base? Assume that the USER and DBN IS commands have already been issued.

Solution: Issue the following commands:

DESCRIBE C4:

DESCRIBE C10:

To list a series of components use the following command --

DESCRIBE {component number} THRU {component number}:

This command causes SYSTEM 2000 to print every component between the first component number and the second. The only restrictions to this command are that the first component must precede the second one in the data base schema, and that both component numbers must exist in the schema. It is not necessary for the second component number to be larger than the first because SYSTEM 2000 does not require component numbers in the data base schema to be in ascending numerical sequence. For example, consider the following simplified data base schema --

1* A
2* B
5* C
74* D
10* E
43* F
106* G
3* H
4* I
6* J

The command DESCRIBE C1 THRU C5: produces the following list:

1* A
2* B
5* C

Component numbers 3 and 4 are not printed because they do not fall between 2 and 5 in the schema. The command DESCRIBE C2 THRU C3: produces the following list --

```
  2* B
  5* C
 74*  D
 10*  E
 43*  F
106*  G
  3*  H
```

The command DESCRIBE C74 THRU C43: is valid even though 74 is a greater number than 43, because component 74 precedes component 43 in the data base schema.

STUDENT EXERCISE 2-1*

Access SYSTEM 2000 and call for the WDSDTEST data base. The required JCL and TSO commands are explained in appendix A under "Requirements for the WDSDTEST and MWDITEST Data Bases".

Issue SYSTEM 2000 commands to --

(1) Print the entire data base schema;

(2) Print the schema for the component ORG_NAME;

(3) Print the schema for components NAWDEX_MBR through ORIENTATION2;

(4) Exit from SYSTEM 2000. Terminate your job according to the instructions in appendix A.

*This problem should be worked on the computer.

II.E. DATA RETRIEVAL COMMANDS

How SYSTEM 2000 Retrieves Data

SYSTEM 2000 provides the user with extremely flexible data retrieval capabilities because the user can construct data retrieval commands in several ways by combining various words and phrases acceptable to SYSTEM 2000. There are two things that must be understood by the user prior to any attempt at data retrieval -- (1) the schema of the data base to be accessed, and (2) the processes that SYSTEM 2000 employs to retrieve data in response to various commands.

Section I.C discussed the importance of the data base schema and the five pieces of information included in every schema. This and subsequent chapters address how SYSTEM 2000 retrieves data.

The sample MWDITEST data base (see fig. 3) has four data records, one for each site. Each data record contains items for the same components according to the data base schema. When a user issues a SYSTEM 2000 data retrieval command, some condition or conditions are specified in the command that are used for qualification of the required data. SYSTEM 2000 considers those data records which satisfy the condition(s) as qualified. With regard to the current simple MWDITEST data base structure these qualified data records may also be termed selected data records. (Note: at a later point in this manual it is shown that this is not always necessarily true with more complex structures.) Next SYSTEM 2000 processes these selected data records to extract data values for the components the user requested, and displays the values in the specified format.

Example 2-3

Problem: If the user commands SYSTEM 2000 to print the NAWDEX_ID and STATION_NAME for all surface water sites (SITE_TYPE = SW) in the MWDITEST data base, how many data records are selected?

Solution: Two - the data records for sites 1 and 2, because they satisfy the condition that the site must be a surface water site.

Example 2-4

Problem: If the user commands SYSTEM 2000 to print STATION_NAME for all groundwater sites (SITE_TYPE = GW), how many data records are selected?

Solution: One - the data records for site 3. It does not matter that STATION_NAME does not exist in this data record. SYSTEM 2000 qualifies any data record satisfying the condition: SITE_TYPE = GW. Then it searches the selected data record for STATION_NAME and finds no values, so nothing is printed.

Occasionally it may happen that no data records are qualified/selected. When this happens the message

- 0 SELECTED DATA RECORDS -

is printed.

For example, if the following is requested from the WDSDTEST data base --

Print the NAWDEX_AGCY, where the TYPE code is A

no data records are qualified because none of the data records in the WDSDTEST data base has a TYPE code of A. Therefore, the above message is printed and, of course, no NAWDEX_AGCY's items are displayed.

Generally, the user who submits a data retrieval command to SYSTEM 2000 does not know how many data records will be selected or how much data will be printed from those data records selected. This is especially true when working with very large or unfamiliar data bases. For example, in the sample MWDITEST data base, if the user requests NAWDEX_ID's for all sites in the United States, three data records are selected and their NAWDEX_ID's are printed. In this case it causes no concern that three data sets are selected. But what happens if the same command is issued while working with a data base of all sites in the United States? Thousands or hundreds of thousands of data records might be selected and the same number of NAWDEX_ID's printed. Such output usually is not desirable, and steps must be taken to prevent that kind of problem.

How to Prevent Display of Excessive Unwanted Data

The USER and DBN IS commands are the first two commands issued upon entry into SYSTEM 2000. The third command that always should be issued is one that prevents SYSTEM 2000 from printing excessive amounts of data if too many data records are selected in response to a data retrieval command. This command is not mandatory like the USER and DBN IS commands; SYSTEM 2000 will proceed normally if it is never issued. But this command is very important and MUST be issued at the beginning of every SYSTEM 2000 run as a safety measure, so assume that it always follows the DBN IS command.

The format of the command is --

```
LIMIT <n1>, {n2}    {/disposition}:
```

Note: The braces { } designate an optional part of the command. The parentheses < > designate a variable to be supplied by the user. These symbols are for documentation purposes only and are not part of the actual command.

Options allow this command to take three forms:

(1) LIMIT <n_1>:

(2) LIMIT <n_1>,<n_2>:

(3) LIMIT <n_1>, n_2 [TRUNCATE / CANCEL]:

The variables are:

n_1 = minimum number of data records acceptable

n_2 = maximum number of data records acceptable

disposition = either TRUNCATE or CANCEL

The only restrictions are that n_1 and n_2 must be zero or a positive integer value and that n_1 must be less than or equal to n_2, with the following exceptions:

1) If n_2=0 the upper limit is infinite.

2) If option 1 is selected, n_2, assumes the value of n_1.

When the user issues a data retrieval command SYSTEM 2000 selects all data records which meet the user's specified conditions for data retrieval. If a LIMIT command was previously issued, as should always be the case, the number of selected data records is compared with n_1 and n_2 of the LIMIT command. If the number of selected data records is fewer than n_1 the message "TOO FEW SELECTED DATA RECORDS" is printed, followed by the message "<nnn> SELECTED DATA RECORDS" where nnn is the number of data records selected. If the number of selected data sets is greater than n_2 the message "TOO MANY SELECTED DATA RECORDS" is printed, followed by "<nnn> SELECTED DATA RECORDS." Then if {disposition} is CANCEL or is omitted nothing more is printed, the selected data records are discarded, and SYSTEM 2000 is ready for another command. If {disposition} is TRUNCATE the n_2 data records are saved, the rest are discarded, and SYSTEM 2000 prints whatever the user requested from the n_2 data records retained.

Following are several examples which illustrate the use of the LIMIT command. All examples refer to the MWDITEST data base (figs. 2 and 3).

Example 2-5

Problem: If there are exactly two surface water sites, print their NAWDEX_ID's. Print nothing if there are more than two or less than two. Assume you just entered SYSTEM 2000.

Solution: The following commands accomplish the required task:

```
USER,TEST:

DBN IS MWDITEST:

LIMIT 2:
```

- Data Retrieval Command (Print NAWDEX_ID where SITE_TYPE is SW) -

The NAWDEX_ID's are printed because two sites have a SITE_TYPE code of SW. If only one site has a SITE_TYPE code of SW, the following messages are printed:

- TOO FEW SELECTED DATA RECORDS -

- 1 SELECTED DATA RECORDS -

If all four sites have a SITE_TYPE code of SW, the following messages are printed:

- TOO MANY SELECTED DATA RECORDS -

- 4 SELECTED DATA RECORDS -

Example 2-6

Problem: Print the NAWDEX_ID's of all sites that are in the U.S. If there are more than two of them do not print anything. Assume you have just entered SYSTEM 2000.

Solution: The following commands accomplish the required task:

```
USER,TEST:

DBN IS MWDITEST:

LIMIT 0,2:

- Data Retrieval Command (Print NAWDEX_ID where STATE_COUNTY <60000)
```

In this case three data records are selected, which is more than the specified maximum of two, so the following message is printed:

- TOO MANY SELECTED DATA RECORDS -

- 3 SELECTED DATA RECORDS -

Example 2-7

Problem: Print the NAWDEX_ID's of all sites that are in the U.S. If there are more than two of them print only the first two. Assume you have just entered SYSTEM 2000.

Solution: The following commands are needed:

```
USER,TEST:

DBN IS MWDITEST:

LIMIT 0,2/TRUNCATE:

- Data Retrieval Command (Print NAWDEX_ID where STATE_COUNTY <60000)
```

Again three data records are selected, more than the maximum of two, and again the following messages are printed:

- TOO MANY SELECTED DATA RECORDS -

- 3 SELECTED DATA RECORDS -

In addition to the above messages the NAWDEX_ID's of the first two sites are also printed, because of the effect of the term TRUNCATE.

Example 2-8

Problem: Print the NAWDEX_ID's of all surface water and ground water sites. If only two or fewer sites meet this requirement, print nothing. If there are more than two, print them all. Assume you are already in SYSTEM 2000 and the USER and DBN IS commands are completed.

Solution: Issue the following commands:

```
LIMIT 3,0:

- Data Retrieval Command (Print NAWDEX_ID where SITE_TYPE is
SW or GW) -
```

In this case three data records are selected and three NAWDEX_ID's are printed. Remember, if n_2 is 0 the upper limit is infinite.

The LIMIT command can be entered at any time to change the effect of a previous LIMIT command. It is used not only as a safeguard against printing too much data but also as a useful data retrieval tool in conjunction with other commands. The LIMIT command will be demonstrated in conjunction with other commands as they are presented to show its usefulness and versatility.

How to Specify Which Data Records are Used in Data Retrieval

In many of the examples that have been presented in this manual, SYSTEM 2000 data retrieval commands were implied but not explicity stated. They have commonly taken the form below:

```
Print NAWDEX_ID  where SITE_TYPE is equal to SW;
      A                      B
```

Part B of the command is executed first by SYSTEM 2000. This part qualifies the data records that satisfy some user-specified conditions; for example, all those data records for which the SITE_TYPE component has a data value of SW. Once the required data records are qualified, Part A is executed. SYSTEM 2000 processes these selected data records, and extracts and prints the NAWDEX_ID components for which a value is found. Part A is called

an Action Clause[1] because it is the part of a command that tells SYSTEM 2000 what action to take with the qualified data records. Part B is called a Conditional Clause[1] because it is the part of a command that states the condition(s) according to which the data records are qualified. Syntactically, the Action Clause always precedes the Conditional Clause.

A typical data retrieval command takes the following form:

<Action Clause> WHERE <conditions exist>:

Conditional Clause

The <conditions exist> part of the Conditional Clause can take a number of forms, but its purpose is always the same; that is, to specify the conditions that must be met by any data record in order for it to be qualified. The term WHERE can be abbreviated as WH in any command, so the following format is also valid --

<Action Clause> WH <conditions exist>:

Every condition contains the name or number of a component, and the component should be a KEY component, although use of NON-KEY components is permitted under certain circumstances. Every Conditional Clause should contain at least one condition using a KEY component. If a command containing all NON-KEY conditions is submitted to the WDSD or MWDI data bases, the command will be rejected with the message --

- 371 - FULL DATA BASE PASS NOT ALLOWED -

0 SELECTED DATA RECORDS

The first format for a Conditional Clause is --

<Action Clause> WHERE <component> [EXISTS / FAILS] :

where <component> is either the name or number of a component that will be tested against a user-supplied condition. The brackets [ ] enclosing the conditions EXISTS and FAILS mean that the user may specify either one of these terms in the command. If EXISTS is used, the Conditional Clause qualifies all data records having a data item for the <component>. If FAILS is used, the Conditional Clause qualifies all data records missing a data item for the <component>. FAILS is treated as a NON-KEY condition, even if the component is KEY.

For example, assume the user wants to know which organizations in the WDSDTEST data base have no ORIENTATION2 value. The following command is entered (the Action Clause is still paraphrased but the Conditional Clause is exactly as it would appear in the actual SYSTEM 2000 command) --

Print the ORG_NAME WHERE C7 FAILS:

Action Clause Conditional Clause

[1]Defined in glossary.

This command tests component C7, which is ORIENTATION2, in each data record in the data base to see if it has a value. If it does not, the data record is qualified because it satisfies the conditions of the Conditional Clause. The data record for ORGANIZATION2 is the only data record qualified, so its name is extracted and printed.

Note: Commands using FAILS require much more computer time than do most other kinds of commands since they necessitate a complete search of the data base. Use these terms judiciously and only to satisfy a specific need which cannot be satisfied by another Conditional Clause. If FAILS is the only condition in the Conditional Clause, it usually will not be permitted.

The second format of the Conditional Clause is --

<Action Clause> WHERE <Component> [EQ NE LT LE GT GE] <value>:

where <component> is either the name or number of a component that will be tested against a user-supplied/<value>. The brackets [] enclosing the six terms mean that the user may specify any one of these terms in the command. The terms are defined as --

EQ - equal to

NE - not equal to

LT - less than

LE - less than or equal to

GT - greater than

GE - greater than or equal to

The tests that can be carried out using these terms do not have do be numeric, but they must be consistent with the data base schema. For example, if a component is defined as INTEGER then it must be tested against integer values. If a component is defined as DECIMAL or MONEY it must be tested against values with no more than the number of decimal places specified in the data base schema. Components defined as TEXT or NAME can be tested against any combination of numeric or alphabetic characters so long as the field specification is not exceeded. In testing alphabetic characters, consider the alphabet to be in ascending order from A to Z with A as the lowest value and Z the highest. Remember that blanks are lower than A and numeric digits are higher than Z. Therefore, A is less than E and K is greater than G. Dates may also be tested against other dates. The complete sequence of alphanumeric characters, in ascending order, is listed in appendix D.

The third format for a Conditional Clause is --

<Action Clause> WHERE <component> [SPANS / EQ / NE] <value$_1$>*<value$_2$>:

where <component> is either the name or number of a component that is tested against user-supplied values. The brackets [] enclosing the three terms mean that the user may specify any one of these terms in the command. The purpose of this command is to test to see if a component data item falls either inside or outside of a range of values supplied by the user. The SPANS and EQ terms are equivalent and are used to test component values for being within a range of values. The NE term tests component values for being outside a range of values. The lower limit of the range is <value$_1$> and the upper limit of the range is <value$_2$>. An asterisk separates the two values. Value$_1$ must be less than value$_2$. The values are inclusive; that is, a data set is qualified if its component value is equal to <value$_1$> or <value$_2$ >or any value in between.

For example, assume that the user wants to know the name of all stations in Texas (FIPS Code 48). Either of the following commands to the MWDITEST data base is correct:

Print the STATION_NAME WHERE C22 SPANS 48000*48999: or

Print the STATION_NAME WHERE C22 EQ 48000*48999:

The STATION_NAME from Data Records 3 and 4 are qualified as a result of either of the above commands. However, since data record 3 does not have a value for STATION_NAME, only STATION_NAME for data record 4 is printed.

STUDENT EXERCISE 2-2

[Refer to schemas on page II-5 or the Data Items on pages II-6 and II-7]

(1) SYSTEM 2000 executes a data retrieval command by:

a. Processing the Action Clause first, then processing the Conditional Clause.

b. Reading through the entire data base sequentially and printing everything specified by the user as it is found.

c. Qualifying data records on the basis of user-supplied values, then printing what the user requested from the selected data records.

d. Processing the Conditional Clause until the maximum number of data records specified by the LIMIT command are selected.

(2) If the user commands SYSTEM 2000 to print the LATITUDE of all sites in the United States, how many MWDITEST data records are qualified?

a. one

b. two

c. three

d. four

e. none

(3) If the user issues a LIMIT 0,2: command, then commands SYSTEM 2000 to print the LATITUDE of all sites in the United States, how many data records are qualified?

a. one

b. two

c. three

d. four

e. none

(4) If the user wants to print a certain kind of data from a data base only if it occurs more than three times and less than ten times, what LIMIT command is required?

(5) If the user desires to find the NAWDEX_ID's of any sites in the MWDITEST data base which are missing a site-type, which command(s) are used to accomplish this?

a. Print the NAWDEX_ID WHERE C12 EQ 0:

b. LIMIT 1,0/TRUNCATE
Print the NAWDEX_ID WHERE C12 LT A:

c. LIMIT 0,999/TRUNCATE
Print the NAWDEX_ID WHERE SITE_TYPE EXISTS:

d. Print the NAWDEX_ID WHERE C12 FAILS:

(6) If the user wants to know the TYPE of all sites that have an ORIENTATION1 code between S and Z inclusive, which command(s) could be used to determin this?

a. Print the TYPE WHERE ORIENTATION1 GT R:

b. Print the TYPE WHERE C6 LE S:

c. Print the TYPE WHERE C6 SPANS S*Z:

d. Print the TYPE WHERE ORIENTATION1 GE S:

How to Print Data from Selected Data Records

The reader has now learned how the Conditional Clause works to qualify data records, and that for the MWDI data base structure the qualified data records[1] are also the selected data records. Once the data sets are qualified, the Action Clause takes over. The Action Clause tells SYSTEM 2000 which data items are to be extracted and printed. The Action Clause may assume a number of different forms according to options chosen by the user, but the most basic form is --

PRINT <components> WHERE <conditions exists>:

Action Clause Conditional Clause

where <components> are the names or numbers of one or more components whose data items are to be extracted from the selected data records and printed. The components must be separated by commas; one or more spaces may also occur between components but they are not required. The components may be in any order desired by the user, and a component may appear more than once, if desired.

Components may be KEY or NON-KEY; this designation is irrelevant in an Action Clause. For example, to print the NAWDEX_ID for all groundwater sites, any of the following commands could be issued;

```
PRINT C1 WHERE C12 EQ GW:

PRINT NAWDEX_ID WHERE C12 EQ GW:

PRINT C1 WHERE SITE_TYPE EQ GW:

PRINT NAWDEX_ID WHERE SITE_TYPE EQ GW:
```

To print the latitude, longitude, type of site, NAWDEX ID, and NAWDEX agency code for all groundwater sites in MWDITEST, the following command could be used --

```
PRINT C2, LONGITUDE, SITE_TYPE, C1, C4 WHERE C12 EQ GW:
```

The steps taken by SYSTEM 2000 to retrieve data in response to the above command are shown in figure 4. It begins with a complete data base consisting of a number of data sets. One or more conditions are supplied by the user in the Conditional Clause of the command, and all the data records that meet those conditions are qualified. Next, SYSTEM 2000 processes these qualified/selected data records and extracts every data item for each component named in the Action Clause. Finally, the extracted components are printed.

[1]Defined in glossary.

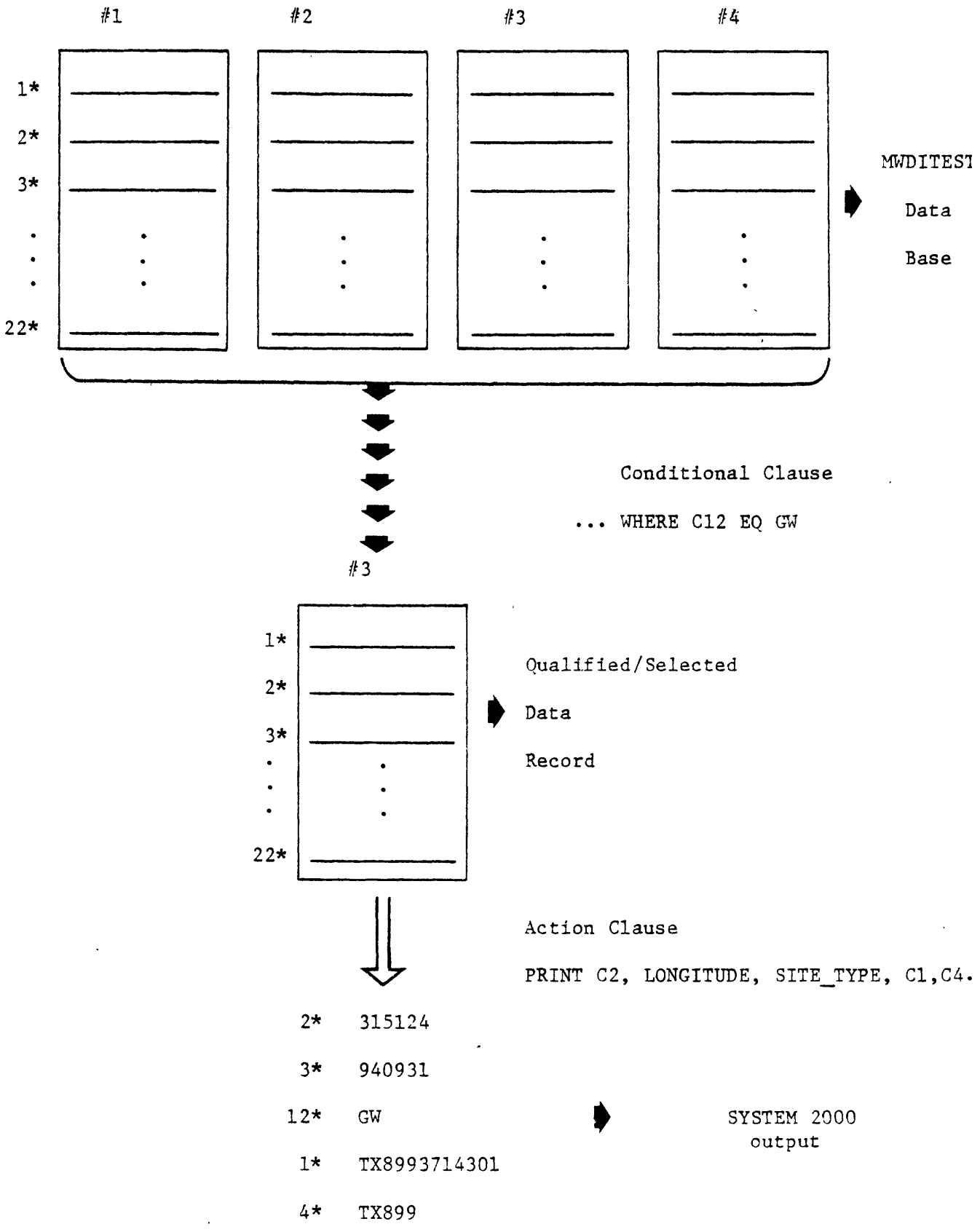


Figure 4.--Qualified/Selected Data Records and Output.

To this point, this chapter has concentrated on commands, which are the user's input to the computer, and little has been said about the output received in response to a command. There are a number of options available to the user for displaying output, most of which will be covered in section II-F. In this chapter, the emphasis is only on the output produced by the basic PRINT command, plus the output produced by a command called TALLY.

When a PRINT <components> WHERE <conditions exist>: command is issued, the output consists of all the data items for each component specified in the Action Clause that were found in the data records qualified/selected by the Conditional Clause. Each value is accompanied by the appropriate component number and an asterisk. The data items are printed in order by data record; that is, all the items from the first selected data record are printed first, followed by all the items from the second selected data record, and so on.

For example, the command

```
PRINT C1, C4,C17,STATE_COUNTY WHERE C1 EQ USWCC15340102:
```

produces the following output from the data base:

```
1* USWCC15340102
4* USWCC
17* 0601
22*   6073
```

Only one data record was selected, and the four components were printed in the order that they appeared in the Action Clause.

The following command produces the same output in different order --

```
PRINT C4, STATE_COUNTY, C17, C1 WHERE C1 EQ USWCC15340102:
```

```
4* USWCC
22*   6073
17* 0601
1* USWCC15340102
```

In the next example, three data records are qualified and the ordering of the output by data record is apparent.

The command --

```
PR C1, C7, C12, C17 WH C12 NE LK:
```

produces the following output --

```
1* USWCC15340102
7* SANTA MARGARITA R
12* SW
17* 0601
1* CA539M01623
7* COLORADO R
12* SW
17* 01
1* TX8993714301
12* GW
17* 02
```

Notice that component 7 is missing in Data Record 3. As can be seen in figure 3, no value was entered for the data record, so nothing is printed.

If in the above example a LIMIT 0,2/TRUNCATE: command precedes the data retrieval command, the following output is produced instead --

```
 - TOO MANY SELECTED DATA RECORDS-
 -       3 SELECTED DATA RECORDS-
1* USWCC15340102
7* SANTA MARGARITA R
12* SW
17* 0601
1* CA539M01623
7* COLORADO R
12* SW
17* 01
```

Data Record 3 was truncated so no data record from it was printed.

How to Specify More Than One Condition in the Conditional Clause

The previous commands have used a single user-supplied condition in the Conditional Clause; such as ... WHERE C4 EQ USWCC: or ... WHERE C17 GT 01:. In other words, only one condition was imposed upon a data record as a requirement for qualification. If C17 is greater than 50, the data record is qualified if not, it is rejected. However, a single condition is often inadequate for qualifying data records, especially when working with a large data base. There may be thousands of sites with a USWCC source agency or an office code greater than 01. To reduce the number of data records qualifed by the Conditional Clause, it may be necessary to specify more than one condition. For example, the user may want to print data on sites in state 48 that are north of 33° latitude.

The number of data records qualified also can be expanded when specifying more than one condition, if the data records must satisfy only one of the conditions in order to be qualified. For example, the user may want to print data on wells that have a TYPE code of either F or I.

Imposing more than one condition for data record selection is accomplished by the use of the operators AND and OR, as follows --

<Action Clause> WHERE <condition> [AND / OR] <condition> ... :

As many conditions as necessary may be specified, each separated by AND or OR. When AND is used, data records must satisfy all the conditions in order to qualify. When OR is used, data records must satisfy only one of the specified conditions in order to qualify. For example, if a command contains five conditions, each separated by AND, a data record must satisfy every one of those conditions before it can qualify. Even if a data record meets four of the conditions, if it fails the fifth, it will not qualify. On the other hand, if five conditions are specified, all separated by OR, then a data record qualfies if it satisfies any one or more of the conditions. A data record fails to qualify only if it does not satisfy any of the conditions.

For example, suppose the user wants to know the NAWDEX_ID of all surface water sites in Mexico (FIPS codes 80 thru 86). The following command is used:

PRINT C1 WHERE C22 SPANS 80000 * 86999 AND C12 EQ SW:

Data Records 3 and 4 do not satisfy either condition, so neither qualifies. Data Record 1 satisfies one condition, the SITE_TYPE, but not the other. Only Data Record 2 satisfies both conditions, so it is the only one that qualifies and its NAWDEX_ID value is extracted and printed. Consider the results of the previous command if OR had been used instead of AND --

PRINT C1 WHERE C22 SPANS 80000 * 86999 OR C12 EQ SW:

Data Records 3 and 4 still do not meet either of the specified conditions so they do not qualify. Data Record 1 meets the first condition, so it qualifies. Data Record 2 meets both conditions, so it also qualifies. The NAWDEX_ID code is then extracted from Data Records 1 and 2 and printed.

The AND and OR operators may be used together in the same Conditional Clause; this combination is covered in section IV. These operators are used together extensively to retrieve data from more complex data bases, and they will be seen in many of the commands in section III along with explanations of how to use them during various data retrieval operations.

How to Obtain Statistical Information About Selected Data

The TALLY command allows the user to obtain statistical information on any KEY component in the data base. It has three formats --

(1) TALLY/ALL/<components>:

(2) TALLY/EACH/<components>

(3) TALLY <components>:

where <components> are the names or numbers of KEY components, separated by commas, that are to be included in the statistical analysis.

The TALLY/ALL/ format produces, for each component, a report of the minimum and maximum data items in the entire data base for that component, the number of unique data values for that component, and the total number of data items for that component in the data base.

For example, the command

TALLY/ALL/C4,C22:

produces the following output from the MWDITEST data base --

```
**********************
 ELEMENT-      NAWDEX_AGCY
**********************
 MINIMUM-      CA539
----------------------
 MAXIMUM-      USWCC
----------------------
        3- UNIQUE VALUES
----------------------
         4 OCCURRENCES
----------------------
**********************
 ELEMENT-       STATE_COUNTY
**********************
 MINIMUM-       6073
-----------------------
 MAXIMUM-       80000
-----------------------
        4 UNIQUE VALUES
-----------------------
        4 OCCURRENCES
-----------------------
```

The component name is printed at the top of the report, and underneath are the minimum and maximum data items. The NAWDEX_AGCY item occurs four times, once for each data record; however, only three of them are unique. The STATE_COUNTY data item occurs four times; all four occurrences are unique.

The TALLY/EACH/ format produces for each component a report of every different data item found in the data base and how many times that data item occurs. The total number of different data items and the total number of occurrences are also printed.

For example, the command

TALLY/EACH/C4,C22:

produces the following output:

```
**********************
 ELEMENT-      NAWDEX_AGCY
**********************
FREQUENCY    VALUE
----------------------

         1      CA539
         2      TX899
         1      USWCC
----------------------

         3 UNIQUE VALUES
----------------------

         4 OCCURRENCES
----------------------
**********************

 ELEMENT-      STATE_COUNTY
**********************

FREQUENCY    VALUE
----------------------

         1       6073
         1       48419
         1       48423
         1       80000
----------------------

         4 UNIQUE VALUES
----------------------

         4 OCCURRENCES
----------------------
```

The component name is printed at the top of the report; each different data item of the component together with its frequency of occurrences is listed under the name. The data items are printed in ascending order.

The TALLY/EACH/ command gives more information than the TALLY/ALL/ command because it lists each different data item of a component. However, it is sometimes best to use the TALLY/ALL/ command to ascertain how many different data items exist before using the TALLY/EACH/ command. For example, suppose the user is just beginning to work with a very large data base, and the TALLY/ALL/ command reports that some component has 1500 different data items and occurs 3000 times. If the TALLY/EACH/ command is used first, all 1500 different data items are printed, which may be undesirable.

The /EACH/ and /ALL/ options of the TALLY command, once set, remain in effect until they are changed by the user or until the session is terminiated. Thus the effect of the third format, TALLY <components> is determined by the option set in previous TALLY commands, or by the system default if no previous TALLY has been issued. The system default option is /EACH/. When a data base is first accessed, the command TALLY <components> is equivalent to TALLY/EACH <component>; when the command TALLY/ALL/ <components> is issued the option is reset, and thereafter TALLY <components> is equivalent to TALLY/ALL/ <components>. The option is reset when TALLY/EACH/ is issued again, and so on. It is prudent, therefore, to specify /EACH/ or /ALL/ with every TALLY command, to assume the desired output.

A Conditional Clause is never used with a TALLY command; that is, TALLY ... WHERE ... is not valid. TALLY reports on the entire data base. However, the LIMIT command can be used in conjunction with the TALLY command to obtain additional information by specifying the number of occurrences a data item must have if it is to be output within the body of the TALLY report.

For example, assume the user wishes to verify that no NAWDEX_AGCY item occurs more than once in the WDSDTEST data base, because each NAWDEX_AGCY should be unique. The following commands are used:

```
LIMIT 2,0:

TALLY/EACH/NAWDEX_AGCY:
```

Only those NAWDEX_AGCY items having two or more occurrences are output, along with their frequency of occurrence.

As another example, suppose the MWDITEST data base contains data for thousands of sites and the user wants to know which counties have between 10 and 20 sites. The following commands produce the required report:

```
LIMIT 10,20:

TALLY/EACH/C22:
```

The TALLY command is not the only method for obtaining statistical information on specific data. A set of operators called SYSTEM FUNCTIONS[1] permit the user to perform various statistical operations on either KEY or NON-KEY components. System Functions are used within a command as follows --

PRINT {System Function} <component> WHERE <conditions exist> :

where {System Function} is any one of the following:

(1) MAX - maximum data item for the component

(2) MIN - minimum data item for the component

(3) COUNT - total number of data items that occur for the component

[1]Defined in glossary.

(4) SUM - sum of the values for the selected occurrences of the component; only components with data types INTEGER, DECIMAL, and MONEY can be used with this function

(5) AVG - average value of the selected occurrences of a component (calculated as X/n) that may be applied only to those with data types INTEGER, DECIMAL, and MONEY; the answer is carried to three decimal places regardless of the component's data type or picture specification.

(6) SIGMA - standard deviation of the selected occurrences of a component, calculated as follows --

$$\text{SIGMA} = \sqrt{\frac{\Sigma X^2 - \frac{(\Sigma X)^2}{N}}{N-1}}$$

it may be applied only to those components with data types INTEGER, DECIMAL, and MONEY; the answer is carried to three decimal places regardless of the component's data type or picture specification.

The output associated with a System Function includes the name of the System Function along with the component number. For example, the command --

```
PRINT MIN C3 WHERE C4 GT CA100:
```

produces the following output:

```
MIN 3*   940931
```

A System Function can reference only one component. If statistical information on several components is required, each component must be preceded by the appropriate System Function. Components without System Functions can also be included, as in the following command:

```
PR MAX C2,MIN C3,C1, COUNT C7, C22 WH C1 EXISTS:
```

When components preceded by System Functions and components without System Functions appear in the same Action Clause, the data values for the components preceded by System Functions are printed first, followed by the components not preceded by System Functions. For example, the above command results in the following output --

```
MAX 2* 330652
MIN 3*   940931
CNT 7*  3
1* USWCC15340102
22*   6073
```

1* CA539M01623
22* 80000
1* TX8993714301
22* 48419
1* TX89907358000
22* 48423

System Functions can be used only in an Action Clause, never in a Conditional Clause. Do not issue a command with a System Function to the right of the term WHERE.

System Functions also can be used to obtain statistics across the entire data base instead of just a subset of it. This is done by omitting the Conditional Clause from the command. For example, to obtain the latest date that an update was made to the WDSDTEST data base, the following command would be issued --

PRINT MAX C9:

Because no Conditional Clause is specified, all data records in the WDSDTE data base qualify and therefore the largest LAST_UPDATE in the data base is computed. Warning: The LIMIT command has no effect on data retrieval commands without a Conditional Clause. Therefore, it is recommended that data retrieval commands always contain a Conditional Clause except in extraordinary circumstances. The SUM and SIGMA functions can also be used to compute the total or standard deviation of all the values for a component over the entire data base. The use of the MIN, MAX, or COUNT operators in this manner is not recommended because the desired result can be obtained via the TALLY/ALL/ command, which is much more efficient in terms of computer time. Use MIN, MAX, or COUNT only with a Conditional Clause, or to obtain statistical information about NON-KEY components, since a TALLY is invalid in this latter case.

The inclusion of the System Functions in a command without a Conditional Clause must be used with caution, because computation of an average or standard deviation over a large data base requires a considerable amount of computer time. Usually, the sum or standard deviation for a particular component's values over an entire data base are of marginal worth, so a restrictive Conditional Clause should be used whenever possible.

Example 2-9

Problem: What command can be issued to print the number of Texas sites in the MWDITEST data base?

Solution: The command

PRINT COUNT C1 WHERE C22 EQ 48000*48999:

produces the following output --

CNT 1* 2

Example 2-10

Problem: Print the NAWDEX_ID, LATITUDE, and LONGITUDE of the easternmost site in the MWDITEST data base. The site cannot be a lake.

Solution: This requires two commands, and the second command cannot be entered until the output from the first command is received. The first command is --

PRINT MIN C3 WHERE C12 NE LK:

The output from this command is --

MIN 3* 940931

Now that the minimum latitude (easternmost) of non-lake sites has been determined, the second command can be entered --

PRINT C1, C2, C3 WHERE C12 NE LK AND C3 EQ 940931:

The results are --

1* TX8993714301

2* 315124

3* 940931

If communication is by batch processing, two separate jobs are required to arrive at the answer to this problem, because the output from the first command must be obtained before the second command is constructed. If communication is by online processing, the output from the first command is received immediately after issuing the command and the second command can then be entered.

Example 2-11

Problem: Print the SITE_ID, LATITUDE, and LONGITUDE of the easternmost site in the MWDITEST data base.

Solution: Again, this requires two commands and the second command cannot be entered until the output from the first command is received. The first command is --

TALLY/ALL/C3:

The TALLY command is used rather than PRINT MIN C3: because it is more efficient. From the report printed as a result of the TALLY command it is determined that the minimum latitude in the MWDITEST data base is 940931.

Then, as in Example 2-10, the command

PRINT C1,C2,C3 WHERE C3 EQ 940931:

is entered, and the results are:

```
1* TX8993714301
2*  315124
3*   940931
```

STUDENT EXERCISE 2-3

(1) The command(s) that will print a report of all different data items for a component and the frequency of occurrence of each value is

a. PRINT COUNT <component>:

b. TALLY/ALL/<component>:

c. TALLY/EACH/<component>:

d. PRINT MIN,MAX,COUNT <component>:

(2) Which of the following commands are incorrect for the MWDITEST data base?

a. PRINT COUNT C1, C2 WHERE C17 GT 10:

b. PRINT C22 WHERE C3 EQ MIN C3:

c. TALLY/SUM/C7, C10:

d. PRINT SIGMA C12:

e. PRINT MAX C4, MIN C4 WHERE C4 EQ USWCC:

f. TALLY/EACH/C22 WHERE C12 EQ SW:

(3) Assume that you have never worked with the MWDITEST data base and are unfamiliar with its data base schema. You have a requirement to find the type of site that occurs most often in the data base, and the number of times it occurs. Print the name and NAWDEX organizatio code for every site of that type and find out if any of those sites are located outside the U.S.

(4) Assume that you have just begun working with the WDSDTEST data base and are familiar with its schema. Print the names of up to 10 organizations whose primary orientation is agriculture and irrigation (C6 is A) What was the latest date that data on one of these organizations was updated? Print the names of all organizations that are not primarily oriented toward agriculture.

II.F. METHODS OF DISPLAYING DATA

Print Formats Available

There are a number of available options called Format Statements[1] that can be used to modify the standard format of output produced by the PRINT command. These Format Statements are used in a command as follows:

PRINT{/<format statements>/}<components> WHERE <conditions>
exist :

where <format statements> consists of one or more Format Statement options separated by commas, with the entire set of options enclosed in a pair of slashes. Format Statements come in pairs, with either one or the other of the options in the pair always in effect. There have been no Format Statements evident in any command presented in the manual up to this point because one of the options in each pair is a default option; that is, it does not have to be explicity stated to be in effect.

There are eight pairs of Format Statements that can be used, but four of them are not introduced until the more complex data bases are discussed in section III. The other four pairs are --

(1) SINGLE SPACE
DOUBLE SPACE

(2) NUMBER
NAME

(3) STUB
STUB SUPPRESS

(4) NULL SUPPRESS
NULL

The default option in each pair is the option on top of the line. Each time a Format Statement is specified it remains in effect for all subsequent commands until it is explicitly overridden by the other Format Statement of the pair. An explanation of each pair of options follows.

(1) The meaning of the SINGLE SPACE and DOUBLE SPACE options is obvious. All sample output from PRINT commands found earlier in this manual was always single spaced, because no option was specified and the default option (SINGLE SPACE) was in effect.

(2) It was shown in previous examples that output from a PRINT command consisted of the component number followed by an asterisk and a data item. The component number appeared because of the default Format Statement NUMBER. If PRINT/NAME/ ... is issued, the component name followed by an asterisk instead of the number precedes each output value.

[1]Defined in glossary.

(3) The component number or name plus the asterisk, which appear to the left of a data value in the output, is called a "stub." The option STUB SUPPRESS causes the stub not to appear and only the data items are printed. Again, in all previous examples, the stub appeared because the STUB default option was in effect.

(4) The NULL option is used to display the fact that a data item is non-existent by printing the characters "-NULL-" to the right of the stub. If the NULL SUPPRESS option is in effect, neither the stub nor a data item will be printed when a data item is missing. For example, suppose two data records within the same data base are qualified by the Conditional Clause and in the first data record C1 has a value of 66 but the values for C2 and C3 are missing. In the second data record C1 is missing and C2 and C3 have values of 100 and 200, respectively. A command to print C1, C2, and C3 from those two data records results in the following output.

```
1*   66
2*  100
3*  200
```

This output erroneously appears to come from one data record since the missing items are not noted in the output. If the NULL option is used, the output appears as follows --

```
1*   66              1* -NULL-
2* -NULL-            2*  100
3* -NULL-            3*  200
```

Now it is evident that the data came from two distinct data records.

Example 2-12*

Problem: Print the hydrologic unit and state/county of all sites in the MWDITEST data base. Double space the output, print component names in the stub, and make note of any missing items.

Solution: PRINT/DOUBLE SPACE,NAME,NULL/C10,C22: produces the following output:

```
HYDROL_UNIT* 18100200

STATE_COUNTY*    6073

HYDROL_UNIT* -NULL-

STATE_COUNTY*   80000

HYDROL_UNIT* 12300050

STATE_COUNTY*    48419

HYDROL_UNIT*  14700200

STATE_COUNTY*   48423
```

Controlling the Sequence by which Data are Displayed

The order in which data items are printed may be unpredictable, so SYSTEM 2000 offers the user the ability to control the ordering of output. With the ORDERED BY phrase, the user can sort the output data items before printing, using one or more components as sorting parameters.

The ORDERED BY phrase is specified in a command as follows --

PRINT <components>{, ORDERED BY <ordering clause>} WHERE <conditions exist>:

where <ordering clause> is a list of one or more component names or numbers, each preceded by either LOW (to indicate ascending order) or HIGH (to indicate descending order). If the LOW or HIGH is omitted, LOW is assumed. A comma must precede the term ORDERED BY. Other items that also may be included in the ordering clause are discussed later in this manual.

The components specified in the ordering clause determine how the output is sorted. Sorting is performed according to the first component specified, then within that sorting sequence further sorting is performed according to the second component specified, and so on. The complete set of alphanumeric characters is listed in ascending order in appendix D. Remember that for sorting purposes, the alphabet is in ascending order from A to Z, and A has the lowest value and Z has the highest. Missing values are the lowest value of all in the sorting sequence.

The best way to explain the ORDERED BY term is through the use of several examples. ORDERED BY can be abbreviated as OB.

Example 2-13*

Problem: Print the NAWDEX_ID's, in descending order, of every site in the MWDITEST data base west of the 92nd meridian.

Solution: The command

```
PRINT C1, ORDERED BY HIGH C1 WHERE C3 GT 920000:
```

produces the following output --

```
1* USWCC15340102
1* TX8993714301
1* TX89907358000
1* CA539M01623
```

The following command produces the same output --

```
PR C1, OB HIGH C1 WH C3 GT 920000:
```

Example 2-14*

Problem: Print a report of the NAWDEX organization code, organization name, type of organization, and primary orientation for the non-agriculturally oriented organization in the WDSDTEST data base that was most recently updated.

Solution: The commands

```
LIMIT 0,1/TRUNCATE:

PRINT C1,C2,C4,C6, OB HIGH C9 WHERE C6 NE A:
```

produces the following output --

```
 - TOO MANY SELECTED DATA RECORDS-
 -       2 SELECTED DATA RECORDS-
 - TRUNCATED -
1* TX899
2* EAST TEXAS ENVIR ASSOC
4* G
6* W
```

Ordering the output by HIGH C9 insures that the organization with the most recent update will be printed first. The LIMIT 0,1/TRUNCATE command insures that only one organization will be printed.

The data items are printed by data record but not in the same sequence as they would have been if the OB clause had not been used. If no OB clause had been used in the command, the data items from data record #1 would have been printed first.

Example 2-15

Problem: What is printed in response to the following command to the WDSDTEST data base --

```
PRINT C6,C7,OB C7  WHERE C1 EXISTS:
```

Solution: The selected data is printed as follows:

```
6*  A
6*  W
7*  B
6*  E
7*  P
```

The data items are printed in order by secondary orientation. Data Record 2 has a missing value, so it comes first in the sorting sequence, followed by the data for other data records in ascending order. If the command is --

```
PRINT/NULL/C6,C7,OB C7  WHERE C1 EXISTS:
```

the output is as follows:

```
6*  A
7*  -NULL-
6*  W
7*  B
6*  E
7*  P
```

Printing Columnar Reports with Titles

Instead of PRINT, the Action Clause may contain the verb LIST, which commands SYSTEM 2000 to display data in columnar format, one column for each component specified in the Action Clause. Commands using LIST also permit the user to specify titles at the top and bottom of each page and, in general, to control the layout of data on each page for the purpose of producing a report.

LIST offers many options; therefore, instructions start with a basic command with no LIST-oriented options, followed by a one-at-a-time introduction and demonstration of how each new option can be used in conjunction with the ones previously discussed.

The basic format of a LIST command is --

```
LIST <components>  WHERE  <conditions exist>:
```

where <components> is one or more names or numbers designating the components to be printed.

This is similar to the PRINT command, except that LIST is substituted for the verb PRINT. As before, component names or numbers are separated by commas.

The difference between using PRINT and using LIST is not the command format but rather the output format; with LIST the data items for each component print in columnar format with the component name at the head of the column and the data items for that component listed below. Asterisks print down the left side of the report (peculiarity of unknown origin).

As an example, suppose the user wishes to display NAWDEX_ID, LATITUDE, LONGITUDE, and STATE_COUNTY for all sites in the MWDITEST data base. One solution is to use the PRINT command

```
PRINT C1, C2,C3, C22:
```

which produces the following results --

```
1* USWCC15340102
2*  330652
3*  1170808
22*   6073
1*  CA539M01623
2*   323021
3*   1145530
22*   80000
1*  TX8993714301
2*   315124
3*    940931
22*   48419
1*  TX89907358000
2*   321017
3*    954320
22*   48423
```

Following is the same command using LIST, and its resulting printout --

```
LIST C1, C2, C3, C22:

  NAWDEX_ID                LATITUDE     LONGITUDE    STATE_COUNTY
***
* USWCC15340102              330652       1170808           06073
* CA539M01623                323021       1145530           80000
* TX8993714301               315124        940931           48419
* TX89907358000              321017        954320           48423
```

This format is often more desirable than the PRINT format because it is easier to read. Therefore, for some applications LIST may be chosen instead of PRINT.

The spacing between the columns is automatically controlled by SYSTEM 2000 under the basic version of LIST. Options that are discussed later allow the user to specify the location of columns. The order in which the columns print is determined by the order in which the components appear in the Action Clause. Titles above each column are the component names in the data base schema.

System Functions and the ORDERED BY clause may be used with LIST, as well as Format Statements SINGLE SPACE/DOUBLE SPACE and STUB/STUB SUPPRESS. The effect of SINGLE SPACE or DOUBLE SPACE is self-evident. The STUB SUPPRESS option will prevent the default titles at the top of each column from automatically being printed. The NULL option has no effect because missing values are always left blank in a report. The ORDERED BY clause, as before, allows the user to sort the output into different sequences according to the values of specified components. The System Functions are utilized in the same manner as with commands using PRINT.

When using the PRINT command (PRINT C1,C2,MAX C3,C10, COUNT C71, for example) components preceded by System Functions are mixed with components

not preceded by System Functions and the values computed by the System Functions are printed first no matter what their position is within the string of components. This is not true when using LIST. The LIST command prints all components in exactly the order they are specified, whether or not they are preceded by System Functions.

For example, the following command is the same command used in an example on page II-34 except that PRINT has been replaced by LIST and C22 is no longer included --

```
LIST MAX C2, MIN C3, C1, COUNT C7  WHERE  C1 EXISTS:
```

Compare the output below with the output produced by the PRINT command on page II-34 --

```
   MAX LATITUDE      MIN LONGITUDE      NAWDEX_ID          CNT STATION_NAME
***
*        330652             940931     USWCC15340102                    3
*                                      CA539M01623
*                                      TX8993714301
*                                      TX89907358000
```

Because LIST produces output in a columnar format across the page, it is possible to specify more components in the Action Clause than will fit in the space available. If the report to be printed by the LIST commands exceeds 132 characters in width, SYSTEM 2000 prints an error message and rejects the command. If the user is communicating in batch mode, output is printed on a high-speed printer on which any report up to 132 characters wide can be accommodated. However, if the user is communicating in online mode, the terminal being used will most likely have only 72 or 80 print positions available across the width of the paper or screen. Thus, even though SYSTEM 2000 will accept a LIST command that produces an output over 80 characters wide, anything past the 72 or 80 character width limit of the terminal is truncated when displayed on the terminal. If the online terminal has 132 print positions, it may or may not be possible to utilize anywhere from 80 to all 132 print positions, depending on constraints established at the computer site where the data base exists. Output up to 128 print positions wide can be produced from the WDSDTEST and MWDITEST data bases; for any other data base, SYSTEM 2000 support personnel should be contacted for information on the maximum report width.

If it is not desirable for SYSTEM 2000 to supply the column titles and to automatically control the location of each column, the user can supply titles and specify column spacing by invoking a special option --

```
LIST{/TITLE <column specifications>/ <components> WHERE
<conditions exist>}
```

where <column specifications> may consist of a single specification or several specifications separated by commas. A specification contains two items of information:

(1) A title to be placed at the top of a column of output.

(2) The width of a column of output plus an indication of whether the title above the column should be right-justified[1] or left-justified[1], or whether the column should remain entirely blank below the column title. Right-justified means that a title is positioned so that its last character always is printed in the rightmost part of a column; left-justified means that a title is positioned so that its first character always is printed in the leftmost part of a column.

Either item (1) or item (2) may be omitted, but at least one of the two items must always be present. The format of a column specification is

<type> (<integer>)<title>

where <type> is R, L, or B, for right-justified, left-justified, or blank, respectively. If B is used the title is left-justified and the column under the title is blank.

<integer> is a number, enclosed in parentheses, specifying the width of the column.

<title> is the title to appear at the top of the column. If the length of the title is greater than <integer>, then the width of the column will be the width of the title, overriding the specified width.

There is normally one column specification for each column to be printed in the report, and there is one column for every component specified immediately after LIST. The data items for the first specified component are printed in the first column, and at the top of the column the first specified title is printed. In other words, the first column specification "matches up" with the first specified component, the second column specification "matches up" with the second specified component, and so on. Examples of four column specifications and their meanings are shown below --

Column Specification	Meaning
L(10)TITLE-1	Left-justify TITLE-1 above a column 10 characters wide
R(15)TITLE-2	Right-justify TITLE-2 above a column 15 characters wide
B(30)TITLE-3	Left-justify TITLE-3 above a blank column 30 characters wide

[1]Defined in glossary.

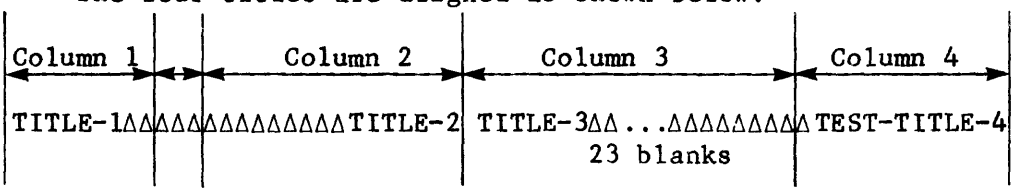
L(4)TEST-TITLE-4 Left-justify TEST-TITLE-4. Since the title is 12 characters wide, but the width is specified as only 4, the column will be 12 characters wide.

Column titles are treated like data type TEXT in the data base schema (p. II-5); that is, leading, trailing, and embedded blanks are retained so that placement of the title is possible. For example, a specification L(20)TITLE-1 causes TITLE-1 to print in positions 1 through 7 of the 20-position-wide column. However, if the specification is L(20)ΔΔΔΔΔΔΔTITLE-1 then TITLE-1 prints in positions 7 through 13 since positions 1 through 6 are occupied by spaces (remember that the Δ is a documentation symbol representing a blank space; it does not appear in the command). The title could be placed in exactly the same position by using R(20)TITLE-1ΔΔΔΔΔΔΔ since in this case the title is situated so that the rightmost character of the title, which happens to be blank, falls on position 20. Then TITLE-1 is in positions 7 through 13 and the blanks are in position 14 through 20. Positions 1 through 6 are also blank.

An example command using the column specifications is explained below--

LIST/TITLE L(10)TITLE-1,R(15)TITLE-2,B(30)TITLE-3,L(4)TEST-TITLE-4/C1,C2,C3 C4 WHERE<conditions exist>:

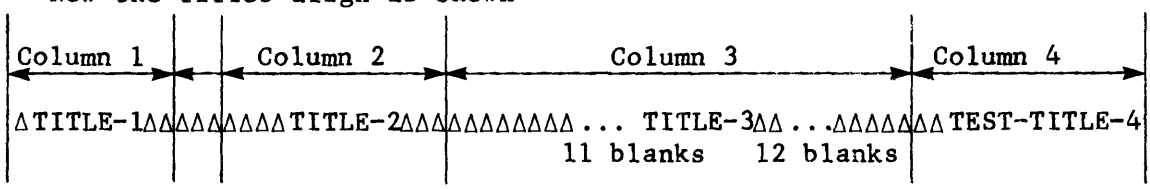
The four titles are aligned as shown below.



If the user wants to center the four titles, the command is structured in a slightly different manner:

LIST/TITLE L(10)ΔTITLE-1, L(15)ΔΔΔTITLE-2, B(30)ΔΔΔΔΔΔΔΔΔΔΔΔTITLE-3, L(4) TEST-TITLE-4/C1,C2,C3,C4 WHERE < conditions exist>:

Now the titles align as shown --



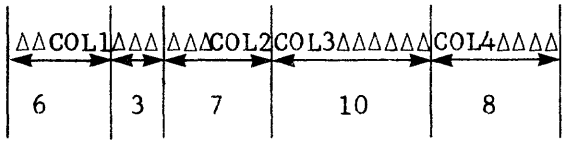
It was stated earlier that either the portions <type>(<integer>) or the portion <title> can be omitted in a column specification. If <type> (<integer>) is omitted, the title is left-justified and the width of the column is equal to the number of characters in the <title>. If <title> is omitted there is no title at the top of the column.

To prevent data from running together in the output, SYSTEM 2000 skips three spaces between columns, unless the columns are defined as blank

(B(10)COMMENTS, for example). The following partial command helps to illustrate this feature --

```
LIST/TITLE R(6)COL1, R(7)COL2, B(10)COL3, L(8)COL4/...
```

An illustration of the output produced by this command is shown below. Column widths in characters are indicated by the number below the arrows.



Notice that SYSTEM 2000 automatically skips three spaces between the first two columns, but no spaces are skipped between columns two, three, and four since column three was designated as a blank column. Keep this in mind when assigning blank columns because titles may run together in adjacent columns, as is shown in the example. To prevent this use B(10)ΔCOL3 for the specification for the third column. If you want two columns to run together, use B(0) between the column specifications.

The position of the data items in a column may help you to determine where you want to place the column title. Data types INTEGER, DECIMAL, MONEY, and DATA are always right-justified; data types NAME and TEXT are always left-justified. This is done automatically by SYSTEM 2000 with no user control permitted.

It may occur that the data item to be printed in a column is wider than the space reserved for that column. When this happens the leftmost digits of data types INTEGER, DECIMAL, and MONEY are truncated and an asterisk indicates that truncation has occured. Data types NAME and TEXT have a "wraparound" capability that allows the data to continue printing on additional lines, up to a maximum of five lines, below the original line. For example, assume a column specification is L(5) DATA. If a DECIMAL data value of 15346.00 prints in this column it appears as *6.00. If a NAME data item of HANDY-DANDY CONTAINER CO. prints in this column it appears as

```
HANDY
-DAND
Y CON
TAINE
R CO.
```

Data type DATE is not truncated if the specified field width is too narrow. The column is automatically widened to 10 characters and the entire data is printed.

Page II-43 contains an example LIST command and its resulting output containing data on NAWDEX_ID, LATITUDE, LONGITUDE, and STATE_COUNTY for wells in the MWDITEST data base. Since no options were specified, the automatic default titles were printed. Now consider a LIST command that produces

the same output, except that the titles are supplied by the user using the TITLE option. The command appears as follows --

```
LIST/TITLE L(20)NAWDEX_ID,LATITUDE,L(7)LONGITUDE,STATE_COUNTY/C1,C2,C3,C22:
```

The output from this command is identical to that shown on page II-43. Following is an explanation of each item specified as part of the TITLE option:

L(20)NAWDEX_ID - Left-justify the title NAWDEX_ID above a column 20 characters wide. This is the first column specification, so it matches up with the first specified component, C1, which is NAWDEX_ID. In the data base schema NAWDEX_ID is defined as data type NAME X(20), meaning a string of up to 20 characters. Therefore, the data item is left-justified within the column.

LATITUDE - No column width is given, so the column is as wide as the title, 8 characters. SYSTEM 2000 skips three spaces between this column and the previous one. This is the second column specification so it matches up with the second specified component, C2, which is defined as data type INTEGER 9(6). Therefore, the C2 data values are right-justified within an 8-character-wide column.

L(7)LONGITUDE - Here the title is greater than the column width, so the column is as wide as the title, 9 characters. This is the third column specification so it matches up with the third specified component, C3, which is defined as data type INTEGER 9(7). Therefore, the C3 data values are right-justified within the column and SYSTEM 2000 skips three spaces between this column and the previous one.

STATE_COUNTY - No column width is given, so the column is as wide as the title, 12 characters. This is the fourth column specification so it matches up with the fourth specified component, C22, which is defined as data type INTEGER 9(5). Therefore, the C22 data values are right-justified within the column and SYSTEM 2000 skips three spaces between this column and the previous one.

Example 2-16

Problem: Print a report of NAWDEX_ID, STATE_COUNTY code, LATITUDE, and LONGITUDE for all U.S. sites in the MWDITEST data base. Print the data in a columnar format with an appropriate title above each column. Reserve 10 spaces between STATE_COUNTY and LATITUDE for comments and center the title COMMENTS above the blank spaces. Reserve 8 spaces between LATITUDE and LONGITUDE, with no title. Print the data in descending order by LATITUDE.

Solution: The following command can be used --

```
LIST/TITLE R(15)NAWDEX_ID,STATE_COUNTY,B(10) COMMENTS ,LATITUDE,
   B(8),LONGITUDE/C1,C22,C2,C3, OB HIGH C2 WH C22 LT 56000:
```

This produces the following output --

```
            NAWDEX_ID    STATE_COUNTY COMMENTS LATITUDE        LONGITUDE

***
     * USWCC15340102          06073          330652          1170808
     * TX89907358000          48423          321017           954320
     * TX8993714301           48419          315124           940931
```

STUDENT EXERCISE 2-4

(1) What happens if a user has previously entered a PRINT command with the DOUBLE SPACE option and later enters a PRINT command with the NULL SUPPRESS option?

a. Output continues to be double spaced and missing values are reported as -NULL-.

b. The DOUBLE SPACE option discontinues and the new option NULL SUPPRESS, becomes effective.

c. Output continues to be double spaced and missing values are not printed at all.

d. SYSTEM 2000 generates an error message because DOUBLE SPACE must be first conteracted by the default option, SINGLE SPACE, before a new option can be specified.

(2) Assume that you have already entered SYSTEM 2000 and have successfully issued the USER and DBN IS commands to call for the MWDITEST data base. What commands would you use in order to accomplish the following task.

a. First, PRINT the NAWDEX_ID and hydrologic unit of all sites that have a site type. Double space the output and print component names.

b. Following that, PRINT the LATITUDE, LONGITUDE, and SITE_TYPE of the same sites as above. The output should be single spaced, the stubs should have component names and missing items should be reported.

c. Next, print the NAWDEX_ID and station name of sites in California or Texas. The output should be single spaced, the stubs should have component names instead of numbers, and missing items should be reported. Print the output data in ascending order by NAWDEX_ID. The FIPS Code for California ia 6 and Texas is 48.

(3) If the user wishes to PRINT several NAWDEX_ID's in one column of a report and several corresponding SITE_TYPE codes in the next column, and the two column titles are to be printed exactly three spaces apart, which commands are correct?

a. LIST/TITLE R(15)NAWDEX_ID,SITE_TYPE/C1,C12 WHERE . . .

b. LIST/TITLE R(15)NAWDEX_ID,B(3),SITE_TYPE/C1,C12 WHERE . . .

c. LIST/TITLE R(15)NAWDEX_ID ΔΔ,L(6)SITE_TYPE/C1,C12 WHERE . . .

d. LIST/TITLE R(15)NAWDEX_ID, L(6) ΔΔ SITE_TYPE/C1,C12 WHERE . . .

e. LIST C1,C8 WHERE . . .

(4) Find two errors in the following command

LIST/TITLE R(20)NAWDEX_ID,ST_CNTY, L(6)LATIT, B(5), R(3)LONGITUDE, L(5)TYPE/C1,C22,C2,C3,C10,OB C7 WH C7 LT 50:

(5) In which order will the components specified in the following command be printed (assume a hypothetical data base)?

PRINT C2,C1, MAX C6,C9, AVG C8, COUNT C1,C6:

a. MAX C6, AVG C8, COUNT C1,C2,C1,C9,C6

b. C2,C1,MAX C6,C9, AVG C8, COUNT C1,C6

c. AVG C8, COUNT C1, MAX C6,C1,C2,C6,C9

d. COUNT C1,MAX C6, AVG C8,C1,C2,C6,C9

e. COUNT C1,C1,C2,MAX C6,C7,AVG C8,C9

(6) Which of the above answers is correct if LIST is used instead of PRINT?

(7) * Select and print a report in columnar format, listing NAWDEX_ID, SITE_TYPE, maximum LONGITUDE, STATE_COUNTY, and hydrologic unit for all sites having a state code greater than 30. Place the title ID NUMBER over the NAWDEX_ID's. Use titles TYPE, MAX_LON,ST_CNTY, and HYD_UNIT over the remaining columns of data. Skip 5 spaces between the NAWDEX_ID's and the SITE_TYPE's. Reserve a 7-space blank column entitled OTHER immediately after MAX_LON. Assume you have just entered SYSTEM 2000 and have not yet called for the MWDITEST data base.

*This exercise should be worked on the computer.

Using Format Statement Options with TITLE

The Format Statement options, as explained previously, may be used with the LIST command as follows --

```
LIST{/<format statements>/}<components> WHERE <conditions exist>:
```

If the TITLE option is used in the LIST command, the Format Statement immediately precedes the term TITLE and a comma separates the two items, as follows --

LIST{/<format statement>, TITLE<column specifications>/}
components ...

For example --

LIST/DOUBLE SPACE,TITLE L(23)ORG-NAME,TYPE/C2,C4 WH C6 EXISTS:

produces a double-spaced report of organization names and type codes.

It was also stated earlier that if LIST is used without the TITLE option, each column has a default title that is the component name in the data base schema. This feature was illustrated on page II-43. There is one other case where the default titles are used--when the total number of column specifications is fewer than the total number of components listed in the action clause. In this event the extra component will receive their default titles. For example, consider the following command --

LIST/TITLE L(23)ORG_NAME,TYPE,MEMBERSHIP/C2,C4,C3,C6,C7 WH...

Five components are to be printed but only three column titles are given. The three titles are assigned to the first three components, C2, C4, and C3, and the other two components receive the default titles ORIENTATION1 and ORIENTATION2.

The default titles always go to the last components without titles in the string of components specified by the user. That is, if three column titles and five components are specified, the first three components receive the user-supplied titles and the last two components have the default titles. For example, consider the following command --

LIST/TITLE R(23)ORG_NAME,TYPE,MEMBERSHIP/C2,C3,C6,C4,C7 WH...

This is similar to the previous example command except that the components are in a different order. The three titles ORG_NAME,TYPE, and MEMBERSHIP will be above the columns for the components C2,C3, and C6 (ORG_NAME, NAWDEX_MBR, and ORIENTATION1), which is obviously an error. The point being made here is that the column titles specified in the TITLE option are not automatically placed above the correct column related to the component name. It is the user's responsibility to maintain column integrity.

Column titles can be up to three lines long. That is, the following two column titles

NAWDEX IDENTIFICATION NUMBER LATITUDE (DEG-MIN-SEC)

can be printed as

NAWDEX IDENTIFICATION NUMBER

LATITUDE (DEG-MIN-SEC)

To accomplish this, titles are split into two or three lines with a plus sign (+) preceding the second and third line parts, to indicate that a new line is to begin. For example, the column specifications for the above titles are L(14) NAWDEX+IDENTIFICATION+ NUMBER,L(13) + LATITUDE+(DEC-MIN-SEC). The first title is split into three parts, each part separated by a + to indicate that the following part is to be printed on the next line. The L(14) indicates that the column is fourteen characters wide and that all three parts of the title will be left-justified. Four spaces are placed before the words NAWDEX and NUMBER in order to center them in the title. The second title begins with a + in order to place the word LATITUDE one line down from where it normally would be. Two spaces are placed before LATITUDE in order to center it over the other line of the title.

There are 11 different ways to arrange titles to fit on one or more of the three available lines --

(a) <pt. 1>

(b) <pt. 1>+<pt. 2>

(c) <pt. 1>+<pt. 2>+<pt. 3>

(d) +<pt. 1>

(e) +<pt. 1>+

(f) ++<pt. 1>

(g) +<pt. 1>+<pt. 2>

(h) <pt. 1>+<pt. 2>+

(i) <pt. 1>++<pt. 2>

(j) <pt.1>+

(k) <pt.1>++

Using the title specifications just given, the following represents the resulting positions of the parts of the title --

	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	(j)	(k)
Line 1	pt.1	pt.1	pt.1					pt.1	pt.1	pt.1	pt.1
Line 2	data	pt.2	pt.2	pt.1	pt.1		pt.1	pt.2			
Line 3	data	data	pt.3	data		pt.1	pt.2		pt.2	data	

Example 2-17

Problem: Print a report of NAWDEX_ID, STATE_COUNTY, HYDROLGIC_UNIT, SITE_TYPE, and NAWDEX_AGCY for all sites in the MWDITEST data base in the U.S. The column titles should be as follows --

NAWDEX IDENTIFICATION NUMBER	STATE COUNTY	HYDROLOGIC UNIT	TYPE	NAWDEX AGENCY CODE

Solution: The desired report can be obtained using the following command, which is long and must be entered on more than one line. Each line of the command shown here is either punched on a separate card if utilizing batch processing, or keyed on a new line if utilizing online processing.

```
LIST/TITLE L(23)    NAWDEX+IDENTIFICATION+    NUMBER,
R(5)+STATE_+COUNTY,+HYDROLOGIC+   UNIT,++TYPE,
R(5)NAWDEX+AGENCY+CODE /C1,C22,C10,C12,C4 WH C22 LT 59000:
```

The following report results --

NAWDEX IDENTIFICATION NUMBER	STATE COUNTY	HYDROLOGIC UNIT	TYPE	NAWDEX AGENCY CODE
***				
* USWCC157340102	06073	18100200	SW	USWCC
* TX8993714301	48419	12300050	GW	TX899
* TX89907358000	48423	14700200	LK	TX899

In the above example a command was issued that had to be entered on three separate lines. That is common when using the TITLE option because title specifications can be lengthy. The user must exercise caution when entering a long LIST command with a TITLE option because it is very easy to make errors that may be undetected until the report is printed. A common error occurs when more than one line is used for entering the titles and all 80 columns of the line are not used: This problem can best be explained by use of an example. Consider the following command with each line entered separately --

```
LIST/TITLE L(23)ΔΔΔΔNAWDEX+IDENTIFICATION+ΔΔΔΔNUMBER,

STATE+COUNTY,ΔLATITUDE/C1,C22,C2 WH C22 EQ 48000*48999:
```

Assuming the command on each line begins in column 1, then the first line occupies columns 1-53 and the second line occupies columns 1-55. The user may have stopped entering the first line after column 53 and begun a new line for some reason of convenience; in any event columns 54-80 of the first line are left blank. This is not a syntactical error, because SYSTEM 2000 does not assume that an entire command has been entered until a colon is reached; therefore, the command is syntactically correct and is accepted

by SYSTEM 2000. However, there is a logical error in this command that will probably go undetected by the user but will be very obvious when the report is printed.

The error is that the 27 blank spaces on the end of the first line are interpreted by SYSTEM 2000 as being part of the first title on the second line. Remember that titles are treated like data type TEXT, with all leading, trailing, and embedded blanks retained in the title. Titles are also separated by commas, so that everything between the last comma on the line 1 and the first comma on line 2 is assumed to be all one title. Since no specification of column width is given for the title STATE+COUNTY, the column width will be the number of characters in the title, which includes the name STATE preceded by 27 blanks.

The way to avoid this type of error is to always precede the first title on a card with a specification of column width, as was done in Example Problem 2-17. This is equally true for both batch and online processing.

For example, the following command is punched onto two cards as shown --

```
LIST/TITLE L(23)ΔΔΔΔNAWDEX+IDENTIFICATION+ΔΔΔΔNUMBER,
R(5)STATE+COUNTY,ΔLATITUDE/C1,C22,C2 WH C22 EQ 48000*48999:
```

As before, columns 1-53 in the first line are occupied and columns 54-80 are blank. But those 27 blank columns are ignored rather than appended onto the title STATE+COUNTY because the specification R(5) precedes the first title of the second line.

One more option is available when using the LIST command, and that is the ability to print headings and footings at the top and bottom of each output page. This option allows the user to specify titles to appear at the top and bottom of each page, to specify the number of lines to be printed on each page, and to print the current date and page number on each page . This feature is used with the TITLE option as follows --

```
LIST{/TITLE <heading>,<footing>,<column specifications>/}..:
```

Both <heading> and <footing> are optional items and either one may be used separately, together, or not at all.

A page heading is specified according to the format --

```
D(<nn>) text
```

where <text> is the title for printing at the top of each page (the current date is automatically centered under the title; the <text> may be omitted if the user wants only a page number and the current date at the top of each page).

and

<nn> is an integer enclosed in parentheses, that specifies which print position across the page (starting from the left margin) the <text> begins in.

A typical heading might be D(20)WATER WELL REPORT, which causes the title WATER WELL REPORT to be printed at the top of every page beginning in print position 20 (meaning that 19 spaces precede the title) and the current date to be centered underneath this title.

A page footing is specified according to the format --

F(<mm>)<text>

where <text> is the wording to appear left-justified at the bottom of each page (it may be omitted if the user wants only to specify page size).

and

<mm> is the maximum number of lines to be printed on a page - this includes the heading and footing and all lines in between (the minimum number of lines per page is 10 and the maximum is 60; if both a heading and a footing are used, there are <mm> minus 5 lines of output per page including column headings, but if only a footing is used, there will be mm minus 2 lines of output per page including column headings).

A page number prints on each page when headings or footings are used. If a heading is used, the page number prints on the heading line over the rightmost print position of the report. If only a footing is used, the page number prints on the footing line under the rightmost print position of the report. The rightmost print position of the report is the last position designated by the column specifications. Page numbers will not exceed 9999.

Example 2-18

Problem: Print the same report required in Example Problem 2-17, except center the following title at the top of the page: NAWDEX SITE REPORT. Also place the following footing at the bottom of the page: DATA COLLECTED BY USGS. Set the page length at 40 lines.

Solution: Each line of the command shown here is considered as entered on a separate line.

```
LIST/TITLE D(18)NAWDEX SITE REPORT,F(40)DATA COLLECTED BY USGS,
    L(23)ΔΔΔΔNAWDEX+IDENTIFICATION+ΔΔΔΔNUMBER,
    R(5)+STATE-+COUNTY,+HYDROLOGIC+ΔΔΔUNIT,++TYPE,
    R(5)NAWDEX+AGENCY+CODE /C1,C22,C10,C12,C4 WH C22 LT 59000:
```

The following report results --

NAWDEX SITE REPORT
08/04/76

NAWDEX IDENTIFICATION NUMBER	STATE-COUNTY	HYDROLOGIC UNIT	TYPE	NAWDEX AGENCY CODE
***				
* USWCC15340102	06073	18100200	SW	USWCC
* TX8993714301	48419	12300050	GW	TX899
* TX89907358000	48423	14700200	LK	TX899

DATA COLLECTED BY USGS

STUDENT EXERCISE 2-5

(1) Which of the following partial commands correctly illustrates the way to combine a Format Statement option and a TITLE option in the same command:

a. LIST/DOUBLE SPACE/,/TITLE (23) NAWDEX_ID/C1 WH . . .:

b. LIST/TITLE/DOUBLE SPACE/L(23) NAWDEX_ID/C1 WH . . .:

c. LIST DOUBLE SPACE/TITLE L(23) NAWDEX_ID/C1 WH . . .:

d. LIST/DOUBLE SPACE,TITLE L(23) NAWDEX_ID/C1 WH . . .:

(2) If the following command is entered while working with the MWDITEST data base, what will be the titles above the three rightmost columns of the output?

LIST/TITLE R(5) NAWDEX_ID,B(12) EXTRA SPACE ,B(10)COMMENTS,

LATITUDE,R(10) SITE_TYPE,B(7)/C1,C2,C12,MAX C3,C22 WH . . .:

a. NAWDEX_ID, EXTRA SPACE, and COMMENTS

b. SITE_TYPE, MAX LONGITUDE, and STATE_COUNTY

c. blank, MAX LONGITUDE, and STATE_COUNTY

d. COMMENTS, LATITUDE, and SITE_TYPE

e. LATITUDE, SITE_TYPE, and blank.

(3) When a command is so long that it must be punched onto more than one card, distorted column titles may appear on the output if the blanks on the end of one card are added onto the first title on the next card. How may this error be prevented?

a. Always precede the first title on a card with a specification of column width.

b. Enter the command in several parts, with each part followed by a colon.

c. SYSTEM 2000 will not allow a command to be punched on more than one card.

d. Always follow the last title specification on a card with a comma.

(4) There is one error in the following command. What is it?

```
LIST/SINGLE SPACE,TITLE D(20) SITE SUMMARY REPORT,
F(50)ALL CURRENT ACTIVE SITES,R(23)++NAWDEX_ID,B(16),
L(20)ΔΔΔLATITUDEΔΔΔ+ΔIN DEGRΔΔ,R(15)+ΔΔLATITUDEΔ+,
R(15)+ΔΔLONGITUDEΔ,ΔB(10)++SPACING,R(3)HYDROL_UNIT/
C1,C4,C2,C2,LONGITUDE,C10, OB C17 WHERE C4 EQ USGS:
```

(5)* Print a report of NAWDEX_AGENCY, STATE_COUNTY, LATITUDE, and LONGITUDE for all sites in the MWDITEST data base. The title of the report should be SITE LOCATION REPORT and it should be centered over the rest of the report. Print LATEST DATA at the bottom of the report and set the page size at 60 lines. Use the following column titles --

SOURCE AGENCY	STATE COUNTY CODE	LATITUDE (DEG-MIN-SEC)	LONGITUDE (DEG-MIN-SEC)

*This exercise should be worked on the computer.

SECTION III. WORKING WITH MORE COMPLEX DATA BASES

 (page III-3 follows)

SECTION III. WORKING WITH MORE COMPLEX DATA BASES

III.A. THE STRUCTURE OF MORE COMPLEX DATA BASES - PART I

Thus far, the SYSTEM 2000 concepts and commands have been presented in a way to be compatible with two example data bases, WDSDTEST and MWDITEST. WDSDTEST contains 7 components in its definition and 3 data records. Each component represents a category of data that occurs a single time for each organization. There is only one NAWDEX_AGCY, one ORG_NAME, one NAWDEX_MBR and TYPE, and so forth, for any organization in the WDSDTEST data base. The next step is to show how SYSTEM 2000 accommodates the type of data that has multiple data items for an organization (or site, in the case of the MWDITEST data base).

First, consider only the WDSDTEST data base. Suppose that each organization in the data base has one or more branch offices which are repositories for water data. Someone who contacts the organization for water data will be directed to the branch office nearest the requestor. Therefore, it is necessary to store data on each of the organization's offices in WDSDTEST. For each office, the office name and address, NAWDEX office code, and telephone number are needed, along with the geographic area encompassed by the data available from the office, and the media used to store the data (publications, computer-readable, microfilm, etc.). If the office does not respond to public requests for data, that fact must also be noted.

The following components must be added to the WDSDTEST data base schema to maintain data on offices:

```
102* OFC_CODE (NAME X(4))
110* OFC_NAME (NON-KEY NAME X(15))
111* OFC_CONTACT (NON-KEY NAME X(15))
112* OFC_ST/POB (NON-KEY NAME X(15))
113* OFC_CITY (NON-KEY NAME X(10))
114* OFC_STATE_ABBR (NAME XX)
115* OFC_ZIP (NON-KEY NAME X(5))
116* OFC_COUNTRY_NAME (NON-KEY NAME X(4))
117* OFC_PHONE (NON-KEY NAME X(12))
120* OFC_AREA (NON-KEY NAME X)
121* OFC_REQUESTS (NON-KEY NAME X)
123* OFC_MEDIA (NON-KEY NAME X)
```

OFC_CODE is the NAWDEX identification code assigned to the office. OFC_NAME is the name of the office, and OFC_CONTACT is the title of the person to contact when requesting data. OFC_ST/POB through OFC_COUNTRY_NAME make up the office's address. OFC_PHONE is the number of the office, including the area code. OFC_AREA is a code designating the area covered by data available at the office (N = nationwide, M = multi-state, S = statewide, A = multi-county). OFC_REQUESTS is "Y" if the office responds to public requests for data and "N" if it does not. If the office does not respond to public requests for data, components OFC_CONTACT through OFC_AREA, and OFC_MEDIA, will be NULL. OFC_MEDIA is also a code (P = publication, G = microform and published, F = computer, published, and microform).

The above group of components defines the characteristics of data values stored for an office. It is different from the group consisting of component 1 through 9, because components 1 through 9 are for an organization and their data values can occur only once per organization. The data values for components 102 through 123 can occur many times per organization but only once for each office of the organization.

The components described above belong to a schema record[1]/(SR) because they are related to a common subject (office) and may have none, one, or multiple data occurrences per organization. A schema record must be distinguished in the data base schema from other components such as NAWDEX_AGCY and ORG_NAME that cannot occur more than once per organization. This is accomplished by the designation of a special component used to define the presence of a schema record. The component is given a name and number like any other component but it has neither a KEY or NON-KEY designation, a data type designation, nor a length of data field designation. In addition, the definition of each of the components within the office schema record contains one extra item of descriptive information that indicates the schema record in which the component is a member. This is the modifier "IN X" where X is the schema record identifier's component number. The complete schema for the schema record is as follows --

```
100* OFFICE (SR)
  102* OFC_CODE (NAME X(4) IN 100)
  110* OFC_NAME (NON-KEY NAME X(15) IN 100)
  111* OFC_CONTACT (NON-KEY NAME X(15) IN 100)
  112* OFC_ST/POB (NON-KEY NAME X(15) IN 100)
  113* OFC_CITY (NON-KEY NAME X(10) IN 100)
  114* OFC_STATE_ABBR (NAME XX IN 100)
  115* OFC_ZIP (NON-KEY NAME X(5) IN 100)
  116* OFC_COUNTRY_NAME (NAME X(4) IN 100)
  117* OFC_PHONE (NON-KEY NAME X(12) IN 100)
  120* OFC_AREA (NON-KEY NAME X IN 100)
  121* OFC_REQUESTS (NON-KEY NAME X IN 100)
  123* OFC_MEDIA (NON-KEY NAME X IN 100)
```

The components of the schema record are indented to set them off visually from the non-repeating components. The entire WDSDTEST data base schema, complete with the new schema record, is shown in figure 5 on page III-15.

Now consider the MWDITEST data base, which presently consists of data on sites at which water data are collected. The components thus far introduced have data items that occur only once per site, such as LATITUDE and SITE_TYPE. To fully describe the site, it is necessary to know what kinds of data are collected there.

For now, assume that the only type of data of any importance is water quality data, and for each MWDITEST site it is necessary to store data about any water quality monitoring program being conducted at the site. If such a program is or has been conducted at the site, the beginning year of the

1/Defined in glossary.

program and its purpose (research, assessment, legal, etc.) must be stored. It is also necessary to know whether or not the program is still active.

The following schema record must be added to the MWDITEST schema:

```
300*  QUALITY_WTR (SR)
   301*  QW_BEGIN_YR (NON-KEY INTEGER 9(4) IN 300)
   345*  QW_PURPOSE (NON-KEY NAME X(4) IN 300)
   350*  QW_ACTIVE (NAME X IN 300)
```

QW_PURPOSE is one or more codes indicating the purpose of the water quality monitoring program at the site. Up to four of the one-letter codes can be combined. For example C = compact or legal, and O = provides data necessary to current operation. If the monitoring program is conducted for both reasons, CO would be stored. Examples of other codes are: H = long-term hydrologic, and R = research or special study.

QW_ACTIVE contains a "Y" if the water quality monitoring program is currently active, and an "N" if the program was once active but is not at the present time. If no water quality monitoring program was ever conducted, there will be no occurrence of schema record (SR) 300 for the site.

The use of the QUALITY_WTR SR in MWDITEST is for a different purpose than the OFFICE SR in WDSDTEST. The OFFICE SR is used to store data that may occur multiple times per organization. The QUALITY_ WTR SR stores data that can occur only once per site, but for many sites it will not occur at all. If water quality was never monitored at a site, SR 300 will not exist for the site, and computer data storage space will be conserved.

The entire MWDITEST data base schema is shown in figure 6 on page III-16.

Note: Because of the addition of new components to WDSDTEST and MWDITEST, a new user password is necessary. Use the password LVL1 until notified differently (USER,LVL1:). The remainder of the problems in this manual cannot be solved if the password TEST is used.

One of the characteristics of schema records is that the data base schema does not have to be altered each time new data items are added. In the example WDSDTEST data base, data on offices may occur twice, one hundred times, or not at all, but the data base schema will remain the same.

On page I-11 a data record was defined as each single occurrence of a group of data items all related to the same subject. In the initial version of WDSDTEST all 7 components, and hence the data items for those components, were related to one subject, the organization. There were three organizations in the data base, each with its own set of data items, so there were three data records. The newest components just added to WDSDTEST also are related to the subject of an organization, but they are more immediately related to the subject of an office. Therefore, each occurrence of a group of data items in schema record 100 also is considered to be data record. Likewise, each

occurrence of a group of data items in schema record 300 of the MWDITEST data base is a data record.

Figures 7 and 8 on pages III-17 and III-18 illustrate the data records in the WDSDTEST and MWDITEST data bases, respectively. Data Records 4-9 in WDSDTEST are occurrences of the OFFICE SR, and Data Records 5-7 in MWDITEST are occurrences of the QUALITY_WTR SR.

The figures illustrate a hierarchial data base structure in which some data records are at a higher level than others. The data records that are occurrences of the OFFICE schema record are related to the organization data records but are considered to be one level lower, and are called schema. In the example, Data Records 4, 5, and 6 are siblings of Data Record 1; they may also be called children of Data Record 1. Data Record 8 is a sibling of Data Record 3. Conversely, Data Record 1 is called an ancestor data record[1] of Data Records 4, 5, and 6. Data Record 3 is a parent of Data Record 8 and Data Record 9.

Data Records 4, 5, and 6 may also be termed siblings of one another, since they are children of the same parent and at the same level. Data Records 8 and 9 are also siblings, and Data Record 7 has no siblings.

The parent-sibling relationship is a vertical relationship between data records because parent data records are at a higher level than their siblings. Vertically related data records belong to both a data subtree[1] and a data family[1].

A data subtree is defined as any data record at a given level plus all o its sibling data records. (This term will become more meaningful as the example data bases grow in complexity.) There are three data trees presently in the WDSDTEST data base, using the top level as a reference point:

1. Data Records 1, 4, 5, and 6
2. Data Records 2 and 7
3. Data Records 3, 8, and 9

There are four data trees in the MWDITEST data base:

1. Data Records 1 and 5
2. Data Record 2
3. Data Records 3 and 6
4. Data Records 4 and 7

Data Record 2 is included even though it has no siblings. The schema of a schema subtree holds for all data records at the referenced level, whether or not they have siblings.

Those data records that are vertically related as one branch of the hierarchical tree are collectively defined as a data family[1]. The families

[1]Defined in glossary.

in the WDSDTEST and MWDITEST data bases are:

WDSDTEST	MWDITEST
Data Records 1 and 4	Data Records 1 and 5
Data Records 1 and 5	Data Record 2
Data Records 1 and 6	Data Records 3 and 6
Data Records 2 and 7	Data Records 4 and 7
Data Records 3 and 8	
Data Records 3 and 9	

Data Records 4, 5, and 6 in WDSDTEST all belong to the same tree but each belongs to a different data family. Data record siblings from a common parent, if they belong to the same schema subtree but not to the same data family, are defined as discrete data records[1]. Therefore, Data Records 4, 5, and 6 are discrete data records because they are descended from a common parent, Data Record 1, but belong to different data families.

The concepts of data subtrees and data families are very important because they are a major factor in determining how SYSTEM 2000 data retrieval commands are processed. One additional term must be introduced and defined at this point before continuing with a discussion of descendant data records and levels. So far, the two example data bases have served to demonstrate the concepts and applications presented in this manual. The MWDITEST data base has been visualized as containing data for four sites and references were made to the data for the first site, or the second site, etc. In section B, when referring to all the data that existed for a single site, the term data record was used because at that point only a single data record existed per site. Since the concept of schema records has been introduced, it can be seen that a single site may have several data records at different levels. Referring to "all the data for the first site" is inadequate terminology because it is making reference to a specific water-data-related data base and a more generalized term is needed that can be used when referring to the structure of any data base. Therefore, the term Schema Entry[1] is defined as all of the data existing at all levels for one of the items of major or overall concern being stored in the data base. In the case of the MWDITEST data base, the major item of concern is a site. In the WDSDTEST data base the major item of concern is an organization. From here on this manual will refer to all the data for the first site or organization as Entry 1, all the data for the second site or organization as Entry 2, and so on. Figures 7 and 8 reflect this terminology. Note that the term Entry is synonymous with a data subtree with the parent data record in the top hierarchical level.

For ease of reference, the different levels in the data base hierarchy are numbered. The top level, which consists of all components for which data items occur only once per entry, is referred to as Level 0. Components 1 through 9 of WDSDTEST and component 1 through 22 of MWDITEST are at Level 0. The next lower level is Level 1, which consists of schema records immediately

[1]Defined in glossary.

descended from Level 0. The OFFICE schema record, consisting of components 102 through 123 is at Level 1. So is the QUALITY_WTR schema record. There even can be lower levels; this is discussed in later chapters.

The block diagrams of the two data bases (figures 7 and 8) show the data records and their values. An identification number is placed at the upper right-hand corner of each data record. This type of diagram of the data base is helpful in representing the relationships of the various data records with each other. However, when working with most data bases, this kind of diagram is not available, and, if there are very many entries and schema records in the data base it is infeasible to draw one. Nonetheless, it is useful to visualize such a diagram of all or parts of a data base from the information conveyed by this data base schema.

Notice that components 100 and 300, the schema record identifiers, are not included in the block diagrams. The block diagrams illustrate data records and their component values, and a schema record identifier cannot have a value; therefore it is not included.

STUDENT EXERCISE 3-1

(1) Components which are related to a common subject and may have multiple occurrences of data within an entry are called a:

a. Data Record

b. Schema Record

c. Related Component Set

d. Entry

(2) As shown in figure 7 on page III-17, Data Record 6 is a _____ of Data Record 1.

a. Schema Record

b. Parent

c. Data Item

d. Sibling

(3) Data Records 3 and 8 are part of the same _____.

a. Level

b. Data Item

c. Schema Record

d. Entry

(4) Data Records 8 and 9 belong to the same _____.

a. Level

b. Data Item

c. Schema Record

d. Entry

(5) Data Records 1 and 6 belong to the same _____.

a. Data Family

b. Level

c. Entry

d. Data Subtree

(6) Data Records 1, 4, and 6 belong to the same _____.

a. Data Family

b. Level

c. Schema Record

d. Data Subtree

(7) Disjoint data records are defined as _____.

(8) There are two errors in the following hypothetical data base definition. Find both of them.

```
 1* SITE-ID (KEY INTEGER 9(15))

 7* STATE (KEY INTEGER 99)

24* WATER-USE (NAME X)

29* WELL-LEVELS (KEY SR)

    40* DATE-MEAS (DATE IN 29)

    30* WHO-MEAS (NON-KEY NAME IN 29)

    35* WATER-LEVEL (KEY DECIMAL 9(5).99 IN 24)
```

Printing Data from Different Levels

The first part of this chapter discussed how SYSTEM 2000 selects data records, and several examples were given to demonstrate the selection process. No output examples were shown because data items extracted from the selected data records may be displayed in a number of ways, using either PRINT or LIST along with appropriate Format Statement options.

Two new Format Statement options are introduced here. They are --

(1) REPEAT
REPEAT SUPPRESS

(2) INDENT
BLOCK

As with the Format Statement options discussed in section II the option on top is the default option and one of the options in a pair is always in effect.

(1) When the REPEAT option is in effect, if data items from a data record and its schema sibling are output, then the values from the schema parent are printed each time the values from one of its schema sibling are printed. For example, the following command --

```
PRINT C1, C9, C112, C113 WHERE C6 NE A:
```

results in the following output from WDSDTEST:

```
1* USWCC
9* 04/12/1976
  112* 7415 KING ST
  113* SACRAMENTO
1* USWCC
9* 04/12/1976
  112* 1401 LAVACA
  113* AUSTIN
1* USWCC
9* 04/12/1976
  112*  3230 R ST NW
  113*  WASHINGTON
1* TX899
9* 06/19/1976
  112* 1234 EAST AVE
  113* DALLAS
1* TX899
9* 06/19/1976
  112* 5678 WEST AVE
  113* HOUSTON
```

Notice that the data items for components 1 and 9 are printed three times. Three data records containing data values for C112 and C113 were selected, and the values of C112 and C113 from each selected data record were printed, accompanied each time by the values of C1 and C9 from the schema parent. This is the effect of the REPEAT Format Statement option. If REPEAT SUPPRESS is specified, the values from the schema parent are printed only once, as follows --

```
PRINT/REPEAT SUPPRESS/C1,C9,C112, C113 WHERE C6 NE A:

    1* USWCC
    9* 04/12/1976
      112* 7415 KING ST
      113* SACRAMENTO
      112* 1401 LAVACA
      113* AUSTIN
      112* 3230 R ST NW
      113* WASHINGTON
    1* TX899
    9* 06/19/1976
      112* 1234 EAST AVE
      113* DALLAS
      112* 5678 WEST AVE
      113* HOUSTON
```

The REPEAT/REPEAT SUPPRESS options also perform the same function with the LIST command.

(2) As seen in the example output above, the data items from Level 1 are indented to the right of those data items from Level 0. The INDENT option causes output from successively lower levels to be indented two spaces each time a new level is reached. This makes it easier to distinguish among data items printed from different levels. The BLOCK option causes all data items to be left-justified, with no indentions. The INDENT/BLOCK options are not used with LIST.

STUDENT EXERCISE 3-2

(1) If the components specified in the Action Clause are at Level 0, then the component specified in the Conditional Clause may be --

a. Only at Level 0

b. Only at Level 1

c. Either Level 0 or Level 1

d. Both Level 0 and Level 1

(2)* Print a Report of NAWDEX Organization Code, the type of organization, the NAWDEX Office Code, and the storage media used by the office for up to to the first three offices whose primary orientation is Evaluation of Resources (Code E). Print the report in columnar format, using the the REPEAT SUPPRESS option. Assume you know nothing about the amount of data in the WDSDTEST data base.

(3)* Print the name of the contact and the phone number for those offices which have water quality data actively being collected at surface water sites in hydrologic unit 18100200. Also print the name of the parent organization of each office. Restrict your output to a maximum of two offices.

HINT: You will need to access both data bases to solve this problem. See appendix A for instructions on accessing more than one data base.

*These exercises should be worked on the computer. Remember that the password is now LVL1.

WDSDTEST AND MWDITEST DATA BASES
Data Base Schema
and
Data Item

For Use With Problems and Examples
In Section III.A.

```
  1*  NAWDEX_AGCY (NAME X(5))

  2*  ORG_NAME (NON-KEY NAME X(23))

  3*  NAWDEX_MBR (NON-KEY NAME X)

  4*  TYPE (NON-KEY NAME X)

  6*  ORIENTATION1 (NAME X)

  7*  ORIENTATION2 (NAME X)

  9*  LAST_UPDATE (NON-KEY DATE)

100*  OFFICE (SR)

  102*  OFC_CODE (NAME XXX IN 100)
  110*  OFC_NAME (NON-KEY NAME X(15) IN 100)

  111*  OFC_CNTACT (NON-KEY NAME X(15) IN 100)

  112*  OFC_ST/POB (NON-KEY NAME X(15) IN 100)

  113*  OFC_CITY (NON-KEY NAME X(10) IN 100)
  114*  OFC_STATE_ABBR (NAME XX IN 100)

  115*  OFC_ZIP (NON-KEY NAME X(5) IN 100)

  116*  OFC_COUNTRY_NAME (NON-KEY NAME XXXX IN 100)

  117*  OFC_PHONE (NON-KEY NAME X(12) IN 100)

  120*  OFC_AREA (NON-KEY NAME X IN 100)

  121*  OFC_REQUESTS (NON-KEY NAME X IN 100)

  123*  OFC_MEDIA (NON-KEY NAME X IN 100)


                User Password - LVL1
```

Figure 5.--WDSDTEST two-level schema.

```
 1*  NAWDEX_ID (NAME X(20))

 2*  LATITUDE (INTEGER NUMBER 9(6))

 3*  LONGITUDE (INTEGER NUMBER 9(7))

 4*  NAWDEX_AGCY (NAME X(5))

 7*  STATION_NAME (NON-KEY NAME X(30))

71*  NON_US_COUNTRY (NON-KEY NAME XX)

10*  HYDROL_UNIT (NON-KEY NAME X(8))

12*  SITE_TYPE (NAME XX)

17*  WDSD_OFC_CODE (NAME XXXX)

22*  STATE_COUNTY (INTEGER NUMBER 9(5))

  300*  QUALITY_WTR (SR)
    301*  QW_BEGIN_YR (NON-KEY INTEGER NUMBER 9999 IN 30

  345*  QW_PURPOSE (NON-KEY NAME XXXX IN 300)

  350*  QW_ACTIVE (NAME X IN 300)
```

User Password - LVL1

Figure 6.--MWDITEST two-level schema.

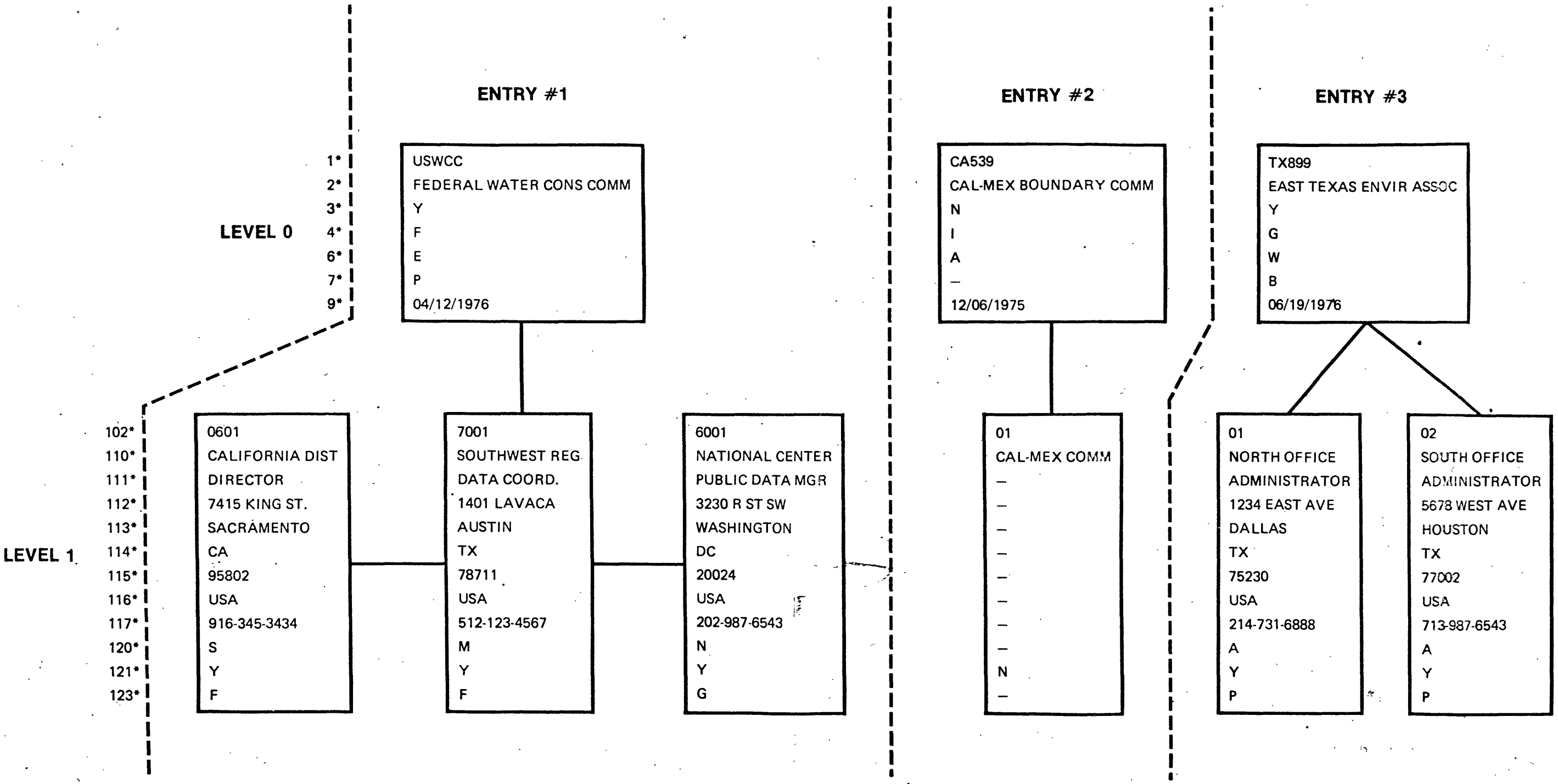


Figure 7.--WDSDTEST two-level structure.

III-17

81-419

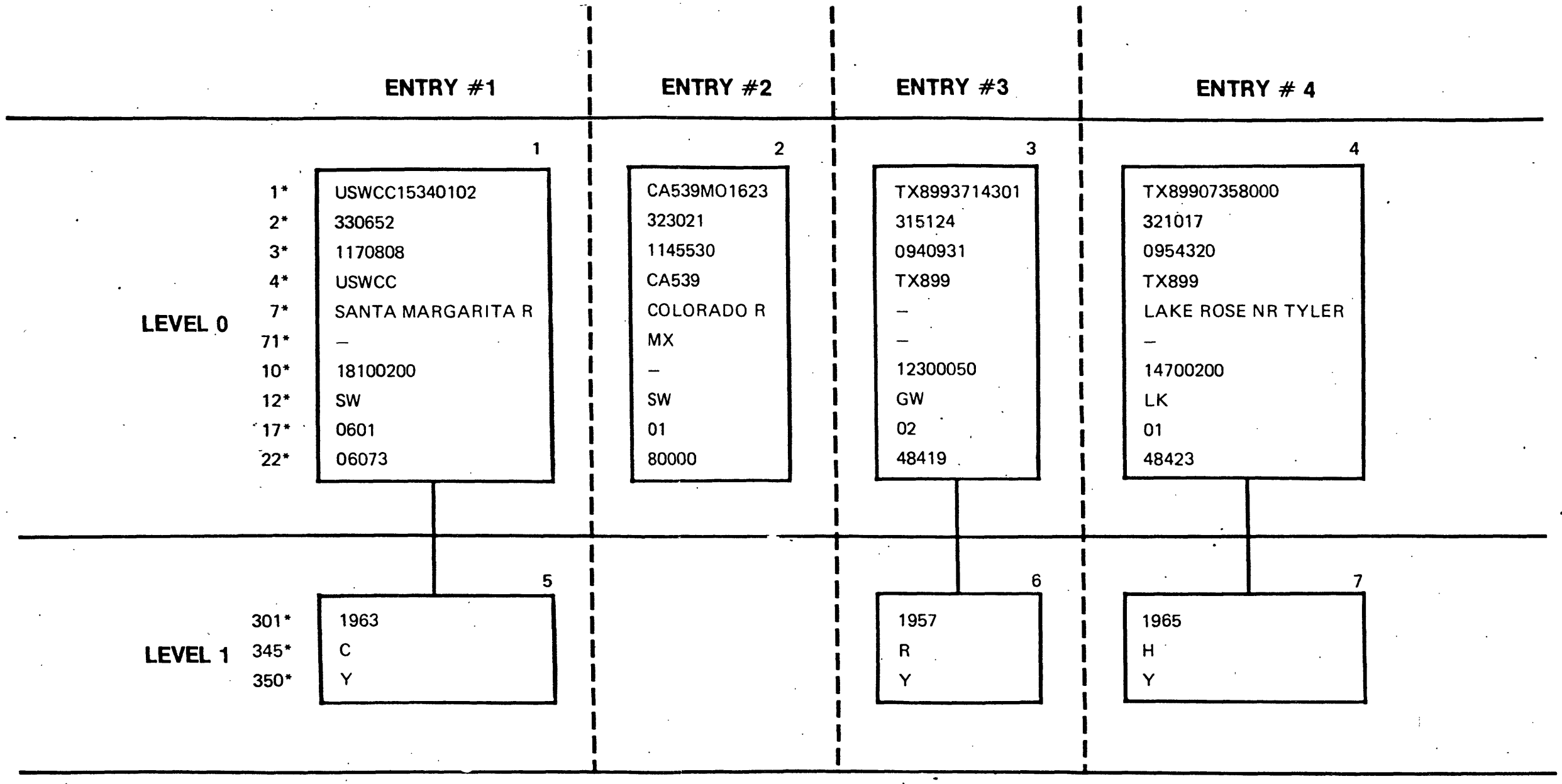


Figure 8.--MWDITEST two-level structure.

III.B. HOW TO RETRIEVE DATA FROM A DATA BASE WITH MULTIPLE-LEVELS

In section III.A the two example data bases were expanded to include a schema record and, therefore, another level. This simple data base structure served as an introduction to schema records and levels, but SYSTEM 2000 data bases are not restricted to one schema record or two levels. Very complex data bases may be created with large numbers of schema records and up to 32 levels. Multiple levels are created when schema records descend from other schema records.

In the WDSDTEST data base, an OFFICE schema record (SR) was defined to permit storing data on more than one office per organization. In order to provide a central clearinghouse for data requests, some offices are designated by NAWDEX as Assistance Centers. A need exists, therefore, to store additional data for any office that is a NAWDEX Assistance Center (AC), namely, its business hours and telephone number.

Assume that WDSDTEST is a new and growing data base that will eventually contain data on hundreds or maybe thousands of offices worldwide. Only a small portion of those offices are AC's. Therefore, to conserve space the AC data will not be stored in the OFFICE SR because few offices are AC's. Instead, AC data will be defined as a new SR sibling from the OFFICE SR. Its data records will occur only for ACs and in those cases will occur only once per office. To aid in the identification of which offices are ACs, a component will be added to the OFFICE SR that will be valued as 1 if the office is an AC and will be null otherwise. The definition of the new components is --

```
100* OFFICE (SR)
    102* OFC_CODE (____)
      .
      .
      .
    124* OFC_AC_FLAG (INTEGER 9 IN 100)
    125* OFC_AC (SR IN 100)
        126* OFC_AC_HRS (NON-KEY TEXT X(20) IN 125)
        127* OFC_AC_PHONE (NON-KEY NAME X(12) IN 125)
```

OFC_AC_HRS contains both the time zone and the business hours of the AC. The time zone is indicated by the number of hours away from Greenwich Mean Time (+06 = Central Standard Time, +08 = Pacific Standard Time, etc). OFC_AC_PHONE includes the area code.

The complete WDSDTEST data base schema is shown in figure 9 on page III-25, and an illustration of the data records is shown in figure 11 on page III-27. Notice that only the USWCC's California and Southwest offices are AC's.

In the MWDITEST data base, a QUALITY_WTR SR was defined to permit storing data on water quality monitoring programs that may exist at a

site. To provide more information about the monitoring program, the kind of water quality data collected should be included. One important kind of water quality data is physical water quality, including such parameters as temperature, specific conductivity, turbidity, color, and odor. Assume there is a need to store data on the frequency of measurement of the above five parameters in MWDITEST. Components for these parameters could be added to QUALITY_WTR, but because not all water quality monitoring programs collect physical water quality data, much space might be wasted. Therefore, a new PHYSICAL_QW SR will be added to MWDITEST as a sibling to QUALITY_WTR, and its data records will occur only for those sites at which physical water quality measurements are made.

The defintion of the new schema record is --

```
500* PHYSICAL_QW (SR IN 300)
   501* TEMPERATURE (NON-KEY NAME X IN 500)
   502* SPEC_CONDUCT (NON-KEY NAME X IN 500)
   503* TURBIDITY (NON-KEY NAME X IN 500)
   504* COLOR (NON-KEY NAME X IN 500)
   505* ODOR (NON-KEY NAME X IN 500)
```

A coding scheme is used to indicate frequency of measurement, as follows: 0 = Daily, W = Weekly, M = Monthly, Q = Quarterly, S = Semi-annual, and Y = Seasonal.

The complete MWDITEST data base schema is shown in figure 10 on page III-26, and an illustration of the data records is shown in figure 12 on p III-28. Notice that physical water quality parameters are measured at three out of the four sites.

Note: To accommodate the new components, a new password is required, LVL2 (USER,LVL2:). Previous passwords TEST and LVL1 should no longer be used.

Printing Data from Different Levels

Printing data from data bases that have three or more levels is no different than from data bases with only two levels, and the effect of the Format Statement option pairs REPEAT/REPEAT SUPPRESS and INDENT/BLOCK is identical. For example, the command

```
PRINT/REPEAT SUPPRESS, INDENT/C9,C113,C126 WHERE C124 EQ 1:
```

produces the following output from WDSDTEST:

```
 9* 04/12/1976
   113* SACRAMENTO
     126* +08 8AM-5PM
   113* AUSTIN
     126* +06 8AM-4PM
```

Format Statement options which are useful when specifying schema records in the Action Clause are TREE and GROUP, with TREE being the default option. The format of a command using these options is

PRINT / [TREE / GROUP] / <schema record> WHERE <conditions exist>:

If the TREE option is in effect, a command to print a schema record displays every data item in the schema record plus every data item in all its sibling data records (the entire schema tree). If the GROUP option is in effect, a command to print a schema record displays every data item in the specified schema record only. Its sibling data records are not printed. These two options have no affect on the LIST command, because a schema record cannot be specified within the Action Clause of a LIST command. For example, the command

```
PRINT/TREE, REPEAT SUPPRESS/C100 WHERE C102 EQ 0601:
```

produces the following output from WDSDTEST:

```
102* 0601
110* CALIFORNIA DIST
111* DIRECTOR
112* 7415 KING ST.
113* SACRAMENTO
114* CA
115* 95802
116* USA
117* 916-345-3434
120* S
121* Y
123* F
124*   1

  126* +08 8AM-5PM
  127* 916-345-3499
```

The same command using the GROUP option instead of the TREE option produces the following output

```
102* 0601
110* CALIFORNIA DIST
111* DIRECTOR
112* 7415 KING ST.
113* SACRAMENTO
114* CA
115* 95802
116* USA
117* 916-345-3434
120* S
121* Y
123* F
124*   1
```

The System Function COUNT was introduced in section II as a means of counting the number of occurrences of a data item for a particular component. COUNT also is used to count the number of occurrences of a schema record. A schema record is counted if it contains at least one data item. For example, the command

```
PRINT COUNT C100:
```

produces the following output:

```
CNT 100* 6
```

The schema record defined by component 100 has six occurrences of data (DATA Records 4, 5, 6, 7, 8, and 9). Each data record contains data items for up to twelve components. But even if eleven out of every twelve data items in each data record are missing, or null, the result is the same.

EXAMPLE 3-1

Problem: For all offices that are Assistance Centers in California, print all the data in the OFFICE schema record. Then for those same offices, if any, print the parent organization's name and print the AC office hours and phone number.

Solution: The following commands and output illustrate one approach to the problem --

```
PRINT/GROUP/C100 WH C114 EQ CA AND C124 EQ 1:

     102* 0601
     110* CALIFORNIA DIST
     111* DIRECTOR
     112* 7415 KING ST
     113* SACRAMENTO
     114* CA
     115* 95802
     116* USA
     117* 916-345-3434
     120* S
     121* Y
     123* F
     124*  1

PRINT/REPEAT SUPPRESS/ C2,C125 WH C114 EQ CA AND C124 EQ 1:

     2* FEDERAL WATER CONS COMM

          126* +08 8AM-5PM
          127* 916-345-3499
```

STUDENT EXERCISE 3-3

(Use WDSDTEST and MWDITEST Data Bases in Figures 11 and 12)

(1) Which data records in the WDSDTEST data base are selected if the following command is entered —

PRINT C113 WHERE C114 EQ TX:

a, 1, 3, 5, 8, and 9

b. 5, 8, and 9

c. 5, 8, 9, and 11

d. 1, 3, 4, 5, 6, 8, and 9

e. None of the above

(2) Which data records are selected if the following command is entered while accessing the WDSDTEST data base --

LIST C100 WHERE C102 EQ 01:

a. 4, 5, 6, 7, and 8

b. 2 and 3

c. 2, 3, 7, and 8

d. 7 and 8

e. None of the above

(3)* Print a report of the name, latitude, and longitude of all sites at which turbidity is measured monthly and color is measured weekly. Can the data for those sites be obtained from an Assistance Center? If so, what is the AC name, business hours, and telephone number?

*This exercise should be worked on the computer. Remember that the password is now LVL2.

WDSDTEST and MWDITEST DATA BASES
Data Base Schemas
and
Data Items

For Use With Problems and Examples
In Section III.B.

```
  1*  NAWDEX_AGCY (NAME X(5))
  2*  ORG_NAME (NON-KEY NAME X(23))
  3*  NAWDEX_MBR (NON-KEY NAME X)
  4*  TYPE (NAME X)
  6*  ORIENTATION1 (NAME X)
  7*  ORIENTATION2 (NAME X)
  9*  LAST_UPDATE (NON-KEY DATE)
100*  OFFICE (SR)
  102*  OFC_CODE (NAME XXXX IN 100)
  110*  OFC_NAME (NON-KEY NAME X(15) IN 100)
  111*  OFC_CONTACT (NON-KEY NAME X(15) IN 100)
  112*  OFC_ST/POB (NON-KEY NAME X(15) IN 100)
  113*  OFC_CITY (NON-KEY NAME X(10) IN 100)
  114*  OFC_STATE_ABBR (NAME XX IN 100)
  115*  OFC_ZIP (NON-KEY NAME X(5) IN 100)
  116*  OFC_COUNTRY_NAME (NON-KEY NAME XXXX IN 100)
  117*  OFC_PHONE (NON-KEY NAME X(12) IN 100)
  120*  OFC_AREA (NON-KEY NAME X IN 100)
  121*  OFC_REQUESTS (NON-KEY NAME X IN 100)
  123*  OFC_MEDIA (NON-KEY NAME X IN 100)
  124*  OFC_AC_FLAG (INTEGER NUMBER 9 IN 100)
  125*  OFC_AC (SR IN 100)
    126*  OFC_AC_HRS (NON-KEY TEXT X(20) IN 125)
    127*  OFC_AC_PHONE (NON-KEY NAME X(12) IN 125)

                User Password - LVL2
```

Figure 9.--WDSDTEST three-level schema.

```
  1*  NAWDEX_ID (NAME X(20))
  2*  LATITUDE (INTEGER NUMBER 9(6))
  3*  LONGITUDE (INTEGER NUMBER 9(7))
  4*  NAWDEX_AGCY (NAME X(5))
  7*  STATION_NAME (NON-KEY NAME X(30))
 71*  NON_US_COUNTRY (NON-KEY NAME XX)
 10*  HYDROL_UNIT (NON-KEY NAME X (8))
 12*  SITE_TYPE (NAME XX)
 17*  WDSD_OFC_CODE (NAME XXXX)
 22*  STATE_COUNTY (INTEGER NUMBER 9(5))
300*  QUALITY_WTR (RG)
  301*  QW_BEGIN_YR (NON-KEY INTEGER NUMBER 9999 IN 300)
  345*  QW_PURPOSE (NON_KEY NAME XXXX IN 300)
  350*  QW_ACTIVE (NAME X IN 300)
  500*  PHYSICAL_QW (RG IN 300)
    501*  TEMPERATURE (NON-KEY NAME X IN 500)
    502*  SPEC_CONDUCT (NON-KEY NAME X IN 500)
    503*  TURBIDITY (NON-KEY NAME X IN 500)
    504*  COLOR (NON-KEY NAME X IN 500)
    505*  ODOR (NON-KEY NAME X IN 500)

              User Password - LVL2
```

Figure 10.--MWDITEST three-level schema.

81-419

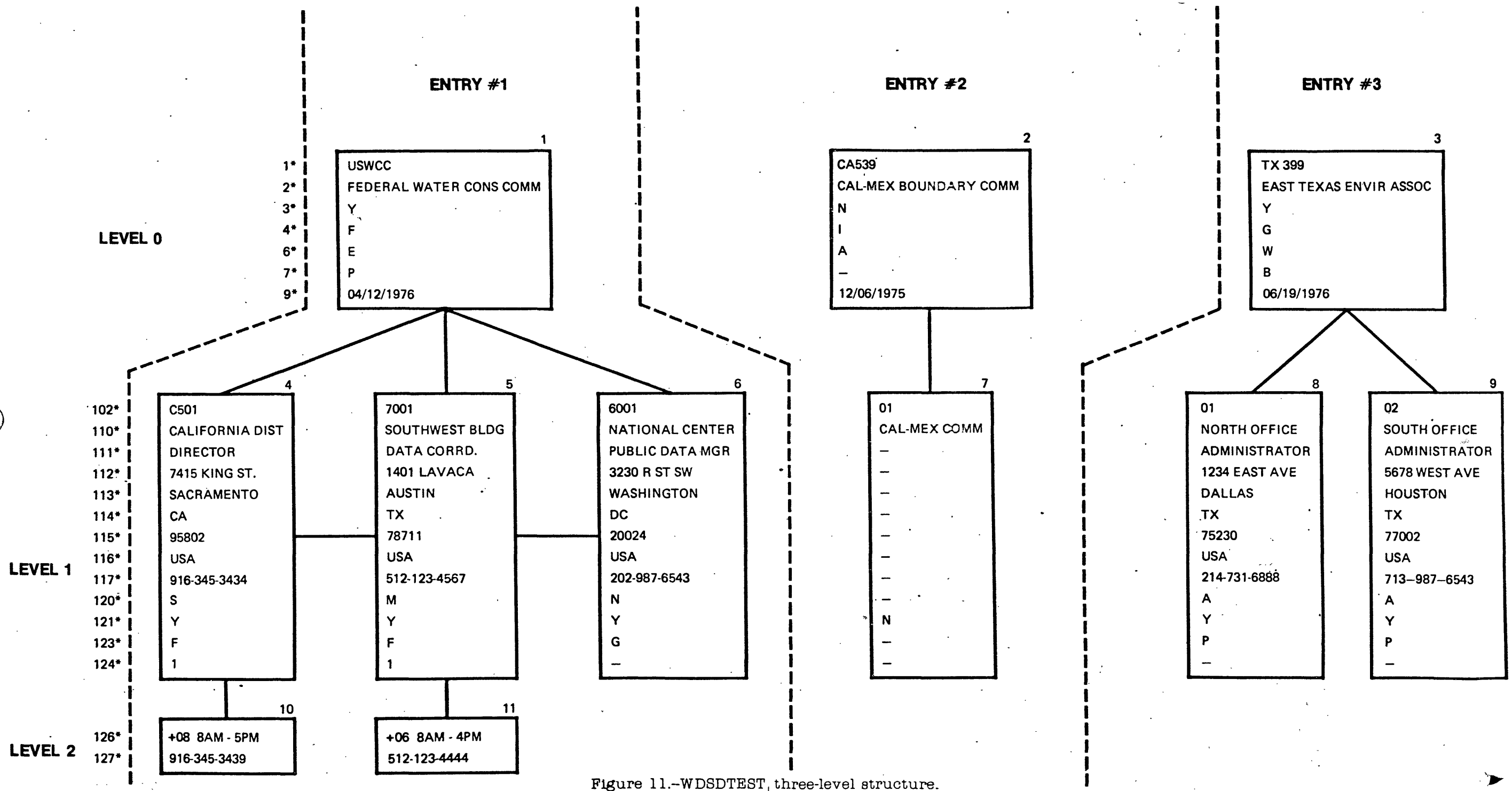


Figure 11.–WDSDTEST, three-level structure.

III-27

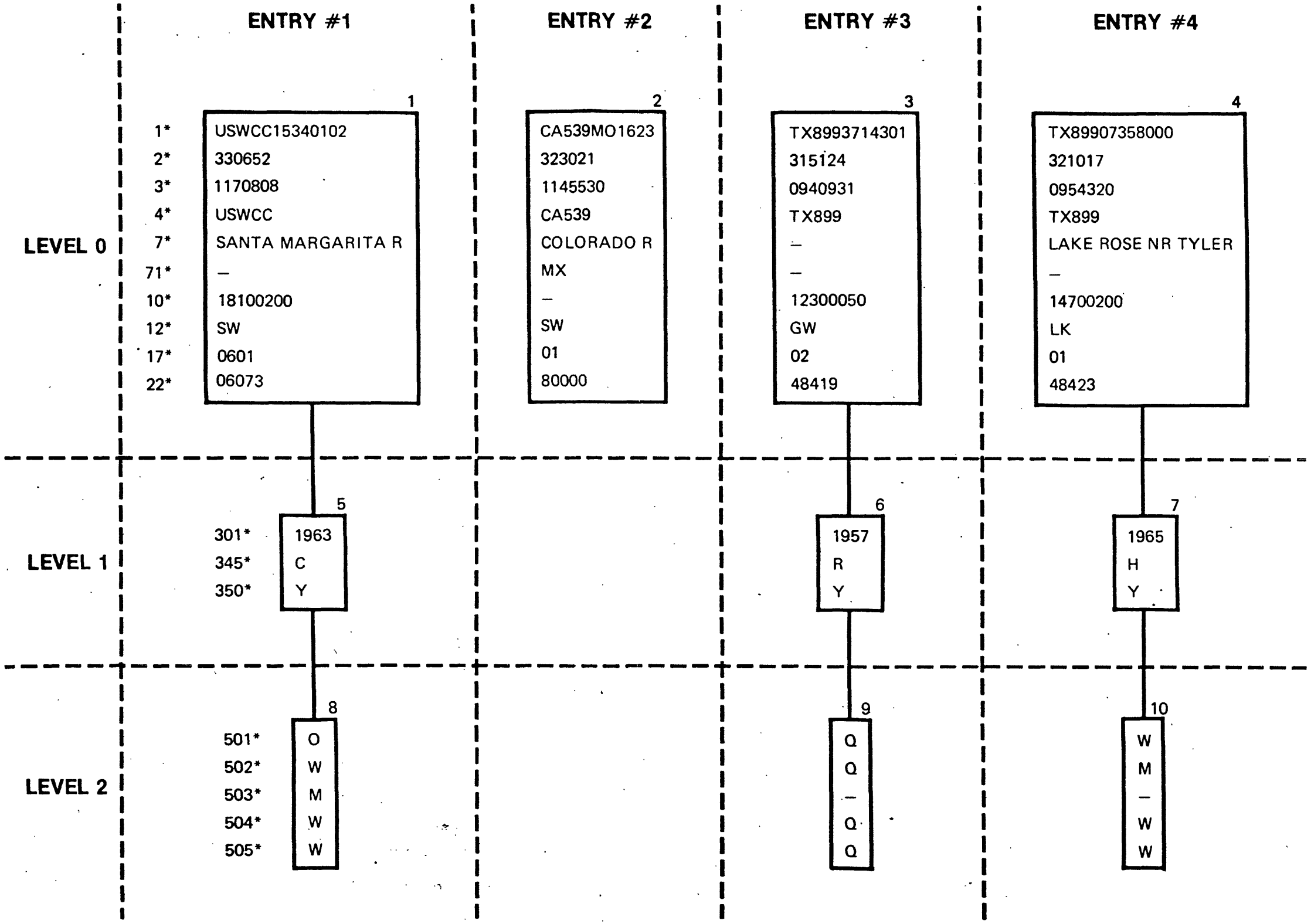


Figure 12.--MWDITEST three-level structure.

III-28

III.C. THE STRUCTURE OF MORE COMPLEX DATA BASES - PART II

The two example data base schemas (figures 9 and 10) currently have two schema records and three levels each. The schema records in WDSDTEST are related in that one is a sibling of the other and both contain data on offices. The schema records in MWDITEST are related in that one is a sibling of the other and both contain data on water quality.

Assume now that other kinds of data about each organization in WDSDTEST also must be stored in a schema record, and the data are unrelated to offices. The new data to be stored pertain to the type and amount of data stored by the organization as a whole. Specifically, it is the data stored by the organization by state (or country, if non-U.S. data are stored) and within each state or country the number of sites operated by the organization for each of the following kinds of data: surface water quality, surface water quantity, ground water quality, ground water levels, ground water pumpage, and the number of sites with geologic descriptions.

Because the new data are not related to any specific office but to the whole organization, the new schema record is not related to the OFFICE or OFC_AC repeating groups. The new SR exists alongside the OFFICE SR at Level 1, since it may occur multiple times per organization (one occurrence per state of stored data). The definition of the new schema record is --

```
400* ORGANIZATION_STORED_DATA (SR)
   401* NON_US_COUNTRY (NAME XX IN 400)
   402* STATE (INTEGER 999 IN 400)
   403* SW_QLTY (INTEGER 9(7) IN 400)
   404* SW_QNTY (INTEGER 9(7) IN 400)
   405* GW_QLTY (INTEGER 9(7) IN 400)
   406* GW_LVL (INTEGER 9(7) IN 400)
   407* GW_PUMP (INTEGER 9(7) IN 400)
   408* GLGC (INTEGER 9(7) IN 400)
```

If the data are for sites in a foreign country, the FIPS alphabetic abbreviation of the country appears in NON_US_COUNTRY and the FIPS numeric country code appears in STATE. If the sites are in the United States, NON_US_COUNTRY is null and the FIPS numeric state code appears in STATE. Each of the other components can contain an integer, which is the number of sites operated by the organization in the state or country. If no sites are operated, the component value is null.

There is a need for one more Level 1 schema record in WDSDTEST. It was stated earlier that only a few offices are designated as Assistance Centers, and that some offices will not respond to public request for data at all. In many instances an organization can provide requestors with an alternate source for data other than its own office. The alternate source may be just an additional source of the same data held by an office, or it may be the source that the organization prefers requestors to contact instead of its own offices.

Whenever an organization has one or more alternate sources of its data, information about those sources should be maintained in WDSDTEST. The information will be the same as is present for any office that responds to requests: NAWDEX code, name, contact, address, and storage media. The kind of data available from the source will also be included.

The definition of the new schema record is --

```
500* OTHER_SOURCE (SR)
   501* OT_SRC_ALT/PRF (NON-KEY NAME X IN 500)
   502* OT_SRC_AGCY (NAME X(5) IN 500)
   510* OT_SRC_NAME (NON-KEY NAME X(23) IN 500)
   511* OT_SRC_CONTACT (NON-KEY NAME X(15) IN 500)
   512* OT_SRC_ST/POB (NON-KEY NAME X(15) IN 500)
   513* OT_SRC_CITY (NON-KEY NAME X(10) IN 500)
   514* OT_SRC_STATE_ABBR (NON-KEY NAME XX IN 500)
   515* OT_SRC_ZIP (NON-KEY NAME X(5) IN 500)
   516* OT_SRC_COUNTRY_NAME (NON-KEY NAME XXXX IN 500)
   522* OT_SRC_DATA (NON-KEY NAME XXXX IN 500)
   523* OT_SRC_MEDIA (NON-KEY NAME X IN 500)
```

With the addition of the ORGANIZATION_STORED_DATA schema record, it is now possible to know the states and countries for which each organization has stored data. Because an organization might have offices in several states, it would be useful to know for which states and countries each office has stored data. Therefore, another schema record must be added as a sibling of the OFFICE SR. It is a separate SR because there may be multiple occurrences of data per office (one occurrence for each state or country).

The defintion of the new SR is --

```
200* OFFICE_STORED_DATA (SR IN 100)
   201* OFC_NON_US_COUNTRY (NAME XX IN 200)
   202* OFC_STATE (INTEGER 999 IN 200)
```

If the office has data for sites in a foreign country, the FIPS alphabetic abbreviation of the country appears in OFC_NON_US_COUNTRY and the FIPS numeric country code appears in OFC_STATE. If the office has data for sites in the U.S., OFC_NON_US_COUNTRY is null and the FIPS numeric state code appears in OFC_STATE.

The full WDSDTEST data base schema is shown in figure 15 on page IV-22 and an illustration of all the WDSDTEST data records is in figure 17 on page IV-24. Data Record occurrences of different schema records at the same level are enclosed in figures of different shapes to set them apart visually.

Figure 13 on page III-32 is a block diagram illustrating the relationships among the various WDSDTEST schema records in the data base schema. At the top of the block diagram is a group of components headed by ORGANIZATION, which is what a schema entry represents. Below the schema entry there are three branches, one headed by OFFICE with its sibling OFC_AC and OFFICE_STORED_DATA, one headed by ORGANIZATION_STORED_DATA, and one headed by

OTHER_SOURCE. The branches of the data base schema are called schema families[1]. A schema family is defined as those schema records vertically related as one single branch of the data base definition. The schema families in WDSDTEST are --

1. ENTRY, C100, C125
2. ENTRY, C100, C200
3. ENTRY, C400
4. ENTRY, C500

Schema records that are not in the same schema family are called discrete schema records[1]. OFFICE and ORGANIZATION_STORED_DATA are discrete schema records, as are OFFICE and OTHER_SOURCE, OFC_AC and OTHER_SOURCE, OFFICE_STORED_DATA and ORGANIZATION_STORED_DATA, etc. Any data base schema may have many discrete schema records, or just a few, or none at all. Discrete schema records can occur at any level.

Discrete schema records can be recognized in three ways:

(1) two or more schema records designated as (SR) are discrete schema records;

(2) two or more schema records designated as (SR IN nnn) where nnn is the same number are discrete schema records;

(3) All schema records that are descended from discrete schema records are discrete from each other.

Do not confuse "data family" with "schema family", or "discrete data record" with "discrete schema record". Schema families and discrete schema records are characteristics of the data base schema. Data families and discrete data records are characteristics of data stored in the data base. There are four schema families in the WDSDTEST data base; there are eighteen data families.

There is also a need to add discrete SRs to the MWDITEST data base schema. At present the only kind of data recognized in MWDITEST is water quality data. To be complete, the data base should also contain information about surface and ground water quantity data collected at sites. This will require two additional Level 1 schema records, because each type of data is related to the site but unrelated to the other types of data.

The surface water quantity SR should contain data on the beginning year of the monitoring program, the frequency of collection of flow data (if a river or stream), the media used to store flow data, the frequency of collection of volume data (if a lake or reservoir), and the media used to store volume data. Also, it is necessary to know if the monitoring program is still active.

[1]Defined in glossary.

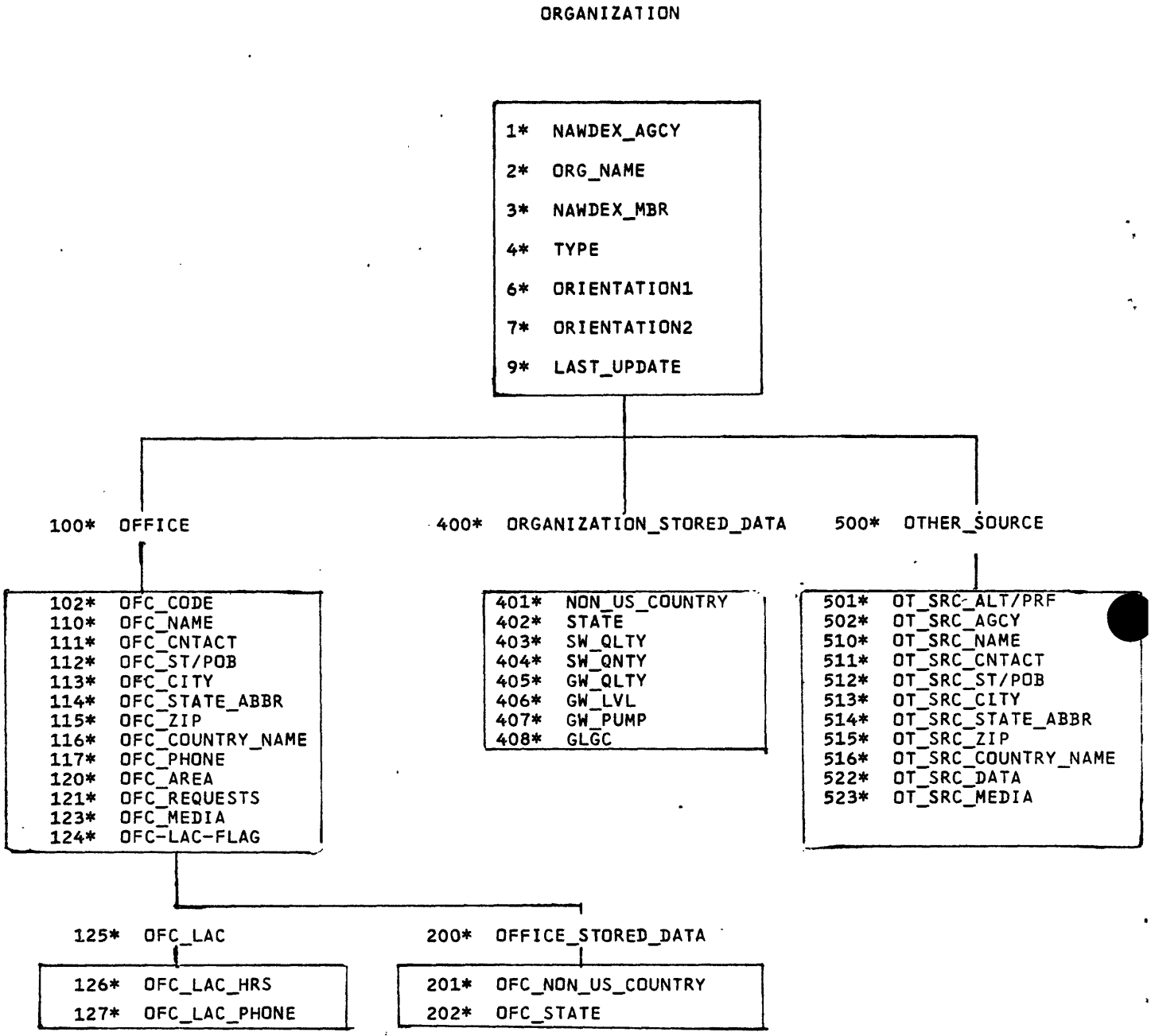


Figure 13.--Block diagram of WDSDTEST data base schema.

The ground water quantity SR should contain data on the beginning year of the monitoring program, the frequency of collection of water level data, the media used to store the level data, the depth of the well, and whether or not the monitoring program is still active.

The definition of the two new schema records is --

```
100* SURFACE_WTR (SR)
   101* SW_BEGIN_YR (NON-KEY INTEGER 9999 IN 100)
   115* COMPLETE_FLOW (NAME X IN 100)
   121* FLOW_MED (NON-KEY NAME X IN 100)
   124* VOLUME (NAME X IN 100)
   126* VOLUME MED (NON-KEY NAME X IN 100)
   150* SW ACTIVE (NAME X IN 100)
200* GROUND_WTR (SR)
   201* GW_BEGIN_YR (INTEGER 9999 IN 200)
   210* LEVEL_FREQ (NON-KEY NAME X IN 200)
   211* LEVEL_MED (NON-KEY NAME X IN 200)
   221* WELL_DEPTH (NON-KEY INTEGER 9(5) IN 200)
   250* GW ACTIVE (NAME X IN 200)
```

COMPLETE FLOW uses a coding system to indicate the frequency of flow measurements (1 = daily year round, 2 = daily seasonal, 3 = monthly year round, etc.). FLOW_MED, VOLUME_MED, and LEVEL_MED use the same coding system as OFC_MEDIA in the WDSDTEST data base to indicate the type of media used to store data (D = computer and published, P = published, etc.). SW_ACTIVE and GW_ACTIVE contain a "Y" if monitoring programs are currently active, and an "N" if the programs were once active but are not at the present time. If no surface water quantity or ground water level monitoring program was ever conducted, there will be no occurrence of SR 100 or SR 200, respectively.

In order to provide a more complete picture of the water quality monitoring program at sites, provision must be made in MWDITEST for more than just physical water quality parameters. Two new schema records should be added as siblings of SR 300, one containing information about the availability of sediment data, and the other containing information about the availability of chemical water quality data. Included will be the frequency of measurement of five sediment parameters: bedload, concentration of suspended sediment, particle size of suspended sediment, particle size of bedload sediment, and total sediment discharge. Also included will be the frequency of measurement of five chemical parameters: dissolved solids, major ions, hardness, silica, and phosphorus.

SR 300 contains a component, QW_ACTIVE, which indicates whether or not a water quality monitoring program is currently active at the site. However, it does not indicate which parameters are being monitored--physical, sediment, and/or chemical. Therefore, a component will be added to each of the three schema records so that their status can be determined.

The definition of the three SRs is --

```
500* PHYSICAL_QW (SR IN 300)
   501* TEMPERATURE (NON-KEY NAME X IN 500)
   502* SPEC_CONDUCT (NON-KEY NAME X IN 500)
   503* TURBIDITY (NON-KEY NAME X IN 500)
   504* COLOR (NON-KEY NAME X IN 500)
   505* ODOR (NON-KEY NAME X IN 500)
   550* PHY_ACTIVE (NAME X IN 500)
600* SEDIMENT_QW (SR IN 300)
   601* BEDLOAD (NON-KEY NAME X IN 600)
   602* CNCNTRTN_SUS (NON-KEY NAME X IN 600)
   604* PART_SIZ_SUS (NON-KEY NAME X IN 600)
   607* SED_DIS_TOT (NON-KEY NAME X IN 600)
   650* SED_ACTIVE (NAME X IN 600)
700* CHEMICAL_QW (RG IN 300)
   701* SOLIDS_DIS (NON-KEY NAME X IN 700)
   702* MAJOR_IONS (NON-KEY NAME X IN 700)
   703* HARDNESS (NON-KEY NAME X IN 700)
   705* SILICA (NON-KEY NAME X IN 700)
   706* PHOSPHORUS (NON-KEY NAME X IN 700)
   750* CHM_ACTIVE (NAME X IN 700)
```

A coding scheme, previously described, is used to indicate the frequency of measurement of each parameter (W = weekly, M = monthly, etc.). The three activity parameters use the same coding scheme as SW_ACTIVE, GW_ACTIVE, and QW_ACTIVE.

The full MWDITEST data base schema is shown in figure 16 on page IV-23 and an illustration of all the MWDITEST data records is in figure 18 on page IV-25. Figure 14 is a block diagram of the MWDITEST data base schema.

Note: Again, another password is necessary after adding more components to the test data bases. Use the password USGS for the remainder of this manual and for all future accesses of WDSDTEST or MWDITEST.

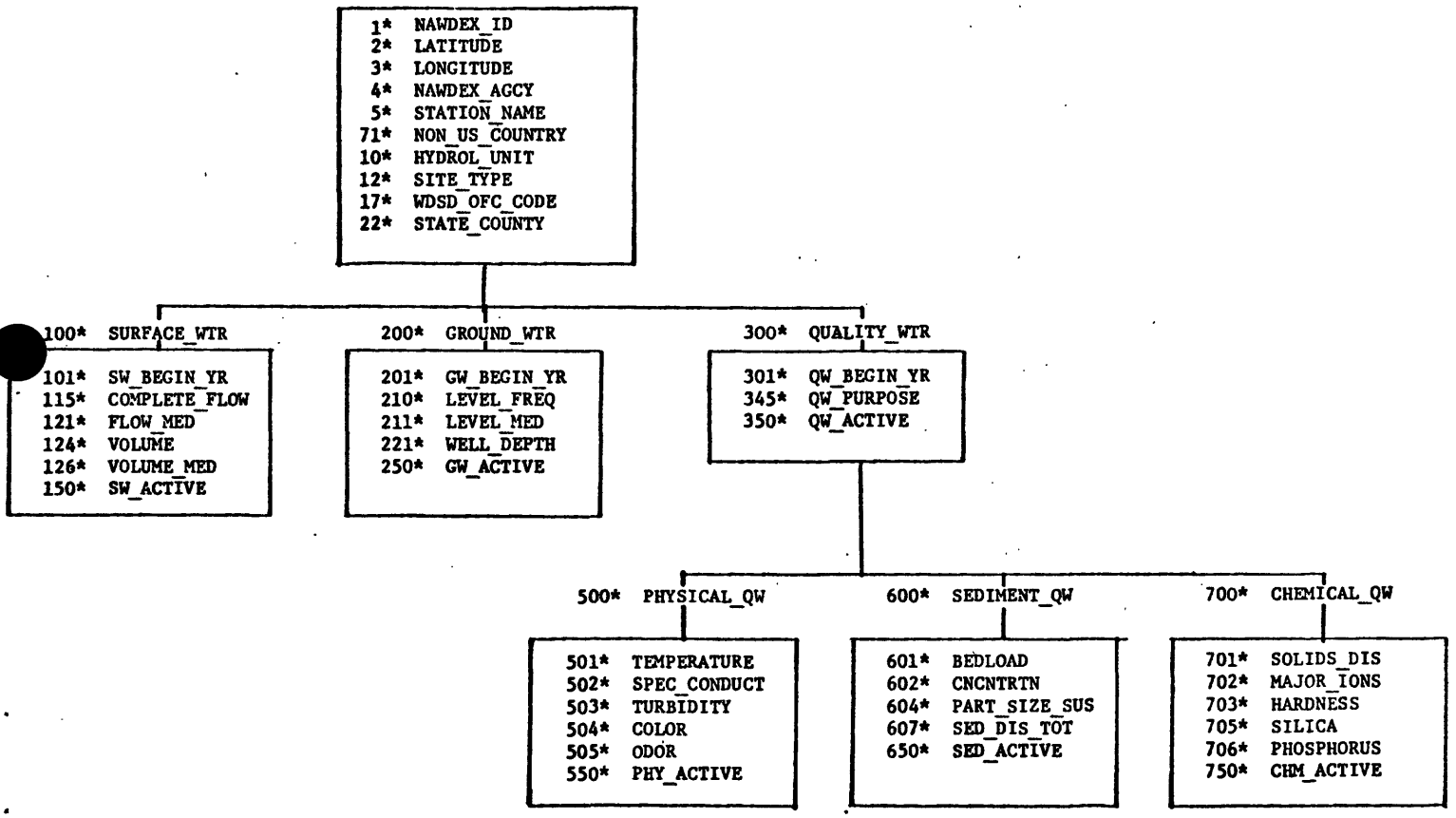


Figure 14.--Block diagram of MWDITEST data base schema.

III.D. HOW TO RETRIEVE DATA WHEN DISJOINT SCHEMA RECORDS EXIST

Method of Qualifying Data Records

In the chapters prior to this the Conditional Clause always qualified data records from the same schema family, because only a single schema family existed. Now a new schema family has been added, but the same principles of data record qualification still apply.

When disjoint schema are present it may be necessary to qualify data sets based on values that do not belong to the same schema family. For example, a user may want a report of the NAWDEX_ID of all sites that have currently active surface water quantity and quality monitoring programs. Retrieving the correct data requires the specification of components C150 and C350 within the same Conditional Clause, and those two components belong to different schema families. Because C150 and C350 belong to different schema families, their data items belong to different data families. Therefore, the Conditional Clause must qualify disjoint data records. Use of the HAS operator is required if disjoint data records are qualified. The HAS operator is used to raise the level of qualification to one common to both families. Since the data set families of components C150 and C350 both originate from Level 0, the level of qualification must be there. Therefore, the correct command is --

```
LIST C1 WHERE ENTRY HAS C150 EQ Y AND ENTRY HAS C350 EQ Y:
```

If one or more components in the Action Clause belong to a different schema family than one or more components in the Conditional Clause, the HAS operator must be used to raise the level of data record qualification up to a level common to both families.

For example, assume the user wants to know the NAWDEX_ID of all sites that have a surface water quantity monitoring program currently active, and for those sites also wants to know how often sediment parameters, if any, are measured. In this case there is a need to display data of one SR family based on the value of a component belonging to another SR family. The correct command is --

```
PRINT C1,C600 WHERE ENTRY HAS C150 EQ Y:
```

The above examples demonstrate two rules to follow when retrieving data from data bases having disjoint schema records.

1. The use of the HAS operator is mandatory if one or more components in the Action Clause are not in the same schema family as one or more components in the Conditional Clause.

2. The use of the HAS operator is mandatory if the Conditional Clause contains AND or OR operators that join conditions that do not belong to the same schema family.

3. HAS must be used to specify a common intersection of the schema families implied within a data retrieval command. The common intersection may occur at any level. The schema specified for the HAS operator may be the schema at the lowest level of intersection between two schema families or it may be a higher level schema ancestor schema of the common intersection.

EXAMPLE 3-2

Problem: Print a report of NAWDEX code and office name for all offices that belong to an organization that stores data for sites in California. Also print the organization NAWDEX code.

Solution: The command

```
LIST/REPEAT SUPPRESS/C1,C102,C110 WHERE ENTRY HAS C402 EQ 6:
```

produces the following output:

```
  NAWDEX AGCY      OFC_CODE     OFC NAME
***
* USWCC           0601         CALIFORNIA DIST
*                 70001        SOUTHWEST REG
*                 6001         NATIONAL CENTER
* CA539           01           CAL-MEX COMM
```

All three USWCC offices are listed even though two of them are not in California. This is because USWCC, the organization, stores California data, and the problem requires that all offices of the organization be printed.

Problem: Print a report from the MWDITEST data base of the NAWDEX-ID and beginning year for all sites that have a currently active chemical water quality monitoring program.

Solution: The command

```
LIST C1,C301 WHERE C750 EQ Y:
```

produces the following output --

```
  NAWDEX_ID                QW_BEGIN_YR
***
* USWCC15340102                  1963
* TX8993714301                   1957
```

There is no need for the HAS operator here because the components named in the command all belong to the same schema family and it is not necessary to raise the level of data record qualification.

Printing Data from Disjoint Schema Records

So far demonstrations have shown how commands must be structured when the Conditional Clause has conditions containing components from different schema families connected by AND or OR, and when components in the Action Clause are in a different schema family than components in the Conditional Clause. One more combination must be discussed -- what to do when components in the Action Clause belong to different schema families.

When printing components from disjoint schema records, the Action Clause must contain a "BY Clause". There are a number of ways that the BY Clause can be used in a command, and these are examined throughout the remainder of this manual. Several different command formats are explained, each demonstrating a specific use of the BY Clause.

Before showing how the BY Clause is used in a command, its format must first be explained. The BY Clause consists of the term BY followed by the component name or number of a schema record. One or more component names or numbers, which identify values to be output, follow the BY Clause and must belong to the schema record specified in the BY Clause or to one of its siblings. For example, if the BY Clause ... BY C200, C501, C601 ... appeared in a command to MWDITEST it would be rejected because components C501 and C601 do not belong to the C200 schema record. The BY Clause ... BY ENTRY, C501, C601 ... is acceptable because both C501 and C601 belong to schema records descended from the ENTRY repeating group.

The format of a command using a BY Clause is --

LIST BY Clause, components WHERE conditions exist :

For example, if the Action Clause of a command to MWDITEST is

PRINT C301, C501 ...

The specified data records are at Levels 1 and 2. But if the Action Clause is

PRINT BY ENTRY, C301, C501 ...

the specified data records are at Level 0 because the BY Clause overrides the levels of components C301 and C501.

A BY Clause overrides the level of every following implicitly or explicitly specified schema record. Components specified prior to the BY Clause are not affected by it. For example, if the Action Clause of a command to MWDITEST is --

PRINT C703, BY ENTRY, C1,C115,C345 ...

the specified data records are Data Records 1 through 4, and 16 and 17. Data Records 1 through 4 are specified because the BY ENTRY clause, which overrides

the schema records levels of C1,C115 and C345, raises the level of data record specification to Level 0. Data Records 16 and 17 are specified because of the presence of C703 which is not preempted by the BY Clause.

When components belonging to different schema families are used in an Action Clause, a BY Clause must also be present. The schema record specified in the BY Clause must be at or above a level common to all implicitly or explicitly identified schema records in the Action Clause.

For example, if a user wants to print a report of the site ID and frequency of measurement of the parameters specific conductivity, concentration of suspended sediment, and major ions for all surface water sites in hydrologic unit 18100200, the following command is required --

LIST C1, BY C300, C502,C602,C702 WH C10 EQ 18100200 AND C12 EQ SW:

The BY C300 Clause is necessary because the QUALITY_WTR SR is common to the schema families of C502,C602, and C702. Data Record 1 is the qualified data record. The specified data records are 1-4 (because of C1 in the Action Clause) and 5-7 (because of BY C300). Data Record 5 is selected because it is the lowest level specified data record in a qualified data family. Then C1 is printed from its ancestor, Data Record 1, and C502,C602, and C702 are printed from its sibling, Data Records 8, 15, and 16.

Because the level of the specified data sets is raised to the level of the schema record explicitly identified in the BY Clause, the BY Clause must be used carefully so that excess unwanted output is not produced.

For example, assume that the user wants to print a report of names of USWCC offices in Texas. The correct command to WDSDTEST is --

LIST C110 WHERE C114 EQ TX AND C1 EQ USWCC:

Data Record 5 is the only data record qualified by the Conditional Clause, and it is selected and its value of C110 is extracted and printed.

Next, assume that the user wants the same report but also wants to include the names of any alternate sources of data in the report. Because items from different schema families are being printed, a BY Clause is included in the command, as follows --

LIST BY ENTRY, C110, C510 WHERE C114 EQ TX AND C1 EQ USWCC:

Without the BY Clause, the components named in the Action Clause produce specified data records at Level 1. With the BY Clause, however, the specified data records are at Level 0, so Data Records 1, 2, 3, and 4 are the specified data records. Again, Data Record 5 is the qualified data record. The lowest level specified data record belonging to a data family containing a qualified data record is Data Record 1, so it is selected. The values for C110 and C510 are extracted and printed from the siblings of Data Record 1, Data Records 4,

5, 6, and 15. The BY ENTRY clause has caused excessive unwanted data to be printed, namely, data items for Data Records 4 and 6, which contain data for non-Texas offices.

The only way to avoid such a problem is to refrain in the Action Clause from naming components that belong to discrete schema records. Otherwise, the ability to be selective at lower levels of the data base hierarchy is lost.

STUDENT EXERCISE 3-4

(1) How many schema families exist in the following data base definition:

```
1*  A (KEY NAME)
2*  B (NON-KEY INTEGER 99)
3*  C (SR)
    4*  D (INTEGER IN 3)
    5*  E (NON-KEY TEXT IN 3)
    6*  F (SR IN 3)
        7*  G (INTEGER 999 IN 6)
        8*  H (DECIMAL 9(4).99 IN 6)
    9*  I (SR IN 3)
       10*  J (NON-KEY INTEGER IN 9)
11* K (SR)
    12*  L (TEXT IN 11)
    13*  M (KEY INTEGER 99 IN 11)
```

a. One

b. Two

c. Three

d. Four

e. Five

(2) Using the above data base schema as a guide, write a command to print a report of data from components C7 and C10 where component C8 has a value greater than 15 and where component C13 has a value less than 15.

(3) In the above data base schema, the schema record defined by component C9 is _____ from the schema record defined by component C3.

a. Once removed

b. Disjoint

c. Normalized

d. Sibling

(4) In the above data base schema, the schema record defined by component C6 is _____ from the schema record defined by component C11.

a. Once removed

b. Disjointed

c. Normalized

d. Siblings

(5) In the MWDITEST data base in figure 18 on page IV-25, data records 7 and 16 are _____.

a. In the same entry

b. Sibling

c. In the same schema family

d. At the same level

(6)* Find out if any MWDITEST sites have disolved solids currently being measured monthly (code M). For those sites, determine if there are also surface water quantity and ground water levels monitoring programs currently active, and the frequency of measurement of streamflow, volume, or water levels. Do this with a single command. Remember than USGS is now the password.

* This exercise should be worked on the computer.

(7) Which data records in the WDSDTEST data base are selected if the following command is entered --

LIST C113 WHERE ENTRY HAS C114 EQ TK:

a. 5, 8, 9

b. 4, 5, 6, 8, and 9

c. 1, 3, 4, 5, 6, 8 and 9

d. 1 and 3

e. None of the above

(8) Which data records are selected if the following command is entered while accessing the MWDITEST data base --

PRINT C345 WHERE ENTRY HAS C505 EQ W AND C350 EQ Y:

a. 5 and 7

b. 1 and 4

c. 8 and 10

d. all of the above

e. none of the above

SYSTEM 2000 RETRIEVAL MANUAL

SECTION IV. HOW TO STRUCTURE EFFICIENT DATA RETRIEVAL COMMANDS

SECTION IV. HOW TO STRUCTURE EFFICIENT DATA RETRIEVAL COMMANDS

IV.A. HOW TO AVOID REPETITIOUS COMMANDS

Conditional Clause

When working with SYSTEM 2000 in the batch mode, it may frequently occur that the user must enter a series of similar commands. It may be that successive commands that use the same qualified data records are required. If the user enters a series of commands that have the same Conditional Clause, doing so is redundant, time-consuming, and increases the chance of error. For example, if the user wants several reports containing different kinds of data for a specific site, many commands containing the following Conditional Clause might be entered --

```
... WHERE C1 EQ USWCC15340102 AND C301 GT 1960 AND C501 EQ 0:
```

When punching the commands in cards, the job is tedious and the chances of punching such a long Conditional Clause a number of times without error are probably small. To alleviate this problem, the SAME operator provides a means of duplicating a previous Conditional Clause without having to reenter it. The format of the SAME operator is --

```
<Action Clause> WHERE SAME:
```

There are several ways that the SAME operator may be used, and the format shown above is the most basic. When a command using SAME in this format is entered, a previous Conditional Clause is repeated, and therefore the same qualified data records are used. The SAME operator is an efficient command because the process of qualifying data records is not repeated.

The SAME operator can also be used as just one part of a Conditional Clause in order to expand on the conditions in the previous command. The SAME operator may precede or follow the other conditions, as shown in the formats below –

```
<Action Clause> WHERE SAME [AND|OR] <conditions exist>:

<Action Clause> WHERE <conditions exist> [AND|OR] SAME:
```

There are two processing modes for the SAME operator, STATIC and DYNAMIC, and SAME has a different effect under each mode. To set mode to DYNAMIC, enter

```
SAME IS DYNAMIC:
```

To set mode to STATIC, enter

```
SAME IS STATIC:
```

If neither of these commands is entered, SYSTEM 2000 assumes DYNAMIC mode is in effect.

In DYNAMIC mode SAME refers to the last Conditional Clause whether or not that Conditional Clause also has a SAME operator. A sequence of commands, some of which include SAME in the Conditional Clause, are shown below. On the left side are the actual commands entered, on the right side are equivalent commands that show the effect of the SAME operator.

Input Command	Equivalent Command (DYNAMIC MODE)
LIST C1 WH C150 EQ Y:	LIST C1 WH C150 EQ Y:
LIST C2,C3,C10 WH SAME:	LIST C2,C3,C10 WH C150 EQ Y:
LIST C1 WH C301 GT 1960:	LIST C1 WH C301 GT 1960:
LIST C4 WH SAME:	LIST C4 WH C301 GT 1960:
LIST C4,C17 WH SAME AND C10 EQ 18100200;	LIST C4,C17 WH C301 GT 1960 AND C10 EQ 18100200:
LIST C1 WH SAME AND C750 EQ Y:	LIST C1 WH C301 GT 1960 AND C10 EQ 18100200 AND C750 EQ Y:
LIST C501 WH ENTRY HAS C150 EQ Y:	LIST C501 WH ENTRY HAS C150 EQ Y:
LIST C501 WH C301 GT 1960 AND SAME:	LIST C501 WH C301 GT 1960 AND ENTRY HAS C150 EQ Y:

The above examples show how successive use of SAME allows the user to expand the Conditional Clause without reentering the previous conditions. The SAME operator also may be preceded by HAS if necessary. For example, if the Conditional Clause of one command contains a component in one schema family, and another condition belonging to another schema family must be added in the next command, HAS may be used to raise the level of data record qualification. This is illustrated by the following commands --

```
LIST C1 WH C101 GT 1955:

LIST C1 WH ENTRY HAS SAME AND ENTRY HAS C650 EQ Y:
```

In STATIC mode, SAME refers to the last Conditional Clause that did not contain a SAME operator or to the last Conditional Clause prior to issuing the SAME IS STATIC command, whichever is most recent. Consider the following commands --

(1) SAME IS STATIC:

(2) LIST C1 WH C150 EQ Y:

(3) LIST C1,C7 WH SAME AND C101 GT 1960:

(4) LIST C1,C7 WH SAME AND C124 EQ 1:

In both commands (3) and (4) SAME refers to the Conditional Clause in command (2). SAME IS STATIC mode is useful because it permits the user to repeat a condition several times, combining it with different conditions in each command.

A sequence of commands, some of which include SAME in the Conditional Clause, is given below. On the left side are the actual commands entered; on the right side are equivalent commands that show the effect of the SAME operator.

Input Command	Equivalent Command
SAME IS DYNAMIC:	SAME IS DYNAMIC
LIST C1 WH C12 EQ SW:	LIST C1 WH C12 EQ SW:
LIST C1 WH SAME AND C650 EQ Y:	LIST C1 WH C12 EQ SW AND C650 EQ Y:
SAME IS STATIC:	
LIST C4,C7 WH SAME	LIST C4,C7 WH C12 EQ SW AND C650 EQ Y:
LIST C1 WH C12 EQ SW:	LIST C1 WH C12 EQ SW:
LIST C1 WH SAME AND C650 EQ Y:	LIST C1 WH C12 EQ SW AND C650 EQ Y:
LIST C1 WH SAME AND C750 EQ Y:	LIST C1 WH C12 EQ SW AND C750 EQ Y:
LIST C4,C7 WH SAME:	LIST C4,C7 WH C12 EQ SW:
LIST C1 WH C350 EQ Y AND SAME:	LIST C1 WH C350 EQ Y AND C12 EQ SW:
SAME IS DYNAMIC:	
LIST C2 WH SAME:	LIST C2 WH C350 EQ Y AND C12 EQ SW:

Action Clause

The DITTO operator eliminates the need to repetitively type or punch the same Action Clause in successive commands. The format of DITTO operator is

DITTO WHERE <conditions exist>:

The use of DITTO in a command causes the entire Action Clause from the previous command to be repeated. Unlike the SAME operator, additional components cannot be added onto DITTO. For example, the following use of DITTO is incorrect:

```
PRINT C1,C4,C22 WHERE C350 EQ Y:

DITTO, C10 WHERE C350 EQ Y:
```

No other items may be placed before DITTO or between DITTO and WHERE. Both DITTO and SAME may be used in the same command, as illustrated by the following sequence of commands.

```
LIST C1 WHERE C12 EQ SW:

DITTO WHERE C12 EQ LK:

DITTO WHERE SAME:

DITTO WHERE SAME AND C350 EQ Y:
```

IV.B. HOW TO SPECIFY SEVERAL CONDITIONS IN A CONDITIONAL CLAUSE

Combination AND and OR

In the Conditional Clause of any command the terms AND and OR may be used to combine as many conditions as are necessary for the user to obtain the proper output. We have already learned how two or more conditions can be combined if each is separated by AND or each is separated by OR. But when OR and AND both must appear in the same Conditional Clause, the user must have an understanding of how SYSTEM 2000 processes the conditions. Conditions may be combined in various ways using AND and OR, and every combination might produce different results. For example, assume that the user wants to print the NAWDEX_ID of every site that meets either of the following conditions: (1) The site has both a current groundwater levels monitoring program and a current water quality monitoring program, or (2) The site has a current surface water quantity monitoring program. The following command is entered --

```
                                    Condition 1
LIST C1 WHERE ENTRY HAS C250 EQ Y AND ENTRY HAS C350 EQ Y
OR  ENTRY HAS C150 EQ Y:
    Condition 2
```

Now assume that the user wants to print the NAWDEX_ID of every site that meets both the following conditions: (1) The site has a current groundwater level monitoring program, and (2) the site has either a current water quality monitoring program or a current surface water quantity monitoring program. The resulting command is --

```
                        Condition 1
LIST C1 WHERE ENTRY HAS C250 EQ Y  AND
      ENTRY HAS C350 EQ Y OR ENTRY HAS C150 EQ Y:
                        Condition 2
```

This command and the previous command are identical even though they are supposed to provide answers to different problems. Obviously, the same command cannot be used to generate answers to two different problems, even though the problems are very similar and require the specification of the same three conditions for data retrieval. Only one of the commands is correct, but which one? The answer to this question can be provided by a study of how SYSTEM 2000 processes a Conditional Clause containing conditions joined by AND or OR. There are two rules for processing a Conditional Clause; after discussion of the rules, this chapter concentrates on how to use those rules to create proper data retrieval commands with AND or OR.

Rule 1: When a Conditional Clause contains both AND or OR, all the conditions linked by AND are processed first, followed by all the conditions linked by OR. For example:

LIST C1 WH <condition$_1$> AND <condition$_2$> OR <condition$_3$> AND <condition$_4$>:

SYSTEM 2000 processes in right-to-left order, and the AND conditions are processed first. Therefore, the data records that satisfy both <condition$_3$> and <condition$_4$> are located first. Then the data records that satisfy both <condition$_1$> and <condition$_2$> are located. Then the OR of the resulting data records is found.

Now the rule can be used to determine which of the two problems on page IV-7 can be solved by the command:

```
LIST C1 WHERE ENTRY HAS C250 EQ Y AND ENTRY HAS C350 EQ Y OR ENTRY HAS
     C150 EQ Y:
```

The conditions separated by AND are processed first, so all the data records that satisfy those two conditions are located. Data Record 3 is the only one that does. Next, the OR part of the Conditional Clause is processed, and as a result Data Records 1, 2, 3, and 4 become the qualified and selected data records. Their NAWDEX_ID'S are extracted and printed. The first of the previous two problems is solved by the above command because the two conditions separated by AND are processed first.

Rule 2 is: All conditions enclosed in parentheses are processed before all conditions not enclosed in parentheses. The second problem on page IV-7 is solved by enclosing the conditions in parentheses.

```
LIST C1 WHERE ENTRY HAS C250 EQ Y AND (ENTRY HAS C350 EQ Y OR ENTRY HAS
  C150 EQ Y):
```

The conditions separated by OR are processed first and Data Records 1-4 satisfy at least one of the two specified conditions. Then the AND of these data records and the ENTRY-level data records having C250 EQ Y is found, resulting in the qualified data records being Data Record 3. The selected data record is Data Record 3.

Sets of conditions enclosed within parentheses can be combined with other conditions by AND or OR and further enclosed with parentheses for combination with other conditions. The process of enclosing successive combinations of conditions within parentheses is known as nesting[1]. An example of a command with nested conditions is shown below:

```
LIST C1 WHERE (C12 EQ SW OR C12 EQ LK) AND (C350 EQ Y AND (C701 EQ M OR
  C702 EQ S)):
```

Because nested conditions are processed first, the data records that satisfy C701 EQ M OR C702 EQ S are located. These are Data Record 16 and 17 at Level 2. Next, the data records that satisfy the C350 EQ Y condition are located (Data Records 5, 6, and 7) and the AND of the two groups of data records is found to be Data Records 5 and 6. Then the data records that satisfy the conditions C12 EQ SW OR C12 EQ LK are found to be Data Records 1, 2, and 4. Their siblings at Level 1 are located and the remaining AND is processed, resulting in Data Record 5 becoming the qualified data record. Data Record 1 is selected, and its value of C1 extracted and printed.

[1]Defined in glossary.

Example 4-1

Problem: Print the name and NAWDEX organization code of all organizations that meet both of the following conditions.

Condition 1. Stores surface water quality data for at least 25 sites in Texas or at least 20 sites in New Mexico.

Condition 2. Has an office in New Mexico and Texas, or else has an office anywhere that is a Local Assistance Center.

Solution: Both Condition 1 and Condition 2 have subconditions that must be nested, as follows --

```
LIST C2,C1 WH ((C403 GE 25 AND C402 EQ 48) OR (C403 GE 20 AND
C402 EQ 25)) AND ((ENTRY HAS C114 EQ TX AND ENTRY HAS C114 EQ
NM) OR ENTRY HAS C124 EQ 1):
```

STUDENT EXERCISE 4-1

(1)* Assume that the following commands have been entered:

```
USER, USGS:

DBN IS MWDITEST:

LIMIT 0,10/TRUNCATE:

LIST C1 WHERE C12 EQ SW:

LIST C1, C2, C10 WHERE SAME AND C150 EQ Y:

SAME IS STATIC:

LIST C121 WH SAME:
```

Enter a command to print the NAWDEX_ID and name of all surface water sites that have both a currently active surface water quantity monitoring program and a currently active water quality monitoring program. Enter a following command to print the same data according to the same conditions except that a water quality monitoring program must never have existed at the site. Use DITTO and SAME wherever possible.

*This exercise should be worked on the computer.

Partial NON-KEY Conditional Clauses

It has been emphasized that the use of KEY components in the Conditional Clause is nearly always more efficient than the use of NON-KEY components (or KEY conditions using FAILS). The difference in efficiency between totally KEY and totally NON-KEY Conditional Clauses is so significant that SYSTEM 2000 allows the Data Base Administrator to prevent the use of NON-KEY Conditional Clauses when accessing a data base. If the DBA chooses to utilize this feature, and a user then enters a command with a totally NON-KEY Conditional Clause, the command is rejected with the message --

FULL DATA BASE PASS NOT ALLOWED

The primary usefulness of the NON-KEY Condition is when two or more conditions are in the Conditional Clause. A partial NON-KEY Conditional Clause, where some conditions are NON-KEY and at least one condition is KEY, might be more efficient than a conditional clause consisting entirely of KEY conditions. This is because of the way that partial NON-KEY Conditional Clauses are processed.

Recall that if all conditions are KEY, SYSTEM 2000 gathers a list of data records that satisfy each condition individually, and then combines the lists according to AND or OR rules. If the lists are large, combining them can take a considerable amount of time. Now consider how SYSTEM 2000 processes partial NON-KEY Conditional Clauses. If only one condition is KEY, a list of data records that satisfy the condition is gathered. If two or more KEY conditions are present, their lists are gathered and combined, just as before, ignoring the NON-KEY conditions. When all the KEY conditions are resolved, a single list of data records is the result. Next, SYSTEM 2000 makes a single pass through the data records in the list, inspecting each one to determine if it satisfies the NON-KEY conditions. The result is a list of data records that satisfy both the KEY and NON-KEY conditions.

Instead of gathering a list of data records for each NON-KEY condition and combining them, SYSTEM 2000 resolves all the NON-KEY conditions at once, operating only on the data records gathered by the KEY conditions.

In many cases, partial NON-KEY Conditional Clauses can be very inefficient. There is no sure way for the average SYSTEM 2000 user to estimate in advance the efficency of a command with a partial NON-KEY Conditional Clause. Therefore, it is advisable to use NON-KEY conditions only when necessary and otherwise to refrain from their use. If you find that it is a frequent requirement to use NON-KEY components for data retrieval, contact the Data Base Administrator and explain the situation. The DBA may be able to suggest a more efficient method of structuring data retrieval commands using the NON-KEY component, or may suggest an alternative way to solve the problem.

Other general guidelines for using partial NON-KEY Conditional Clauses are as follows:

(1) Place NON-KEY conditions to the left and KEY conditions at the right in the Conditional Clause.

(2) Order the NON-KEY conditions left-to-right according to the ascending level numbers of the components or schema records.

(3) Order the KEY conditions left-to-right by ascending level numbers of the components or schema records.

(4) Move any KEY condition likely to qualify no or very few data records to the rightmost position in the Conditional Clause.

IV.C. HOW TO USE PREVIOUSLY DEFINED COMMANDS AND FUNCTIONS

Stored Commands

Given any SYSTEM 2000 data base, certain types of data retrieval activities are carried out more often than others, and therefore some commands are used quite frequently. If it is possible to anticipate commands that are used often, these commands can be formulated in advance by the Data Base Administrator and included in the data base schema. Later, users can recall, or invoke, the command just by entering the component name or number of the command. Such a procedure can save time and reduce the chance of user error, especially if the command is long and complex. Commands set up in this manner are called Strings[1].

Two kinds of strings can be defined, Simple Strings[1] and Parametric Strings[1]. A Simple String is a command that always performs the same function and cannot be varied by the user in any way. A parametric String contains variables for which the user must supply values at the time the String is invoked, and the effect of the String depends on the variables supplied by the user.

Every String has a unique component number and name, just as any other component of the data base schema. Strings are defined by the Data Base Administrator and cannot be altered by the user, nor can the user add new Strings to the definition.

The user has the capability of displaying the Strings existing in the data base schema by invoking the following command -

```
DESCRIBE STRINGS:
```

This command causes every String in the data base schema to be printed. To print individual strings one at a time, enter

```
DESCRIBE <String number>:
```

or `DESCRIBE <String number 1> THRU <String number 2>:`

Two Strings that are part of the WDSDTEST definition are --

1. `2000* AC_REPORT (STRING? LIST/REPEAT SUPPRESS,NULL, TITLE R(6)NAWDEX+AGENCY+ CODE,L(15)++OFFICE NAME,L(15)++CITY, R(5)++STATE,L(20)+AC+HOURS/C1,C110,C113,C114,C126 WHERE C124 EQ 1:?)`

2. `2001* NO_OF_SITES (STRING? PRINT C1, *1* WHERE C1 EQ *2* AND C402 EQ *3*:?)`

String 1 is a Simple String. Its component number is 2000 and its name is AC_REPORT. The String is enclosed within parentheses and preceded

[1] Defined in glossary.

by the word STRING, with a special character (delimiter) placed immediately prior to and immediately following the String. SYSTEM 2000 requires that Strings be set off by some special character. In this case question marks are the delimiters, although other special characters could be used instead, as long as those characters do not also appear within the String.

String 2 is a Parametric String. Its component number is 2001 and its name is NO_OF_SITES. Question marks are again used to set off the String. As an Extended String, it is designed to accept user-supplied values when it is invoked. In this case, the String is set up to accept three values from the user to fill the places denoted by *1*, *2*, and *3*. An integer enclosed within asterisks in a String definition indicates where a value is to be supplied by the user when the String is invoked. From looking at the command within String 2, it is evident that a component name or number must be supplied for *1*. For *2*, some value must be supplied against which values of C1 in the data base will be compared as a basis for qualifying data records. For *3*, a data value must be supplied for comparison with values in the data base for C402.

There is also a Parametric String in MWDITEST, as shown below --

```
3000*  FREQ_OF_MEAS(STRING? PRINT C1,C301, *2*, *3* WHERE C1 EQ *1*:?)
```

Component names or numbers must be supplied for *2* and *3*, and for *1* a NAWDEX SITE_ID must be supplied. The significance of placing *2* and *3* before *1* in the String is explained later.

To invoke a Simple String, an asterisk followed by either a C and the String number or the String name, followed by another asterisk, with no intervening spaces is issued. If either of the following two commands is issued, the command in WDSDTEST String 1 is executed exactly as if the entire command itself has been given explicitly --

```
*C2000*:
```

```
*AC_REPORT*:
```

The output from C2000 is as follows --

```
  NAWDEX
  AGENCY                                                  AC
    CODE   OFFICE NAME        CITY                 STATE  HOURS
***
* USWCC    CALIFORNIA DIST    SACRAMENTO           CA     +08 8AM-5PM
*          SOUTHWEST REG      AUSTIN               TX     +06 8AM-4PM
```

To invoke a Parametric String, enter an asterisk followed by either a C and the String number or the String name, followed by the data items, enclosed in parentheses and separated by commas, that are required by the String. Always enter in the exact number of data values required, and in the correct format.

For example, WDSDTEST String 2 is a Parametric String requiring three data items. The first one must be a component name or number; the second must be a NAWDEX organization code; the third data item is compared with C402, which is a state code, so the user-supplied value must also be a state code. String 2 may be invoked by entering either of the following commands --

```
*C2001 (C404,CA539,6):

*NO_OF_SITES (C404,CA539, 6):
```

The items in parentheses are options and their values are dependent on the user's needed.

Entering either of the commands shown above is equivalent to entering --

```
PRINT C1, C404 WHERE C1 EQ CA539 AND C402 EQ 6:
```

and the output is:

```
1* CA539
404* 25
```

Instead of designating C404 as the first variable to be supplied to the String, the user could have specified a clause, such as AVG C404 BY ENTRY. Although this type of data item is acceptable, it is not acceptable to specify a data item that contains commas. For example, the following command --

```
*C2001 (AVG C404 BY ENTRY, CA539, 6):
```

is correct, but the next is not --

```
*C2001 (BY ENTRY, AVG C404, COUNT C405, CA539, 6):
```

This command is not correct because a comma can be used only as a separator between the user-supplied data items. SYSTEM 2000 interprets the above command as containing five data items. Furthermore, String 2 expects the first item to be a component name or number, the second to be five-character name field, and the third, a two-digit integer, but the first three items in the command are BY ENTRY, AVG C404, and COUNT C405, respectively. A syntax error is the result.

The value of using Strings should be clear. String 1 is lengthy and not only would it be quite time-consuming to repeatedly enter the equivalent command, but the chances of making an error when issuing it explicity are great. Also, especially in the case of a very complicated command, a command need only be constructed once, then stored as a String; the same lengthy process need not be repeated many times, with the risk of improperly structuring the command.

There is one feature of Parametric Strings that the user must take care to always remember: the integer number between the asterisks in a String definition represents the relative position of the required data value in the list of values supplied by the user. In other words, a *1* means that the first user-supplied value is inserted at that point, a *2* refers to the second user-supplied value, and so on. For example, consider the MWDITEST String --

```
3000* FREQ_OF_MEAS (STRING?PRINT C1, C301, *2*,*3* WHERE C1 EQ *1*:?)
```

In this case, when the String is invoked, NAWDEX SITE_ID supplied first, followed by two component numbers. A command to invoke the String might be --

```
*C3000 (TX8993714301, C501,C504):
```

When working with an unfamiliar data base, it is useful to print out the Strings in the data base schema to check if there are any that might be pertinent. Which are Simple Strings and which are Parametric should be noted. If any Parametric Strings are used, be certain to determine the proper order in which data items must be supplied.

STUDENT EXERCISE 4-2

(1) Which of the following commands for invoking Strings are correct? Look for syntax errors only - the commands have no relation to the WDSDTEST or MWDITEST data bases.

a. *C607*:

b. *C399*(15, 395.66, ABC, 01/12/1974):

c. *C700(AB,CD,EF,GH4):

d. *C895(XYZ, 3.2, 9500000)*:

e. *PRINT-IT*:

(2) Given the String --

```
699*REPORT (STRING$LIST BY ENTRY, C1, C110,*3*,OB *2*
 WHERE C6 EQ *4* AND C100 HAS C201 EQ *1*:$)
```

which of the following commands correctly invoke it? All components in the String refer to ones in the WDSDTEST data base.

a. *699*:

b. *699(50.00,C1,C102,F):

c. *699(80,C402,C402,E):

d. *699(75,C4,C127,R)

e. *699(111, C121, C123, USWCC)

f. *699(87, SUM C403 BY ENTRY, AVG C403 BY ENTRY, A):

(3)* Print the Strings in the WDSDTEST data base. Invoke the Simple String.

(4)* Using the Parametric String in the WDSDTEST data base, print a report of the number of surface water quantity sites in Mexico (State code = 80) for which organization CA539 has stored data.

(5)* Using the Parametric String in MWDITEST, print the frequency of measurement of silica and phosphorus at site USWCC15340102.

*These exercises should be worked on the computer.

Stored Functions

Some functions may be used very frequently in connection with a particular data base. In the same way that frequently-used commands can be stored in the schema as Strings, frequently used functions can be stored in the data base schema as Stored Functions[1]. Like Strings, Stored Functions cannot be altered by the individual user, nor can a user add new ones to the data base schema. These activities are performed only by the Data Base Administrator.

The user does have the ability to print the Stored Functions in the schema with the following command --

DESCRIBE FUNCTIONS:

This command causes every Stored Function in the data base schema to be printed. To print individual Stored Functions, enter DESCRIBE <function number>.

There are two kinds of Stored Functions, Simple and Parametric. They operate in the same manner as Simple and Parametric Strings, in that the user supplies data values to an Parametric Stored Function. The following Stored Function is part of the WDSDTEST data base definition:

2010* SITE_TOTAL (INTEGER FUNCTION ? (C403+C404+C405+C406+C407+C408)?)

The function is a Simple Stored Function. Its component number is 2010 and its name is SITE_TOTAL. The term INTEGER tells SYSTEM 2000 that the result of the mathematical calculation should be printed in INTEGER format (no decimal places), and the term FUNCTION designates that it is a Stored Function. The purpose of this function is to calculate the total number of sites for which an organization has stored data in a single state.

Stored Functions are used only in the Action Clause of a command; never in the Conditional Clause. A Simple Stored Function is called by specifying the component name or number of the function, enclosed between a pair of asterisks. Following are several example Action Clauses that illustrate how to call a Simple Stored Function --

PRINT *C2010* WHERE ...

LIST C1, C402, *C2010* WHERE ...

LIST C1, C402, *C2010*, C403, C404, OB C1 WHERE

LIST C1, BY ENTRY, MAX *C2010* WHERE ...

The last example shows a legal syntax construction in which a Stored Function is used with a System Function.

The following command --

PRINT/ZERO/C1,C402,*C2010* WHERE C1 EQ TX899 AND C402 EQ 48:

[1]Defined in glossary.

produces this output --

```
1*TX899
402* 48
2010* 135
```

A Parametric Stored Function is called by specifying an asterisk followed by the component name or number, in turn followed by the user-supplied data values enclosed in parentheses. No asterisk is placed at the end, as it is with a Simple Stored Function. Dates must be separated by dots instead of slashes because a slash is confused with a divide sign. Parametric Stored Functions can also be used with System Functions. Several examples of how hypothetical Parametric Stored Functions are called are shown below --

```
PRINT *C503(01.07.1975)  WHERE ...

LIST  *C666(28.15,25,17.52)  WHERE ...

LIST  C1, C60, *C777(12.20.1946,97.68,15)  WHERE ...

LIST  C1, BY ENTRY, MAX C60, *C503(07.15.1975)  WHERE ...
```

As in Parametric Strings, the user-supplied variables for a Parametric Stored Function must be in proper sequence according to the integers specified within the function. The following two Parametric Stored Functions are identical except for the order in which user-supplied values are given when the function is called --

```
666*  FUNCT-1  (INTEGER FUNCTION ((C84/*1*)-(C83/*2*)*(*3*))

666*  FUNCT-2  (INTEGER FUNCTION ((C84/*3*)-(C83/*1*)*(*2*))
```

Stored Functions may also be used within an ORDERED BY clause. For example, the Stored Function C2010 could be used in an ORDERED BY clause to sort the output data items according to the largest number of sites per organization. The command --

```
LIST/REPEAT SUPPRESS,ZERO/C1, C402, *C2010*, OB MAX *C2010*
  BY ENTRY WHERE C4 GT B AND C1 EXISTS:
```

produces the following output --

NAWDEX_AGCY	STATE	SITE_TOTAL
***		
* CA539	80	45
*	6	72
* USWCC	48	87
*	35	22
*	6	124
* TX899	48	135

STUDENT EXERCISE 4-3

(1) Which of the following commands using hypothetical Stored Functions are correct? Check for syntax errors only.

a. PRINT C4, *C789(15,100.00,12.12.1945), C24, OB C1, MIN *C789(10,5.75, 05.05.1955) BY ENTRY WHERE C4 EQ USGS:

b. PRINT C1,*C555(66.6, C16)*, C24 WHERE C4 EQ USGS:

c. LIST *USER-FUNCT(89.00), C83 WHERE SAME:

d. LIST C1, *600*, *601*, C16 WHERE SAME:

e. LIST C1, *C897(1,2,3), *C444*, C28, OB MAX *C555* BY ENTRY WHERE C59 GT *C333* AND C16 GE 150.00:

(2) **Print the Stored Function in the WDSDTEST data base by two different methods. Print the total number of sites in each state for which organization USWCC stores data, and print the name of any alternate sources of the data.

**This exercise should be worked on the computer.

WDSDTEST and MWDITEST DATA BASES

Data Base Schema

and

Data Items

For Use With Problems and Examples

In Section III.C through IV.C

```
  1*  NAWDEX AGCY (NAME X(5))
  2*  ORG_NAME (NON-KEY NAME X(23))
  3*  NAWDEX_MBR (NON-KEY NAME X)
  4*  TYPE (NAME X)
  6*  ORIENTATION1 (NAME X)
  7*  ORIENTATION2 (NAME X)
  9*  LAST_UPDATE (NON-KEY DATE)
100*  OFFICE (SR)
  102*  OFC_CODE (NAME XXXX IN 100)
  110*  OFC NAME (NON-KEY NAME X(15) IN 100)
  111*  OFC CONTACT (NON-KEY NAME X(15) IN 100)
  112*  OFC ST/POB (NON-KEY NAME X(15) IN 100)
  113*  OFC_CITY (NON-KEY NAME X(10) IN 100)
  114*  OFC STATE ABBR (NAME XX IN 100)
  115*  OFC_ZIP (NON-KEY NAME X(5) IN 100)
  116*  OFC COUNTRY NAME (NON-KEY NAME XXXX IN 100)
  117*  OFC_PHONE (NON-KEY NAME X(12) IN 100)
  120*  OFC AREA (NON-KEY NAME X IN 100)
  121*  OFC_REQUESTS (NON-KEY NAME X IN 100)
  123*  OFC MEDIA (NON-KEY NAME X IN 100)
  124*  OFC_AC_FLAG (INTEGER NUMBER 9 IN 100)
  125*  OFC AC (SR IN 100)
    126*  OFC_AC_HRS (NON-KEY TEXT X(20) IN 125)
    127*  OFC AC PHONE (NON-KEY NAME X(12) IN 125)
  200*  OFFICE_STORED DATA (SR IN 100)
    201*  OFC_NON US COUNTRY (NAME XX IN 200)
    202*  OFC STATE (INTEGER NUMBER 999 IN 200)
  400*  ORGANIZATION STORED DATA (SR)
    401*  NON_US_COUNTRY (NAME XX IN 400)
    402*  STATE (INTEGER NUMBER 999 IN 400)
    403*  SW_QLTY (INTEGER NUMBER 9(7) IN 400)
    404*  SW_QNTY (INTEGER NUMBER 9(7) IN 400)
    405*  GW_QLTY (INTEGER NUMBER 9(7) IN 400)
    406*  GW_LVL (INTEGER NUMBER 9(7) IN 400)
    407*  GW_PUMP (INTEGER NUMBER 9(7) IN 400)
    408*  GLGC (INTEGER NUMBER 9(7) IN 400)
  500*  OTHER SOURCE (SR)
    501*  OT SRC ALT/PRF (NAME X IN 500)
    502*  OT_SRC_AGCY (NAME X(5) IN 500)
    510*  OT SRC_NAME (NON-KEY NAME X(23) IN 500)
    511*  OT_SRC_CONTACT (NON-KEY NAME X(15) IN 500)
    512*  OT_SRC_ST/POB (NON-KEY NAME X(15) IN 500)
    513*  OT SRC CITY (NON-KEY NAME X(10) IN 500)
    514*  OT_SRC_STATE_ABBR (NAME XX IN 500)
    515*  OT SRC ZIP (NON-KEY NAME X(5) IN 500)
    516*  OT SRC COUNTRY NAME (NAME XXXX IN 500)
    522*  OT_SRC DATA (NON-KEY NAME XXXX IN 500)
    523*  OT SRC_MEDIA (NON-KEY NAME X IN 500)
```

Figure 15.--WDSDTEST data base complete schema.

```
  1*  NAWDEX_ID (NAME X(20))
  2*  LATITUDE (INTEGER NUMBER 9(6))
  3*  LONGITUDE (INTEGER NUMBER 9(7))
  4*  NAWDEX AGCY (NAME X(5))
  7*  STATION_NAME (NON-KEY NAME X(30))
 71*  NON US COUNTRY (NON-KEY NAME XX)
 10*  HYDROL UNIT (NON-KEY NAME X(8))
 12*  SITE TYPE (NAME XX)
 17*  WDSD_OFC_CODE (NAME XXXX)
 22*  STATE COUNTY (INTEGER NUMBER 9(5))
100*  SURFACE_WTR (SR)
  101*  SW_BEGIN_YR (INTEGER NUMBER 9999 IN 100)
  115*  COMPLETE_FLOW (NON-KEY NAME X IN 100)
  121*  FLOW_MED (NON-KEY NAME X IN 100)
  124*  VOLUME (NON-KEY NAME X IN 100)
  126*  VOLUME MED (NON-KEY NAME X IN 100)
  150*  SW_ACTIVE (NAME X IN 100)
200*  GROUND WTR (SR)
  201*  GW_BEGIN YR (INTEGER NUMBER 9999 IN 200)
  210*  LEVEL_FREQ (NON-KEY NAME X IN 200)
  211*  LEVEL_MED (NON-KEY NAME X IN 200)
  221*  WELL DEPTH (NON-KEY INTEGER NUMBER 9(5) IN 200)
  250*  GW_ACTIVE (NAME X IN 200)
300*  QUALITY_WTR (SR)
  301*  QW_BEGIN YR (NON-KEY INTEGER NUMBER 9999 IN 300)
  345*  QW PURPOSE (NON-KEY NAME XXXX IN 300)
  350*  QW_ACTIVE (NAME X IN 300)
  500*  PHYSICAL_QW (SR IN 300)
    501*  TEMPERATURE (NON-KEY NAME X IN 500)
    502*  SPEC CONDUCT (NON-KEY NAME X IN 500)
    503*  TURBIDITY (NON-KEY NAME X IN 500)
    504*  COLOR (NON-KEY NAME X IN 500)
    505*  ODOR (NON-KEY NAME X IN 500)
    550*  PHY ACTIVE (NAME X IN 500)
  600*  SEDIMENT_QW (SR IN 300)
    601*  BEDLOAD (NON-KEY NAME X IN 600)
    602*  CNCNTRTN_SUS (NON-KEY NAME X IN 600)
    604*  PART_SIZ_SUS (NON-KEY NAME X IN 600)
    607*  SED_DIS_TOT (NON-KEY NAME X IN 600)
    650*  SED_ACTIVE (NAME X IN 600)
  700*  CHEMICAL_QW (SR IN 300)
    701*  SOLIDS_DIS (NON-KEY NAME X IN 700)
    702*  MAJOR IONS (NON-KEY NAME X IN 700)
    703*  HARDNESS (NON-KEY NAME X IN 700)
    705*  SILICA (NON-KEY NAME X IN 700)
    706*  PHOSPHORUS (NON-KEY NAME X IN 700)
    750*  CHM_ACTIVE (NAME X IN 700)
```

Figure 16.--MWDITEST data base complete schema.

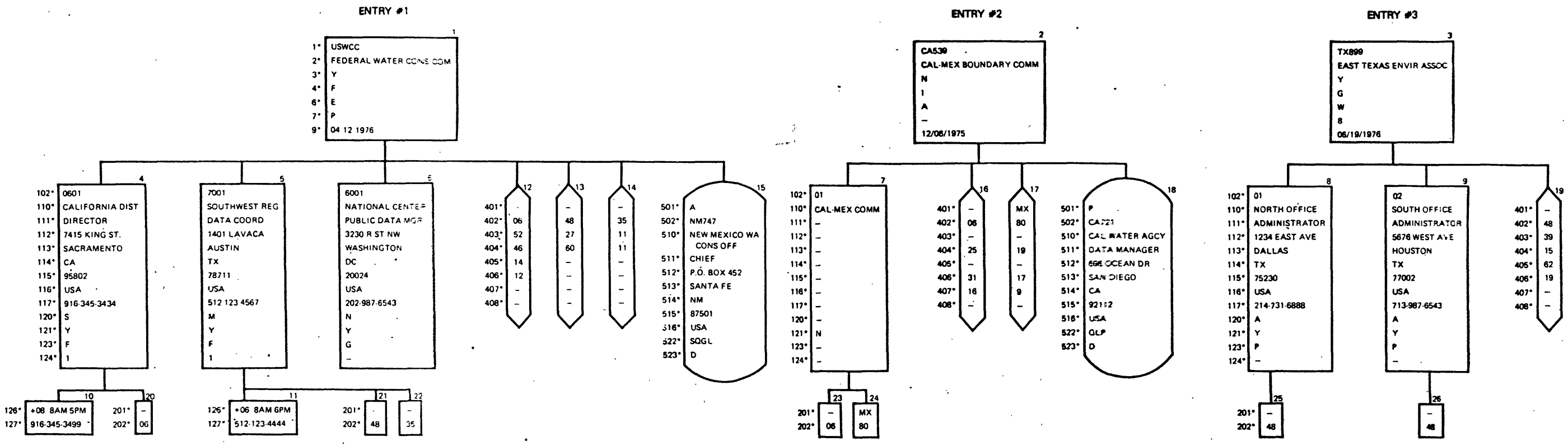


Figure 17.--WDSDTEST final data base structure

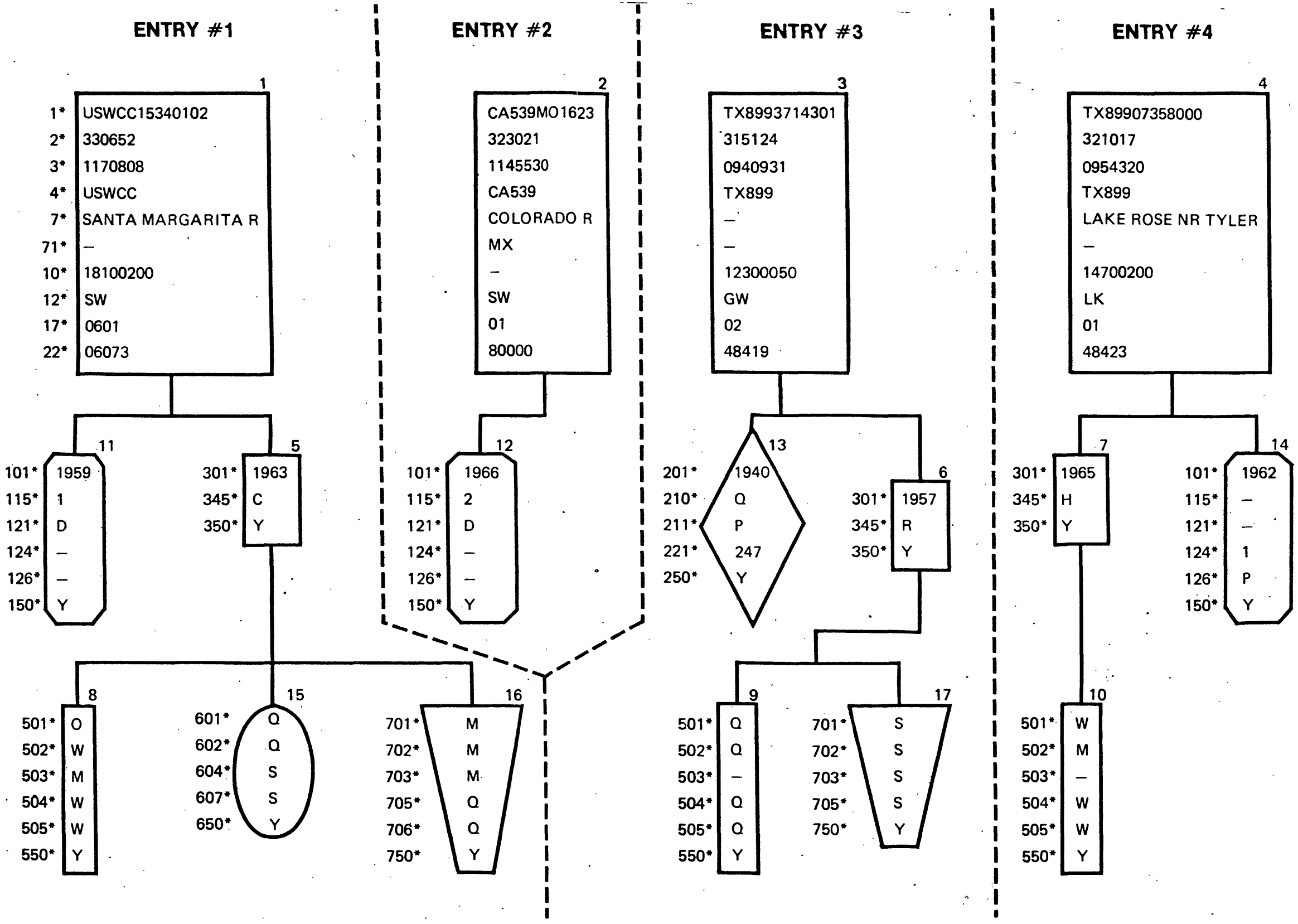


Figure 18.--MWDITEST final data base structure.

IV-25

SYSTEM 2000 RETRIEVAL MANUAL

SECTION V. SUMMARY OF DATA RETRIEVAL TECHNIQUES

SECTION V. SUMMARY OF DATA RETRIEVAL TECHNIQUES

V.A. THE LOGIC OF DATA RETRIEVAL

The goal of this manual is to teach the use of SYSTEM 2000 Natural Language to retrieve data from a SYSTEM 2000 data base. In order to reach that goal, a number of things must be learned:

The nature of a SYSTEM 2000 data base.

Elements and syntax of the SYSTEM 2000 Natural Language.

How SYSTEM 2000 displays the selected data.

How to solve problems using SYSTEM 2000.

The first three items above are covered in sections I-IV of this manual. The Natural Language elements are explained separately and then combined into commands, using many examples and visual aids such as block diagrams. Block diagrams of the entire data bases, including their data values, are illustrated. All these things are necessary parts of a tutorial manual, but the reader should not get the idea that such an abundance of information will be available when actual problems are confronted with "real-world" data bases.

The purpose of section V is to approximate the conditions that exist when the reader begins to access "real-world" data bases, the last item listed above. In most cases the SYSTEM 2000 user has a limited amount of knowledge concerning the contents of a data base - how many entries, how many data sets, etc. - beyond the schema. A block diagram of all the individual data sets, such as is shown in this manual, almost certainly will not be available simply because most data bases are so large that it is impossible to draw a block diagram.

The process of solving problems using SYSTEM 2000 and working with an unfamiliar data base can be simplified into a logical series of steps. Each step can be further subdivided into a series of activities, giving the user a checklist to follow from the inception of a problem to its solution. There are four major steps to follow in solving problems with SYSTEM 2000:

Learn the data base schema well

Become familiar with the data stored in the data base

Structure the data retrieval commands carefully

Access the data base

Learning the Data Base Schema

When beginning to work with any SYSTEM 2000 data base, the first step is to become throughly familiar with the data base schema. There are three activities to pursue when doing this:

Study the contents of each schema record

Learn the hierarchial relationships of all schema records with each other

Study the component definitions

Schema records are the key to what is stored in the data base. When looking at a data base schema for the first time, look at the schema records to get an overall picture of the components of the data base. It might be beneficial to make a list of all the schema records names to serve as an outline of the data base. Study the components in each schema record, especially in those schema records that are of major interest. Become familiar with which components belong to which schema records.

Next, learn the hierarchial relationships of all schema records with each other. Discern the different families in the schema and determine which schema records are directly related and which are discrete. Draw a block diagram of the entire data base schema, or if the schema is unusually large, of the parts of special interest. Such a block diagram is especially useful when structuring data base retrieval commands.

Finally, inspect the component schemas to determine the nature of the data for each component. If the data base schema is large, concentrate on the components of special interest. Make a mental note of which components are KEY components. Check the data types and field lengths and determine which components are likely to have data stored in coded or abbreviated form. If the component names are not clear about what data is stored, obtain documentation which explains each component.

Becoming Familiar with the Data Stored in the Data Base

After studying the data base schema, the next step is to familiarize oneself well with the data stored in the data base. There are four activities that can assist in doing this:

Find out the approximate size of the data base

Determine the range of data values for each component

If data are stored in abbreviated or coded form, obtain an explanation of the abbreviations and codes

Find out how often the data base is updated

Before making any attempts at data retrieval from an unfamiliar data base, obtain as much information as possible on the data which are stored within it. Knowing the number of entries in the data base aids in estimating the number of selected data records, and therefore the amount of output, that could be produced in response to a data retrieval command. Users working with very large data bases might want to specify narrower conditions for retrieval than they would if a much smaller data base were being accessed.

Often, this information on data base size is available in some sort of written documentation. The TALLY command also is very useful in obtaining information on the exact data values that are present. The first thing to determine is the approximate size of the data base, for which the number of entries is a good indicator. If this information is not documented, discern some Level 0 component which must be present in each entry and perform the command TALLY/ALL/ on it.

Next, become familiar with the ranges of data that are present for KEY components. Again, the TALLY/ALL/ and TALLY/EACH/ commands can provide this information if it is not available from a documented source. Because data records are normally selected by comparing the values in the data record with user-supplied items, knowing the range of values for KEY components will help the user to formulate conditions for data retrieval.

If some data values are stored in abbreviated or coded form, obtain an explanation of the abbreviations and codes. This information must be obtained from written documentation. The TALLY/EACH/ command can print out all codes in the data base for any component, but it cannot list all possible codes or the meaning of the codes. This kind of information is vital because the codes themselves are usually meaningless. It is impossible to intelligently use a code as a user-supplied value within a Conditional Clause if its meaning is not known.

Another item of possible importance is how often the data base is updated. Updates can change the number of entries, the range of data values for components, and the number of data records present. The same data retrieval command might produce two different answers is issued both before and after an update. The Data Base Administrator can supply information about frequency of updates.

Structuring Data Retrieval Commands

Once the user is familiar with the schema and the contents of the data base, the task of problem-solving can begin. There are four activities to pursue in problem solving:

Define the problem

Create the Conditional Clause

Create the Action Clause

Verify the command

The first step in problem-solving is to define the problem. Determine which data items must be extracted from the data base and the conditions that must be imposed for data retrieval. If reports must be printed, design their layout. The order in which commands will be issued should be considered; this can be important if the nature of one command is dependent upon the output from another command. Group similar commands so the DITTO and SAME Operators can be used.

Next, structure the data retrieval commands individually. Formulate the Conditional Clause of a command first. If multiple conditions are present, separate them with OR or AND, and nest them within parentheses if necessary. Once all the conditions have been established, determine whether or not the level of data record qualification must be raised with the HAS operator. Use SAME whenever possible.

Once the Conditional Clause of a command is completed, develop the Action Clause. Begin with LIST if a report in columnar format is desired, and specify the page heading, footing, and column titles to conform to the desired report layout. Specify any necessary Format Statement options. Write out the list of components to be extracted and printed from the selected data records, along with any necessary System Functions and schema records. Use an ORDERED BY clause to sort the printed output into a desired sequence. Insert BY clauses where necessary to order the output, control the number of selected data records, or raise the level of the specified data records. Use DITTO whenever possible.

When each data retrieval command is completely structured, check it over to make sure that it is correct. Verify the syntax, then compare the components specified in the command with the block diagram of the data base schema to insure that the command treats any discrete schema records correctly and that the level of data record qualification is correct. Verify whether components in this Conditional Clause are KEY or NON-KEY, since NON-KEY components should be used only with good reason. Check the user-supplied values in the Conditional Clause to make sure that they conform to the data type and field length defined for each component, and that they accurately represent your problem constraints. Above all, make sure that the command will provide the right answer to the problem to be solved.

Accessing the Data Base

When the first three steps in problem-solving have been completed, the user should have a list of verified data retrieval commands. There are three remaining activities:

Set up all other necessary SYSTEM 2000 commands

Set up the appropriate Job Control Language or Time Sharing Option commands

Enter the commands

The first three SYSTEM 2000 commands in every job are USER, DBN IS, and LIMIT. The proper user password and data base must be known in order to correctly write the first two commands. Establish appropriate lower and upper limits and write out the LIMIT command. Insert other LIMIT commands in the list of the data retrieval commands as needed. Add TALLY or SAME IS commands wherever needed, and place an EXIT command after the final data retrieval command.

Next, determine which JCL or TSO commands must precede and follow the SYSTEM 2000 commands. This information should be available in written documentation. If not, obtain the help of data processing personnel.

The final activity is to actually enter all the commands. If processing in batch mode, punch the commands on cards and read them into the computer. If processing in online mode, type in the commands one by one and wait for a response after each command.

Going through all of the steps presented in this chapter may at first seem to be cumbersome and time comsuming. However, it is a good practice to be as thorough as possible because repetition of these recommended steps encourages good habits and prevents errors. As the user gains experience in working with SYSTEM 2000, these steps become second nature and are performed as a matter of routine.

SECTION VI

APPENDIXES

APPENDIX A

HOW TO SUBMIT A SYSTEM 2000 JOB TO THE
U.S. GEOLOGICAL SURVEY's COMPUTER IN RESTON

General Requirements for any SYSTEM 2000 Data Base

Communication between the USGS computer system and its users is accomplished by means of Job Control Language (JCL) commands if in batch mode, or Time Sharing Option (TSO) commands if processing online. JCL and TSO commands are used to initiate and terminate each job, to allocate facilities (tapes, disks, memory storage) required for the job, and to begin execution of computer programs, as including Data Base Management System packages such
SYSTEM 2000.

Prior to issuring any SYSTEM 2000 commands, the user must accomplish the following functions by the user of JCL or TSO commands:

Begin the job

Allocate "scratch files" required by SYSTEM 2000

Allocate the files on which the SYSTEM 2000 data base resides

Begin execution of SYSTEM 2000

After the SYSTEM 2000 commands are all issued, including the final EXIT command, an additional JCL or TSO command is required to end the job. Although the forementioned functions are identical requirements for any SYSTEM 2000 job, the actual JCL or TSO commands needed ususally differ, depending on the data base being accessed, the computer site at which the job is being processed, and whether communication is via batch or online processing. Inasmuch as there is no standard set of JCL or TSO commands that will work for every SYSTEM 2000 job, it is necessary to obtain the assistance of ADP support personnel or the Data Base Administrator in preparing the JCL or TSO commans for your specific application; these support personnel should be informed of every data base that will be accessed during the job, so that proper provisions are made.

Requirements for the WDSDTEST Data Base

Following are the cards required when submitting a job in batch processing (all commands begin in column 1):

```
/*RELAY PUNCH RE3
//       Job card
/*PROCLIB  NWDX.PROCLIB
//WDSDTST EXEC WDSDTEST
```

```
 //SYSIN DD *
USER,USGS:
DBN IS WDSDTEST:
   .                    } SYSTEM 2000 Commands
   .
   .
EXIT:
/*
//
$$$
```

NOTE: The '/*RELAY' card is required to relay all following cards to the computer system on which the data base resides. It is anticipated that all data bases will be resident at the central computer system in Reston, Va., sometime in June 1981 after which time the '/*RELAY' card and the '$$$' will be no longer be required.

When communicating online, enter the following TSO commands to access the data base:

```
OO (for 300 baud) or^(for 1200 baud)
RE3TSO
LOGON USER-ID/password  PROC(WRDPROC)
EX 'VG4A44E.S2KTEST.CLIST'
WDSDTEST
USER,USGS:
DBN IS WDSDTEST:
   .                    } SYSTEM 2000 Commands
   .
   .
EXIT:
LOGOFF
```

Enter the EX 'VG4A44E.S2KTEST.CLIST' commands only after the computer responds to the previous LOGON with READY. Following the exit from SYSTEM 2000, the word READY is also printed, at which time LOGOFF should be issued to terminate the job.

Requirements for the MWDITEST Data Base

Following are all the cards required when submitting a job in batch processing (all commands begin in column 1):

```
/*RELAY PUNCH RE3
//    Job Card
/*PROCLIB  NWDX.PROCLIB
//MWDITST EXEC MWDITEST
//SYSIN DD *
USER,USGS:
DBN IS MWDITEST:
      .
      .
      .                 } SYSTEM 2000 Commands
EXIT:
```

```
/*
//
$$$
```

NOTE: The '/*RELAY' card is required to relay all following cards to the computer system on which the data base resides. It is anticipated that all data bases will be resident at the central computer system in Reston, Va., sometime in June 1981 after which time the '/*RELAY' card and the '$$$' will be no longer be required.

When communicating online, enter the following TSO commands to access the MWDITEST data base:

```
OO (for 300 baud) or^ (for 1200 baud)
RE3TSO
LOGON USER-ID/password  PROC(WRDPROC)
EX 'VG4A44E.S2KTEST.CLIST'
MWDITEST
USER,USGS:
DBN IS MWDITEST:
      .
      .
      .                      SYSTEM 2000 Commands
EXIT:
LOGOFF
```

Switching Data Bases

Because WDSDTEST and MWDITEST are interrelated, it is sometimes necessary to query one data base, then switch to the other. When running in batch, just combine the cards needed to access each data base into a single deck. For example, to first access MWDITEST and then WDSDTEST, the commands are --

```
/*RELAY  PUNCH RE3
//     Job Card
/*PROCLIB  NWDX.PROCLIB
//MWDITST EXEC MWDITEST
//SYSIN DD *
USER,USGS:
DBN IS MWDITEST:
      .
      .
      .
EXIT:
//WDSDTST EXEC WDSDTEST
//SYSIN DD *
USER,USGS:
DBN IS WDSDTEST:
      .
      .
      .
EXIT:
```

```
/*
//
$$$
```

NOTE: The '/*RELAY' card is required to relay all following cards to the computer system on which the data base resides. It is anticipated that all data bases will be resident at the central computer system in Reston, Va., sometime in June 1981 after which time the '/*RELAY' card and the '$$$' will not be required.

When running under TSO, use the following procedures:

(1) If currently in WDSDTEST, to switch to MWDITEST issue the following commands:

```
EXIT:
WDSDFREE
MWDITEST
USER,USGS:
DBN IS MWDITEST:
```

(2) If currently in MWDITEST, to switch to WDSDTEST issue the following commands:

```
EXIT:
MWDIFREE
WDSDTEST
USER,USGS:
DBN IS WDSDTEST:
```

APPENDIX B

QUICK REFERENCE LIST OF SYSTEM 2000 NATURAL LANGUAGE SYNTAX

SYSTEM 2000 Natural Language offers the user a great amount of flexibility but certain rules of syntax must be followed. The purpose of appendix B is to explain and demonstrate the rules in a manner such that the syntax of any command, function, operator, or other part of a command can be rapidly located. The terms in this appendix are grouped into three sections. The first section consists of terms found in the Action Clause, the second section consists of terms found in the Conditional Clause. And the third section consists of commands other than the Action Clause - Conditional Clause data retrieval commands.

Within each section, terms are listed in alphabetical order. For convenience, terms that share a common syntax are grouped together under the same title, as shown below

To Find the Term --	Look Under --
BLOCK, DOUBLE SPACE, GROUP, INDENT, NAME, NULL, NULL SUPPRESS, NUMBER, REPEAT, REPEAT SUPPRESS, SINGLE SPACE, STUB, STUB SUPPRESS, TREE, ZERO, ZERO SUPPRESS	Format Statement Options
SUM, COUNT, AVG, MIN, MAX, SIGMA	System Functions
EXISTS, FAILS	Unary Operators
EQ, NE, LT, LE, GT, GE	Binary Operators
AND, OR, NOT	Boolean Operators

If an abbreviation can be used for a term, its proper form is shown within parentheses immediately following its full spelling.

Special symbols used in illustrating commands are listed below:

Symbol	Explanation
< >	The enclosed item is mandatory and its contents are user-supplied item.
[ ]	The enclosed item is mandatory and restricted to the contents shown. When a choice of items is shown the user must specify one of them.
{ }	The enclosed item is optional.
Δ	Designates a blank.

ALL CAPS	Terms in all capital letters must appear as shown.
Lowercase	Terms in lowercase letters are supplied by the user.

Action Clause

Action Clause - That part of a SYSTEM 2000 data retrieval command which tells the System what action to take with the selected data records. The Action Clause is always the first part of a data retrieval command. Two entities must be present in every Action Clause: (1) either PRINT or LIST, which is always the first entity in the clause, and (2) one or more component names or numbers.

A number of options can be used in the Action Clause, either alone or in combination with each other. If Format Statement options are used, they are enclosed within slashes and immediately follow PRINT or LIST. If one or more BY Clauses are used, they may precede or follow the specified components. If an ordering statement is used, it is the final item in the Action Clause, immediately preceding WHERE.

The following four examples of possible structures of Action Clauses, from simple to complex, illustrate syntax when using various options:

(1) [PRINT / LIST] <components> ...

(2) [PRINT / LIST] /format options/ <components>...

(3) [PRINT / LIST] /format options/ <components>, ORDERED BY <components> ...

(4) [PRINT / LIST] /format options/ <BY Clause> <components>, ORDERED BY <components> ...

AVG - See System Functions

BLOCK - See Format Statement Options

BY - This word is always followed by one or more schema record names or numbers, which jointly form a BY Clause. There are two possible constructions for a BY Clause. In the first, the BY Clause is followed by a comma, and then by one or more component names or numbers which are modified by the BY Clause. The second format is used when a system Function is present in the Action Clause. In this second case, the BY Clause follows the System Function and its component.

The two formats are --

(1) [PRINT / LIST] BY <schema record>, <components> ...

(2) [PRINT / LIST] <system function><component> BY <schema record> ...

COUNT - See System Functions

DITTO - (DL) - The DITTO operator duplicates the entire Action Clause of the preceding command. No additional Action Clause options are permitted. The structure of such a command is DITTO followed by the Conditional Clause.

DOUBLE SPACE - See Format Statement Options

FORMAT STATEMENT OPTIONS - These options are used to control the format of output. When used, they alway immediately follow the term LIST or PRINT and are enclosed within a pair of slashes. There are eight sets of Format Statement options, shown below with the default option on top. All options can be used with PRINT. Only options (1), (7), and (8) can be used with LIST.

(1)	SINGLE SPACE DOUBLE SPACE	(2)	INDENT BLOCK	3)	NUMBER NAME	(4)	STUB STUB SUPPRESS
(5)	NULL SUPPRESS NULL	(6)	TREE GROUP	(7)	REPEAT REPEAT SUPPRESS	(8)	ZERO SUPPRESS ZERO

Format Statement Options are used as shown in this structure:

[PRINT / LIST] /format options/ <components> ...

GROUP - See Format Statement Options

INDENT - See Format Statement Options

LIST - (LI) - This command SYSTEM 2000 to print a report in columnar format, with a separate column for each specified component. When used, it is the first term in an Action Clause. Three title options are available for use with LIST. If none of these options are used, the list of components immediately follows the LIST command verb. The entire set of options, if any are used, is enclosed in a pair of slashes, occurring between the LIST command verb. The entire set of options, if any are used, is enclosed in a pair of slashes, occurring between the LIST verb and the list of components. The word TITLE must appear immediately after the first slash and the options follow, separated from each other by commas.

The column title option consists of a column type (either L, R, or B, for left-justified, right-justified, or blank, respectively) followed by an integer in parentheses, and a column title of up to three lines. A new line is indicated by a plus sign preceding the title for that line. The column type and integer are optional and have default settings, which are, respectively, the component name in the schema and the width of the component name, or the picture width of the component plus 3 characters, whichever is greater.

Column Title Option: {[L / R / B] (integer)}<title specification>

The page heading option consists of the letter D followed by an integer in parentheses representing the column in which the title begins, and the text of the heading. The page heading option must precede the column title option.

Page Heading Option: D(<integer>)<heading>

The page footing option consists of the letter F followed by an integer in parentheses representing the total number of lines on a page, and the text of the footing. The page footing option must precede the column title option but follow the page heading option, if present.

Page Footing Option F(<integer>)<Footing>

Format Statement options are specified immediately following the first slash and are separated from the word TITLE by a comma.

A summary of the format of the LIST command as used with its options is given below. Note that Format Statement options precede title options in the slash enclosed list.

(1) LIST <components> ...

(2) LIST/TITLE <column title option>/ <components>...

(3) LIST/TITLE <page heading options>, <column title option> / <components> ...

(4) LIST/TITLE <page heading option>, <page footing options>, <column title options>/<components> ...

(5) LIST/<print format options>, TITLE <page heading option>, <page footing options>,<column title options>/<components>

NAME - See Format Statement Options

NULL - See Format Statement Options

NULL SUPPRESS - See Format Statement Options

NUMBER - See Format Statement Options

MAX - See System Functions

MIN - See System Functions

ORDERED BY - (OB) - The ORDERED BY Clause sorts the output according to a specified order before it is printed. It is always followed by one or more component names or numbers which determine the sorting sequence. It is the last item in an Action Clause, immediately following the components to be printed and immediately preceding the Conditional Clause. A comma always follows the component specified immediately before ORDERED BY. The ORDERED BY Clause may contain System Functions, User-Defined Functions, or In-Line Functions.

```
[PRINT]
[LIST ]   <components>, ORDERED BY <components> ...
```

PRINT - (PR) - This commands SYSTEM 2000 to print the data items sequentially for a specified set of components. It is always the first item in an Action Cluase when it is used.

```
PRINT <components> ...
```

REPEAT - See Format Statement Options

REPEAT SUPPRESS - See Format Statement Options

SIGMA - See System Functions

SINGLE SPACE - See Format Statement Options

STUB - See Format Statement Options

STUB SUPPRESS - See Format Statement Options

SUM - See System Functions

SYSTEM FUNCTIONS - These functions permit the user to obtain arithmetic statistics about selected data from the data base. A System Function immediately precedes the component name or number that is affected by the action of the function. A System Function acts on only one component and only a single System Function may be used with a specified component. The six System Functions are MAX, MIN, COUNT, SUM, AVG, and SIGMA. The syntax for using a System Function is:

```
[PRINT]   <components>,<system function><components>,<system
[LIST ]    function><component> ...
```

TREE - See Format Statement Options

ZERO - See Format Statement Options

ZERO SUPPRESS - See Format Statement Options

Conditional Clause

AND - See Boolean Operators

BINARY OPERATORS - The purpose of binary operators is to define, limit or bound the range of qualifying data records. The six binary operators are EQ, NE, LT, LE, GT, and GE. A binary operator is immediately preceded by a component name or number and immediately followed by a user-supplied value with which the component's data items are compared

... WHERE <component><binary operator> <specific value> :

BOOLEAN OPERATORS - The purpose of boolean operators is to indicate the relationship between multiple conditions for qualifying data records. The two boolean operators are AND and OR. The AND operator placed between two conditions requires that both conditions must be met for a data record to qualify; the OR operator placed between two conditions requires that either condition may be met for a data record to qualify. A boolean operator must always be located between two conditions.

... WHERE <condition$_1$> [AND / OR] <condition$_2$> :

CONDITIONAL CLAUSE - Latter part of a SYSTEM 2000 data retrieval command that qualifies data records from the data base according to one or more conditions. The Conditional Clause always begins with WHERE and immediately follows the Action Clause. It always contains at least one condition, which consists of a component name or number plus one or more binary operators, unary operators, or the term SPANS. If a binary operator or SPANS is used, the Conditional Clause also must contain user-supplied value or values as a stipulation for data retreival. Multiple conditions must be joined by boolean operators. The HAS operator is used to qualify data records based on their logical position in the data base.

Below are a number of structures possible for Conditional Clause that illustrate the syntax when using with various options.

(1) ... WHERE <component> [EXISTS / FAILS] :

(2) ... WHERE <component> [EQ / SPANS / NE] <value$_1$>*<value$_2$>:

(3) ... WHERE <components> [EQ / NE / LT / GT / LE / GE] <value>:

(4) ... WHERE <condition> [AND / OR] <condition> :

(5) ... WHERE <repeating group> HAS <condition>:

(6) ... WHERE NOT <condition>:

EQ - See Binary Operators; also see SPANS

EXISTS - See Unary Operators

FAILS - See Unary Operators

GE - See Binary Operators

GT - See Binary Operators

HAS - The purpose of the HAS operator is to qualify a higher level data record for selection based on one or more values that occur in its data descendants. It follows a schema record component name or number which is at the desired level of qualification. It is immediately followed by a condition containing the name or number of a component that must be in the same schema family as the specified schema record.

... WHERE <schema record> HAS condition

LE - See Binary Operators

LT - See Binary Operators

NE - See Binary Operators; also see SPANS

OR - See Boolean Operators

SAME - (SA) - This operator is used to duplicate the Conditional Clause of a previous command. It may be used by itself or in combination with other conditions, as shown below.

(1) ... WHERE SAME :

(2) ... WHERE <condition> [AND / OR] <SAME>:

(3) ... WHERE SAME [AND / OR] <condition>:

(4) ... WHERE (SAME [AND / OR] <condition>) [AND / OR] (SAME [AND / OR] <condition>) :

SPANS - The purpose of the SPANS operator is to qualify data records on the basis of a range of values. SPANS is always preceded by a component name or number and is followed by a pair of user-supplied values separated by an asterisk. The term EQ may be used in place of SPANS if desired; they are equivalent. If the term NE is used instead of SPANS, data records are qualified on the basis of values which lie outside the specified range.

... WHERE <component> [SPANS / EQ / NE] <value>*<value>:

UNARY OPERATORS - The purpose of the unary operators EXISTS and FAILS is to test for the existence or nonexistence of data values. A unary operator is immediately preceded by a component name or number. Any condition containing FAILS is treated as a NON-KEY condition.

... WHERE <component> [EXISTS / FAILS] :

WHERE (WH)- This is always the first entity in a Conditional Clause. All Conditional Clauses begin with WHERE.

Other Commands

DESCRIBE - The purpose of the DESCRIBE command is to print out all or a portion of a data base schema. If the entire schema is wanted, just enter DESCRIBE. If a single component is wanted, that component number should immediately follow DESCRIBE. If a range of components is wanted, follow DESCRIBE with the beginning component number, then either THROUGH or THRU, and the ending component number. To print out Strings or Functions, follow DESCRIBE with STRINGS or FUNCTIONS. Below are the formats that can be used.

(1) DESCRIBE:

(2) DESCRIBE <component number>:

(3) DESCRIBE <component number> [THROUGH / THRU] <component number>:

(4) DESCRIBE [STRINGS / FUNCTIONS] :

DATA BASE NAME IS - (DBN IS) - This is the second command issued upon entering SYSTEM 2000. It names the data base from which data are to be retrieved.

DBN IS <data base name>:

EXIT - This command causes an exit from the SYSTEM 2000 program.

EXIT:

LIMIT - The purpose of the LIMIT command is to control the output from selected data records, to set the minimum/maximum boundary of data records which may be selected, and to set the specific number of data records which may be retrieved. LIMIT has three formats. It may be followed by a single positive integer, in which case the number of selected data records must be equal to that integer in order to be accepted. It may be followed by two positive integers separated by a comma, in which case the number of selected data records must not lie outside the two integer values in order to be acceptable. Or, it may be followed by two positive integers separated by a comma, which in turn are followed by a slash and either CANCEL or TRUNCATE. CANCEL is the default value and orders the command to be cancelled if values lie outside the specified range. TRUNCATE allows the command to be performed on a truncated portion of the selected data records, as determined by the specified range. LIMIT has no affect on commands without a Conditional Clause. The three formats are shown below.

(1) LIMIT <number>:

(2) LIMIT <number> , <number> :

(3) LIMIT <number> ,<number> / [CANCEL / TRUNCATE] :

SAME IS - The purpose of this command is to change the processing mode of the SAME operator. There are two options, STATIC and DYNAMIC, which can be used in the command. The default option is DYNAMIC.

SAME IS [STATIC / DYNAMIC] :

STRING - A String is a command that is stored within the data base schema and may be invoked by the user. There are two kinds of Strings, Simple and Parametric. To invoke a Simple String, enter an asterisk followed by the component number or name of the String, followed by another asterisk.

<String name or number>:

To invoke a Parametric String, enter an asterisk followed by the component number or name of the String, and followed by one or more data values, enclosed in parentheses, as required by the String.

* <String name or number>(<data values>):

TALLY - (TA) - The purpose of the TALLY command is to provide a means of obtaining statistical information about the unique values of components stored in the data base (minimum, maximum, and distribution). One of two options, EACH or ALL, must be enclosed in a pair of slashes immediately following TALLY. After the options, one or more component names or numbers, all separated by commas, must be specified.

TALLY / [EACH / ALL] / <components>:

USER - This command is used to issue the security password required for access to the data base named in the DBN IS command. This is always the first command issued upon entering SYSTEM 2000. The term USER is followed by a comma and then the password.

USER, <password>:

APPENDIX C

SYSTEM 2000 PROBLEMS AND THEIR SOLUTIONS

Kinds of Problems

A summary of the most common SYSTEM 2000 problems along with an explanation as to why they occur, how to avoid them, and how to recover from them is presented in this appendix.

In general, problems encountered when working SYSTEM 2000 can be classified as follows:

(1) Inability to begin the SYSTEM 2000 run or unwanted exit from SYSTEM 2000;

(2) Errors in SYSTEM 2000 commands:

(3) Incorrect output.

Inability to Enter SYSTEM 2000 or Unwanted Exit from SYSTEM 2000

The command that allocates the SYSTEM 2000 files and begins the SYSTEM 2000 run (see appendix A) occasionally fails and an error message is printed. An unwanted exit from SYSTEM 2000 can be caused by a number of things; for example, certain commands entered incorrectly, interrupting the output during online processing, or system malfunctions not caused by the user.

If the unwanted exit occurs during a batch run, inspect the last command processed by SYSTEM 2000 to see if it is valid or whether SYSTEM 2000 printed an error message stating that an exit from SYSTEM 2000 was about to occur. If an error is detected, correct it and resubmit the run. If the error message SYSTEM ERROR CODE is found, report the error to SYSTEM 2000 technical support personnel. If no reason for the unwanted exit is apparent, resubmit the run.

Procedures for recovery from an unwanted exit are different when processing online. The exit is normally accomplished by some sort of error message, but if the computer returns the message READY it is certain that an exit from SYSTEM 2000 has occurred. To recover, issue the following command--

```
S2KAGAIN
```

If unwanted exits persist and no cause can be found, contact:

National Water Data Exchange
Support Services Unit
421 National Center
Reston, VA 22092

Telephone: (703) 860-6031
FTS 928-6031

Errors in SYSTEM 2000 Commands

SYSTEM 2000 checks each command for validity before executing the command. If an error is found the command is rejected and an appropriate error message is printed. Erroneous commands often cause an exit from SYSTEM 2000 during batch processing, but exit from SYSTEM 2000 is rare during online processing.

SYSTEM 2000 denotes an error by printing the letter C underneath the point at which the program detected an error in the command. The letter is preceded by a string of dots, and underneath is printed the error message. An example is shown below --

```
PRINT C1 WH C100 EQ 1959:

..................C

- ILLEGAL USE OF SCHEMA RECORD -
```

The error occurred because C100 is a schema record and a schema record cannot be tested for being equal to a value. Because EQ cannot follow a schema record number, the C appears under EQ. Note that the C is printed at the point where the SYSTEM 2000 recognized that an error exists; the erroneous value is generally to the left of the C. Error messages that can occur while using SYSTEM 2000, along with an explanation of each message and the action to take in response to the message are listed beginning on the next page. The messages are in alphabetical order according to the first word in each message. Some error messages may be preceded by a three-digit number, but the number may be ignored when looking up errors in this appendix.

ERROR MESSAGE	EXPLANATION	RECOMMENDED ACTION
CLAUSE CONTAINS NON-SIBLINGS OF CONTROLLING SR	A component in a BY Clause is not a descendant of the specified schema record.	Remove the component from the BY Clause.
DATA BASE CURRENTLY IN USE	Someone else is using the data base, and no one else may access it.	Try again later.
DATA BASE DCB OPEN FAILED	SYSTEM 2000 coul' not access the data base specified in the DBN IS command.	Try again Reenter SYSTEM 2000 and reissue the USER and DBN IS commands.
DATA BASE DOES NOT EXIST	The DBN IS command specified a data base that does not exist.	The wrong data base name was specified; correct the error and try again.
ERROR IN DATE OR NUMERIC VALUE	A date was specified incorrectly.	Make sure the date is of the form mm/dd/yyyy.
F OR D SPECIFICATION IS OUT OF LEGAL RANGE	The F or D specification in the title field of a LIST command must contain an integer between 1 and the maximum number of columns permissible for the output device being used. The user has specified something outside the legal range.	Specify an integer within the legal range.
FIRST LIMIT GREATER THAN SECOND	Parameters in the LIMIT command must be given in ascending order to be equal, unless the second parameter is zero.	Make sure that the first parameter is not greater than the second.
FULL DATA BASE PASS NOT ALLOWED	A command containing a fully NON-KEY Conditional Clause has been entered, which is illegal.	There must be at least one KEY command in the Conditional Clause.
FUNCTION ITEMS MUST BE IN ONE FAMILY	Components belonging to different schema families have been specified within an In-Line Function.	Make sure that all components within the the function belong to the same schema family.

ERROR MESSAGE	EXPLANATION	RECOMMENDED ACTION
HAS CONTAINS NON-SIBLINGS OF PARENT RG	The condition following a HAS may contain only components that are siblings of the schema record preceding the HAS.	Either eliminate the component which is a non-sibling of the schema record or raise the level of the schema record to accomodate the component.
HEADER AND/OR FOOTING SPECIFICATION MUST APPEAR FIRST	The F and D specifications in the title of a LIST command must precede any other title specifications.	Place the F and D specifications immediately after the word TITLE.
ILLEGAL PASSWORD	The user password specified in the USER command is not acceptable for the data base specified in the DBN IS command.	Reenter the USER command with the correct password, then reenter the DBN IS command.
ILLEGAL USE OF SCHEMA RECORD	a) An SR cannot precede EXISTS, FAILS, GE, GT, LE, LT, EQ, or NE in a Conditional Clause. b) An SR cannot be used with any System Function except COUNT. c) An SR cannot be used with LIST. d) An SR cannot be used in a function.	Specify individual component numbers instead of a schema record number. Use PRINT instead of LIST to print all components in a schema record.
MAJOR ITEMS MUST BE IN ONE FAMILY	Components specified in an ORDERED BY Clause belong to different schema families.	Use one or more BY Clauses to raise the level of specified data records.
MISSING <COMP ID>	A component name or number was omitted where one was expected.	Include the proper component name or number.
NO PRIOR LEGAL COMMAND	DITTO may be used only if preceded by a valid command	When communicating online, do not use DITTO after an erroneous command.

ERROR MESSAGE	EXPLANATION	RECOMMENDED ACTION
NO PRIOR LEGAL WHERE CLAUSE	WHERE SAME may be used only if preceded by a valid command with a Conditional Clause.	Spell out the entire Conditional Clause.
NO USER SPECIFICATION	The USER command was not issued as the first SYSTEM 2000 command.	Enter the USER command followed by the DBN IS command.
NUMERIC TYPE ELEMENT IS REQUIRED	a) A schema record type NAME component, or type TEXT component cannot be used in an IN-LINE FUNCTION. b) A type NAME or TEXT component cannot be used with SUM, AVG, or SIGMA.	Components used in IN-LINE FUNCTION or with SUM, AVG, or SIGMA must be type DECIMAL, MONEY, INTEGER, or DATE.
OBJECT SR UNRELATED TO WHERE-CLAUSE	One or more components in the Action Clause belong to a different SR family than one or more components in the Conditional Clause.	Use only components in the same SR family, or use HAS or BY to raise the level of the qualified or specified data records.
POS INTEGER REQ	A negative number or a non-numeric character is used where a positive integer is required.	Replace the erroneous character with a positive integer.
PRINT CLAUSE TOO COMPLEX	a) More than the maximum allowable number of components specified in an Action Clause. b) Too many distinct constants referenced in a function. c) Too many syntactic units included in a function.	Reduce the number of components specified in the Action Clause, or reduce the complexity of an In-Line Function.
REDEFINITION OF HEADER OR FOOTING	D or F specified twice in the TITLE field of a LIST command.	Specify a heading and footing only once.
SCHEMA RECORD MUST PRECEDE 'HAS'	The component specified immediately prior to HAS was not a schema record.	Always specify a schema record just prior to HAS.

ERROR MESSAGE	EXPLANATION	RECOMMENDED ACTION
SECOND VALUE MUST BE GREATER THAN FIRST	A SPANS, EQ, or NE operator is followed by two values. The second value must be greater than the first.	Correct one of the values or reverse their order.
SYNTAX ERROR IN COMMAND	This is a general error message denoting that a syntax error is present somewhere in the command.	Check for misspelled terms, clauses out of order, missing punctuation, etc.
TOO FEW SELECTED DATA RECORDS	Fewer data records selected than allowed by the lower limit of a LIMIT command	To prevent this, decrease the lower limit of the LIMIT command.
TOO MANY COLUMNS SPECIFIED IN FORMAT	Exceeded the limit of 136 print positions in the title of a LIST command.	Reduce the width of columns or eliminate columns so that fewer than 136 positions are required.
TOO MANY LINES SPECIFIED IN TITLE	Exceeded the limit of three lines of the title of a LIST command.	Do not specify column titles of more than three lines.
TOO MANY OPERANDS FOR OPERATOR	Operator in Conditional Clause followed by more operands than permissible.	Make sure that the GE, GT, LT, LE operators have one operand; FAILS and EXISTS have no operand; and SPANS, EQ, and NE have no more than two operands.
TOO MANY OPERANDS IN WHERE CLAUSE	Exceeded system limitations for a Conditional Clause.	Break up the command into several commands using SAME AND or SAME OR to obtain the same results.
TOO MANY OPERATORS IN WHERE CLAUSE	Exceeded system limitation for the number of operators (e.g., EQ, NOT) in a single Conditional Clause.	Do not specify more than 58 operators.

ERROR MESSAGES	EXPLANATION	RECOMMENDED ACTION
TOO MANY SELECTED DATA RECORDS	More data records selected than allowed by the upper limit on a LIMIT command.	To prevent this, specify a higher upper limit in a LIMIT command. Or specify TRUNCATE in a LIMIT command so that limited amount of data is printed.
TYPE OR WHERE CLAUSE DIFFERS FROM LAST USE	If DITTO is followed by a Conditional Clause the previous command must also have a Conditional Clause.	Resubmit the command with a Conditional Clause instead of DITTO.
UNBALANCED PARENTHESES	An unequal number of right and left parentheses exists in the Conditional Clause.	Check to see that every left parenthesis has a corresponding right parenthesis.
UNDEFINED VALUE	A division by zero in an In-Line Function has been specified. This occurs only when PRINT is used with the ZERO option.	To keep this message from being printed use LIST or PRINT with the ZERO SUPPRESS option, or division by ZERO can be eliminated by specifying in the Conditional Clause that the component used as divisor cannot equal zero.
UNRECOGNIZED <COMP ID>	A component name or number has been encountered which is not part of the data base schema, or is defined as an inappropriate type for its use in the command.	Make sure all component names and numbers are in the data base schema. Look for missing commas between component ID's or misspelling of the terms ORDERED BY or WHERE.
UNRELATED CONDITIONS ENCOUNTERED	Conditions contain components belonging to different SR families.	Use HAS to raise the level of qualifiction up to one common to both families.

ERROR MESSAGE	EXPLANATION	RECOMMENDED ACTION
VALUE EXCEEDS PICTURE WIDTH	Data value for the type NAME or TEXT component is longer than required by the data base schema.	Shorten the value to the length required by the data base schema.
WARNING: DATA BASE DAMAGED. NO UPDATES PERMITTED.	The data base has been damaged during a previous usage of SYSTEM 2000.	This problem normally occurs when another user is attempting to update the data base. Data retrieval is still possible, but the results may not be valid. Try again later.
WHERE CLAUSE ILLEGAL WHEN UNRELATED ITEMS ARE IN PRINT LIST	Components specified in the Action Clause belong to different schema families.	Use only components in the same schema family or use one or more BY Clauses to raise the level of specified data records.
WHERE CLAUSE REQUIRED WITH ORDERING	ORDERED BY is used in a command without a Conditional Clause.	Always use a Conditional Clause in a command with ORDERED By.

APPENDIX D

ALPHANUMERIC SYMBOL SORTING SEQUENCE

All special symbols, alphabetic characters, and numeric digits recognized by the computer are listed below. They appear in ascending order from top to bottom. Special Symbols are "less than" alphabetic characters, which are "less than" numeric digits in the sorting sequence. This sorting sequence is in effect for the terms ORDERED BY, LT, GT, LE, and GE.

Special Symbols	Alphabetic	Numeric
blank		
¢	A	0
.	B	1
<	C	2
(	D	3
+	E	4
|	F	5
&	G	6
!	H	7
$	I	8
*	J	9
)	K	
;	L	
¬	M	
‾	N	
/	O	
,	P	
%	Q	
_	R	
>	S	
?	T	
:	U	
#	V	
@	W	
'	X	
=	Y	
"	Z	

APPENDIX E

ANSWERS TO STUDENT EXERCISES

Answers to all Student Exercises in the manual, are given below. Explanations accompany the answer when appropriate.

STUDENT EXERCISE 1-1

(1) b

(2) d

(3) c

(4) a

STUDENT EXERCISE 1-2

(1) Component number, component name, KEY or NON-KEY designation, item type, and size of the data field.

(2) b

(3) KEY components require extra storage space for their index tables, and storage space is usually limited. Also, each time a KEY component is updated, the index tables for that component must also be updated, requiring the use of additional computer time Large numbers of KEY components mean that updates require large amounts of computer time.

(4) a

(5) b, d and e

(6) d - Data type TEXT retains all blanks.

(7) b - Data type NAME discards leading and trailing blanks, and retains only single embedded blanks.

(8) a, d, and e - Answer d also is correct because data type NAME can be used to store any alphanumeric characters.

(9) c and d

(10) 1. In component 3, INTEGER and X(6) are incompatible.

2. In component 12, no field length should be specified with data type DATE. The xx/xx/xxxx must be omitted.

3. In component 33, the data type is missing.

4. In component 17, the data type DECIMAL requires a decimal point in the field length. The 9(7) must be 9(7).9 or 9(7).99 or some designation including a decimal point.

5. In component 19, the field length should be 99.9 instead of XX.X to be compatible with data type DECIMAL.

6. In component 10, no field length should be specified with data type DATE. The 9(6) must be omitted.

7. Component 12 appears twice.

8. In the second occurrence of component 12, NUMBER is not a valid data type.

9. In component 13, the field length 9999 is incompatible with data type NAME. It should be XXXX.

10. In component 16, the field length X(2000) is greater than the 250 character maximum.

    Incidentally, there is no error in component 11 even though it appears to be incomplete. If KEY or NON-KEY designation is not specified, the component becomes KEY. If field length is not specified, the default for data type DECIMAL is 9(6).9(2).

STUDENT EXERCISE 2-1

Enter into SYSTEM 2000 as described in appendix A. Call for the WDSDTEST data base by issuing the following commands:

```
USER,TEST:

DBN IS WDSDTEST:
```

Following are the answers to the four questions:

(1) DESCRIBE:

(2) DESCRIBE C2:

(3) DESCRIBE C3 THRU C7:

(4) EXIT:

Following the EXIT: command, terminate your job according to the procedures described in appendix A.

STUDENT EXERCISE 2-2

(1) c - Answer d is not correct because the LIMIT command has no effect on how many data records are selected. It limits the amount of data that is extracted and printed from the selected data records.

(2) c - Data Records 1, 3, and 4.

(3) c - As stated in answer (1) above, LIMIT has no effect on the number of data records selected.

(4) LIMIT 4,9/TRUNCATE:

(5) d - Answers a and b are not correct because a missing value cannot be compared to a value such as 0 or A.

(6) c - Answers a and d are incorrect because they qualify data records with a numeric value for ORIENTATION1 (Look at appendix D - numeric values are "greater than" Z).

STUDENT EXERCISE 2-3

(1) c - Answer d is not correct because it prints only the maximum and minimum values, not all unique values.

(2) b - The System Function MIN cannot appear in a Conditional Clause.

c - There is no such command as TALLY/SUM/.

d - Component C12 is of data type NAME and therefore no standard deviation can be computed.

f - A Conditional Clause cannot be used with a TALLY command. Command e is syntactically correct but is useless because it tells SYSTEM 2000 to find all data records having a value of USWCC for component C4, then to print the maximum and minimum values of C4. These values can only be USWCC!

(3) USER,TEST:

DBN IS MWDITEST:

DESCRIBE:

At this point the data base schema is printed, and the user may determine which components to work with.

TALLY/EACH/C12:

The TALLY command prints the frequency of occurrence for each

TYPE code in the data base. SW occurs twice and GW and LK appear once each.

PRINT C1, C7, C71 WHERE C12 EQ SW:

From the output it is evident that site CA539M01623 is located in Mexico.

(4) USER, TEST:

DBN IS WDSDTEST:

LIMIT 0,10/TRUNCATE:

PRINT C2 WHERE C6 EQ A:

PRINT MAX C9 WHERE C6 EQ A:

LIMIT 0,0:

PRINT C2 WHERE C6 EQ A:

The LIMIT command had to be changed prior to the PRINT command so that all non-NAWDEX members could be printed. If the LIMIT had not been changed, a maximum of ten non-members could have been printed.

STUDENT EXERCISE 2-4

(1) c

(2) 1. PRINT/DOUBLE SPACE, NAME/C1, C10 WHERE C12 EXISTS:

2. PRINT/SINGLE SPACE, NULL/C2,C3, C12 WH C12 EXISTS:

(The NAME option is still in effect from the first command)

3. PRINT C1, C7, OB C1 WHERE C22 SPANS 6000*6999 or C22 SPANS 48000*48999:

No Format Statement options are required because the previous ones are still in effect.

The SPANS operator is necessary in order to include the entire state (the use or EQ is also permissable). Since C22 is the concatenation of state and county codes, and since no county code will be less than 000 or greater than 999, 48000*48999 qualifies the entire state of Texas.

(3) a, b, and e - SYSTEM 2000 skips three spaces between columns unless the columns are designated as blank. For the titles printed by the command in answer a, three spaces will automatically be

skipped between the columns. In answer b, the command instructs SYSTEM 2000 to skip three spaces between columns. In answers c and d, SYSTEM 2000 skips three spaces between columns but the titles will not be three spaces apart because of the blanks embedded in the titles. In answer e, the default titles are three spaces apart.

(4) Error 1 - The field L(6)LATIT is not wide enough to contain a value for LATITUDE, so the value is TRUNCATED.

Error 2 - The title specification L(5)TYPE implies that site-type data is being printed, but data for component C10, HYDROL-UNIT, is printed in this column. The column also is too narrow to contain data on hydrologic units, so the data values are truncated.

(5) a - System Functions are printed first in the order specified by the user, followed by the other components in the order specified by the user.

(6) b - When LIST is used, everything is printed in exactly the same order specified by the user.

(7) USER,TEST:

DBN IS MWDITEST:

LIMIT 0,50/TRUNCATE:
(Don't forget the LIMIT command!)

LIST/TITLE L(20) ID NUMBER, B(5), TYPE, MAX-LON,

B(7) OTHER, ST_CNTY,HYD UNIT/ C1, C12, MAX C3, C22, C10

WHERE C22 GE 31000:

This is but one way to print the desired report. You may have used other methods to place the titles over the columns.

The Conditional Clause C22 GE 31000 is one way to qualify states with a FIPS code greater than 30. Another way would be ... C22 GT 30999. Because C22 includes both the state and county codes, the Conditional Clause ... C22 GT 30 would qualify all states, since the lowest State_County code is 1001.

STUDENT EXERCISE 2-5

(1) d

(2) c - The report lists values for component C1 under the title NAWDEX_ID at the far left of the report, then next appear two blank columns entitled EXTRA SPACE and COMMENTS. Next over is

a column entitled LATITUDE containing values for C2 (inconsistent but that is what happens), and next to that is a column entitled SITE_TYPE containing values for component C12. The final three columns are a blank column 7 spaces wide with no title, and two columns with the default titles MAX LONGITUDE and STATE_COUNTY.

(3) a

(4) The report is more than 132 characters in width and therefore part of it is TRUNCATED.

(5) LIST/TITLE D(20) SITE LOCATION REPORT, F(60) LATEST DATA,

L(6)+SOURCE+AGENCY,STATE_+COUNTY+ CODE,+ LATITUDE+(DEG-MIN-SEC),

L(13)+ LONGITUDE+(DEG-MIN-SEC)/C4, C22,C2,C3:

STUDENT EXERCISE 3-1

(1) b

(2) d

(3) d

(4) a, c and d

(5) a, c, and d

(6) d

(7) Data records that are siblings from a common parent but are members of different schema families.

(8) Error 1 - In component 29, the terms KEY SR are incorrect. No other designation other than just SR is permitted. It should say 29* WELL-LEVELS (SR).

Error 2 - In component 35, the phrase IN 24 is incorrect because component 24 does not define a schema record. The correct phrase is IN 29.

STUDENT EXERCISE 3-2

(1) c and d

(2) LIMIT 0,3/TRUNCATE:

LIST/REPEAT SUPPRESS/C1,C4,C102,C123 WHERE C6 EQ E:

(3) First, access the MWDITEST data base and find out the organization and office codes associated with hydrologic unit 18100200.

```
USER,LVL1:

DBN IS MWDITEST:

PRINT C4,C17 WH C10 EQ 18100200 AND C12 EQ SW AND C350 EQ Y:

The answer is

  4* USWCC

  17* 0601
```

Now switch to WDSDTEST and find the contact name and phone number at the office.

```
USER,LVL1:

DBN IS WDSDTEST:

PRINT C111,C117 WH C102 EQ 0601 AND C1 EQ USWCC:

  111* DIRECTOR

  117* 916-345-3434
```

STUDENT EXERCISE 3-3

(1) b

(2) e - A schema record may not be used with LIST.

(3) First, access the MWDITEST data base to get the name, latitude, and longitude of the desired sites. Also get the organization and office code for use in accessing the WDSDTEST data base.

```
USER,LVL2:

DBN IS MWDITEST:

LIST C7,C2,C3,C4,C17 WHERE C503 EQ M AND C504 EQ W AND C350 EQ Y:
```

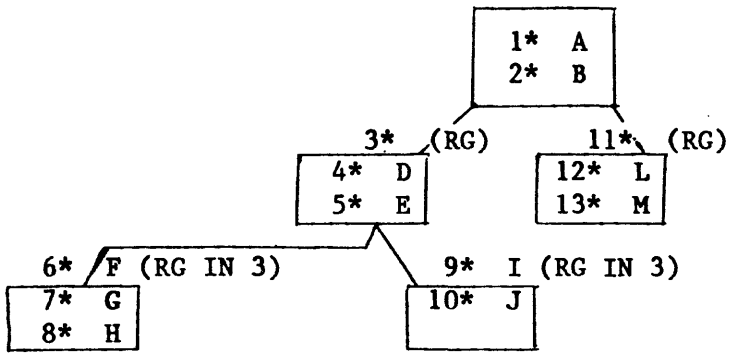
Only one site, the Santa Margarita River, is selected. Office 0601 of organization USWCC has data on this site. The WDSDTEST data base must be accessed to ascertain whether or not the office is a an Assistance Center.

```
PRINT C110,C126,C127 WHERE C102 EQ 0601 AND C1 EQ USWCC:
```

The answer shows that office 0601 of USWCC is an Assistance Center.

STUDENT EXERCISE 3-4

(1) c - Below is a block diagram of the data base schema. One schema record family is formed by schema records headed by components 1 and 11. A second schema family is formed by schema records headed by components, 1, 3, and 6. The third schema family is for by schema records headed by component 1, 3, and 9.



(2) LIST BY ENTRY, C7, C10 WHERE ENTRY HAS C8 GT 15.00 AND ENTRY HAS C13 LT 15:

BY C3 could have been used instead of BY ENTRY, if desired.

(3) d

(4) b

(5) c - On page III-35 is a block diagram of the MWDITEST data base schema. Schema record 700 is sibling from schema record 300 and therefore in the same schema family. Data Record 7 belongs to schema record 300 and Data Record 16 belongs to schema record 700, so they are in the same schema family.

(6) LIST C1, BY ENTRY, C150,C115,C124,C250,C210 WHERE C701 EQ M AND C350 EQ Y:

(7) b - Because of ENTRY HAS, the level of qualification is at Level 0, and Data Records 1 and 3 are the qualified data records. Data Records 4, 5, 6, 8, and 9 are selected because they are specified data records in the same schema families as Data Records 1 and 3.

(8) a - Component C345 is at Level 1. Therefore, the selected data records are at Level 1, Data Records 5 and 7.

STUDENT EXERCISE 4-1

(1) LIST C1,C7 WHERE SAME AND ENTRY HAS C350 EQ Y:

DITTO WHERE SAME AND ENTRY HAS C350 FAILS:

STUDENT EXERCISE 4-2

(1) a, c, and e are correct. Answers b and d are wrong because of the second asterisk.

(2) d and f are correct. A is wrong because it has no user-supplied values; b is wrong because C201 can be compared only to integer, not a decimal value; c is wrong because C402 is in a different SR family than C110 and C100; e is wrong because 111 and USWCC are too large for the size of the field.

(3) DESCRIBE STRINGS:

C2000: or *LAC_REPORT*:

(4) *C2001 (C404,CA539, 80):

(5) *C3000(USWCC15340102, C705, C706):

STUDENT EXERCISE 4-3

(1) a and c are correct. b is incorrect because there should be no asterisk following the User-Defined Function; d is incorrect because a C must precede the component numbers - they should be *C600*,*C601*, instead of *600*,*601*; e is incorrect because Stored Functions cannot appear within a Conditional Clause.

(2) DESCRIBE FUNCTIONS: or DESCRIBE C2010:

PRINT/ZERO/BY ENTRY, *C2010*,C502 WHERE C1 EQ USWCC:

APPENDIX F

NATIONAL WATER DATA EXCHANGE
SYSTEM 2000 DATA BASES

Overview of NAWDEX

A National Water Data Exchange (NAWDEX) has been established to assist users of water data in the identification, location, and acquisition of the data. NAWDEX consists of water-oriented organizations throughout all levels of the Federal, State, and local governments as well as and private sectors of the water-data community. These organizations work together to make their water data more readily available in a timely and efficient manner. All activities and services are coordinated by a central program office located within the U. S. Geological Survey. Assistance in the location of desired water data and the referral of requests to the most expedient source from which data can be acquired is be provided by the NAWDEX Program Office and a nationwide network of Local Assistance Centers established within the NAWDEX membership in nearly all States and major population centers. Membership in NAWDEX is voluntary and open to those who wish to take an active role in its activities.

NAWDEX Data Bases

A computerized system has been established for a Water Data Sources Directory (WDSD) containing information pertinent to the sources, location, types and availability of water data and for a Master Water Data Index (MWDI) containing more detailed information pertinent to the identification and definition of the types of water data held by NAWDEX members.

The purpose of this manual is to provide instruction in SYSTEM 2000 data retrieval capabilities to those individuals who will be accessing the MWDI and WDSD data bases. For this reason, the definitions of the two test data bases used as examples throughout the manual are subsets of the actual MWDI and WDSD data bases and the reader who completes the manual will, therefore, gain substantial knowledge about the MWDI and WDSD. As is demonstrated by the example problems in the manual, the two data bases are designed to compliment each other. Either may be the starting point for inquiries about the availability of water data for a specific geographical area. Either data base may provide all the information required to answer the inquiry, or both data bases may have to be accessed.

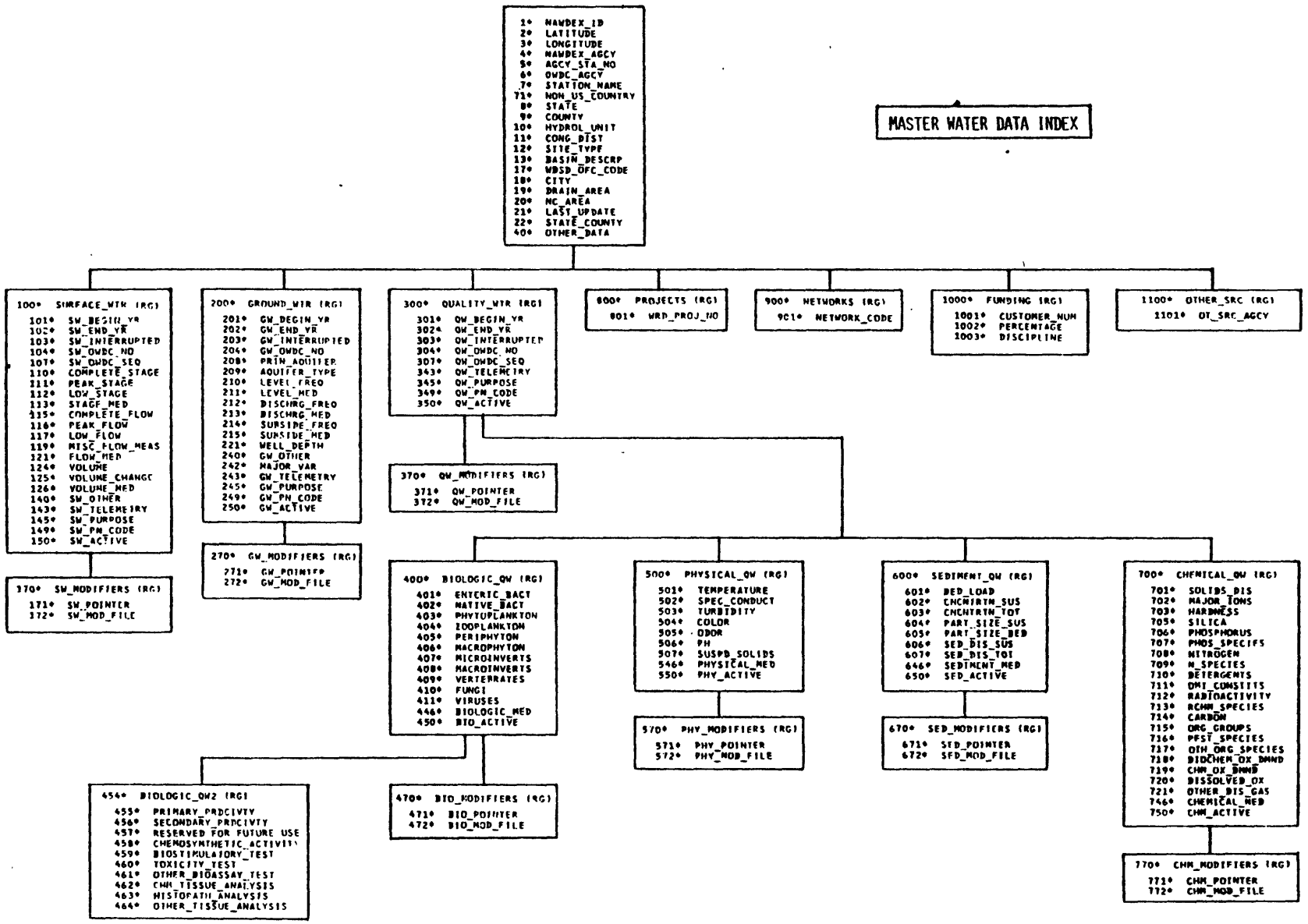
Data Retrieval

In some cases, all the information required to respond to an inquiry is available from the MWDI or WDSD only. Frequently both the MWDI and WDSD must be accessed to respond to the inquiry. Requests for information about sources of water data typically supply some or all of the following as search criteria: type of data, geographic area, period of record, and/or frequency of measurement. Information typically requested includes: who can supply the

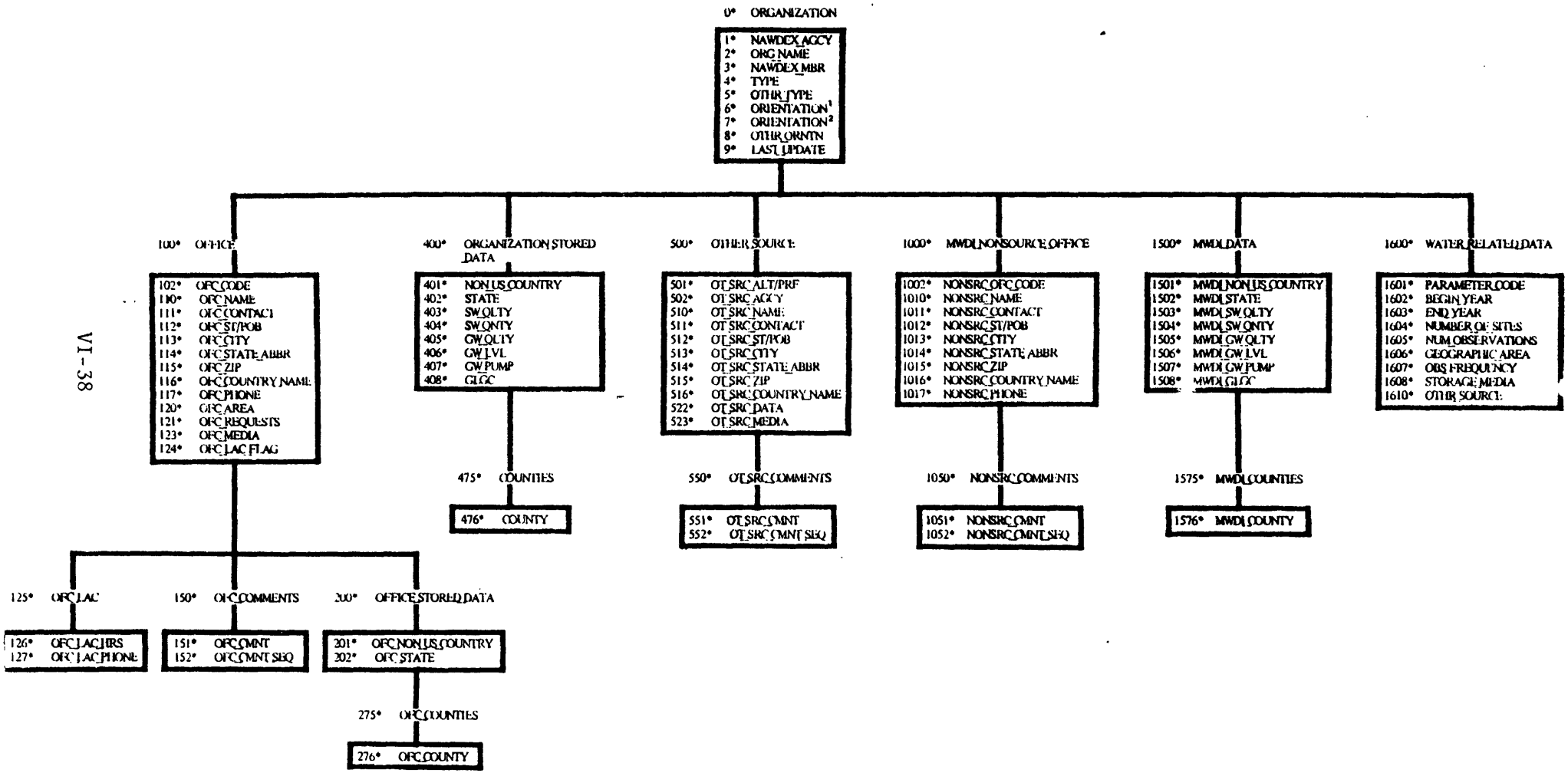
data, at how many stations are the data collected, what is the frequency of measurement, how much does the data cost, and/or what kind of related data are also available?

Therefore, upon receiving an inquiry, the first task is to determine specifically what the requestor wants to know, and whether or not the requested information is available from NAWDEX. If all or part of the information might be found in the MWDI or WDSD, the next task is to determine what criteria to use to retrieve the requested information. In some cases, further clarification from the requestor may be necessary. The next task is to determine which data base(s) must be accessed to retrieve the information, and to construct the SYSTEM 2000 commands that will accomplish the retrieval. The final task is to perform the retrieval and respond to the requstor.

In order to properly use the WDSD and MWDI data bases, whether for one's own use or for answering inquiries, a knowledge of both SYSTEM 2000 and the contents of each data base is required. To assist the NAWDEX user, appendix G describes the Master Water Data Index and appendix H describes the Water Data Sources Directory.



APPENDIX G.--HIERARCHICAL STRUCTURE OF THE MASTER WATER DATA INDEX (MWDI)



APPENDIX H.--HIERARCHICAL STRUCTURE OF THE WATER DATA SOURCES DIRECTORY (WDSD).

NAWDEX

SYSTEM 2000 Retrieval Manual

STUDENT'S EVALUATION FORM

Please complete this form and return to -

National Water Data Exchange
U.S. Geological Survey
421 National Center
Reston, Virginia 22092

1. Level of student's data processing experience:

_____ None

_____ Limited technical knowledge

_____ Proficient with one or more programming languages

2. Have you used a Data Base Management System before?

_____ Yes _____ No

If so, which one? _____

3. Have you taken a SYSTEM 2000 training course?

_____ Yes _____ No

4. How far did you read in this manual?

_____ Completed it

_____ Section A

_____ Section B

_____ Section C

_____ Section D

5. Did you notice any errors? Comment.

6. Did the manual hold your interest?

_____ Yes _____ No

7. Did you use the computer to solve any of the Student Exercises?

_____ Yes _____ No

If yes, do you think that the experience was beneficial?

8. After reading the manual, do you feel that you have a good understanding of SYSTEM 2000? Comment.

9. According to the graduated scale given here, use a number from 1 to 5 to evaluate this manual on the following points:

5 - Excellent
4 - Good
3 - Adequate
2 - Poor
1 - Useless

_____ Easy to Understand

_____ Continuity (Logical progression from one topic to the next is smooth)

_____ Organization

_____ Explanation of terminology and complexities

_____ Example Problems and Student Exercises

_____ Overall quality of Instruction

10. In your opinion, is classroom instruction or other documentation necessary to supplement this manual?

11. a.) In your opinion, what are the strong points of this manual?

b.) The weak points?

12. Do you have any suggestions as to how the quality of this manual might be improved?

13. Would you recommend this manual to someone else who needs to learn about SYSTEM 2000?