A DATA MANAGEMENT SYSTEM FOR TIME-SHARED FILE PROCESSING USING A CROSS-INDEX FILE AND SELF-DEFINING ENTRIES

E. W. Franks
System Development Corporation
Santa Monica, California

INTRODUCTION

The Time-Shared Data Management System (TDMS), under development at System Development Corporation (SDC) for use in its Research and Technology Laboratory, is intended to provide the users of the SDC Time-Sharing System with a powerful set of tools for the manipulation of large volumes of formatted, that is, not free text data. The functions to be provided include the description of data, the storage of files or data bases into the computer environment, the retrieval of the data either in response to human query or under program control for processing by other programs of the system, and the maintenance of data already loaded.

TDMS is for the use of subscribers to the Research and Technology Laboratory's facility; many of these users are not professional programmers. This imposes the requirement that the system be controlled by a nonprogrammer-user-oriented language. The data management function for which TDMS is the instrument is by no means the sole function of the computer system in the laboratory. Furthermore, it operates on a time-shared basis with the other functions performed by the computer and may therefore be used simultaneously by several users. This aspect of the environment imposes a requirement to provide responses acceptable to on-line users of the time-sharing system in circumstances where there may be many users and where the volume of data from which responses are required is very large. The organization of the data is designed to optimize on-line retrieval of this kind where the criteria for selecting data from the file are not known in advance. The two aspects of TDMS emphasized in this paper are the user-orientation features and the file organization scheme.

STRUCTURE OF TDMS

TDMS is designed to operate under the control of a Time-Sharing executive *program* using IBM S/360 computers. One feature of the system is that it will be easy to adapt the system to various models of the IBM S/360 computers. Although the system will at first be designed for model *65* IBM S/360 computers, as advances are made to the computers, the system can be adapted to increasingly sophisticated versions, such as the proposed model 67. Communication with the programs is by means of the TDMS language, which is a reactive, or dialogue, type of language, whose rules of use are always available to the user on request to the program.

The control program of TDMS can be contacted through the time-sharing system by any user from either remote or local stations. The control program of TDMS will enter a dialogue with the user. After the control program has communicated the

user's requirements to the time-sharing executive and to the computer console operators, as appropriate, the user will be granted access to various functions of the system without having to be aware of the program structure of TDMS. From the user's point of view, he is dealing only with the program he called through the time-sharing system, although, in fact, in the course of performing data management operations, he may have used and communicated with as many as half a dozen different programs.

In addition to the control program, TDMS includes a data description translator, a data load program, an on-line query and update program, a report-generator program, and a maintenance program which, in addition to performing batched updating, permits the user to create a subset of a file, to merge files and to restructure files including the computation or generation of new data elements. Space and time limitations preclude presentation of a more detailed description of the programs incorporated in TDMS. However, to describe the user orientation features and file organization scheme, it is necessary to first say something about the underlying philosophy of data upon which TDMS is based.

CONCEPT OF THE ENTRY

A TDMS data file or data base, as it is often called, is a collection of information sets or entries. Each entry contains information about one object. The object itself need not be named, but may be understood from the description provided by the name in the data base itself. Thus, a data base describing the personnel in a corporation might have one entry for each employee, and yet, within that entry, it would never be necessary to state explicitly "this is the description of a person." The kinds of data collections encountered in Command Control problems and other management problems are seldom as straightforward as the example just given. Each entry in a data base describes an object, but the objects are not all of the same kind. For example, a resources file may contain entries dealing with factories and other entries dealing with training schools. It is essential, in such circumstances, for each entry to identify the object which it describes. What TDMS allows, in fact, is the accomodation of more than one logically consistent file in the same file structure. This makes the cross-coordination of different kinds of data much easier and more efficient in an on-line environment than would be the case if separate files had to be run together and matched.

The logical structure of the TDMS entry is a collection of predefined elements or descriptors. Each entry will have a subset of these elements appropriate to the object being described. For example, the entry describing factories would have an element "GROSS PRODUCT IN THOUSANDS OF DOLLARS," but would not have the element "AVERAGE SIZE OF GRADUATING CLASS OVER LAST 10 YEARS." The reverse would be true of an entry describing a training school.

The TDMS data base is not organized into sort hierarchies such as COUNTY within STATE within COUNTRY. Provision is made, however, to accomodate naturally occurring hierarchies in the data. For example, in a data base defining tactical military organization, an entry might exist for a group. The Group Headquarters, names of staff officers, mission, and so forth, would pertain to the whole group. Each company of the group, however, might be in a different location, and each might have a specific subordinate mission. One possibility of handling this situation would be to establish separate entries for each company, each containing an element labeled "GROUP TO WHICH ASSIGNED." But, because TDMS is a general system, there would be no special magic in that particular label which would enable the system to know that these entries were really part of the group description. To ensure retrieval of the whole set, the retriever would have to know of the existence of this element and to use it as part of the retrieval key expression. To solve this problem, TDMS permits the automatic association of data connected by a natural hierarchical relationship through the device called a repeating group. A repeating group is, in effect, a set of subentries which are part of an entry. Thus, the elements in an entry which belong to one of its subordinate repeating groups may have several values within that entry, but only one value within each of the subentries. The flexibility of this device is such that, on the one hand, it will accomodate a simple multivalue element like "PROFESSIONAL ASSOCIATION MEMBERSHIP"; on the other hand, an order-of-battle file with only three basic entries, ARMY, NAVY and AIR FORCE, would contain all the subordinate organizations appearing as repeating groups within the three basic entries. As this statement implies, repeating groups may themselves contain repeating groups to any level of nesting.

DESCRIPTION OF DATA

The best way to understand how data is described is to take a hypothetical example and show the process being performed. Let us imagine that our hypothetical user is a mail-order merchant who handles a variety of merchandise. Let us suppose he has access to TDMS and the SDC Time-Sharing System through a teletype machine in his office. Let us further suppose that he is so committed to using a system that he does all of his paper work on that one teletype.

His first task is to describe his data to the system. He contacts the TDMS control through the Time-Sharing Executive. He may request a list of the functions available, but, in this case, we assume that he knows that the function he wants is called DEFINE. The DEFINE program then asks him to name his data base. He responds by typing in

COMPANY OPERATIONS

From now on he is able to refer to the descriptions he will supply and to the collected data itself by this name.

The data our merchant is about to describe will exist in two forms: outside the computer system, the data will exist as data input; inside the system the data will exist as the stored file. The following conventions exist for input data. The data input to TDMS always exists as card images on tape, or as input entered by teletype, but the scheme is the same in either case. Each data element is preceded by an identifying number field, and the set of elements and repeating groups constituting one set or entry is terminated by a special symbol selected by the user. The sequence of the elements is immaterial except that the elements in a repeating group must all be listed before the elements of the next repeating group or nonrepeating element. Thus, after receiving the name of the data base, the DEFINE program asks the user for the terminating symbol. In this case, let us say that the user chooses the term ALL. When the data is loaded, the system will know that whenever the term ALL is encountered in the input, and the term is not preceded by an identifying number, the last of the data for a particular entry has been received.

The user now proceeds to name the various elements of data he will be dealing with. He may request the system to spell out the rules for the description process, and, if he is uncertain of his typing skill, he may request the ECHO function, under which the program types back what he has input, giving him a chance to make corrections to what he has just typed before proceeding.

The elements of data are listed one at a time on the teletype. First the identifying number which will appear on the input is given. Then the name of the data element is stated. Following the name is the specification of the data type in one of the following three possible types:

NAME	(alphanumeric character string)
INTEGER	
DECIMAL	

The names of repeating groups, or subentries, are given in the same way—first the identifying number, then the name of the repeating group, then the term REPEATING GROUP (abbreviated as RG). Data elements within a repeating group are specified like other data elements, except that, following the data type specification, the name of the repeating group to which the element belongs appears. Thus the order in which elements are described does not matter. Finally, as an option, input legality check information may be inserted. This information may be a list of acceptable values, one or more ranges of values for numeric data, or a data format description. A list of some of the data input elements entered by the hypothetical merchant is shown in Example 1, below.

Example 1

```
 1  ENTRY TYPE (NAME) VALUES
        CUSTOMER PRODUCT
 2  CUSTOMER NAME (NAME)
 3  CUSTOMER CATEGORY (NAME)
        VALUES ACCOUNT PROSPECT
 4  ACCOUNT SYMBOL (NAME) FORMAT
        L999
 5  PRODUCT (NAME)
 6  ACCOUNT HISTORY (RG)
61  DATE OF ORDER (NAME IN
        ACCOUNT HISTORY) FORMAT
        09[/]99[/]99
62  AMOUNT OF ORDER (DECIMAL IN
        ACCOUNT HISTORY)
63  BILL OF MATERIAL (RG IN
        ACCOUNT HISTORY
631 MERCHANDISE (NAME IN BILL OF
        MATERIAL)
632 QUANTITY (INTEGER IN BILL OF
        MATERIAL)
633 UNIT PRICE (DECIMAL IN BILL OF
        MATERIAL)
634 ACTION (NAME IN BILL OF
```

MATERIAL) VALUES SHIPPED BACK/ORDER
635 SUBTOTAL (DECIMAL IN BILL OF MATERIAL)
636 STOCK CODE (NAME IN BILL OF MATERIAL) FORMAT LL999
7 ADDRESS (NAME)
8 CURRENT STATUS (NAME) VALUES PAID OPEN
9 BALANCE (DECIMAL)
10 CODE (NAME) FORMAT LL999
11 WAREHOUSE (NAME)
12 UNITS ON HAND (INTEGER)
13 UNITS ON ORDER (RG)
131 NUMBER ORDERED (INTEGER IN UNITS ON ORDER)
132 SOURCE (NAME IN UNITS ON ORDER)
133 ORDER NUMBER (NAME IN UNITS ON ORDER) FORMAT 009LLL
134 COST (DECIMAL IN UNITS ON ORDER)

The list is, of course, by no means the complete set of elements which would be required for the hypothetical operation. It is sufficient, however, to illustrate the features of the descriptive language.

The data base contains two types of entries—customer entries and product entries—so that billing and mailing operations and inventory control operations can be performed from the same file. The first element described, ENTRY TYPE, specifies which kind of data is included in a particular entry. Only two values are possible for this item of data, namely CUSTOMER and PRODUCT, and these values are listed, following the word VALUES for checking the legality of input. Input Element 2, identified as input by a field containing the number "2" preceding the data value, occurs only if the entry is the customer-type entry. The third element, also applicable only to the customer-type entry, shows a distinction between actual customers (value ACCOUNT) and hoped-for customers (value PROSPECT). The PROSPECT entries would be entered for mailings or in response to queries. Although no such elements are shown in the example, data about correspondence, brochure mailings, and areas of interest would probably be included in such entries.

Element 5, PRODUCT, is the first element described which would be applicable to the inventory type of entry. The next element in this category does not occur until Element 10, the product code. The legality check for this element is a format check. The L's stand for letters and the nines for numbers. Thus value AA010 would be a legal product code and value A33 would not. An additional example of format control is shown in Element 61, where the slashes in the data are enclosed in square brackets, indicating that these exact characters must occur. The example shows several repeating groups. The first of these, ACCOUNT HISTORY, occurs in customer-type entries. The user has chosen to number the inputs for the repeating group 6 as 61, 62, etc. This is an example of a user-devised convention, and is not required by TDMS. The repeating group, ACCOUNT HISTORY, itself contains a repeating group, occurring for each order recorded; this repeating group is BILL OF MATERIAL. The third repeating group is Element 13, UNITS ON ORDER, which relates to the inventory type of entry.

The list given is merely the description of the data given to TDMS. It is not the data itself. To clarify the significance of the description, two examples of input data are given below, one for a customer and one for stock.

Example 2

Input Data for a Customer-Type Entry

1) CUSTOMER 2) JOHN Q JONES
3) ACCOUNT
4) J021 6)
61) 5/21/64 62) 205.63 63)
631) TABLECLOTH 632) 17
633) 5.40 634)
SHIPPED
635) 91.80 636) TC301 63)
631) PLACE SETTINGS 632) 3
633) 37.81 634)
SHIPPED
635) 113.43 636) SV002
7) 2000 LONDELIUS ST LOS ANGELES CALIFORNIA
8) OPEN
9) 105.00

Example 3

Input Data for a Product-Type Entry

1) PRODUCT
2) TIKI FIGURES
10) TK000 11) ZELZAH 12) 205 13)
131) 50 132) PORYNESHA KK YOKOHAMA
133) 127PKK 134) 410.00

When the user has finished entering his description, he may have it presented to him for checking. He may add, delete, or change a description already made at any time by calling the REDEFINE function. He needs to know very little about the operation of the program or about computers. Technically, he need only know that data may be numeric or nonnumeric. The logic of the organization is the logic of the data itself as it appears to him. Once the data is described in these user-oriented terms, he may load the data into the TDMS system at any time by calling the LOAD program. The LOAD program expects inputs which agree with the description given. Discrepancies are logged, and the user may have them logged on-line for immediate correction. Once the data is entered, it may be called by name, and again the user does not need to know how it is stored or accessed. He may perform spot queries for fact retrieval. He may describe formats of output and call a report generator to make up bills or bookkeeping summaries. In the example given, the inventory is presented as it would appear if listed by product within a warehouse. He may want to obtain summaries by product, regardless of warehouse location, or he may want a summary of all products by warehouse. He is not restricted by the organization implicit in the way he has chosen to define his data. This freedom results from the way the data is actually organized in the computer.

TDMS DATA BASE STRUCTURE

The data structure created by TDMS when the input data is loaded is designed to optimize retrieval in an on-line environment if it is assumed that the user has no prior knowledge about what data is most likely to be retrieved or what criteria will be used to select data for retrieval. It is also assumed that retrieval will be requested from the file on the basis of some Boolean expressions given in terms of data elements and values. Such Boolean expressions define a subset of the data base, namely, those entries in the data base for which the Boolean expression is true. Frequently, however, instead of requiring the entire contents of this data subset only certain values from the qualifying will be needed. Thus, the selection path is entered with a combination of element names and values associated with them. This defines a list of entries which meet the criteria. The retrieval path is then entered with a list of entries and a list of element names for which values are required from these entries. The data base organization is designed to optimize both the selection of qualifying entries and the retrieval of the specified element values.

From the data user's point of view, the data base appears to be a collection of values to which he may wish to refer. These values have two sets of associations. In the first place each value is part of the total description of one of the objects in the data base. In the second place each value is both a value for a specified element and a member of the set of all values for that element. The TDMS organization of the data base reflects both types of value association, the element set and the object set.

The actual values are stored according to the element set relationship. That is, for each element there is a block of storage for the unique values occurring for that element. Each value is stored only once, regardless of how many entries of the input data may contain it. Associated with these lists of unique values are two other groups of lists. The first of these has the function of ordering the raw value list algebraically, or in the case of symbolic values, alphabetically. The items on the value list are stored in a random arrangement; the value list orders the values in the sequence in which they appear in the input. The ordering list is generated to speed up search by permitting the use of binary search techniques. The second group of lists associated with the blocks of values is the entry group. For each entry there is a list of the elements which were found in the entry and a reference to the place in the value list for that element where the specific value for that entry may be found. This part of the organization is represented schematically in Fig. 1.

For the purposes of selection, the ordering lists, in addition to pointing to values in the value list, also point to occurrence lists. The occurrence lists are lists of entries in which each value of each element occurs. Thus, in order to make a selection on the basis of a Boolean expression, the ordering list is searched for values which meet the various criteria. When a matching value is found, a list of entries containing this value is obtained. The various lists obtained for different parts of the Boolean expression are merged, using AND or OR logic; the result is a final list which represents a subset of the data base that meets the criteria for selection. This list of entries is then used in combination with the names of the elements to be retrieved to obtain the values of these elements from the appropriate value lists.

The entire retrieval process becomes clearer if we take an example. Let us imagine that our mail-

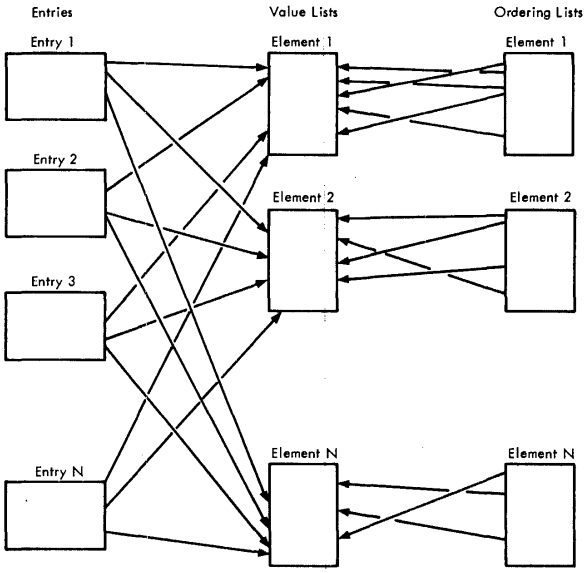

Figure 1.

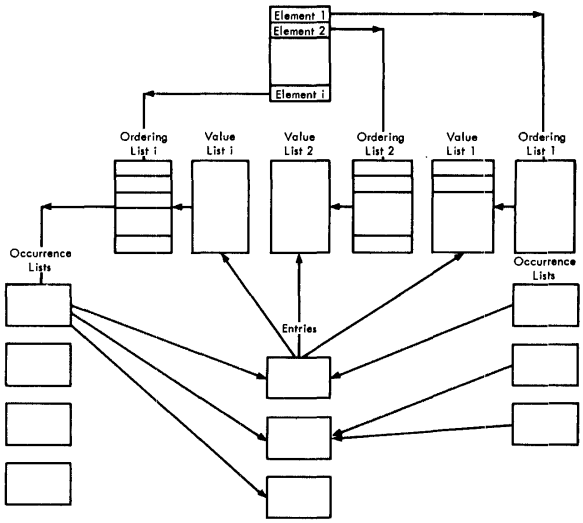

Figure 2.

order merchant wishes to obtain a list of the customers who currently owe him money—an accounts receivable list. He might do this through the TDMS QUERY program with the following request:

PRINT CUSTOMER NAME, BALANCE
WHERE CURRENT STATUS EQUALS OPEN

The subset of the data base to be selected is the set of entries which have the value OPEN for the element called CURRENT STATUS. Through the Data Definition Table the program converts the name CURRENT STATUS to the address of its ordering list. This list is accessed, and is found to have only two entries, one for the value PAID and one for the value OPEN. The value OPEN points to a list of the entries which have the value for CURRENT STATUS. Then entry references are converted to entry list addresses by means of a directory table. The qualifying entries are accessed, and, for each one, the pointers to the value lists for the elements CUSTOMER NAME and BALANCE are followed, and the values are recovered and printed.

The query in the above example is a simple and straightforward one, not involving AND and OR logic, and not concerned with the complexities of nested repeating groups. It does serve, however, to introduce the entire data base structure as created by TDMS when the data is loaded. Figure 2 shows this structure schematically.

The route is from definition table to ordering list. The ordering list permits an efficient examination of the value list for qualifying values. For each qualifying value there is an occurrence list. The occurrence list points to the data base entry where the value occurs. Each entry, in turn, points to the value lists for all the elements present in it. The occurrence list actually points indirectly to the entry via a directory of entry addresses. In cases where a value occurs only once for a given element, the ordering list bypasses the occurrence list and points immediately to the directory to save storage space.

So far little has been said about the directory table, which, in its simplest form, simply lists the address of each entry in the sequence in which the entries were loaded. In cases where repeating groups are involved, however, the directory assumes greater importance, since it is here that the hierarchical restrictions imposed by repeating groups are observed. This concept is best explained by means of an example. Again using the merchandising data base, let us imagine that the user wishes to obtain a list of customers who have ordered sardines in quantities of 100 cans or more as part of an order totaling $100 or more. In this case, "sardines" is a potential value for the element MERCHANDISE and "100 cans or more" represents a potential value for the element QUANTITY. Both elements occur in the repeating group BILL OF MATERIAL. Both must occur in the same group. In other words a value of 100 for QUANTITY is not sufficient to qualify the entry unless it is directly as-

sociated with the value SARDINES for the element MERCHANDISE. Furthermore, the co-occurrence of these two values does not qualify the entry unless the total order of which the sardines are a part equals or exceeds $100. It is necessary then to find entries in which one of the values for AMOUNT OF ORDER in the repeating group ACCOUNT HISTORY is equal to or greater than 100 at the same time as the values for MERCHANDISE and QUANTITY in the BILL OF MATERIAL for that particular order meet the criteria respectively of "sardines" and "equal to or greater than" 100.

This complex matching problem is solved by carrying an entry for each repeating group, as well as for each data base entry proper in the directory table. In the case of directory entries for repeating groups, instead of an entry address there is a reference to the directory entry for the next higher level in the hierarchy. In this way it is possible to determine whether values for two elements of the same repeating group (in the example SARDINES and 100 or more cans) actually occur together. If they do, the references from their respective occurrence tables will be the same. Then, by following the pointer from this directory entry to the next higher level, namely to the particular order in ACCOUNT HISTORY to which it belongs, it is possible to see whether or not the total order was equal to or greater than 100 dollars. The entry containing this information will qualify if an entry number in the directory for an occurrence of AMOUNT OF ORDER greater than or equal to 100 dollars is the same as the next higher entry pointed to by the directory entry meeting the MERCHANDISE and QUANTITY criteria.

The fact that the elements dealt with are parts of repeating groups is determined by the selection program from the definition table, as is the relative level in the hierarchy of each repeating group. In summary, the following is the path followed in response to a request phrased

```
PRINT CUSTOMER NAME WHERE
    AMOUNT OF ORDER GQ
100 AND MERCHANDISE EQUALS
    SARDINES
AND QUANTITY GQ 100
```

The program determines that the selection criteria elements are members of repeating groups. Starting at the highest level where the element appears, it accumulates a list of entry numbers in the directory table, for which AMOUNT OF ORDER qualifies. It then accumulates a list of entries for MERCHANDISE equal to SARDINES and a list of entries for QUANTITY greater than or equal to 100. These last two lists are ANDed together to eliminate entries with insufficient sardines and entries with a sufficient quantity but the wrong merchandise. The resulting intersection list is then converted to the next higher level by substituting the "up" pointers from the directory. The converted list is then ANDed with the AMOUNT OF ORDER list to produce a list of fully qualifying entries. This is not, however, the final step, since the AMOUNT OF ORDER list contains entries for a repeating group, ACCOUNT HISTORY. This must be converted to actual entry references again by substituting the "up" links, which now results in a list of basic entries. These entries are retrieved, and the output values, in this case, CUSTOMER NAME, are retrieved and printed exactly as in the first simple example.

BACKGROUND

The data handling techniques of TDMS have evolved over several years of research and experiment conducted at SDC, and the new system benefits from experience gained elsewhere. In particular, the idea of the cross-reference file was developed and tested in an experimental data management system called LUCID, and was refined and expanded in TSS-LUCID (Time-Sharing LUCID) which is currently in operation at SDC on the IBM ANFS/Q32 computer under the Time-Sharing System. The cross-reference file is that part of the data structure which consists of the value lists, the ordering lists and the entry directory table. The concept, and indeed, the name of the repeating group is derived from the ADAM system of the MITRE Corporation. The inadequacy of LUCID in dealing with the natural hierarchies occurring in data prompted this borrowing. What is entirely new in TDMS is the entry structure which has been termed the "self-defining entry." In LUCID the entry association is determined solely from storage juxtaposition. The values are tightly packed in the entries. The values also occur in the value list, thus duplicating storage requirements. Furthermore, much additional storage space is required to accommodate bits of storage assigned to data elements not actually present. In the case of multiple value elements with assigned bit locations, this arrangement requires a great deal of space for empty data

base storage. Most important, however, in motivating the development of the new structure, are time considerations. In a general system the locations of packed data elements are known to the program through parameter tables. An average of more than 100 machine instructions required to convert such parameters to an actual retrieval, and, in iterative operations, the process is likely to be very slow. TDMS does away with packing parameters so that everything is standardized, and in all probability, the number of instructions required to move a data element is always fewer than ten.

OTHER OPERATIONS

The heart of TDMS has been described in some detail. The data description language has been presented to give some idea of the essential simplicity of the approach to the system and, thus, of its suitability for nonprogrammer users. The operation of retrieval has been explained with some examples of the on-line query language being used as illustrations. The retrieval mechanism is the same throughout the system, whether it is triggered by an on-line query or by the execution of a report-generator function. What has not been covered is the arithmetic capability of the system. The extent of this capability is illustrated by the following query which shows that the system accepts arithmetic expressions involving elements of data both as output specifications and as selection criteria.

```
PRINT SUM OF HOURS WORDS
    * HOURLY WAGE
WHERE 1966 - BIRTHDATE
GR 21
```

To optimize operations such as the above without sacrificing efficiency in cases of simple retrieval, the value lists for numeric elements contain both the symbolic form of the values originally input as well as binary representations of them in either integer of floating point format. In this way the original value can be retrieved and printed without going through a conversion routine, and arithmetic and magnitude comparisons can be made in the binary mode.

CONCLUSIONS

TDMS is a generalized system which makes no a priori assumptions about the way in which the data will be used. In cases where this is known, the data can be converted to the more conventional hierarchical format by the maintenance program so that the efficiency of specific usages can be maximized. Nevertheless, the basically general approach is sound. The life expectancy of a special-purpose data management program is short, and in terms of cost effectiveness, likely to be very poor. Our experience has been that the collectors and users of data approach their problems initially with a somewhat vague and largely intuitive notion of the uses to which a data base will be put. It is only as they begin to use the data that its full utility becomes apparent. TDMS is an attempt to give users a facility which does not preclude the easy and inexpensive evolution of data management procedures, and which, at the same time, is remarkably efficient as generalized programs go. It is designed for the nonprogrammer user. We do not like to say it is for the unsophisticated user, because the more sophisticated he is in the terms of his own data and his own problems, the better TDMS will serve him.