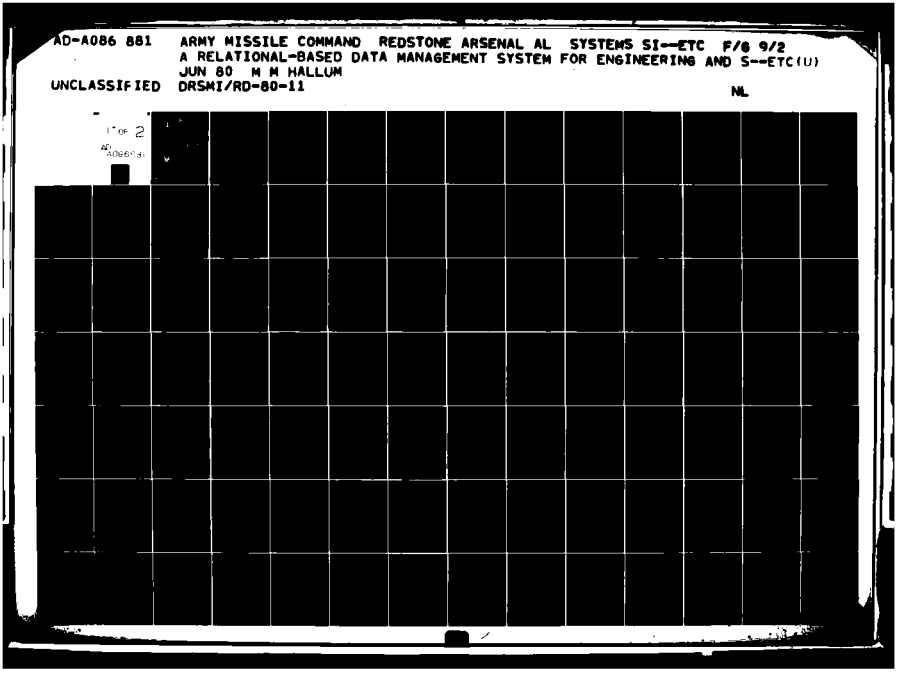
AD-A086 881 ARMY MISSILE COMMAND REDSTONE ARSENAL AL SYSTEMS SI--ETC F/G 9/2
A RELATIONAL-BASED DATA MANAGEMENT SYSTEM FOR ENGINEERING AND S--ETC(U)
JUN 80 M M HALLUM
UNCLASSIFIED DRSMI/RD-80-11 NL



LEVEL II

12

TECHNICAL REPORT RD-80-11

A RELATIONAL-BASED DATA MANAGEMENT SYSTEM FOR ENGINEERING AND SCIENTIFIC APPLICATION

Maurice M. Hallum, III
Systems Simulation and Development Directorate
US Army Missile Laboratory

June 1980

U.S. ARMY MISSILE COMMAND
Redstone Arsenal, Alabama 35809

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

DISPOSITION INSTRUCTIONS

DESTROY THIS REPORT WHEN IT IS NO LONGER NEEDED. DO NOT RETURN IT TO THE ORIGINATOR.

DISCLAIMER

THE FINDINGS IN THIS REPORT ARE NOT TO BE CONSTRUED AS AN OFFICIAL DEPARTMENT OF THE ARMY POSITION UNLESS SO DESIGNATED BY OTHER AUTHORIZED DOCUMENTS.

TRADE NAMES

USE OF TRADE NAMES OR MANUFACTURERS IN THIS REPORT DOES NOT CONSTITUTE AN OFFICIAL INDORSEMENT OR APPROVAL OF THE USE OF SUCH COMMERCIAL HARDWARE OR SOFTWARE.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: Technical Report RD-80-11	2. GOVT ACCESSION NO.: AD-A086 881
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): A RELATIONAL-BASED DATA MANAGEMENT SYSTEM FOR ENGINEERING AND SCIENTIFIC APPLICATION	5. TYPE OF REPORT & PERIOD COVERED: Technical Report
6. PERFORMING ORG. REPORT NUMBER	
7. AUTHOR(s): Maurice M. Hallum, III	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Commander, US Army Missile Command, Attn: DRSMI-RD, Redstone Arsenal, AL 35809	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Commander, US Army Missile Command, Attn: DRSMI-RPT, Redstone Arsenal, AL 35809	12. REPORT DATE: June 1980 13. NUMBER OF PAGES: 130
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

(17) DRSMI/RD-80-11

18. SUPPLEMENTARY NOTES

This report was previously published as a dissertation for Southeastern Institute of Technology, Huntsville, Alabama, dated 15 July 1979.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Data base management system
Relational data model
Relational data base management system
Inquiry language
Computer data base system

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The description and development of a user friendly Data Management System based on the relational data model is presented. The system is a self-contained Data Management System programmed in FORTRAN for the Interdata 8/32. A file system, inquiry language, and a demonstration of the operational system is also presented. The system is intended for use in a scientific and over

(Continued)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED 417706

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. Abstract (Continued)

engineering environment designed for the casual user. Background data and a general discussion of data models are also included.

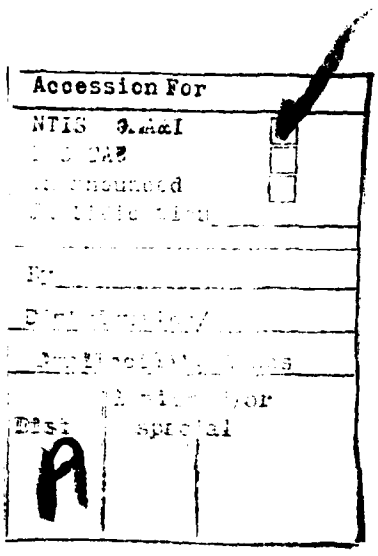


TABLE OF CONTENTS

Table of Contents (Continued)

Table of Contents (Concluded)

LIST OF FIGURES

List of Figures (Concluded)

CHAPTER I

INTRODUCTION

A. Overview of Scientific and Engineering Data

The engineering and scientific community generates data from many varied sources. These sources are characterized by results generated in areas such as testing, designing, and simulating. The data generated also vary greatly in content, controls that are applied to the data, degree of data validity (correctness), and intended use.

The disciplines which generate the data also have multiple uses to which the data is to be put. Many types of an analysis are to be performed. The actual form the data will take at the conclusion of its generation and analysis is not certain until the end of the process. The large scale testing of equipments or systems, the gathering of data for basic and applied research, and the large scale simulation and analysis of systems are some examples of areas that generate this type of data.

The scientific and engineering forms of data differ from the business area. In a conference on data management, panel discussions on the differences between the business

data and scientific data best defines the essential differences, [Lopatka, 1978], [Fenves, 1978], [Sobieski, 1978].

1) Terminology (FORTRAN - COBOL)
2) Record Size Orientation (Salary - ARRAY)
3) Iteration
4) Data Ownership
5) Unpredictable Growth and Organization
6) Multiple Networks of Data

B. Environment

The environment that typifies the generation and analysis of scientific and engineering data is illustrated by a large simulation facility at the US Army Missile Command [Systems Simulation Directorate, 1979]. The user must function in a real-time environment to define test to be performed, establish parameters to be varied and their ranges, and perform preliminary analysis to establish additional investigative areas.

In order to be prepared to properly react in this research testing environment of large, complex, sophisticated equipment, it is necessary that a complete plan of anticipated tests be generated well in advance of actual testing. This plan is then used to have all the appropriate test environments available in minimum time when the tests are to be made. In general, not all the planned tests are performed, nor are all the actual tests required planned

ahead of time. This testing environment leads to a most dynamic situation during the time the tests are being performed. Tests can be performed that are not configured properly; tests can be performed more than once; tests combinations that are deemed needed at test time, but thought not to be planned, may indeed be planned; set up parameters can not be located in the maze of paper during the pressures of actual test performance. These are but a few of the data dilemma that can arise.

After the tests are performed, the results must be dealt with in an orderly fashion. In general, there are two types of results from tests. First, there are the immediate results that appear as identifiable quantities or states at the end of the tests. There are both quantitative and qualitative results. An example of qualitative results are the real-time observations made and reported during the tests. They are typified by such terms as REMARK or COMMENTS. Often times the qualitative results dictate the fate of the total tests. The second result type is that recorded by various techniques to be reduced (or not reduced) as need be.

C. Data Characteristics

The data generated in the environment just described is in a tabular form. The size and content of the tables are established by the particular environment.

A typical test matrix form is shown in Table 1. This is generated well in advance of actual testing. The number of tests that will be performed or the number added is a function of many things. Suffice it to say that it is dynamic in the testing environment. It can be altered by adding columns of test parameters and/or rows of test conditions.

The run-time test observations are represented by those shown in Table 2. The results here are generated on the "fly." Again, columns can be added during test and the rows are added as test proceeds. Table 3 can be generated later during test analysis from recorded data or it can be an additional table similar to Table 2. To date the data in Tables 1, 2, and 3 have been maintained manually.

The specific test environment that creates this application is the time-critical-real-time simulation of missile systems, which include missile flight hardware-in-the-loop (HWIL) as part of the s mulation. Other "on-line" testing environments have similar situations; wind tunnel testing and structural testing are examples.

In addition to the data as shown in Tables 1, 2, and 3, other data is taken in forms of strip chart recording, digital tape records, or analog tape recordings. These data are analyzed in various ways, depending on the test. Fast Fourier Analysis, non-stationary time-varying series analysis, and power spectral analysis are a few examples. These

analyses lead to other data to be analyzed, creating more tables similar to Tables 1, 2, and 3 or to additional entries into these same tables.

The magnitude of the data generated in the simulation environment is large. The test matrix typified by Table 1 can easily have 500 cases. Many of these cases will not be run because the planned parameter of interest is having no effect on the observed results. The number of runs that may be made and entered into Table 2 is in the thousands. The amount of data grows with time as the system is tested in ever-varying environments. The growth continues as long as the system is actively in the field.

After the data is taken, any of several things may happen to the data. A cursory analysis is performed with more detailed analysis to be performed later; the data is put away to be added to later; or the testing is continued with additional hardware types.

In the past, notebooks of data depicted by Tables 1, 2, and 3 have been generated. Manual reduction and analysis of the data is out of the question. The most trivial question such as "Have we run a case where the temperature was __?__, the range was __?__ , and the speed __?__?" draws responses of "I think so", "Yes, but the data cannot be found", or worse yet - "Who knows?" Much time can be spent rerunning

TABLE 1. TEST-CONDITIONS

RUN CODE	TEMPER-ATURE °C	WEIGHT GRAMS	SPEED ft/sec	RANGE ft	FLUSH TYPE
100791	0.0	100.2	50.0	20.0	ROD
100792	10.0	100.6	10.0	20.0	ROD
100793	20.0	100.9	50.0	20.0	RAKE
100794	30.0	100.2	10.0	20.0	RAKE
100795	40.0	100.6	50.0	20.0	Plunger
100796	50.0	100.9	10.0	20.0	Plunger
100797	60.0	100.2	50.0	20.0	Plunger
100798	70.0	110.0	100.0	20.0	ROD

TABLE 2. TEST-OBSERVATIONS

RUN CODE	FINAL COLOR	FINAL ACCELERATION	DATE RUN	REMARKS
100791	Red	10.0	30 Jun 79	Flip Valve Leakage
100797	Red	10.1	30 Jun 79	
100794	Blue	----	30 Jun 79	
100798	Clear	11.0	30 Jun 79	Bad Run
100795	Black	10.1	3 Jul 79	
100798	Clear	12.0	30 Jul 79	Repeat of 100798

TABLE 3. TEST RESULTS

RUN CODE	TIME sec	TEMPERATURE °C	FLOW RATE in^3/sec
100797	0.0	60.0	10
100797	1.0	55.0	9
100797	2.0	51.0	8
100797	3.0	48.0	7
100797	4.0	46.0	6
100797	5.0	45.0	5
100797	6.0	44.5	4
100797	7.0	44.25	3
100797	8.0	44.125	2

data to provide some baseline results because the original results cannot be found or recovered.

The characteristics of the data can be modeled according to the basic structure of the data.

D. Data Models

There are three basic types of structures or functional classifications of data models. These are the hierarchical, the network, and the relational types. Each of these classes have their appropriate applications that will be discussed later. The data model, discussed in the following sections, is defined to be the type of data structure used to represent information.

1. Hierarchical Model. The hierarchical model is probably the original form of the data model. As its name implies, it is a hierarchical structure where the data occurrences are represented as nodes of a tree in a strictly owner-member relationship.

2. Network Model. In the network model occurrences are represented as modes of a network, chained together by named, directed arcs. The directed arcs are logical links between the various entities of the data. These logical links are traversed in specific directions in order to navigate through the data. The network model is aligned with the work of Bachman [1964, 1965]. Strictly speaking the hierarchical model is a restricted form of the network model

where a member can have one and only one owner. The naming of the link is unnecessary since there is only one link.

3. Relational Model. The relational data model is founded in the landmark paper by Codd [1970]. The relational model views data as a group of tables (flat files). The ordered relationship of the columns (attributes) containing the data is not of significance. There is no hierarchical or linking relationship between tables.

These models are presented and discussed in a rigorous fashion by both Date [1975] and Martin [1975]. An excellent tutorial is also provided by the Institute of Electrical Engineers [Bernstein, 1978].

E. Managing Data

The management of scientific and engineering data has several unique considerations. First, in general there is not a staff of programmers available to support the management of data. The user of the data establishes the path by which the data arrives at its ultimate objective. Whether the end-product is worthwhile or the degree to which it is worthwhile is a function of many things. Some of the factors which effect the user's ability to manage the data is basic to the nature of the data. The primary factors that establish the nature are:

1) Unpredictable Growth

2) Multiple Networks of Data

3) User Controls (Data Ownership)

4) Data Record Types and Organization

5) Volume of Attributes

There are three trend areas in the field of data management (and somewhat in computer hardware) that are influencing the way in which data will be managed. First is the casual user, second is the relational model, and third is the availability of relatively inexpensive, yet powerful computer hardware.

1. Computer Hardware. The advancement in computer hardware is so rapid at this time that whatever may be written today is obsolete tomorrow. Some general observations may be made that are valid, however. First and probably the most significant is the emergence and growth in the minicomputer and microcomputer field. The physical size and cost are on the decrease with the computing power on the increase. This trend is tending to counter the trend toward larger operating systems and large computer systems.

2. Casual User. Codd [1974] and others [Lough, 1977] have established five classes or types of users in the field of data management. They are:

a) Systems Analysts/Data Base

b) Application Programmers

c) On-Line Job-Trained Users

d) Researchers

e) Casual Users

The casual user includes lawyers, accountants, analysts, engineers, planners, managers, etc. In the future, this list may indeed include everyone.

The trend to this casual user has vastly shifted in the past decade. Some commercial Data Base Management System (DBMS) vendors view developing the casual user as a necessity for survival. Sherman [1978] in his panel remarks to the Engineer and Scientific Data Management Conference stated:

> ...and now there seems to be a trend toward smaller systems. I think one of the reasons you have that trend is because a smaller operating system or a smaller computer is essentially less complex. It might not do everything the big system does, but it doesn't take [you over and], you don't feel like you're dealing with a monster....

Sherman's remarks were directed equally toward the increase in minicomputer usage.

3. Relational Modeling of Data. The characteristics of Scientific and Engineering data coincide with the relational model of data. The relational concept of data is new, in a formal sense, as documented by Codd [1970]. The relational concept is the most pleasing from a formal, theoretical point of view. The rigorous mathematical basis for the relational data model is founded in elementary relation theory and in set theory. This foundation allows development of

manipulation definitions and a sound theoretical bases for various implementations.

The relational data model provides the capability to achieve "data independence" [Date, 1975], [Lough, 1977], [Popa, 1976]. Data independence is the aspect of being able to store and retrieve data without regard to where it went, how it's stored, or how it gets back. The basis for this is the user's logical view of the data without regard to the physical view. This data independence provides the means to develop general retrieval and manipulation languages.

F. Data Management Systems

A Data Base Management System (DBMS) is a system for storing, maintaining, and retrieving data. There are many "systems" that satisfy this definition - a library, a cabinet filing system, a computer file system. The ease of use of these systems depends on the user's familiarity and the system's own inherent properties. The storage, retrieval, and maintenance of data by a DBMS says nothing of the reporting, analyzing, safeguarding, and sharing of that data. The retrieval of data on a selective basis implies that the data has some special significance that deserves reporting, analyzing, or sharing in some understandable fashion. The selection process may or may not be part of the DBMS. The addition of reporting, analyzing, safeguarding, etc., features to a DBMS then makes it a Data

Management System (DMS). What features are added depends on the organization using the DMS. The decision of "when and how to go "data base" is aptly discussed by Davis [1976]. Though Davis' discussion is not in detail, it provides an excellent primer.

1. Data Base Models and Implementation. The DBMS as one might suspect has its genesis in the business or commercial data processing world. Stock control, payrolls, customer list, inventory control, etc., are typical of the list of uses to which automated processing techniques were first applied. These applications take the entire data set and perform operations, i.e., everyone was paid, everyone was billed, etc. The next stage of development is exemplified in the airlines reservation systems developed in the early 1960's. Specific application programs were developed based on a well defined set of inputs and outputs. The data systems of the 1960's were specific processing applications based on each task to be performed. The primary objective was very high processing efficiency. Response of these systems to even simple, non-standard queries was intolerable. The surveys of data management and file systems such as those by MITRE corporation [Fry, 1969 and Koehr, 1973] are numerous.

The implementation of DBMS has begun to change directions. Instead of the specific application schemes, the

systems are emerging as more general software packages. These packages, though more general than previous applications, are still oriented to a specific type of enterprise and are tailored for specific installations. Though less efficient computationally, these systems have and are gaining widespread implementations in such products as IBM's IMS, CINCOM's TOTAL, Informatic's MARK IV, MRI's Systems 2000, and Honeywell's I-D-S/III.

As noted previously, there are three types of data models on which the majority of DBMS are based. These are the hierarchical, network, and the relational models.

The hierarchical approach is implemented as the IBM Information Management System (IMS), Informatics MARK IV, MRI's Systems 2000, and others. A network DBMS has been announced or implemented by every major computer manufacturer with the exception of IBM. The Honeywell I-D-S/II (developed from 1964 General Electric I-D-S [Bachman, 1965, 1966]) and CINCOM's TOTAL are representative of the presently available network based systems. The relational model has several research implementations, such as MacAims [Goldstein, 1970], INGRES [Held, 1975], SQUARE [Boyce, 1975], and SEQUEL [Chamberlin, 1974]. With the possible exception of Relational Inquiry and Storage System (RISS) [McLeod, 1975], there are no commercially produced DBMS based on the relational approach. The qualification,

possible, attached to the RISS is because it does not contain all relational operations to make it complete in a relational sense. The set operations intersection, union, and difference are not present. The foundation of RISS is basically relational in concept however.

An excellent overview of the status of DBMS is provided by Fry [1975, 1976]. In these papers Fry further divides DBMS on the bases of whether it is self-contained or dependent on a host language. Self-contained capabilities are tools that do not require programming skills to perform.

2. DMS/User Considerations. The DBMS user interface is generally an applications programmer of some type. The term user, as used here, excludes the airlines terminal reservation system, department store sales-clerk type of user, but refers to the non-standard query user. The term non-standard query user implies queries that require quantitative bounds to be applied before the query can be satisfied. The standard query user has available to him only preprogrammed retrievals selected by a terminal push-button or sense switch.

A new interface is now being studied, proposed, and implemented. This interface is termed the "Query Language." Several research efforts have been cited such as INGRES [Held, 1975], SEQUEL [Chamberlin, 1974], and CUPID

[McDonald, 1975]. In addition to these based on the relational algebra and calculus, there are attempts at the "Natural Language" approach. An example is "RENDEZVOUS" [Codd, 1974]. An excellent treatise on the query languages and their relative worths is given in the paper by Lough and Burns [1977]. This paper provides a method for evaluating various query languages in terms of such parameters as:

a) Completeness

b) Mathematical Sophistication

c) Learnability

d) Procedurality

e) Level*

The combination of minicomputers and query languages are beginning to appear in such areas as RISS [McLeod, 1978], DB85 [Lien, 1977], and the announcement by DEC of the DBMS-11.

The area of data ownership is unique in the engineering and scientific environments. While the term ownership may be a mis-label, it is representative of the unique situation. Take, as an example, a set of test data. These data, as a test, may belong to an organization. But the data has limitations, restrictions, and qualifications that must, in some fashion, forever accompany the test data. Retrieval on

*Refers to the number of decisions required of the user as contrasted to those made by the language.

the data qualifications is unique to this type of application.

One final interesting discussion in DBMS is the proposition of data base management system architectured computers. While this is a viable concept and, from a theoretical point of view, has a defendable position, from a practical standpoint, it is beyond the state-of-the-art today. The most probable step prior to actual data base architecture is the back-end concept. The principal feature of the back-end DBMS is that mini-computers act as an interface and perform the data management functions. Bell Labs [Canaday, 1974] has configured an experimental system termed the XDMS. Software requirements to support the back-end data base computer (DBC) is reported as work performed for the Office of Naval Research [Benerjee, 1977]. The Office of Naval Research effort is to develop the software requirements for the hierarchical, network, and relational models. Only the first two have been generated thus far. These studies assert that the back-end DBC can provide significantly improved performance over such systems as IMS.

3. Scientific and Engineering DMS Configuration. A Data Management System providing storage, maintenance, retrieval, reporting, analysis features, query language, and other manipulative features are required for the engineering

and scientific research and development environment. The DMS environment for the engineering application must allow for unpredictable growth and organization, multiple networks of data or data interaction, nonrepeating retrieval criteria, data that requires caveats to maintain credibility. This environment strongly supports the development of the DMS to be used by the "casual user", thus requiring a high degree of data independence.

While the "casual user" implies any user, the opportunity to even have such a DMS available suggests professionalism. This line of thinking leads to a DMS philosophy that provides the user with a great deal of freedom of interaction with the DMS. The user is then given the tools to perform various data management functions; he is free to use the basic tools in various ways to accomplish the task at hand. The development and implementation of such a DMS can best be realized through the relational data model concept.

The recent advancements in the area of DMS centers around the relational data model spawning many designs for relational data base systems. The primary drawback to the relational DBMS is, as yet, inefficient retrieval to queries. This inefficiency must be weighed against the data independence achieved to access the desire to implement a relational data base system. The relational view of data

greatly facilitates the casual user and associated query languages.

The objective of this study was to provide a Relational-Data-Model-Based Data Management System which meets these requirements. The system provides a stand-alone, interactive data base management capability to the nonprogrammer. The general concept of the implementation is to provide simple, yet powerful, tools to the user. This will maximize the user's heuristic abilities while utilizing the enormous manipulative abilities of the computer.

A second primary requirement is to provide "user level" independence such that changes at a given level do not affect the levels above or below it. This will allow for growth and alterations as necessary to accommodate new hardware or implementation techniques as they become available.

The system developed is a self-contained DMS. A functional block diagram of the system is shown in Figure 1. The query language provides the interface between the user and the DMS. The data sub-language facilities are shown in their functional form and depict the capabilities afforded the user. The actual data storage and retrieval is accomplished using the Data Base Management System (DBMS). The DBMS provides the interface to the physical data storage devices used. Data reporting is accomplished through the

interaction of the data selection and data analysis features via the Inquiry Language.

The relational data model developed by Codd [1970] is presented in Chapter II and provide the foundation for this development. The background provided by this section makes the DMS development in the succeeding section readily linked to the theoretical basis. The overview of the DMS operational concept, target machine, user interface, and file and data structures are described in Chapter III. The DMS capabilities that provide the maintenance, manipulation, and retrieval functions are then presented. A demonstration of the salient features of the DMS is given using the actual system as developed. Limitations of the implementation are then presented in terms of hardware associated with the specific target machine.

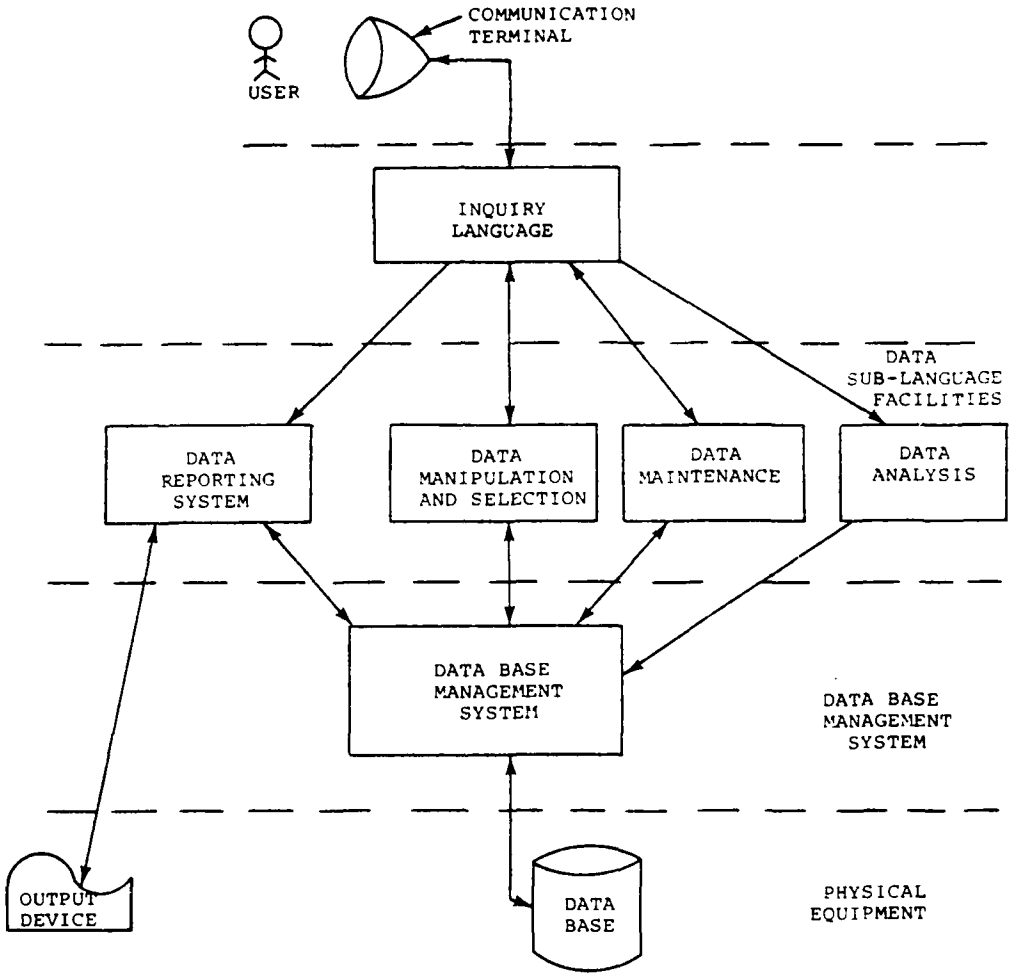


Figure 1. Data management system.

CHAPTER II

RELATIONAL DATA BASE CONCEPT

A. General

The overall trend in data base management is toward data content and not the structure of the data. The methods of storing and representing and other implementation features are becoming of ever-decreasing concern. The trend is away from these details and is termed "data independence." As mentioned previously, the hierarchical and network concepts of data base systems have been developed to a high degree of proficiency. These two concepts tie the data in a data base system to a structure. As described by Chamberlin [1976], information may be represented in at least three ways: 1) by contents of records (i.e., Smith's employee record has Dept. No. in it), 2) by the connection between records (i.e., Smith's employee record occurs within a hierarchy below his department record), and 3) by the ordering of records (i.e., all sales records occur in chronological order). User query requests then come in the form of: Find the next record of given set. Fundamentally it requires the

User to know the structure of the data and to maintain a knowledge of where he is in the data base.

The relational concept or model frees the user of the requirement to know the structure. This allows the use of higher level, non-procedural languages through which he may communicate with the data. This also allows basic data storage and implementation changes to be made without impacting the user.

In his original paper developing the relational concept, E. F. Codd [1970] introduced the key concepts defining a data sub-language as a set of facilities which allow the retrieval of subsets of data from a data bank. The paper introduces a set of relationally functional operators such as, join, projection, etc., which have been developed into the relational algebra. These operators allow the manipulation of data modeled in the relational form. The concepts of redundancy and consistency are also introduced by Codd which lay the ground work for normalization. The redundancy that may exist in the representation of relations leads to the problem of consistency between these relations. The problem then becomes a practical matter or properly updating, deleting, and inserting data in the data base.

B. Definitions

A relation (R) is a set of n-tuples (rows) where the first element is an element of C1, the second element is from C2, and so on, where C1, C2, C3,...Cm (not necessarily distinct) are given sets. The set C represent the various "domains" of R. The magnitude of m is the "order" or "degree" of the relation. The number n (number of rows) is called the relations "cardinality."

A relation is generally viewed conceptually as a table (Tables 1, 2, and 3) or array of data. The array which represents a relation exhibits the following properties:

1) No two rows are identical.
2) The ordering of rows is not significant.
3) Each row represents an n-tuple of the relation.
4) The ordering of columns is significant for columns with unnamed attributes.

The domain of the columns of the relation is partically conveyed by the naming of the column. It is important to distinguish between the column attributes (role-name as sometimes called) and the domain. The example below of the presidential election relation serves to illustrate.

YEAR	WINNER-NAME	LOSER-NAME
1960	Kennedy	Nixon
1964	Johnson	Goldwater
1968	Nixon	Humphrey
1972	Nixon	McGovern
1976	Carter	Ford

While the domain of columns 2 and 3 are both NAME, each column has an attribute name which describes its meaning in the relation. If columns are unnamed, then their order is of significance. If the names were reversed in the table above without reversing the role name, the table would convey a completely different (and incorrect) meaning.

The individual entries of each tuple are called components or sometimes elements of a relation. While relations tuples are viewed as rows, they are sometimes called records. The term record is to be avoided (although it was not successfully avoided in all cases here) because of its association with a physical computer record. This association is incorrect. If the term record is used, it should be viewed as a logical record at best.

The concept of keys is the means of specifying a column component as a unique identifier for that relations tuples. In Table 1, the column RUN CODE uniquely identifies each tuples. Such a domain is termed a primary key. When a relation has more than one domain that uniquely identifies each tuple, one of the domains can be arbitrarily selected as the primary key.

Keys provide a technique for cross-referencing data of one relation (R) to a second relation (S). If the primary

key of relation S is also contained in relation R and is not the primary key of R, this key in relation R is called a foreign key. The primary key of Table 1 is contained in Table 3 as a foreign key. The primary key in Table 3 is the time domain which uniquely identifies each tuple.

C. Normalization

The normalization of a relation is to allow it to be properly represented as a simple, two-dimensional, column-homogeneous array. There are three levels of normalization. Simply stated, the first normal form states that no component of a relation can itself be a relation in addition to the four properties already mentioned. If this occurs, the offending component should be broken out as a relation itself, or the relations should be reconstructed so that the components of neither relation can be decomposed.

Normalization has its most profound impact in the area of Update, Insertion, and Deletion of relational data in the array form. The normalization process evolves to three levels, termed the first normal form, second normal form, and the third normal form. The normalization process yields a relation that is easier to use because the three anomalies are gone. The relation is also more informative because of its simpler representation in that it has fewer columns and table entries. The normalization process fosters more but simpler relations.

The second normal form is primarily just a step toward achieving the third normal form. A relation is in second normal form if it is in first normal form and each non-candidate key domain is fully dependent on every candidate key in that relation. In the presidential table illustration, both the winner-name and loser-name role-names are meaningful and dependent on the year domain.

The third normal form is the final stage of normalization. To be in a third normal form, a relation must be in second normal form and every role-name that is not a candidate key is not transitively dependent on each candidate key of the relation. If columns C1, C2, and C3 are members of a relation R and C2 is fully dependent on C1 and C3 is fully dependent on C2, then C3 is transitively dependent on C1. For example, if there were a column with attribute LOSER-ADDRESS in the presidential election illustration, this column is fully dependent on the LOSER-NAME, which is in turn fully dependent on the YEAR; thus the LOSER-ADDRESS is transitively dependent on the YEAR. This points out the need for a second relation for presidential candidate addresses instead of including it in this relationship.

The updating anomaly is that the modification of one particular attribute (component) of a given tuple may require the updating of additional tuples. For example, if the table below were found to be in error and the test for

Run Code 1003 were performed in Otherplace-USA instead of Someplace-USA, then two tuples would require changing.

The deleting anomaly is that when a tuple is deleted, data is lost that is unrelated to the reason for the deletion. If test 1004 were not run for some reason, and the tuple deleted in our trivial example, reference to a test facility at Someotherplace-USA is wiped out.

RUN CODE	TEMP °C	PLACE PERFORMED
1001	10	Someplace-USA
1002	20	Someplace-USA
1003	30	Someplace-USA
1003	40	Someplace-USA
1004	50	Someotherplace-USA

The insertion anomaly is the counterpart of the deletion anomaly. If data relating to a new test facility at Someotherplace-USA needs to be inserted in the data base, it cannot be entered until a test is performed there.

The normalization process is not something that should be undertaken after a data base is constructed, but should be considered in the initial design of its relations. Once a user is aware that a certain relation is stored, he will expect to be able to query, update, and delete any combination of attributes as knowns and the remaining ones as unknowns because he knows the information is there. While the normalization process is to eliminate the aforementioned anomalies, the third normal form is the desired end product. The third normal form has been defined in a variety of ways.

Two equivalent definitions are quoted here:

> [Codd, 1974]
> A relation R is in third normal form if it is in first normal form and, for every attribute collection C of R, if any attribute not in C is functionally dependent on C, then all attributes in R are functionally dependent on C.
>
> [Sharman, 1975]
> A relation is in third normal form if every determinate is a key.

D. Languages and Relational Algebra

One of the primary advantages of the relational data concept is the easy way in which high-level, non-procedural languages can be defined. The salient features of such languages were addressed by Codd [1970] in his initial paper. While not his aim to describe such languages, the underlying features remain basically unchanged. The basic features ascribed are:

1) Permits the declaration of relations and their domains.
2) Declared relations are added to the system catalog for use by any authorized member of the user community.
3) Permits the specification for retrieval of any subset of data from the data bank.
4) Arithmetic functions may be defined for qualification of or retrieval of subsets

invoked on the relation by the language.

5) Insertions may take place without regard to any ordering that may be present in their machine representation.

6) Deletions take the form of removing elements from declared relations.

The language development encompasses the development of a "data sublanguage" to define the relational domains and their data types. A "query language" refers to a stand-alone language where the user interacts directly with the data base management system. The query language provides the facilities to update, create, and delete (performed by the data sublanguage) relations in addition to the query capabilities and is usually of a higher-level, less procedural, and intended for the casual user.

The query languages for relational data manipulation are based for the most part on the relational algebra developed by Codd [1970 and 1971]. The major operators of the relational algebra are:

1) Projection: returns only specified columns of a relation and eliminate duplications.

2) Restriction (select): selects only those tuples of a relation which satisfy a given condition.

3) Join: combines two relations into a third relation by concatenating the tuples of the two relations where a given condition holds between them.

4) Set-Theoretic Operators: provides for the union, intersection, and set-difference of two relations of compatible attributes producing a single relation.

As stated earlier there have been many languages based on the relational concept. The discussion as to when and where to use what concept will doubtlessly go on for some-time to come. As uses evolve and hardware continues to improve, the pivot points of decision will also change. The final summary by Burns and Lough [1977] best puts the discussion ground rules to rest.

> ...procedural languages (1) are intended for experienced programmers, (2) require more source code, (3) decrease programmer efficiency, (4) give less logical data independence, and (5) increase machine efficiency (provided queries are well designed). The more non-procedural languages (1) may be used by a broader spectrum of users, (2) require less source code, (3) increase programmer productivity, (4) provide increased logical data independence, and (5) decrease total problem solution time. Thus non-procedural languages increase human productivity at the expense of machine time due to the additional layers of software.

Whether all of these particular points are self-evident truths or not, the overall reduction of time to get the answer is supportable. The question becomes person-hour

dollars compared to machine-time dollars. As equipment becomes faster and cheaper and person-hour costs are ever on the increase for an approximately constant output level, the direction to go seems well-lighted. The Distributed Data Base Management System [Bernstein, 1978] provides further momentum to the relational data model concept.

CHAPTER III

FUNCTIONAL OVERVIEW AND OPERATIONAL CONCEPT

A. General

The development of the Data Managemert System (DMS) provides the salient capabilities described by Codd and enumerated in Chapter II D. In addition, the DMS provides a backup facility, reporting output capability, a reorganization facility, and a data base statusing capability.

The operations of the relational algebra will be provided with the exception of the set-theoretic operators. The projection function is limited in that it does not, as of this writing, eliminate duplicate residue tuples after the projection.

The system is designed for the casual user as outlined in the proposed solution (Chapter I B). The accomplishment of various maintenance functions, retrieval, and manipulation features is designed for maximum user participation in the various operations.

The system is of a query language type. It is a stand-alone, non-procedural, interactive program that is programmed basically in FORTRAN VI. A structured programming

language pre-compiler FLECS [Beyer, 1975] to be used with FORTRAN programs is used for the actual source programming.

The target machine is the Interdata 8/32. The user interfaces with the Interdata 8/32 in the Multiple-Terminal-Monitor (MTM) environment and interface with the DMS through the Interdata Multi-Task Operating System (MT/OS).

B. Functional Description: Data Management System

The Data Management System (DMS) utilizes the Interdata Multi-Task Operating System, Multi-Terminal-Monitor, Operating System File Support System to support its various capabilities. It stands alone with an Inquiry Language that prompts the casual user for appropriate responses. The DMS then fulfills the user's request through either self-contained functions or makes use of appropriate system capabilities.

The casual user functions within the Multiple-Terminal-Monitor (MTM) environment. The user enters the system with an individual account. Once in the system, the Data Management System (DMS) is immediately entered via a single call to a high-level system Command Substitution System (CSS) procedure. This CSS procedure brings in the DMS, makes appropriate logical unit assignments, and starts the task. A block diagram sketch of the user and his relationship to machine through the system support software and the DMS is shown in Figure 2.

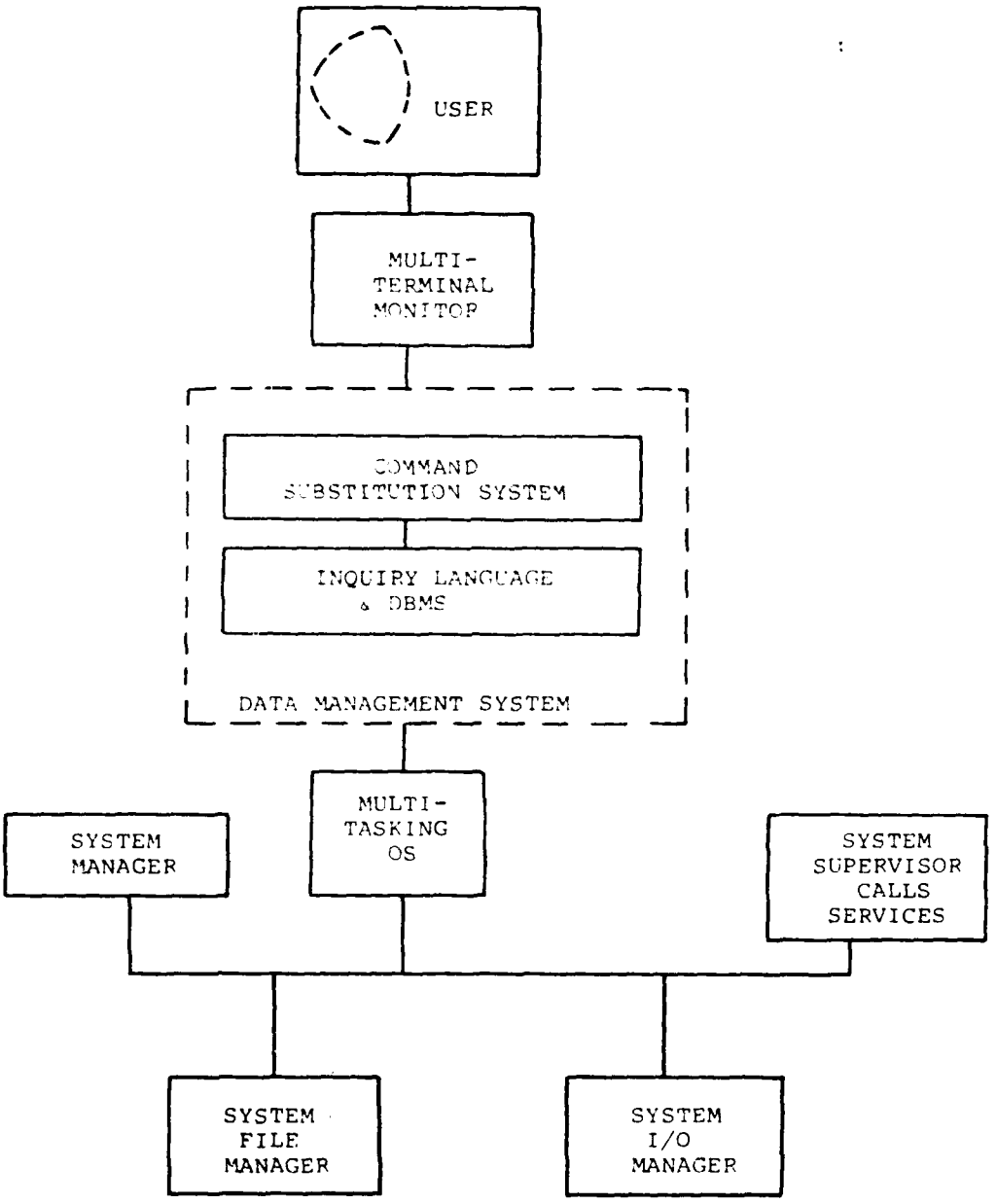


Figure 2. Data management system/interdata 8/32 functional interface diagram.

C. Target Machine Description

1. Hardware. The target machine for the DMS is the Interdata 8/32. The hardware configuration is as follows:

a) Minicomputer Interdata 8/32 with 528 KB (expandable to 1056 KB) Core Memory

b) 15 CRT Terminals

c) 1 - Card Reader

d) 1 - 600 LPM Printer

e) 1 - 9 Track Tape Drive

f) 1 - 7 Track Tape Drive

g) 2 - Phone Modem

h) 1 - Electro Static Plotter/Printer

A functional block diagram of the hardware configuration is shown in Figure 3.

2. Software. The system software available to support the Interdata 8/32 is as follows:

a) Multi-Task Operating System (Command Substitution System (CSS))

b) Multi-Terminal-Monitor (MTM)

c) FORTRAN VI Compiler

d) BASIC Level II Compiler

e) File Support System

- Indexed Files
- Contiguous Files

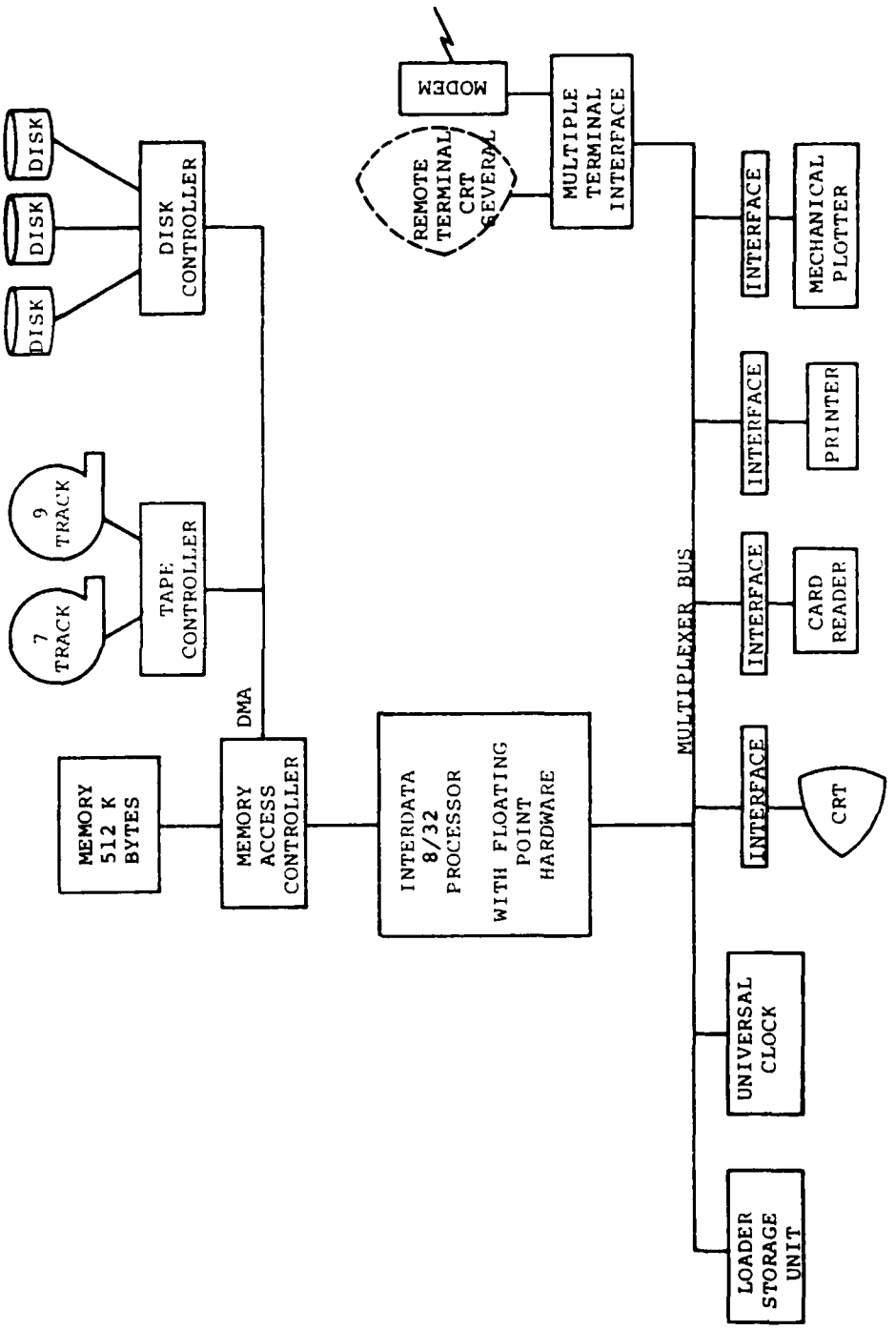


Figure 3. Interdata 8/32 configuration.

The Interdata 8/32 executive software is the Multi-Task Operating System (MT/OS). The MT/OS utilizes a high-level language to accomplish job execution. The Command Substitution System (CSS) provides for loading and execution of system routines such as the FORTRAN compiler and task establisher. It is also used to direct the output of the completed task, assign the logical units for accepting input, and generally carrying out the desired task.

The only compilers available on this particular Interdata system is the BASIC II and FORTRAN VI. The working environment for this Interdata 8/32 is purely research and development and at the time of acquisition there is no need for a COBOL type compiler. While COBOL may have facilitated, the development of the DMS, the use of FORTRAN was no great hindrance. On the contrary FORTRAN has the advantage of having a much greater number of users who have familiarity with it. This familiarity allows greater portability and utilization of previously generated FORTRAN data analysis packages (integration of these packages will be addressed later).

The DMS is programmed using the FORTRAN pre-compiler FLECS [Beyer, 1975] developed at the University of Oregon. A FORTRAN source file is a parallel output to the FLECS listing. The FORTRAN file is then used as the input or source to the regular FORTRAN compiler. From this

stage on the process is the same as any other FORTRAN compilation.

There are two sections that are not in the FLECS format. These are the string handling and some basic machine language operations.

The basic machine language routines consist of the following procedures:

<NAME>	<FUNCTION>
FIELD	CREATES A FIELD DESCRIPTOR WORD
LOC	RETURNS THE CORE ADDRESS OF A VARIABLE
CONT	RETURNS THE CONTENTS OF A LOCATION BY ADDRESS
FETCH	RETURNS AS 32 BIT RT JUSTIFIED VALUE IN AN *ADDRESS* AS DESCRIBED BY *FIELD*
STORE	STORES DATA IN *ADDRESS* AND ACCORDING TO *FIELD*
INTV	RETURNS VARIABLE IN INTEGER REGISTER
REALV	RETURNS VARIABLE IN FLOATING POINT (SINGLE PRECISION)

The detail development and description of these procedures are presented by Pfaltz, [1977].

The string handling routines are in FORTRAN. These routines are the FLECS pre-compiler subroutines for handling STRING variables. The routines were developed by Beyer, [1975] and are used in the DMS. The routines and their functions are listed below:

<ROUTINE>	<FUNCTION>
CATNUM	CONCATENATE NUMBER TO STRING
CATSTR	CONCATENATE STRING TO STRING
CATSUB	CONCATENATE SUBSTRING TO STRING
CHTYP	CHARACTER TYPE i.e., LETTER, DIGIT
CPYSTR	COPY STRING
CPYSUB	COPY SUBSTRING
GETCH	GET CHARACTER i.e., ASCII CODE IN STRING
HASH	HASH FUNCTION
PUTNUM	PUT NUMBER (OPPOSITE OF GETCH)
STREQ	STRING EQUALITY CHECK
STRLT	STRING LESS THAN ANOTHER STRING

The program is divided into groups for ease of development. Each group can be compiled separately, then the total established as a system executable task. The breakdown was established for convenience. The primary functional areas of the DMS are divided into these groups, however.

The program in either the FLECS or FORTRAN form is available from:

Systems Simulation Directorate
DRSMI-TDF(R&D)
Redstone Arsenal, AL 35809

The Interdata file support system allows use of three types of files: indexed files, contiguous files, and chained files. The chain file system is useable, but is no

longer supported by Interdata. The indexed file system is a random access system that is open ended and records may be added as necessary. The indexed file records are linked with a forward and backward pointer system to maintain the file. The contiguous file is, as its name implies, a fixed size file that must be allocated with sufficient size to hold the desired file. The contiguous file has the advantage of accepting variable length records where the index file record size is fixed at allocation time.

The system also has resident various supervisory routines for I/O to the peripheral devices as well as supervising the various user task. A function block diagram of the operating system is shown in Figure 4. The Interdata also has a library of utility routines that is not available on this particular installation to perform such task as SORTING.

The Interdata system software supports Re-Entrant programs such that multiple users may use a single copy of a program. This feature is not exercised with this development, but will probably be one of the first improvements made to the DMS.

D. User/DMS Communication Description

The user enters the DMS task into the Interdata as described in Chapter III B. This entry into the machine begins execution of the program or places the user in the

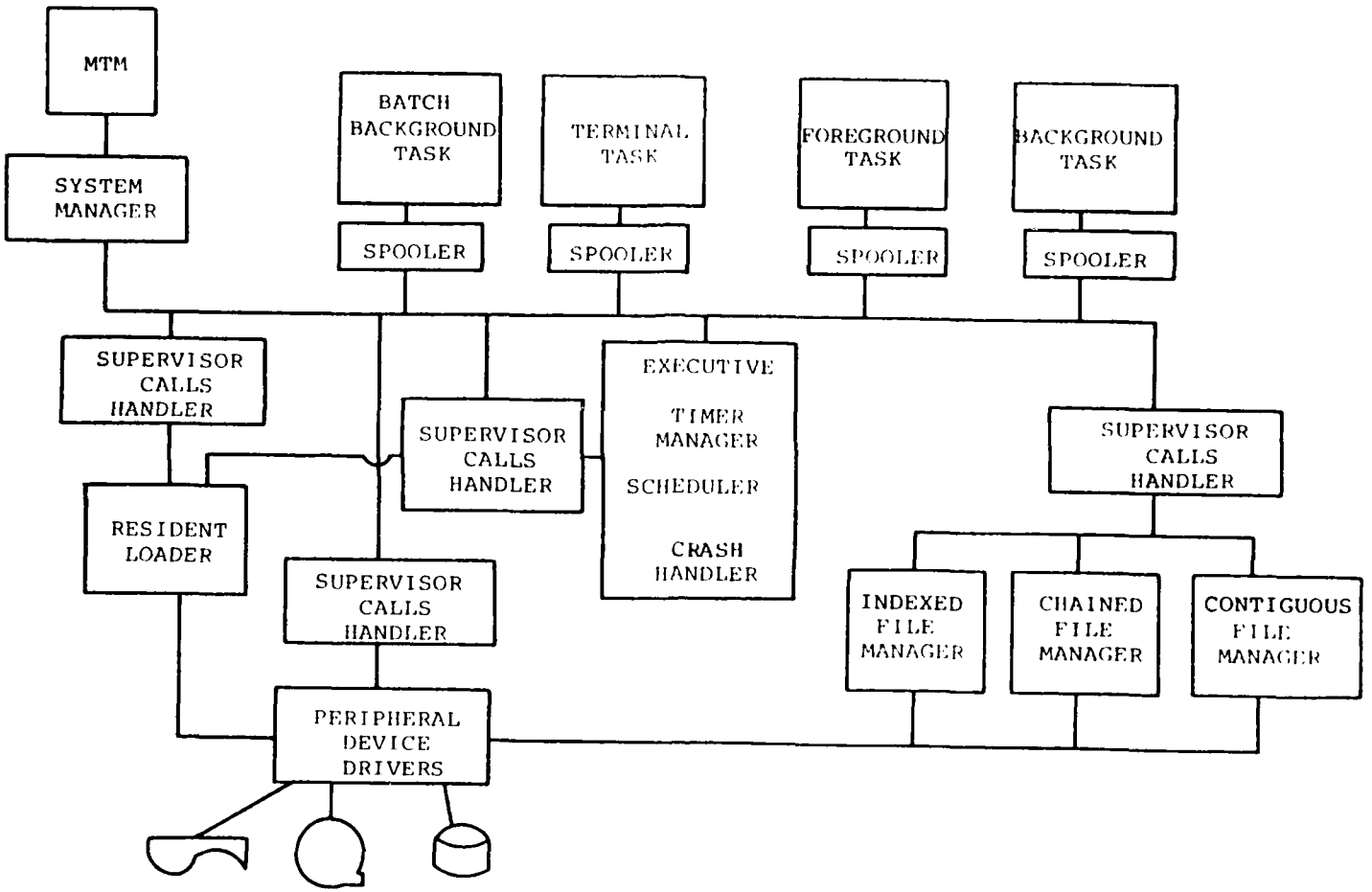


Figure 4. Multi-task operating system functional block diagram.

Data Management System. The user is then asked what action he wishes to perform. The DMS is functionally segmented into the following areas:

1) CREATE A RELATION
2) DELETE A RELATION
3) EDIT A RELATION
4) MODIFY RELATION COLUMNS
5) MERGE TWO RELATIONS
6) BACKUP A RELATION
7) RETRIEVE AND MANIPULATE A RELATION
8) REORGANIZE A RELATION
9) STATUS A RELATION

The user indicates what action he wishes to take. That indication enters him into that particular operational portion of the DMS. The user then performs the desired function. After the action is complete (the term complete will be described later), the user is then given the option of exiting the DMS (and consequently the MTM system thus canceling his task) or performing further operations. An overall functional block diagram is shown in Figure 5.

Each of the functional areas provides a different portion of the overall DMS support. The general execution of each segment is the same however. The overall functional flow is indicated in Figure 6. The user is asked to identify the relation he wishes to exercise, where it is

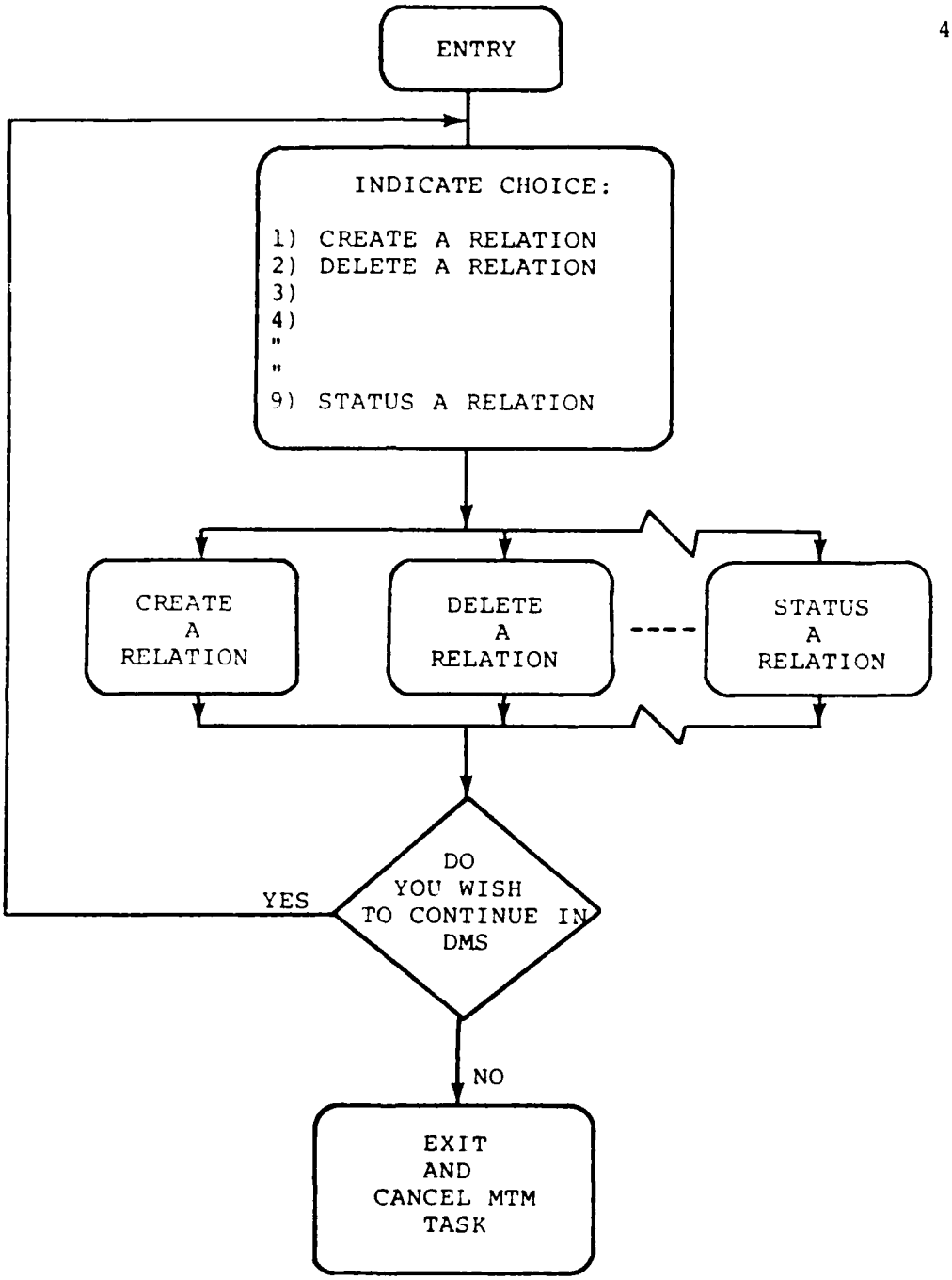


Figure 5. Overall functional block diagram.

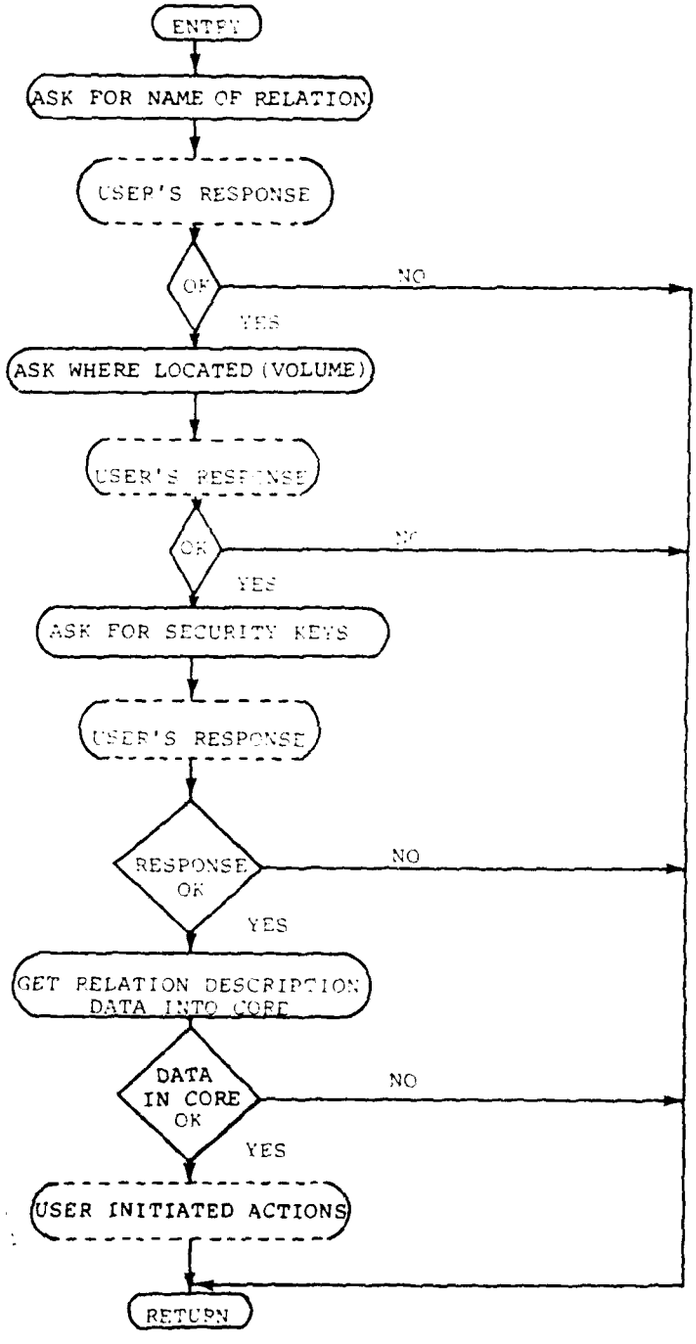


Figure 6. Overall functional flow.

(volume), and access privilege keys for read and write. (The privilege keys request is not functional at this time. The individual accounts in MTM provides some protection. The reason it is not implemented is because anyone may obtain the keys by asking the operating system to provide them. To alter this requires an operating system change which has not been implemented at this time.) If the user's responses are not satisfiable because that specific volume is not mounted, no such relation, etc., the action is terminated and the user is returned to the main stream of the DMS. The user having satisfied his portion of the sequence the machine must then actually get the appropriate relations data description (to be explained later). If this is satisfied, the user then carries out his relational operations. If the system cannot obtain the proper relations description (which can happen for various reasons; hardware failure, operating system failure, etc.), the user is so informed and he is again returned to the main stream of the DMS.

E. Data Management System/Machine Interface

The primary interface between the DMS and the Interdata is the Interdata File Support System. As can be seen in Figure 4, the system has a supervisory call (SVC) handler for the files system. Each user's (actually, user account) files are available only to him. While the system

supports a group access file arrangement where a user may put files into a shared access state for use by other members of his group, the DMS is not allowing this feature at this time. Data can easily be given to other users of the DMS through the backup feature of the DMS.

The Interdata [INTERDATA, 1976] system supports FORTRAN file operations that provide the capability to:

1) CREATE A FILE
2) DELETE A FILE
3) OPEN A FILE (assign a file to a logical unit)
4) RENAME A FILE
5) MODIFY FILE ACCESS PRIVILEGES

The DMS utilizes these file facilities a great deal in its operation. In fact, the use of the file system is the heart of the DMS.

F. User Operational Capabilities

1. General. The DMS functions, to the user, as a relational model of the data. He is free to create, delete, and manipulate the relation in a multitude of ways. The operations will be discussed in detail in the demonstration section. The functional capabilities were discussed in Chapter III D.

The operation of the DMS requires the user exercise some responsibility. There are two areas where the user must pay particular attention. First is the area of normal forms,

and the second is the accidental deletion of data. While neither of the concerns are major, they could pose problems if the user is unaware of them.

The DMS system has no inherent normalization functions. The user is entirely responsible for maintenance of normalcy or should be prepared to accept the possible consequences. In many cases the requirement for normalization is nominal at best or only the first normal form is required (i.e., no component or element in the relation is itself a relation). The third normal form is primarily required when the relational operations of join, project, or select are invoked. This is probably the greatest impact area to the "casual user". The degree of this impact is yet to be evaluated. It probably cannot be evaluated until some hands-on casual user interaction with the DMS is obtained.

The second area of loss of data is the inadvertent deletion of tuples or possibly entire relations. The DMS has been developed to minimize loss of data due to software failure or machine malfunction. The two primary concepts of design to help avoid DMS data loss is minimum relation file opening time and copies of relations are used when manipulations are being performed.

The relation to the user appears as the theoretical array (given normality is observed) described in Chapter II.

The DMS provides sufficient dimensions to such an array that to the user it should be boundless.

2. Data - Model (Psuedo-Schema). The data model in the DMS is provided for inherently by the use of the relational data model. The relation is defined by a relation name. The column role-names of attributes are declared as the columns are created in the initial construction of the relation. The number of columns supportable by the system is either 250 or 500 depending on whether the column domain is a string data type or not. If a relation consist of columns whose domains are all in the string domain then the maximum is 250 columns. If all column domains are non-strings, the maximum is 500 columns. For combinations the maximum lies somewhere between the 250 and 500 column number.

The number of tuples allowable is limited by the disk space available. For example, a 80 Meg byte disk can contain a maximum of 312,500 tuples in one relation. This diminishes as the ratio of string to non-string domains changes. If a larger size relation is required, it can be broken into two parts or the system file handler could be modified to get files from two physically different disks. The best solution is to get a larger disk.

The domains that are supported are a combination data-scheme or data-declaration and data type. Each column must

have declared, at the time of creation, a data strategy. There are 4 data strategies available. These are:

a) Integers

b) Single Precision Decimal Numbers

c) A Single Character (any type)

d) Character String

The column role-name combined with the strategy defines the column.

3. Language Facilities. The creation and deletion function are provided for as basic features. The deletion simply eliminates the existence of all reference to the relation. It is non-recoverable.

The creation of a relation entails declaring the degree (may be changed later) of the basic relation, defining the role-name or attribute with an associated domain or strategy.

The projection is accomplished with the column modification facility. The facility provides for deleting columns. The projection is performed not by forming a new relation of specific columns, but forms a new relation by deleting undesired columns.

The join of two relations is accomplished by the merge operation. The merge is the concatenation of two relations into a third based on the same component(s) of columns with the same role-name and domain.

After the join, the third relation has the same operational features and permanency in a user's relational repertoire as any relation.

The select operation is performed as an "ANDING" or "ORING" of component values in a specific column. If the value in the column meets the criteria equal to, greater than, or less than (=, >, or <) specified, the tuple is selected. The ANDING or ORING processes may be repeated (providing nesting) as often as desired to achieve the desired selection.

The capability to update components, insert a new tuple, or delete a tuple is provided for in the DMS. Aids to locate the desired tuple to update or delete are provided.

The user is required to think in terms of currency when performing the update, insert, and delete operations. The currency is required for locating or moving to a new location in the relation. The current tuple and current column then define a current component that can be altered. The current tuple may be deleted or a tuple may be added after the current tuple.

The list of manipulation features and analysis function could be almost endless. The facilities provided here are few, but felt to be most powerful.

The features provided are:

- Print to user format (of selected data).
- Sort relation on specified column.
- Copy and rename a relation.
- Move selected tuples to a new relation.
- Sum the selected components of a specified column.
- Find the Root Mean Square and Standard Deviation of selected components of a specified column.

As mentioned previously, security is accomplished through the user account and password system feature. A somewhat more secure feature in a long term sense is a back-up feature. Entire relations can be dumped to tape and stored in a safe. The relation is deleted from the system and restored as necessary. Of course a user with a private disk pack can achieve the same thing. The tape does provide a backup to some data in the case of a system failure.

The features of status and reorganization are available to determine the amount of unavailable (dead) file space that has accumulated in a user's relation space. The reorganize feature eliminates the dead space. An additional feature is available, but is not oriented to the casual user. This feature is a dump of some of the relations

pointer, counters, and status data used by the DMS in the accomplishment of various operational capabilities.

The language features provide in the DMS fulfill the basic requirements outlined as salient to a Data Management System [Codd, 1970 and Bernstein, 1978]. The objective is to provide powerful yet simple relational operative tools to allow the maximum use of the user heuristic capabilities combined with the computer's manipulative capabilities.

CHAPTER IV

DATA STORAGE AND FILES ORGANIZATION

A. Data Organization Concept

A pictorial representation of a relation is shown in Figure 7. As mentioned previously, each column must have a strategy defined for its use as it is created. The column strategies are one of the following:

1) SINGLE CHARACTER
2) SINGLE INTEGER
3) SINGLE FLOATING POINT (NOT DOUBLE PRECISION)
4) CHARACTER STRING OF ARBITRARY LENGTH

The concept to allow for storage and retrieval of the data was developed using three files per relation. The technique is patterned after the implementation used in the development of Relation Inquiry and Storage System (RISS) [Meldman, 1978].

When a relation is created, three (3) files are actually created in which to store the relation. The three files are:

RELATION NAME

	COL1	COL2	COL3	COL4	----	COLi	----	COLn
TUPLE 1								
TUPLE 2								
"								
"								
"								
"								
TUPLE j								
"								
"								
"								
"								
"								
"								
TUPLE L								

Figure 7. Relational configuration.

1) Tuple Descriptor File (.TDF)

2) Tuple File (.TF)

3) Alpha Data File (.ADF)

These files are oriented toward a rapid method of accessing and storing data in an interactive environment.

Other files are established on a temporary basis as needed. These will be discussed as required in later sections as development proceeds.

1. TDF File Concept. The TDF contains the basic descriptors and pointers for any given relation, as well as overall descriptive data about the relation such as number of columns, number of tuples, etc. The TDF is an array that is made core resident. Core residence allows the manipulation task to have rapid access to the relations data. Two data arrays are in this file. One is a two dimension array containing information about each column in the relation. The second is a linear array which contains the general description about the relation.

2. TF File Concept. The TF file records are all of the same length. Each is formed according to the form of the relation creation with the exception of a STATUS indicator in the first element.

Two possible organizational concepts can be used in the TF record. Record of basic FORMAT (i.e., integer, floating

point, etc.) can be used. Or all data can be converted to one data type and made one long array.

For RISS development Meldman [1978] converts all data to a character string. This approach allowed use of the BASIC-PLUS compiler STRING handling capabilities to insert or delete columns or alter data components.

The approach taken there will be to represent all data as INTEGER types. This will require altering floating point and ASCII* types to INTEGER acceptable types. Reconversion to the proper type is then made on extraction of the data for use by the Data Management System.

3. ADF File Concept. The ADF is the simple storage of character string data in a record in the ADF file. Each string will have its own record. An alternate storage technique is to concatenate the strings of any one tuple into a record. The first method is fastest in terms of storage and access for retrieval, but consumes a minimum of one 256 Byte disk sector when stored on a contiguous file. The second method requires packing and un-packing strings and adding another location identifier in the TDF or requires a method to determine its location in the desired string.

*ASCII character strings are handled as integers anyway.

The technique used here will be to store each string in its individual record. This method can be altered later and not impact the user of this system.

B. File Description

The three files used as permanent relation storage are contiguous files allocated, stored and retrieved (record retrieval) under the Interdata 8/32 [Interdata, 1978] FORTRAN real-time file management system.

The file for the TDF is static. The file is created at relation creation and remains unchanged for the life of the relation. The descriptor file has considerable space for growth. The IDENT Array has sufficient room to double its present information content. The descriptor (ITDF) for each column has room for an additional column identifier.

The TF and ADF files are also contiguous files. These files are dynamic in size as the need to add data arises. These files are checked as data is entered into the DMS. The files are increased in size in increments of approximately 10 percent of their present size.

As stated previously each relation has three unique files that describe the relation configuration. These files have the following functions and file name organization:

1) <VOL: Relation Name .TDF> - Tuple Description File - This file contains pointer and keys used in storing

and retrieving the relation data in a tuple file and alpha data file.

2) <VOL: Relation Name .TF> - Tuple File - This file contains data for each tuple in the file. The column data in each file is represented here in either its exact form (according to column data strategy) or as an indicator or pointer to the data's location in the alpha data file.

3) <VOL: Relation Name .ADF> - Alpha Data File - The alpha data file contains the variable-length alpha-numeric character-string data. The location of the logical record within this file where the data is contained is in the Tuple File.

C. Record Description Concept

1. Tuple Descriptor File Records. There is an ITDF array element for each column in the relation. In addition there is an array IDENT that holds general status information about the relation. The format and contents of the IDENT ARRAY and the two dimensional array ITDF are as follows:

The ITDF ARRAY portion of the TDF has the following configuration definition:

ITDF (I, N)

Where I = is the column information index
N = number of columns relation

The ITDF I element definitions are as follows for the i^{th} column:

ITDF (1, i) = Relation column strategy

ITDF (2, i) = Record number in ADF where column name is located

ITDF (3, i) = Length of column name in bytes

ITDF (4, i) = ARRAY element in TF record of i^{th} columns' data

ITDF (5, i) = Unused (spare)

The IDENT array elements are defined as follows:

IDENT (1) = Next record to be used in ADF

IDENT (2) = Total length of ADF in bytes

IDENT (3) = Total length of TDF in bytes

IDENT (4) = Total length of TF in bytes

IDENT (5) = Allocated file size of ADF in 256 byte sectors

IDENT (6) = Allocated file size of TDF in 256 byte sectors

IDENT (7) = Allocated file size of TF in 256 byte sectors

IDENT (8) = Number of tuples in relation

IDENT (9) = Number of columns in relation

IDENT (10) = Number of deleted columns still in files

IDENT (11) = Number of deleted columns still in files

IDENT (12) = Maximum allowable number of columns, i.e., N dimension of ITDF

IDENT (13) = Next element in TF record to be used

IDENT (14) = TF allocation safety factor in bytes

IDENT (15) = Number of columns with STR strategy

IDENT (16) = Number of columns with FP strategy

IDENT (17) = Number of columns with INT strategy

IDENT (18) = Number of columns with A strategy

IDENT (19) = DIMENSION OF IDENT ARRAY

IDENT (20) = DIMENSION OF ITDF ARRAY COLUMNS

IDENT (21) = DIMENSION OF ITDF ARRAY ROWS

IDENT (22) = DIMENSION OF ADF RECORD IN CORE

IDENT (23) = DIMENSION OF TF RECORD IN CORE

IDENT (24) - IDENT (64) = Unused (spare)

The IDENT array is dimensioned to a size of 64. The choice is more for convenience because the 64 elements matches a 256 byte disk sector. The TDF is made CORE resistant anytime a relation is active.

2. Tuple File Records. The TF is a file of records where each record corresponds to a tuple (row) in the relation.

Unlike the TDF the TF is not CORE resident. Only a portion of the file is in resident at one time. The residents of a TF record in CORE is for the current relation tuple only.

The TF record is organized as follows:

Element 1 | Element 2 | Element 3 | Element 4 | Element 5 | Element 6

Each element is an INTEGER word. As a record is called into CORE, it becomes an ARRAY. The elements of the record at that time become equivalent to ARRAY elements.

With the exception of Element 1 of the record, the remaining elements correspond to the relation columns. The column strategy determines the number of record (or ARRAY) elements required for a column. There is not a one to one correspondence between the TF record elements and the column number of the relation.

The integer (INT), floating point (FP), and single character (SC) strategies are directly stored in the TF records. The character string (STR) strategy puts two pieces of information in the TF record. One element contains the record number in the ADF where the string is stored. The second element contains the string length in bytes.

The TF is in an INTEGER format. The floating point numbers must then be converted to INTEGER acceptable (Psuedo Integers) formats for storage. They are then reconverted back to floating point format for use.

The TF record is described in the TDF in the ITDF array. The TF record length is a function of the number of columns and the distribution of the column strategies. Each of the TF records in one relation are the same length.

An example of a TF record is shown below.

INTEGER*4 INTEGER*4 INTEGER*4 INTEGER*4 INTEGER*4

STATUS	INTEGER	FLOAT PT	ADF REC#	STRING LENGTH

Element 1 Element 2 Element 3 Element 4

With the exception of Element 1 of TF, the data for each column in the data has an element in TF.

When the TF record is written onto the storage media, only the record length presently in use will be stored. The TF record length in use is equal to IDENT (13) -1.

3. Alpha Data File Record. The ADF is used to file column heading names and all string data. The record format is

This format is consistent with the string manipulation routines. These routines require the string length be in element one of the string array.

The ADF records are similar to the TF records in that they are not CORE resident. The only record in CORE (if any) is the current required record.

While there are to be no string length limitations, in practice a length to dimension the IADFRC is required. For now the length will be 64 words or 256 bytes or one disk sector. It then becomes only necessary to have a file allocated to the number of tuples times the number of columns with a STR strategy.

4. TF, ADF, and TDF Records Relationship. In order to provide a better understanding of how the records and files relate to one another, a pictorial of a simple relation is shown in Figure 8.

The example for column 1 is either a Floating Point (FP), Integer (INT), or Single Character (SC), representing strategies of 2, 1, or 3 respectively. For illustration purposes these three strategies appear the same to the DMS. Column 2 is a character string (STR) or strategy 4.

The TF file is for an arbitrary number of tuples (record for each tuple). Record 2 has an empty string for column 2 thus a null indicator of FFFF in the pointer location and a 4 in the string length location. A column could have been deleted and the name is no longer meaningful. If a

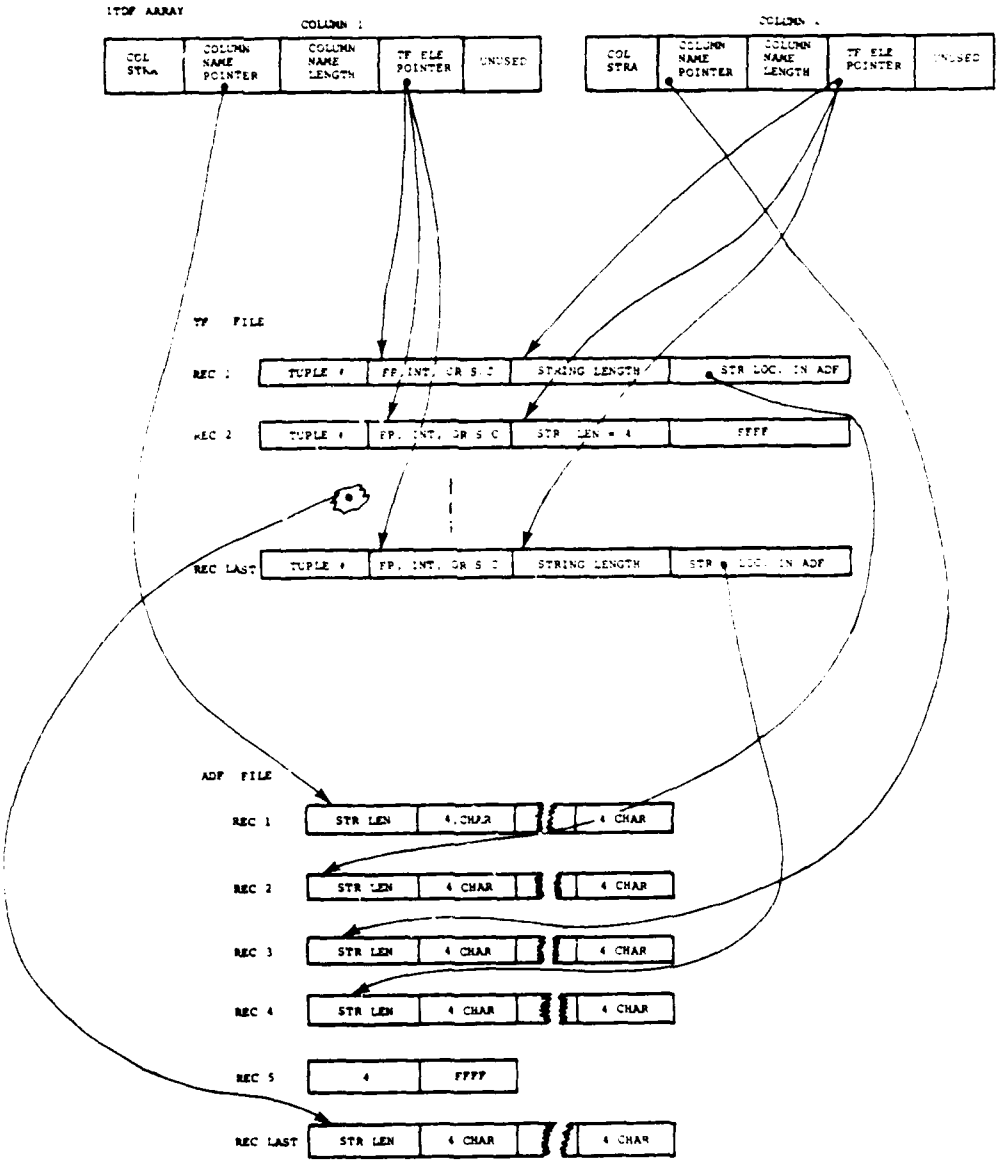


Figure 8. Relation between records and files.

correlation is made to a character string component, a new record is assigned and the old record loaded with the NULL indicator.

The pointers are examples to show typical linkages. As can be seen, the order of the records in either the TF or ADF have no significance to the relations organization.

CHAPTER V

DATA MANAGEMENT SYSTEM DESCRIPTION

A. Overall Usage Concept

The DMS is designed to encourage (in fact demands) user interaction. The overall flow concept is an option choice or command on the part of the user with a response by the DMS. The response may be the action requested by the user or a request for more information or a clarification request (an integer of the form 1029TA would require a clarification).

The DMS has three functional aspects. These are: 1) Relation Maintenance, 2) Data Structure Maintenance, and 3) Relational Operators. The nine user selections that compose the DMS operational aspects are related to the functional areas as follows:

Relation Maintenance

- CREATE A RELATION
- DELETE A RELATION
- EDIT A RELATION

Data Structure Maintenance

- REORGANIZE A RELATION

STATUS A RELATION

BACKUP A RELATION

Relational Operators

RETRIEVE AND MANIPULATE

MODIFY A COLUMN

MERGE TWO RELATIONS

Each of these operational areas will be described in later sections.

The DMS implementation presented here is to be a prototype system for evaluation purposes. While it in intended that it be a permanent Data Management System, this version is not to be optimized in terms of CORE usage, data retrieval, or manipulative operations. These areas will be improved with time. The primary issue is to evaluate the concept and the human factors. The human interaction is prime in fostering the man-machine combination for an optimum mix of capabilities. The objective is to have man suppling the heuristic demands and the machine suppling the manipulative demands.

B. Create a Relation

This allows the user to create a relation. The user specifies the relation name, initial number of columns, and names (role-names) and their associated strategies (domains) for each column. These are supplied on request from the DMS. The strategies supported are: 1) integer numbers,

2) decimal numbers, 3) single characters, and 4) character strings. After all columns have been defined, all column specifications are reviewed for the user's approval. Alterations to all specifications can be made at this time. Once the strategies are defined, they are unalterable after the relation has been created.

C. Delete a Relation

This function provides the user with the capability to delete an existing relation. The space utilized is returned for system use. All reference to the relation is deleted. Backup data information of the relation is left intact. The user shall be responsible for deletion of BACKUP relations or he may maintain it for future reference or long-term storage. While not apparent, the deletion function plays a part as a relational operator. During the relational operations, intermediate relations are created that are deleted at the end of the operation.

D. Edit a Relation

The relation editor allows the user to enter, locate, modify, and examine data in a relation. The editor also provides for the addition and deletion of tuples (rows) to a relation. The editor is based on the maintenance of the currency of a column and tuple. This makes the current data component available for modification or examination. The basic functions of insert, delete, update, find first, etc.,

are available. The editor requires the user to initiate action after he is given the READY TO EDIT> indication.

The following edit commands are available to the user:

<COMMAND>	<FUNCTION>
?(STATUS)	GIVES CURRENT VALUES OF ROW AND COLUMN POINTERS, RELATION NAME, NUMBER OF TUPLES AND NUMBER OF COLUMNS CURRENTLY IN RELATION.
Q(QUIT)	TERMINATES EDITING
C(COLUMNS)	LIST NAMES OF EACH COLUMN BY NUMBER IN THE RELATION.
+[]*(PLUS)	MOVES CURRENT ROW POINTER FORWARD [X] ROWS (LAST GIVEN IF TOTAL EXCEEDED) 1 IS UNDERSTOOD IF [0] OR [ ] USED.
-[](MINUS)	SAME AS + EXCEPT IN OTHER DIRECTION.
F(FIND)	FIND THE FIRST OCCURENCE OF DATA ITEM IN SPECIFIED COLUMN.
S(SUBSTITUTE)	USER SUBSTITUTES NEW DATA INTO LOCATION POINTED TO BY CURRENT ROW AND CURRENT COLUMN POINTERS.
E(EXAMINE)	DISPLAYS TO USER CONTENTS OF CURRENT ROW AND COLUMN POINTER LOCATION.
D[](DELETE)	DELETES [X] NUMBER OF TUPLES STARTING WITH CURRENT TUPLE. NONE DELETED IF NONE SPECIFIED.
I(INSERT)	INSERTS TUPLE AFTER CURRENT TUPLE POINTER. PROMPTS USER FOR COLUMN VALUES.
B(BOTTOM)	SAME AS INSERT EXCEPT PLACES TUPLE AT BOTTOM.

*[ ] are not actually used in system. Used here to denote location relative to command function.

P[](PRINT)	PRINTS [X] TUPLES OF RELATION STARTING WITH CURRENT TUPLE. 1 ASSUMED IF 0 OR BLANK.
R(RE-START)	USER MAY RESTART EDIT PROCESS TO CHANGE DISPLAY FORMAT.

The editor dynamically fulfills the file requirements unbeknownst to the user. The user is responsible for his own actions, however. While every effort has been made to guard against data destruction, the user can, by being careless, negate any of the safety features.

The retrieval of data is for the purpose of editing only and not for data retrieval in general. Data output is accomplished in the retrieval and manipulation capability.

E. Reorganize a Relation

The user is provided the capability to reorganize a given relation. This function compresses the relation by ejecting deleted records from the storage structure. The active records are then made contiguous. This is a manual decision because of the many variables that can come into play in making the decision to reorganize. Examples are: overall size of relation, ratio of empty (deleted) records to full (active) records, type or size of volume in use, etc.

The user issues the decision. All file allocations and size changes are taken care of by the DMS.

F. Status of a Relation

The user is provided this statusing capability to gain information about a relation, such information as the active to deleted record ratio, relation size in terms of characters, etc. Additional status features can be added as the need appears. This facility should provide the user with information sufficient to make the decision whether or not to reorganize a relation.

The statusing facility also provides the capability to interrogate the IDENT and ITDF arrays for a given relation. This capability is accessed by entering the selection code 10 when the selection code is requested. While not normally needed, this can aid in relation maintenance and trouble-shooting problem areas. It is not recommended for the casual user, thus the choice of 10 is not shown on the selection list.

G. Backup a Relation

The user is provided the capability to backup his individual relations on magnetic tape. The backup frequency and determination of need is the user's responsibility. The backup feature also provides the capability to restore a relation. This is the only data protection provided for by this system.

The backup facility to tape is the most hardware-oriented operation with which the user is required to

interact. Basically the three relation files are stored on tape. In addition a header record is added containing the relation name, read/write protection keys, and the date of the backup. The header is then used to restore the relation files as requested. The restoration is made by relation name and proper identification of access authority with provision of the matching read/write keys.

More than one relation can be stored on one tape space permitting. The storage configuration is sketched below.

HEADER*TDF*TF*ADFHEADER*TDF....

RELATION 1 RELATION 2

Where the * indicate an END-OF-FILE mark.

H. Retrieval and Manipulation of a Relation

The retrieval and manipulation of the data functionally provides the "select" capability of the relational algebra. The reporting of data is also provided. Report generation is in the form of tabular data selected from the data base and ordered according to a specific column.

The user may selectively obtain data by:

1) Selecting subsets of relation tuples (records) by specifying combinations of values and/or restriction on the relations data.

2) Printing selected columns of retrieved tuples (rows).

3) Moving selected records from one relation to another.

4) Copy and rename (if desired) the entire relation.

The retrieve and manipulation functions are evoked in a fashion similar to that of the edit function.

The available functions are:

<COMMAND>	<FUNCTION>
Q(QUIT)	TERMINATES RELATION MANIPULATION.
C(COLUMNS)	LIST NAMES OF EACH COLUMN BY NUMBER IN RELATION.
R(REPRODUCE)	COPIES RELATION UNDER MANIPULATION INTO A RELATION SPECIFIED BY USER.
P(PRINT)	PRINTS RELATIONS' DATA ACCORDING TO USER'S SPECIFICATIONS.
S(SORT)	SORTS A RELATION FOR PRINTING PURPOSES. BASIC RELATION ALTERED TO LAST SORT STATUS.
I(RE-INITIALIZE)	INITIALIZES OR RE-INITIALIZES ALL RECORDS TO ACTIVE STATE.
A(AND)	KEEPS ACTIVE THOSE ACTIVE RECORDS THAT MEET THE SELECTION CRITERIA.
O(OR)	MAKES ACTIVE ALL RECORDS THAT MEET THE SELECTION CRITERIA.
M(MOVE)	MOVES ACTIVE RECORDS TO RELATION DESIGNATED BY USER.
F(FUNCTION)	TAKES USER INTO AREA WHERE OPERATIONS SUCH AS SUM, AVERAGE MAY BE PERFORMED ON DATA.

The F(FUNCTION) supplies the user a list of available analysis operations. The user then selects the desired operation. There are only two functions presently in the system. These are:

1) Summing of the active components in a relation.

2) Calculating the mean and standard deviation of the active components in a relation.

I. Modify Columns of a Relation

This allows the user to alter headings, add new columns, or delete columns to an existing relation. Strategy changes will not be allowed because this implies data type change. This would imply new data in which case a new column should be added.

The projection relational algebra operation is performed with this feature in combination with the R(REPRODUCE) function in the retrieve and manipulate sequence. The relation targeted for projection is first reproduced (and subsequently renamed). The columns not desired in the final result are then deleted.

If a column is added, the components of that column are made EMPTY. The motive for this was to maintain the statistical significance of the data. If zero were loaded, then the possibility exists for the zero to be included in a statical calculation where really no data exist.

J. Merge Two Relations

This function is the JOIN operation of the relational algebra. Two relations are combined on the bases of comparison of data elements in like columns in each relation. A third relation is thus created with a user specified name.

The column(s) on which the comparison of data is made must have the same role-name and like domain (strategies). For a tuple to qualify for merging, all components of the like columns must match. If this condition is passed, the data are entered into the third relation. The third relation consists of columns of like role-names and the unique columns of each of the two individual relations.

The merged relation has all the attributes features of a relation created with the CREATE A RELATION function.

CHAPTER VI

DEMONSTRATION OF DATA MANAGEMENT SYSTEM

The demonstration of the DMS will examplify the life of some typical relations encountered in a testing environment. The major features of the DMS will be exercised from creation to deletion. The relations in Chapter I B will be used in the demonstration. To avoid uninformative redundancy, not all stages in the lives of all relations will be shown. In the demonstration figures the user response to the DMS are underlined.

A. Relation Creation and Maintenance

1. Creation. The relation is created by making the appropriate responses to the DMS to establish the relations name, degree, column role-names, and their domains or strategies. The creation of the TEST-CONDITIONS relation is shown in Figure 9. The figure clearly shows interaction of the user and the DMS.

2. Editing. The EDITING of the relation is the process of manipulation which allows updating, inserting, and deleting to be carried out on the relation. This facility is responsive to the user's lead. The user requests one of the

```
 NAME RELATION
>TESTCOND
 CONFIRM CREATION OF RELATION TESTCOND
 <YES OR NO>
>YES
 IS THE SYSTEM VOLUME BEING USED?
 <YES OR NO>
>YES
 CONFIRM READ KEY  0 AND WRITE KEY  0 ON STORAGE VOLUME MT32
 <YES OR NO>
>YES
 NUMBER OF COLUMNS<ORDER> OF RELATION?>
>5
 SUPPLY COLUMN NAMES & STRATEGIES AS INDICATED.
  COLUMN STRATEGY CHOICES ARE:
     1-INTEGER NUMBER
     2-FLOATING POINT NUMBER
     3-SINGLE CHARACTER
     4-CHARACTER STRING

  COLUMN      1
       COLUMN NAME?>
             |
             |
             |

  COLUMN      5
       COLUMN NAME?>
 >RANGE - FT
       STRATEGY?>
 >2

  COLUMN      6
       COLUMN NAME?>
 >FLUSH TYPE
       STRATEGY?>
 >4
```

Figure 9. Creation of the TEST-CONDITIONS relation.

```
COLUMN    1
      COLUMN NAME> RUN CODE
      STRATEGY> 1-INTEGER
COLUMN    2
      COLUMN NAME> TEMPERATURE - DEG CENTIGRADE
      STRATEGY> 2-FLOATING POINT
COLUMN    3
      COLUMN NAME> WEIGHT - GRAMS
      STRATEGY> 2-FLOATING POINT
COLUMN    4
      COLUMN NAME> SPEED - FT/SEC
      STRATEGY> 2-FLOATING POINT
COLUMN    5
      COLUMN NAME> RANGE - FT
      STRATEGY> 2-FLOATING POINT
COLUMN    6
      COLUMN NAME> FLUSH TYPE
      STRATEGY> 4-CHARACTER STRING

ANY CORRECTIONS? <YES OR NO>
>NO

RELATION TESTCOND HAS BEEN CREATED.
```

Figure 9. Concluded.

actions shown in Figure 10. After each requested action is fulfilled, the DMS responds with the user selected output of the tuple values. The establishment of the response is shown in Figure 11. Each tuple element for the columns shown in Figure 11 will be presented after each user action is satisfied.

The relation is initially empty. This is shown by the status display of the relation when the editor is first entered.

The insertion of data is in the form of a tuple. This is accomplished by the I(Insertion) or B(Bottom) commands. The Insertion command inserts the tuple immediately following the current tuple. The Bottom places the tuple at the bottom or end of the relation.

The insertion of a tuple is shown in Figure 12. The system responds requesting data for each column by name. The system then awaits an input. After all data has been entered, the selected column data is displayed.

The update is accomplished with the S(Substitution) command. The value of the current tuple and column is substituted with the new value. To get the value to be changed to become the current element, the E(Examine) request must be invoked. Figure 13 shows the correction of a data entry FLUSH TIME in our example. First the incorrect data is examined. This assures that the proper relational element

EDITING

THE FOLLOWING EDIT COMMANDS ARE AVAILABLE TO THE USER:

<COMMAND>	<FUNCTION>
?(STATUS)	GIVES CURRENT VALUES OF ROW AND COLUMN POINTERS,RELATION NAME, NUMBER OF RECORDS AND NUMBER OF COLUMNS
Q(QUIT)	TERMINATES EDITING
C(COLUMNS)	LIST NAMES OF EACH COLUMN BY NUMBER IN THE RELATION
+[](PLUS)	MOVES CURRENT ROW POINTER FORWARD [X] ROWS(LAST GIVEN IF TOTAL EXCEEDED) 1 IS UNDERSTOOD IF [0] OR [ ] USED.
-[](MINUS)	SAME AS + ACEPT IN OTHER DIRECTION
F(FIND)	FIND THE FIRST OCCURANCE OF DATA ITEM IN SPECIFIED COLUMN.
S(SUBSTITUTE)	USER SUBSTITUTES NEW DATA INTO LOCATION POINTED TO BY CURRENT ROW AND CURRENT COLUMN POINTERS.
E(EXAMINE)	DISPLAYS TO USER CONTENTS OF CURRENT ROW AND COLUMN POINTER LOCATION.
D[](DELETE)	DELETES [X] NUMBER OF TUPLES STARTING WITH CURRENT TUPLE. NONE DELETED IF NONE SPECIFIED.
I(INSERT)	INSERTS TUPLE AFTER CURRENT TUPLE POINTER. PROMPTS USER FOR COLUMN VALUES.
B(BOTTOM)	SAME AS INSERT ACCEPT PLACES TUPLE AT BOTTOM.
P[](PRINT)	PRINTS [X] TUPLES OF RELATION STARTING WITH WITH CURRENT TUPLE. 1 ASSUMED IF 0 OR BLANK.
R(RE-START)	USER MAY RESTART EDIT PROCESS TO CHANGE DISPLAY FORMAT.

Figure 10. User commands.

```
 NAME OF RELATION TO BE EDITED?>
>TESTCOND
 IS THE SYSTEM VOLUME BEING USED?
 <YES OR NO>
>YES

 RELATION TESTCOND HAS:   0 RECORDS     6 COLUMNS.
 CURRENT RECORD:        0
 CURRENT COLUMN:        1

 DISPLAY COLUMN NAMES?<YES OR NO>
>NO

 DISPLAY HOW MANY COLUMNS?>
>6
 DISPLAY COLUMN NUMBER  1?>
>1
 DISPLAY COLUMN NUMBER  2?>
>2
 DISPLAY COLUMN NUMBER  3?>
>3
 DISPLAY COLUMN NUMBER  4?>
>4
 DISPLAY COLUMN NUMBER  5?>
>5
 DISPLAY COLUMN NUMBER  6?>
>6
 READY TO EDIT>
  >
>
```

Figure 11. Setup response.

```
I
   1.RUN CODE>
>100791
   2.TEMPERATURE - DEG CENTIGRADE>
>0.
   3.WEIGHT - GRAMS  >
>100.2
   4.SPEED - FT/SEC  >
>50
   5.RANGE - FT  >
>20
   6.FLUSH TYPE  >
 >
>ROD
 >     100791
 > 0.00
 >        100.20
 >         50.00
 >         20.00
 >ROD
 >
>
```

Figure 12. Inserting a tuple.

```
P2
 >100791     >0.0     >100.20     >50.00     >20.00     >ROD
 >100792     >10.00     >100.60     >10.00     >20.00     >RAKE
 >
>I
 RELATION TESTCOND HAS:          2 RECORDS     6 COLUMNS.
 CURRENT RECORD:        2
 CURRENT COLUMN:        6
 >
>E
    COLUMN?>
>E
 >RAKE
 >
>S
 COLUMN:
 >RAKE
    ENTER NEW STRING:>
 >
>ROD
 >     100792
 >        10.00
 >       100.60
 >        10.00
 >        20.00
 >ROD
 >
>
```

Figure 13. Update tuple data.

is current. To make sure it is the proper one, the ?(status) is examined. The substitution of the correct value is then made.

The P(Print) feature in the editor is merely an editing aid and not for generalized output. The P(Print) of a number of tuples starting with the current tuple shows the correction to been made to run code 100792.

The deletion of a tuple is accomplished by specifying the number of tuples to be deleted beginning with the current tuple. The deletion of an entire tuple is shown in Figure 14. The user is responsible for making sure the deletions are what he wants. The system will not assume any deletions if a D followed by any form except an integer is encountered. The deletion is performed directly on the data of interest, so, all deletions are permanent unless the user takes precaution. One such precaution would be to copy the original relation (accomplished in the retrieve and manipulate area), make corrections, etc., then delete the original relation and copy the corrected relation as the original if all updates were made satisfactorily.

3. Column Modification. There are three types of column modifications available to the user. These modifications are:

```
P5
 >100791     >0.0      >100.20     >50.00     >20.00     >ROD
 >100792     >10.00     >100.60     >10.00     >20.00     >ROD
 >100793     >20.00     >100.90     >50.00     >20.00     >RAKE
 >100000     >1000000.00     >EMPTY     >10000.00     >9999999.00     >RUBBLE
 >100794     >30.00     >100.20     >10.00     >20.00     >RAKE
 >
>-
 >     100000
 >   1000000.00
 >EMPTY
 >     10000.00
 >   9999999.00
 >RUBBLE
 >
->D1
 >     100794
 >       30.00
 >      100.20
 >       10.00
 >       20.00
 >RAKE
 >
>
P5
 >100791     >0.0      >100.20     >50.00     >20.00     >ROD
 >100792     >10.00     >100.60     >10.00     >20.00     >ROD
 >100793     >20.00     >100.90     >50.00     >20.00     >RAKE
 >100794     >30.00     >100.20     >10.00     >20.00     >RAKE
 >100795     >40.00     >100.60     >50.00     >20.00     >PLUNGER
 >
>
```

Figure 14. Deleting a tuple.

a) Rename a Column

b) Delete a Column

c) Add a Column

The rename operation is a change in role-name only and not a strategy change. A strategy change would imply a data type change. The change in data type means that the data in the existing column is to be changed which requires a new column. For this reason no strategy changes are allowed. The rename does not effect any data in the relation. An example of a column name change is shown in Figure 15.

The deletion of a column eliminates that column and all data in that column from the relation. The repeated deletion of selected columns then can accomplish the PROJECTION function in the sense of a relational operation. An example of a column deletion process is shown in Figure 16.

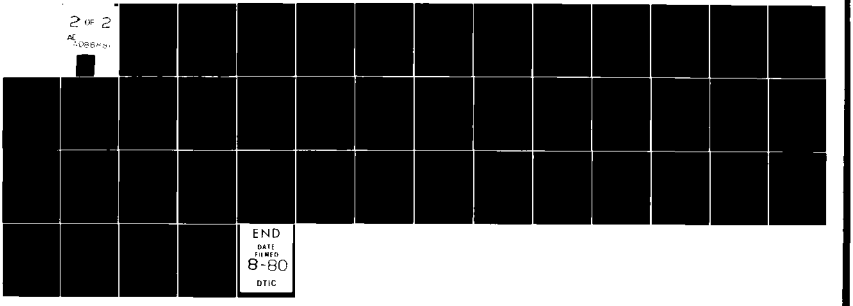
The addition of a column provided for in the maintenance area of the DMS is not the relational JOIN function. It is strictly a process to provide for the inclusion of an additional column. The column can be added in any position in the relation. The process is similar to the initial creation of the relation. The addition of a column is shown in Figure 17. When a column is added, the data is shown as EMPTY. The BEFORE and AFTER data for the TEST-OBSERVATION relation after the column addition shows the column data as EMPTY.

COLUMN MODIFICATION

```
 RELATION TO BE MODIFIED?>
>TESTCOND
 IS THE SYSTEM VOLUME BEING USED?
  <YES OR NO>
>YES

 DO YOU WISH TO:
    R> RENAME A COLUMN.
    D> DELETE A COLUMN.
    A> ADD A COLUMN.
 OPTIONS <R,D, OR A>>
>R

 COLUMN TO RENAME?>
>2
 NEW COLUMN NAME?>
>TEMPERATURE - DEG CENTIG

 COLUMN    3 IS NOW NAMED:
    TEMPERATURE - DEG CENTIG

 ANY MORE OPERATIONS ON THIS RELATION?>
 <YES OR NO>
>
```

Figure 15. Renaming.

AD-A086 881 ARMY MISSILE COMMAND REDSTONE ARSENAL AL SYSTEMS SI--ETC F/G 9/2
A RELATIONAL-BASED DATA MANAGEMENT SYSTEM FOR ENGINEERING AND S--ETC(U)
JUN 80 M M HALLUM
UNCLASSIFIED DRSMI/RD-80-11 NL



COLUMN MODIFICATION

```
 RELATION TO BE MODIFIED?>
>TESTOBS
 IS THE SYSTEM VOLUME BEING USED?
 <YES OR NO>
>YES

 DO YOU WISH TO:
    R> RENAME A COLUMN.
    D> DELETE A COLUMN.
    A> ADD A COLUMN.
 OPTIONS <P,D, OR A>>
>D

 COLUMN TO BE DELETED?>
>4
 CONFIRM DELETION OF COLUMN    4 NAMED:
   >FINAL SPEED - FT/SEC
   <YES OR NO>
>YES

 ANY MORE OPERATIONS ON THIS RELATION?>
 <YES OR NO>
>NO
```

Figure 16. Deleting.

COLUMN MODIFICATION

```
 RELATION TO BE MODIFIED?>
>TESTOBS
 IS THE SYSTEM VOLUME BEING USED?
 <YES OR NO>
>YES

 DO YOU WISH TO:
    R> RENAME A COLUMN.
    D> DELETE A COLUMN.
    A> ADD A COLUMN.
 OPTIONS <R,D, OR A>>
>A

 COLUMN AFTER WHICH TO ADD THE NEW COLUMN?>
>3

      NEW COLUMN NAME?>
>FINAL SPEED - FT/SEC

    STRATEGY?>
      1-INTEGER NUMBER
      2-FLOATING POINT NUMBER
      3-SINGLE CHARACTER
      4-CHARACTER STRING
 <ENTER BY NUMBER>
>2

 ANY MORE OPERATIONS ON THIS RELATION?>
 <YES OR NO>
>
```

Figure 17. Adding.

COLUMN MODIFICATION

```
>100791     >RED      >10.00      >30 JUN 79      >FLIP VALVE LEAKAGE
>100797     >RED      >10.10      >30 JUN 79      >EMPTY
>100794     >BLUE      >EMPTY      >30 JUN 79       >EMPTY
>100798     >CLEAR      >11.00      >30 JUN 79       >BAD RUN
>100795     >BLACK      >10.10      >3 JUL 79       >EMPTY
>100798     >CLEAR      >12.00      >30 JUL 79       >REPEAT OF 100798
```

BEFORE

```
 >100791     >RED      >10.00      >EMPTY      >30 JUN 79      >FLIP VALVE LEAKAG
E
 >100797     >RED      >10.10      >EMPTY      >30 JUN 79      >EMPTY
 >100794     >BLUE      >EMPTY      >EMPTY      >30 JUN 79       >EMPTY
 >100798     >CLEAR      >11.00      >EMPTY      >30 JUN 79       >BAD RUN
 >100795     >BLACK      >10.10      >EMPTY      >3 JUL 79       >EMPTY
 >100798     >CLEAR      >12.00      >EMPTY      >30 JUL 79       >REPEAT OF 10079
8
```

AFTER

Figure 17. Concluded.

4. Status of a Relation. The status function of the DMS is to provide the user with information pertaining to the size of a given relation. The most meaningful measure from the user's point of view is the percentage of dead space to active space. The DMS presents this information in addition to the total size of the relation upon request. An example of a status report is shown in Figure 18. The size of the relation is not the actual storage space used by the DMS, but the actual number of bytes of data in the relation.

The status information gives the user insight as to whether a relation should be reorganized. In addition the status feature is implemented with a DMS systems programmer dump of the IDENT and ITDF arrays containing pointer information and other data describing the relation. This data is obtained by entering 10 as a response to the select operation when the DMS is first entered. In general, this information is not useful to the user. Its primary use is as an aid to recovery from a system failure of some type. The details of such an operation are best left to an operations and maintenance manual for the DMS. An example of the status dump is included as Figure 19, for completeness.

5. Backup a Relation. The backup feature of the DMS provides the user with the capability to save desired relations on magnetic tape. The relation is stored in the format of the TF, TDF, and ADF files. In addition a header

```
9

 NAME OF RELATION TO BE STATUSED?>
>TESTCOND
 IS THE SYSTEM VOLUME BEING USED?
 <YES OR NO>
>YES

 IS PRINTER COPY DESIRED?>
      <YES OR NO>
>NO

   **** STATUS DATA FOR RELATION TESTCOND****

 RELATION CONSIST OF        578 CHARACTERS

 RELATION HAS:    6 COLUMNS AND      8 ROWS.

  DEAD TO ACTIVE SPACE % =  27.27

 DO YOU WISH TO CONTINUE IN DATABASE SYSTEM?
 <YES OR NO>
>
```

Figure 18. Status report.

RELATION MAINTENANCE

**** STATUS DATA FOR RELATION TESTCOND****

RELATION CONSIST OF 578 CHARACTERS

RELATION HAS: 6 COLUMNS AND 8 ROWS.

DEAD TO ACTIVE SPACE % = 27.27

IDENT(1)= 17
IDENT(2)= 202
IDENT(3)= 120
IDENT(4)= 256
IDENT(5)= 18
IDENT(6)= 21
IDENT(7)= 19
IDENT(8)= 8
IDENT(9)= 6
IDENT(10)= 0
IDENT(11)= 0
IDENT(12)= 250
IDENT(13)= 9
IDENT(14)= 24
IDENT(15)= 1
IDENT(16)= 4
IDENT(17)= 1
IDENT(18)= 0
IDENT(19)= 64
IDENT(20)= 5
IDENT(21)= 250
IDENT(22)= 64
IDENT(23)= 501
IDENT(24)= 0
IDENT(25)= 0
IDENT(26)= 0
IDENT(27)= 0
IDENT(28)= 0
IDENT(29)= 0
IDENT(30)= 0

COLUMN	ITDF(1,ICC)	ITDF(2,ICC)	ITDF(3,ICC)	ITDF(4,ICC)	ITDF(5,ICC)
1	1	0	8	2	0
2	2	1	24	3	0
3	2	2	14	4	0
4	2	3	14	5	0
5	2	4	10	6	0
6	4	5	10	7	0

Figure 19. DMS system status dump.

is provided for security (write/read key), dating, and identification. This procedure is the only one where the user must directly interact with the computer hardware. The interaction is the requirement to setup a tape on the tape drive. A typical example of this interaction is shown in Figure 20.

The backup allows relations to be put on tape for long-term storage, transporting relations to other installations or other individual users. More than one relation can be put on a tape. The number that can be put on a tape is a function of the size of the relations to be stored. An example of the restore function is shown in Figure 21.

6. Reorganization. The determination of the need for a relation to be reorganized is left to the user. The reorganization's primary function is to eliminate dead records in the ADF that have gotten there by deletion operations. In certain cases the reorganization process can repair or aid in repairing damage to relations induced by either user or machine hardware.

An example of a reorganization is shown in Figure 22. The status of the relation is shown before and after the reorganization. The reorganization process can cause the relation to consume more space after the reorganization than before. The cause for this phenomena is a large number of string elements in a relation being empty. The user is

```
8

 DO YOU WISH TO BACKUP AN EXISTING RELATION?
      <YES OR NO>
>YES

 PLACE TAPE ON DRIVE IN ON-LINE STATE
 WHEN READY TYPE * CO * & RETURN *
PAUSE
TASK PAUSED
*CO

 NAME OF RELATION?>
>TESTCOND
 IS THE SYSTEM VOLUME BEING USED?
 <YES OR NO>
>YES

 ARE OTHER RELATIONS ON THIS TAPE?>
>NO
 TESTCOND HAS BEEN STORED IN POSITION  1

 DO YOU WISH TO CONTINUE IN DATABASE SYSTEM?
 <YES OR NO>
>
```

Figure 20. Backup a relation.

RELATION MAINTENANCE

```
S

 DO YOU WISH TO BACKUP AN EXISTING RELATION?
      <YES OR NO>
>NO
 DO YOU WISH TO RESTORE A RELATION?
      <YES OR NO>
>YES

 PLACE TAPE ON DRIVE IN ON-LINE STATE
 WHEN READY TYPE * CO * & RETURN *
PAUSE
TASK PAUSED
*CO

 NAME OF RELATION?>
>TESTCOND
 IS THE SYSTEM VOLUME BEING USED?
 <YES OR NO>
>YES
 HOW MANY RELATIONS ARE ON THIS TAPE?>
>1
 RELATION TESTCOND HAS BEEN RESTORED WITH
 DATA DATED  7-13-79.

 DO YOU WISH TO CONTINUE IN DATABASE SYSTEM?
 <YES OR NO>
>
```

Figure 21. Restore a relation.

RELATION MAINTENANCE

```
6

 NAME OF RELATION TO BE REORGANIZED?>
>TESTCOND
 IS THE SYSTEM VOLUME BEING USED?
 <YES OR NO>
>YES

 RELATION TESTCOND REORGANIZED

 DO YOU WISH TO CONTINUE IN DATABASE SYSTEM?
 <YES OR NO>
>YES
          |
          |
          |

9

 NAME OF RELATION TO BE STATUSED?>
>TESTCOND
 IS THE SYSTEM VOLUME BEING USED?
 <YES OR NO>
>YES

 IS PRINTER COPY DESIRED?>
     <YES OR NO>
> NO

  **** STATUS DATA FOR RELATION TESTCOND****

 RELATION CONSIST OF       480 CHARACTERS

 RELATION HAS:   6 COLUMNS AND     8 ROWS.

  DEAD TO ACTIVE SPACE % =   0.00
```

Figure 22. Reorganization of a relation.

unaware of this and should not be concerned about it. If the ratio of dead to active space is greater than zero in the statusing of a relation, it will not happen.

7. Deletion of a Relation. The deletion of a relation is demonstrated in Figure 23. The confirmation request shown in the figure gives the user his only opportunity to negate his deletion if he has made an error.

B. Retrieval and Manipulation

The retrieval and manipulation of the relation is carried out in two areas. These are the merge and retrieve and manipulate sections of the DMS.

1. Merge Two Relations. The merge process concatenates tuples of like conditions using the columns of common domains and role-names. If no role-names and domains are equal for the two relations, no merge takes place.

The merge is shown in Figure 24. The original relations are shown in the upper portion of the continued portion of the figure. The merged relation is shown as the lower relation. The process shown in Figure 24 is not a continuous process. Intervening steps of naming the relation, entering the editing process, etc., have been omitted for clarity.

The relation resulting from the merge is a permanent relation in the user's inventory of relations. The system does not attempt to delete duplicate tuples resulting from the merge. The merge is the relational equivalent of the

```
2
ENTER NAME OF RELATION TO BE DELETED?
>COPYREL
 CONFIRM DELETION OF COPYREL <YES OR NO>
>YES
 IS THE SYSTEM VOLUME BEING USED?
 <YES OR NO>
>YES

 RELATION COPYREL  HAS BEEN DELETED ON VOLUME, MT32

 DO YOU WISH TO CONTINUE IN DATABASE SYSTEM?
 <YES OR NO>
>
```

Figure 23. Relation deletion.

```
5

 FIRST RELATION?>
>TESTCOND
 IS THE SYSTEM VOLUME BEING USED?
 <YES OR NO>
>YES
 SECOND RELATION?>
>REDCLEAR
 IS THE SYSTEM VOLUME BEING USED?
 <YES OR NO>
>YES
 MERGED RELATION?>
>MERGREL
 IS THE SYSTEM VOLUME BEING USED?
 <YES OR NO>
>YES

         3 RECORDS RESULTED FROM THIS MERGE

 DO YOU WISH TO CONTINUE IN DATABASE SYSTEM?
 <YES OR NO>
>
```

Figure 24. Merge two relations.

RELATION TEST OBSERVATIONS

RUN CODE	FINAL COLOR	FINAL ACCELERATION	DATE RUN	REMARKS
100791	RED	10.00	30 JUN 79	FLIP VALVE LEAKAGE
100797	RED	10.10	30 JUN 79	EMPTY
100794	BLUE	EMPTY	30 JUN 79	EMPTY
100798	CLEAR	11.00	30 JUN 79	BAD RUN
100795	BLACK	10.10	3 JUL 79	EMPTY
100798	CLEAR	12.00	30 JUL 79	REPEAT OF 100798

RELATION TEST CONDITIONS

RUN CODE	TEMPERATURE - DEG CENTIG	WEIGHT - GRAMS	SPEED - FT/SEC	RANGE	FLUSH TYPE
100791	0.0	100.20	50.00	20.00	ROD
100792	10.00	100.60	10.00	20.00	ROD
100793	20.00	100.90	50.00	20.00	RAKE
100794	30.00	100.20	10.00	20.00	RAKE
100795	40.00	100.60	50.00	20.00	PLUNGER
100796	50.00	100.90	10.00	20.00	PLUNGER
100797	60.00	100.20	50.00	20.00	PLUNGER
100798	70.00	110.00	100.00	20.00	ROD

Figure 24. Concluded

RED AND CLEAR TEST CONDITIONS & OBSERVATIONS

RUN CODE	TEMPERATURE - DEG CENTIG	WEIGHT - GRAMS	SPEED - FT/SEC	RANGE	FLUSH TYPE	FINAL COLOR	FINAL ACCELERATION	DATE RUN	REMARKS
100791	0.0	100.20	50.00	20.00	ROD	RED	10.00	30 JUN 79	RED
100797	60.00	100.20	50.00	20.00	PLUNGER	RED	10.10	30 JUN 79	EMPTY
100798	70.00	110.00	100.00	20.00	ROD	CLEAR	11.00	30 JUN 79	s 0

JOIN with the exception of the removal of the duplicate tuples. The two original relations are left intact.

2. Retrieve and Manipulate. The retrieval and manipulation section of the DMS provides the relational operation of SELECT. In addition the relation analysis functions are provided here. The DMS report generation features are also contained here.

The operation of the retrieve and manipulate features are operationally similar to the EDIT facility. As in the EDIT facility the operations on a relation are initiated by the user. A list of the available operations are shown in Figure 25.

The Q(Quit) and C(Column) functions are the same as the EDIT. The C(Column) is shown in Figure 26. Each column role-name is displayed as the column number the DMS is using to reference that particular column.

The manipulation of a relation is provided for in three ways. These are the R(Reproduce), S(Sort), and M(Move) functions.

The R(Reproduce) simply copies a relation under a new name. The R(Reproduce) is examplified in Figure 27. The reproduction function allows for such operations as the projection operation without destroying the basic relation.

The S(Sort) operation sorts the entire relation in ascending or descending order on a specified column. Any of

USER COMMANDS

THE AVAILABLE MANIPULATION FUNCTIONS ARE

<COMMAND>	<FUNCTION>
Q(QUIT)	TERMINATES RELATION MANIPULATION
C(COLUMNS)	LIST NAMES OF EACH COLUMN BY NUMBER IN RELATION.
R(REPRODUCE)	COPIES RELATION UNDER MANIPULATION INTO A FILE SPECIFIES BY USER.
P(PRINT)	PRINTS RELATIONS' DATA ACCORDING TO USERS SPECIFICATIONS.
S(SORT)	SORTS A RELATION FOR PRINTING PURPOSES. BASIC RELATION ALTERED TO LAST SORT STATUS.
I(RE-INITIALIZE)	INITIALIZES OR RE-INITIALIZES ALL PECORDS TO ACTIVE STATE.
A(AND)	KEEPS AVTIVE THOSE ACTIVE RECODS THAT MEET THE SELECTION CRITERIA.
O(OR)	MAKES ACTIVE ALL RECORDS THAT MEET THE SELECTION CRITERIA.
M(MOVE)	MOVES ACTIVE RECORDS TO RELATION DESIGNATED BY USER.
F(FUNCTION)	TAKES USER INTO AREA WHERE OPERATIONS SUCH AS SUM,AVERAGE MAY BE PERFORMED ON DATA.

Figure 25. Manipulation commands.

COLUMN NUMBER PRESENTATION

```
RELATION
>TESTCOND


C
      1= RUN CODE
      2= TEMPERATURE - DEG CENTIG
      3= WEIGHT - GRAMS
      4= SPEED - FT/SEC
      5= RANGE - FT
      6= FLUSH TYPE
  >
 >
```

Figure 26. Display Column.

RELATION MANIPULATION

```
 NAME OF RELATION TO BE MANIPULATED?>
>TESTCOND
 IS THE SYSTEM VOLUME BEING USED?
 <YES OR NO>
>YES
  READY FOR MANIPULATION>
>
>R
    COPIED RELATION NAME?>
>COPYREL
>
>
```

Figure 27. Copy a relation (reproduction).

the four strategies may be sorted. The sort is performed in a synthetic fashion. The relation itself is not sorted. If after a sort the relation is brought into the EDITOR, no sort would be apparent. The sort is active only while the relation is active in the retrieve and manipulation section. The sort is used to order the print output of the relation. An example of the SORT operation is shown in Figure 28. A P(Print) of the relation sorted is also shown in Figure 28.

The M(Move) operation is similar to the R(Reproduce) operation except that only the active records result from the SELECTION process (A(And) and O(Or) discussed later). The M(Move) is very instrumental in performing relational operations. A somewhat sterile example of the M(Move) is shown in Figure 29. The motive behind the move and its continued use in other operations are left to the imagination of the user.

The selection process is carried out with the A(And), O(Or), and I(Re-Initialize). The I(Re-Initialize) simple negates any of the previous selection processes imposed on the relation. Its use is to allow the user to initiate a new selection criteria or simply to start over after a mistake or not obtaining the expected result. No example of the I(Re-Initialize) is shown.

RELATION MANIPULATION

```
 NAME OF RELATION TO BE MANIPULATED?>
>TESTCOND
 IS THE SYSTEM VOLUME BEING USED?
 <YES OR NO>
>YES
 READY FOR MANIPULATION>
>
            |
            |

>S
 COLUMN ON WHICH TO SORT?>
>4
 <ASCENDING OR DESCENDING SORT?>
      <OPTIONS A OR D>
>A
 SORT ON COLUMN  4 COMPLETE
>
>
```

DEMONSTRATION SORT ON SPEED

RUN CODE	TEMPERATURE - DEG CENTIG	WEIGHT - GRAMS	SPEED - FT/SEC	RANGE	FLUSH TYPE
100792	10.00	100.60	10.00	20.00	ROD
100794	30.00	100.20	10.00	20.00	RAKE
100796	50.00	100.90	10.00	20.00	PLUNGER
100791	0.0	100.20	50.00	20.00	ROD
100793	20.00	100.90	50.00	20.00	RAKE
100795	40.00	100.60	50.00	20.00	PLUNGER
100797	60.00	100.20	50.00	20.00	PLUNGER
100798	70.00	110.00	100.00	20.00	ROD

Figure 28. Sort a relation.

RELATION MANIPULATION

```
>M
 RELATION TO MOVE RECORDS TO?>
>REDCLEAR
 IS THE SYSTEM VOLUME BEING USED?
 <YES OR NO>
>YES
 ERASE RECORDS AFTER MOVING?>
      <YES OR NO>
>NO
     4 RECORDS MOVED TO REDCLEAR
>
```

FINAL COLOR - RED OR CLEAR - DATA

RUN CODE	FINAL COLOR	FINAL ACCELERATION	DATE RUN	REMARKS
100791	RED	10.00	30 JUN 79	FLIP VALVE LEAKAGE
100797	RED	10.10	30 JUN 79	EMPTY
100798	CLEAR	11.00	30 JUN 79	BAD RUN
100798	CLEAR	12.00	30 JUL 79	REPEAT OF 100798

Figure 29. Move active tuples.

The A(And) function makes the selection of a tuple active, if it was active, and the desired column element meets the selection criteria.

The O(Or) function makes a tuple active if the column element meets the selection criteria regardless of whether the tuple was previously active. The A(And) and O(Or) operations can be repeatedly applied to a relation, in any order, until the desired process is complete. The resulting "active" tuples can then be P(Printed) or M(Move) as the desired result dictates. An example of a single A(And) and O(Or) are shown in Figure 30 and 31, respectively. The selection process can be made on criteria of greater than (>), less than (<), or equal (=). The print of the selected tuples is shown at the bottom of each figure. The O(Or) in Figure 31 is a follow-on selection from the original A(And) shown in Figure 30.

The analysis functions are invoked with the F(Function) operator. There are two functions available through the DMS. These are summing a column and calculating the mean and standard deviation of a column. The tuples included in these two operations are the active tuples resulting from the selection process. An example of the summing of a column is shown in Figure 32. The sum is of all the tuples in the relation.

RELATION MANIPULATION

```
A
COLUMN ON WHICH TO SELECT?>
>2
IS SUBSTRING MATCH DESIRED?
     <YES OR NO>
>NO
SELECTION CRITERIA <OPTIONS <,> OR =>
     <WHICH ONE>?>
>=
VALUE?>
>RED
ADDING ON COLUMN    2 COMPLETE>
SELECTION PROCESS COMPLETE-ACTIVE RECORDS =      2
```

```
     FINAL COLOR - RED - DATA
RUN     : FINAL : FINAL        : DATE RUN   : REMARKS
CODE    : COLOR : ACCELERATION:
100791  : RED   : 10.00        : 30 JUN 79 : FLIP VALUE LEAKAGE
100797  : RED   : 10.10        : 30 JUN 79 : EMPTY
>>OUTPUT COMPLETE>
>
>
```

Figure 30. Adding for selection.

RELATION MANIPULATION

```
>
>O
 COLUMN ON WHICH TO SELECT?>
>2
 IS SUBSTRING MATCH DESIRED?
     <YES OR NO>
>NO
 SELECTION CRITERIA <OPTIONS <,> OR =>
     <WHICH ONE>?>
>=
 VALUE?>
>
>CLEAR
 ORING ON COLUMN   2 COMPLETE>

 SELECTION PROCESS COMPLETE-ACTIVE RECORDS =      4
>
>
```

```
 FINAL COLOR - RED OR CLEAR - DATA
RUN    : FINAL : FINAL        : DATE RUN  : REMARKS
CODE   : COLOR : ACCELERATION:
100791 : RED   : 10.00        : 30 JUN 79 : FLIP VALVE LEAKAGE
100797 : RED   : 10.10        : 30 JUN 79 : EMPTY
100798 : CLEAR : 11.00        : 30 JUN 79 : BAD RUN
100798 : CLEAR : 12.00        : 30 JUL 79 : REPEAT OF 100798
>>OUTPUT COMPLETE>
>
>
```

Figure 31. Oring for selection.

RELATION ANALYSIS

```
>
>F
 THE FOLLOWING FUNCTIONS ARE AVAILABLE:
      1) TOTAL COLUMN DATA
      2) MEAN & VARIANCE
 <<ENTER SELECTION BY NUMBER?>
>1
 COLUMN TO USE?>
>2

   COLUMN  2 TOTALS =       280.00
   WITH           8 ACTIVE RECORDS
>
>
```

Figure 32. Summing a column.

An example of the statistical function is shown in Figure 33. The DMS allows empty elements in a column. If an element is empty, it is not included in the statistical result even if the tuple is active.

It is anticipated that other functions such as regression analysis and data plotting will be added later. The DMS is designed to make this area adaptable to the user's needs.

```
                    RELATION ANALYSIS
>
>F

 THE FOLLOWING FUNCTIONS ARE AVAILABLE:
      1) TOTAL COLUMN DATA
      2) MEAN & VARIANCE
 <<ENTER SELECTION BY NUMBER?>
>2
 COLUMN TO USE?>
>3

   MEAN & VARIANCE OF COLUMN  3 FOR:

        RECORDS  =          9
        MEAN     =      48.65
        STD.DEV. =       5.62
>
>
```

Figure 33. Statistical analysis of a column.

CHAPTER VII

LIMITATIONS

There are three types of limitations. First, there are basic hardware limitations. These encompass such things as CORE memory, disk size, etc. Secondly, there are system software support limitations including such things as file management facilities, compiler availability (even though written in FORTRAN all machines may not support the level required). Third and lastly are the inherent design limitations which are such things as dimension sizes of CORE resident arrays, allowable string length for character strings (even though stated 'any length'), and the limit on the number of columns in a given relation. Each of these classes of limitations will be discussed indicating some of the specific limitation of this particular implementation. It should be emphasized at the outset that this is a prototype development to evaluate the casual user DMS concept in support of a R&D testing environment. It has not been optimized for I/O, storage nor user convenience. As time passes the information required to make meaningful corrections will

be gathered. After evaluation of the data, correction will be made where appropriate.

A. Hardware

The hardware limitations are not discussed with the view toward what cannot be done with the hardware available, but what hardware is required to support the DMS.

The memory requirements are 167.5K bytes with a 20K byte system buffer area for I/O. The buffer size is a function of some of the other design choices, so not all limitation or independent of other areas.

It is felt that CORE requirements can be reduced to approximately 100-110K bytes for the basic system. The buffer reduction is possible only through redesign of the DMS.

The system is random access device oriented. Disks provide this facility now. If other random access mass storage became available, it could be substituted with no user knowledge required.

Other limitations are easily altered such as the backup medium being tape. A backup disk pack would serve just as well.

B. Computer System Software Support

The system on which the DMS is implemented requires a direct access file management system. The file system for this implementation requires the support of the CONTIGUOUS and INDEXED type of file. The CONTIGUOUS file is the basic

storage form of the relation. The INDEXED file system is used in supporting some of the manipulative operations. The CONTIGUOUS file could be used entirely, thus eliminating the need for the INDEXED file support. The space requirements for the temporary files used in the manipulations would be much larger for CONTIGUOUS files (256 byte sectors for each record) as opposed to the smaller fixed record sizes of INDEXED files. The open-ended feature of the INDEXED file also affords the generally smaller file space requirements. This is because rarely are all tuples represented in the manipulation of data.

The compiler is the FORTRAN VI level. The primary compiler features used that require this level are the ENCODE and DECODE functions. All other operations can be supported by much lower level FORTRAN. Since even the most austere mini-system appears to support FORTRAN VI this limitation does not appear a true restriction.

The final support limitation is the real-time support for file renaming, allocation, deletion, access privilege modification, and opening and closing. The dynamic nature and data safety features of the DMS require that the files supporting a given relation be continually opened and closed, and expanding. These file limitations can also be negated for systems not supporting these features, but considerable redesign would be required.

C. DMS Limitations

The limitations of the DMS are for the most part design choices or trade-offs. The DMS features display the relation to the user, in a logical sense, will be used to define the DMS limitations. The logical view is formed from the following constituence:

1) Number of Relations Allowable (per user)
2) Number of Tuples
3) Number of Columns
4) Data Type Restrictions

The number of relations allowed a user is a function of the number of files afforded a user. While there is obviously some limit, the author has been unable to locate the exact number. It is a function of the number of users, number of disk associated with a given machine, etc. Users with 150 files presently exist in the system environment. This suggests that at least 50 relations can be allowed to each user.

The number of tuples allowed is a function of the hardware disk size, the combinations of data types, and number of columns. For example, if all data is non-string data and only one column exists on a 80 megabyte disk, there is then room for approximately 312000 tuples (Each record written in the CONTIGUOUS file system is placed on 256 byte sectors).

If the number of columns exceeds 256, the number of allowable tuples will drop to approximality 156000 columns. Each additional group of 256 columns further reduces the number of tuples that can be placed on a disk. The number of allowable tuples for a given number of columns is shown in Figure 34 for relation domains of no string strategies and for a relation with 1 and 2 columns with string strategies. As can be seen the string strategies greatly reduce the disk volume useable for tuple storage. These data hold if for each tuple with string strategies a string is actually entered in the relation. The DMS developed here does not use an additional space if a string is not actually placed in the relation. Thus the allowable number of tuples lies between the no string curves and the curve for the number of columns with string strategies. It thus behooves the user to avoid string strategies except where necessary. The number of columns that are allowable for each relation was established as a design choice. Since an active tuple has a tuple file (TF) record in CORE memory, the number of columns is determined by the dimension of the array in CORE. The dimension selected was a 501 element ARRAY. This allows for a 500 column relation with no string strategies or 250 columns for all string strategies. Other allowable numbers of columns are a function of the mix of string to non-string

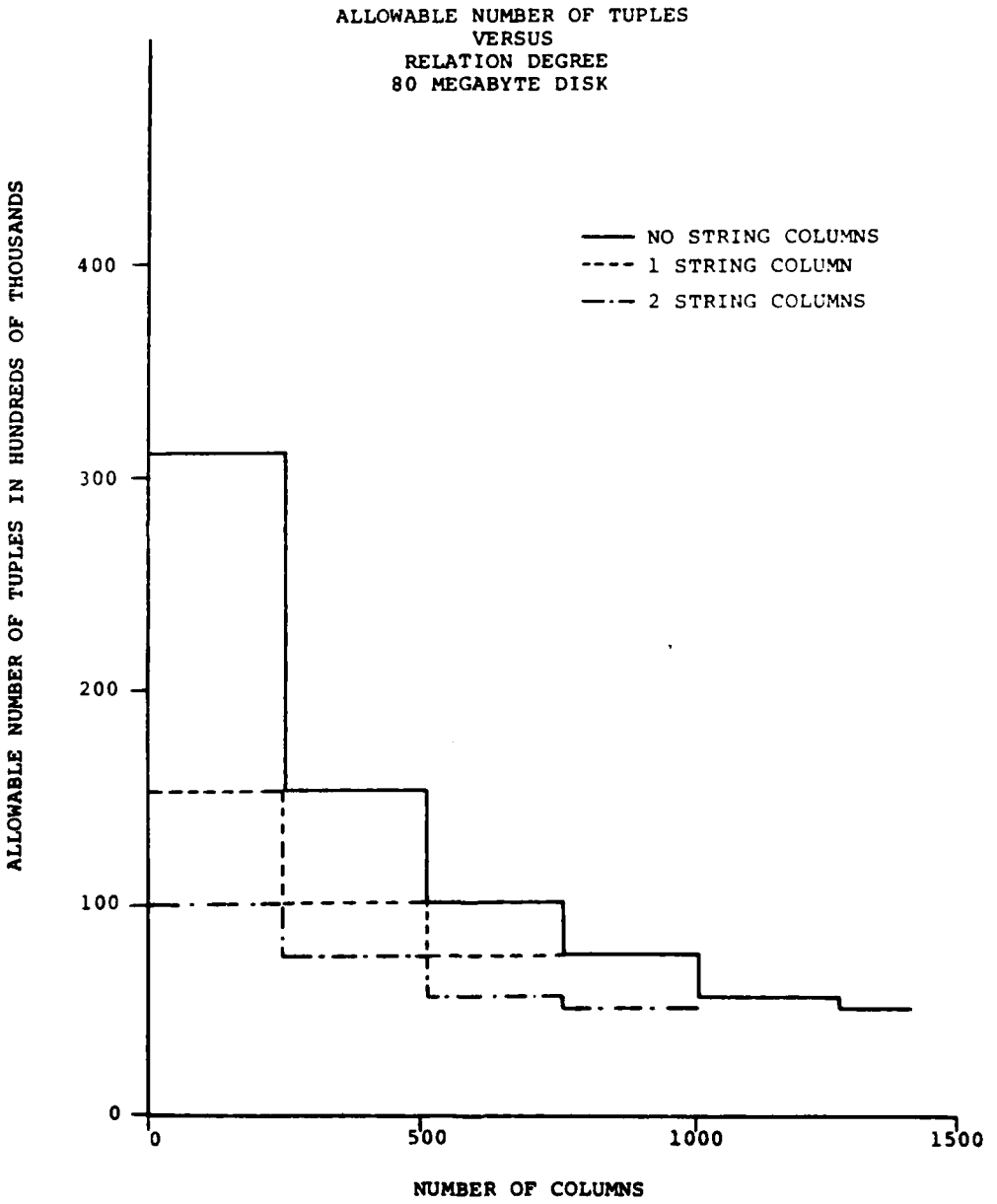


Figure 34. Allowable number of tuples for a given number of columns.

strategies. The first element is reserved as a tuple element counter for DMS system use.

The data types allowable are identified by the strategies that the system supports. These are:

1) Integers
2) Decimal Numbers
3) Single Character
4) Character String

The data types excluded are double precision, complex, and logical representations. With the exception of double precision, these data can be placed in the relation in an alternate form. For example, the logical can be represented as a single character T(True) or F(False). As stated at the outset, these limitations are the result of design selections for a given set of circumstances. Other choices can be made with different rationale. These selections have a strong foundation in logic coupled with hardware and system software capabilities.

CHAPTER VIII

AREAS OF FURTHER DEVELOPMENT

These areas of further study are confined to the context of the DMS developed here. They are not intended to encompass the entire field of data management systems.

The DMS developed here is oriented to the casual user. The user is the designer, developer, implementer, and controller of his data in the DMS environment. In his use of the DMS he interacts directly with the DMS to perform the transactions necessary to accomplish his goals. This direct interaction eliminates the need of an intermediary through which to accomplish his transaction. The concept and performance of data base systems has been evaluated in terms of computer time or query response time. In many applications, in terms of a broader system view, the manhours of participation in a transaction can be a more important considerations. The evaluation schemes for data base systems should be extended to included comparisons to include user cost. For example; contrast a company executive completing a transaction in his office in a non-procedural language, to that of his completing the same transaction

through programmers, to whom he must explain what he wants with no ambiguities and then await the response. This broader systems view will provide better insight for system selection.

The DMS described in this study was developed with little or no emphasis on optimization of retrieval methods or CORE storage requirements. The DMS is presently a single tasked configuration and much of the same program code is duplicated in everal subprograms. Subprogram optimization and operating task storage requirements can be reduced by careful review of all subprograms to eliminate duplicate code, multiply defined array definitions, and extraneous code. The storage required for the execution of the task can be reduced by applying overlay techniques. Overlays permit only that portion of code required for the particular transaction to be CORE resident for that transaction.

The retrieval techniques and the storage efficiency used to store relations data has used straightforward methods. No data packing, or multiple record retrieval methods have been employed. Studies should be made to determine if performance in terms of data storage and retrieval can be improved. In addition, specific storage configurations should be defined to provide that improvement. Disk optimization to improve storage efficiency has many ramifications. For example, if data is packed specifically

for a given number of columns and a column is added, what is the best procedure for including this column?

The security provided in this DMS system is minimal at best. Users should be able to deny access to particular relations without having to physically remove the data from the system as is now required. The primary security provided in the DMS is through the individual user's machine account. While for many applications this is sufficient, for many others where multiple users have access to the same relation, this is inadequate. Another feature that is required allows many users to have access to only portions of relations data. The system as presently implemented allows any authorized user access to a total relation. Methods need to be developed to provide security of data for individual relations besides the accounting method now available. Techniques should also be developed to allow privileged access to only portions of a relation by authorized users.

CHAPTER IX

SUMMARY AND CONCLUSIONS

The basic nature and characteristics of scientific and engineering data have been discussed. The testing and simulation engineering discipline is described in relation to the generation of data and the environment in which it is created. The structure of data and the models used to describe this structure have also been presented. The various types of DMS were discussed. Tradeoffs options were examined to select the optimum data model to represent the scientific and engineering data characteristics. Additional requirements of a DMS were included to provide the necessary data manipulation and analysis features.

The storage, retrieval, and analysis of scientific and engineering data has been shown to be characterized by unpredictable growth, multiple data record types, multiple networks of data, loose data controls, number of attributes, and necessity for data clarification or data restrictions. It has been shown that the basically tabular nature of the data combined with the variable structure,

analysis, and retrieval requirements can best be satisfied by the relational [Codd, 1970] concept of data. The DMS developed, based on the relational concept, fulfills the engineering and scientific needs of data storage, retrieval, and manipulation that has been described. The system provides the capability to create, delete, update, select, join, analyze, and selectively output relational data.

This effort has produced a user-friendly relationally-complete data management system for the Interdata 8/32 minicomputer. The system is operational in a multi-terminal, interactive environment provided by the Interdata operating system. The DMS is a stand-alone system programmed in FORTRAN VI through a structured programming pre-compiler language called FLEC [Beyer, 1975].

The system provides data reporting capabilities which allows selective output of both tuples and columns. In addition, two fundamental data analysis features which allow summing a column of data and performing some basic statistical analyses are provided.

The system can support a minimum of 50 relations per user, with each relation containing between 250-500 columns depending on data types. As the DMS is designed, the number of tuples that can be supported is unlimited. The actual number is a function of many variables such as disk size,

data types, and other data already on a given disk. Four data types are allowable in the DMS. These are integers, decimal numbers, single characters, and character strings.

The user is able to interactively create relations in a real-time environment. While many applications are possible, the system was developed to support the scientific and developmental testing community. The operations available allow selective retrieval, updating, and inserting of data as tests are being performed.

The design of the DMS will allow for changes in the computer hardware support capabilities without affecting the user. The design concept attempted to maximize the user's heuristic abilities by providing a group of powerful yet simple tools for his use. More sophistication to reduce the user's load leads to negation of this concept.

The use of the system to date indicates the overall objectives of the DMS have been achieved. There have been some problems uncovered. At the same time the system has demonstrated some capabilities that far exceed the initial expectations. The primary problems have been in data storage and maintaining the status indicators of the various relations. The ability of the system to recover from computer system failures has been particularly pleasing. The data retrieval and manipulation features have also proven more versatile and powerful than originally envisioned.

While not a completed product, the DMS does provide a powerful facility for data management. Continued use will provide the user requirements on which to make improvements.

REFERENCES

Bachman, C. W., Integrated Data Store Data Base Study, General Electric Co., Application Manual Pub No. CPB-481B, November 1966.

Bachman, C. W., Integrated Data Store, DPMA Quarterly, January 1965.

Bachman, C. W., and Williams, S. B., A General Purpose Programming System for Random Access Memories, Proceedings AFIPS FJCC, 1964, pp. 411-422.

Banerjee, J., Hsiao, D. K., and Kerr, D. S., DBC Software Requirements for Supporting Network Databases, Office of Naval Research, Report No. OSU-CISRC-TR-77-4, June 1977.

Bernstein, P. A., Rothnie, J. B., and Shipman, D. W., Tutorial: Distributed Data Base Management, IEEE Catalog No. EHO 141-2, IEEE Computer Society, 1978.

Beyer, T., FLECS: Users Manual, University of Oregon Edition, Dept of Computer Science, University of Oregon, October 1975.

Boyce, R. F., Chamberlin, D. D., King, W. F., III, and Hammer, M., Specifying Queries as Relational Expressions: The SQUARE Data Sublanguage, Communications of the ACM, Vol. 18, No. 11, 1975.

Canaday, R. H., et. al., A Back-End Computer for Data Base Management, Communications of the ACM, Vol. 17, No. 10, October 1974.

Chamberlin, D. D., and Boyce, R. F., SEQUEL: A Structured English Query Language, Proceedings of the ACM, Ann Arbor, Michigan, May 1974.

Chamberlin, D. D., Relational Data-Base Management Systems, ACM Computing Surveys, Special Issue Data-Base Management Systems, Vol. 8, No. 1, March 1976.

CODASYL Data Base Task Group, October 1969 Report, ACM, New York, NY.

Codd, E. F., A Relational Model of Data for Large Shared Data Banks, Communications of the ACM Vol. 13, No. 6, June 1970, pp. 377-387.

Codd, E. F., Relational Completeness of Data-Base Sublanguages, Courant Computer Science Sumposia 6, Data Base Systems, May 1971, Prentice-Hall, pp. 65-98.

Codd, E. F., Recent Investigations in Relational Data Base Systems, Information Processing 74: Proceedings of the IFIP Congress 74, 1974, Vol. 5, pp. 1017-1021.

Codd, E. F., Seven Steps to Rendezvous with the Casual User, Proceedings of the IFIP Working Conference on Data Base Management, 1974.

Date, G. J., An Introduction to Data Base System, Addison-Wesley, Reading, Massachusetts, 1975.

Davis, B., When and How to go Data Base, Data Processing (UK), Vol. 18, No. 6, September 1976, pp. 18-22.

Elliott, L., Kunii, H. S., and Browne, J. C., A Data Management System for Engineering and Scientific Computing, Engineering and Scientific Data Management Conference, Hampton, Virginia, May 1978.

Fenves, S. J., and Eastman, S. J., Design Representation and Consistency Maintenance Needs, Engineering and Scientific Data Management Conference, Hampton, Virginia, May 1978.

Fry, J. P., et. al., Data Management Systems Survey, MITRE Corp. Report MTP 329, January 1969.

Fry, J. P., and Sibley, E. H., The Development of Data-Base Technology, ACM Computing Surveys, Special Issue, Vol. 8, No. 1, March 1976.

Fry, J. P., Significant Developments in Data Base Management Systems, Proceedings of the ACM, Waterloo, Canada, September 1975.

Goldstein, R. C., and Strand, A. L., The MacAims Data Mangement System, Proceedings of the ACM, Houston, Texas, November 1970.

Held, G., Stonebraker, M., and Wong, E., INGRES: A Relational Data Base System, Proceedings of National Computer Conference, Anaheim, California, May 1975.

Hiromoto, H., Observations of a Firm's Information Processing with a Data Base Management System, International Journal of Computer and Information Sciences (GB), Vol. 5, No. 3, September 1976.

Interdata Inc., FORTRAN VI 32-Bit Run Time Library - Real Time Extensions, Pub # 29-564, July 1976.

Koehr, G., Connolly, J., Rhymer, P., Gerken, B., and Sahr, E., Data Management System Catalog, MITRE Corp. Report MTP 139, 1973.

Lien, Y. E., Design and Implementation of a Relational Data Base on a Minicomputer, Proceedings of the ACM, Seattle, Washington, October 1977.

Lopatka, R. S., and Johnson, T. G., CAD/CAM Data Management Needs, Requirements and Options, Engineering and Scientific Data Management, Hampton, Virginia, May 1978.

Lough, D. E., and Burns, A. D., An Analysis of Data Base Query Languages, Naval Post Graduate School, Monterey, California, March 1977.

Martin, J. T., Computer Data-Base Organization, Prentice-Hall, Englewood Cliffs, New Jersey, 1975.

McDonald, N. H., CUPID: A Graphics Oriented Facility for Support of Non-Programmer Interactions with a Data Base, Electronics Research Lab Report ERL-M563, University of California, Berkeley, California, November 1975.

Meldman, M. J., McLeod, D. J., Pellicore, R. J. and Squire, M., RISS: A Relational Data Base Management System for Minicomputers, VAN NOSTRAND REINHOLD, Co., 1978.

Pfaltz, John L., Computer Data Structures, Dept of pplied Math and Computer Science, University of Virginia, McGraw-Hill Book Co., 1977.

Popa, J. H., Relational Data Management, Department of Defense, Washington, DC, May 1976.

Price, C., APL/VAAM: Associative Programming Language and Virtual Associative Access Manager, General Motors Corp., May 1978.

Sharman, G. C. H., A New Model of Relational Data Base and High Level Languages, Technical Report TR 12.136, IBM Hursley Park Laboratory, England, February 1975.

Sherman, S., Panel Discussion - Current Research and Development Efforts, Engineering and Scientific Data Management Conference, Hampton, Virginia, May 1978.

Sibley, E. H., On the Equivalence of Data Based Systems, Proceedings of the 1974 ACM, Ann Arbor, Michigan, May 1974.

Systems Simulation Directorate, Description of the Advanced Simulation Center, US Army Missile Laboratory, Redstone Arsenal, Alabama, undated.

DISTRIBUTION

	No. of Copies
Defense Technical Information Center Cameron Station Alexandria, VA 22314	12
IIT Research Institute Attn: GACIAC 10 West 35th Street Chicago, IL 60616	1
US Army Materiel Systems Analysis Activity Attn: DRXSY-MP Aberdeen Proving Ground, MD 21005	1
DRSMI-LP, Mr. Voigt	1
-EX, Mr. Owen	1
-R, Dr. Kobler	1
-RPR	3
-RPT (Record Copy)	1
(Reference Copy)	1
-RDF, Dr. Hallum	25