

A COMPARISON OF RELATIONAL AND NETWORK DATA BASE  
REPRESENTATIONS OF A MEDICAL REPOSITORY SYSTEM

BY

PAULA S. BOSWELL

B.S., NORTHWEST MISSOURI STATE UNIVERSITY

MARYVILLE, MISSOURI

1976

-----

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

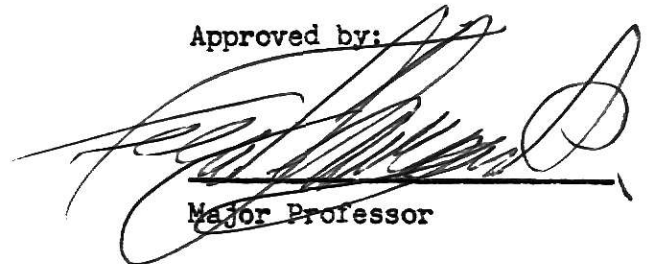
Department of Computer Science

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1978

Approved by:

A large, stylized handwritten signature in black ink, likely belonging to a faculty member, is written over a horizontal line.

Major Professor

Document  
LD  
2668  
.R4  
1978  
B68  
C.2

## TABLE OF CONTENTS

ILLUSTRATIONS . . . . .	iv
CHAPTER	
1 INTRODUCTION . . . . .	1
2 THE CURRENT SYSTEM . . . . .	4
2-1 Evolution and Use of the Current System . . . . .	5
2-2 Repository Definitions. . . . .	8
2-2-1 Clinical Repository Information Filing System (CRIFS) . . . . .	8
2-2-2 Electrocardiogram Repository (ECG) . . . . .	9
2-2-3 Treadmill Exercise Tolerance Test (TDM). . . . .	9
2-2-4 Clinical Laboratory Determinations (LAB) . . . . .	11
2-2-5 Aeromedical Evaluation Summary Cover Sheet (CS). . . . .	11
2-2-6 Cardiac Catheterization Repository (CAT) . . . . .	12
2-3 Foreseeable Developments. . . . .	15
3 DB4 DATA BASE DESCRIPTION. . . . .	16
3-1 Overview. . . . .	17
3-1-1 The Relational Data Base . . . . .	17
3-1-2 Data Set Selection . . . . .	19
3-2 Justification of the Chosen Definition. . . . .	23
3-2-1 CRIFS. . . . .	23
3-2-2 ECG. . . . .	24
3-2-3 TDM. . . . .	25
3-2-4 LAB. . . . .	25
3-2-5 CS . . . . .	26
3-2-6 CAT. . . . .	27
4 DMS 1100 DATA BASE DESCRIPTION . . . . .	28
4-1 Overview. . . . .	29
4-1-1 A Network Data Base. . . . .	29
4-1-2 Data Set Selection . . . . .	39
4-2 Justification of the Chosen Definition. . . . .	44
4-2-1 Area Definitions . . . . .	45
4-2-2 Record Definitions . . . . .	49
4-2-3 Set Definitions. . . . .	51
4-2-4 QLP Information. . . . .	54

5	COMPARISONS. . . . .	56
5-1	Basis of General Queries. . . . .	57
5-2	Queries . . . . .	58
5-3	Storage Requirements. . . . .	66
5-4	System Overhead . . . . .	68
5-5	Data Base Integrity . . . . .	69
5-6	Flexibility . . . . .	72
6	CONCLUSIONS. . . . .	75
	APPENDIX A - DB4 DATA BASE DEFINITION . . . . .	A-1
	APPENDIX B - DMS 1100 DATA BASE DEFINITION. . . . .	B-1
	APPENDIX C - DERIVATION OF SIZE REQUIREMENTS. . . . .	C-1
	BIBLIOGRAPHY. . . . .	D-1

## ILLUSTRATIONS

1.	Clinical Repository Information Filing System (CRIFS) . . . .	8
2.	Electrocardiogram File Structure (ECG) . . . . .	10
3.	Treadmill File Structure (TDM) . . . . .	10
4.	Clinical Laboratory Determinations (LAB) . . . . .	11
5.	Aeromedical Evaluation Summary Cover Sheet (CS) . . . . .	13
6.	Cardiac Catheterization Repository (CAT) . . . . .	14
7.	Set Relationships. . . . .	30
8.	Example of a Record Definition . . . . .	31
9.	Use of Calc-chain Pointers and Overflow Pages. . . . .	32
10.	Example of a Set Definition. . . . .	34
11[a].	Singly Linked Set Occurrence . . . . .	34
11[b].	Doubly Linked Set Occurrence . . . . .	34
12.	Set Occurrence Using POINTER Array . . . . .	35



## CHAPTER 1

### INTRODUCTION

This report deals with the computerized processing of medical repository data by the United States Air Force School of Aerospace Medicine (USAFSAM). Currently the USAFSAM is using the MARK IV file management system for its information storage and retrieval. The case records on file are in sufficiently large numbers such that clinical studies employing their use are able to gain statistical validity. Yet it has been found that these data are not being fully utilized due to the difficulty of retrieval of the desired information. The goal of this report is to define the current system in two different data base management systems, namely the UCC Data Management System, hereafter to be referred to as DB4, and the Data Management System 1100 (DMS 1100), and to demonstrate how these definitions and their subsequent use could help to alleviate both the present and possible future problems arising from the use of the data base.

The problems to be dealt with are many and varied. It is of prime concern to keep the logical structure simple for two basic classes of users: first, for those attempting to retrieve information for clinical studies, whom the author does not assume to be well versed in the technical aspects of information retrieval, and second, for those who, in the future, will find it necessary to modify the currently existing definition in order to meet changing information needs. Many of the systems currently in use have ignored this second consideration, causing subsequent modifications to be both costly and time consuming, if not impossible. Other problems include recovery from failure, redundancy in stored data, storage requirements, future growth of the system, and tunability as specific needs become known.

Initially one must understand the system currently in use. Chapter 2 describes the evolution of the system, its present file structure, and what is expected of the system in the future, in terms of growth and expansion.

The resolution of the problem is divided into three chapters. Chapters 3 and 4 explain the DB4 and DMS 1100 systems, respectively. Both these chapters assume the same format. First, the basic characteristics of each system are explained, including such factors as the organization of the data within the system, terminology peculiar to the system, and methods of data retrieval. Next, a justification of the data base definition derived by the author is given. Frequent references to the appendices, which contain the detailed file descriptions, will be necessary during the reading of these sections. Chapter 5 is the final chapter in the problem resolution. Here direct comparisons of the two systems are made, taking into consideration such factors as storage requirements, system overhead, flexibility, recovery, and integrity. Here one will also find specific queries that could be used in each system, based on the type of queries that are now used in the current system. Chapter 6 states the conclusions of the author.

## CHAPTER 2

### THE CURRENT SYSTEM

## 2-1 Evolution and Use of the Current System

Until the implementation of the current system, individual clinical repositories were maintained independently of each other with little or no cross-referencing between files. Due to the advent of file management systems, in this case MARK IV, it was found feasible to redesign the repositories in order to facilitate the cross-referencing desired in data retrievals for statistical analyses. The following information and file structures were derived from the report SAM-TR-77-21 as noted in reference [1].

The initial conversion effort revealed that many discrepancies already existed in the currently existing files with respect to serial numbers, social security account numbers (SSAN), names (NAME), dates of birth (DOB), sex and race. In order to maintain the integrity of the keys to be used throughout the system a master file, the Clinical Repository Information Filing System (CRIFS), was created. Input transactions are checked against this file, which is assumed to be correct, in order to verify the validity of the patient identifiers before any update to the large repositories occurs. This check includes the automatic correction of errors in the input data in the event of a keypunch error. For example, if the NAME field matches an existing name in the CRIFS file but the SSAN's do not agree, the input SSAN is checked for the possible inversion of two numbers or the mispunch of a single number. If this is found to be the case and the date of birth matches, the input transaction SSAN is modified to match the CRIFS file, the transaction is allowed to continue being processed, and a message is sent to the printer for human validation of the modification.

An individual's SSAN is used as the primary key, interconnecting the repositories, but military dependents have their sponsor's SSAN as part of their identification. Thus, one's name is used as a secondary key, with total identification employing the use of DOB, sex and race.

The current configuration allows the individual files to be used either concurrently or individually, dependent upon the particular application. Also, redundancy of data fields among the repositories is eliminated.

After CRIFS was established the other repositories were gradually added to the system. The first to be incorporated was the Electrocardiogram Repository (ECG). Following the resolution of any key incompatibilities, the ECG transaction data values are validated for correct range. In the event of an error, the entire record is printed and further processing bypassed. After the necessary corrections are made the record will be reprocessed in the next update cycle, which generally occurs every six to eight weeks. Any abnormal ECG code in a valid record causes a flag to be set in both CRIFS and the ECG record for that individual.

Next to be added was the Treadmill Exercise Tolerance Test (TDM). The necessity of weekly output requirements causes the data field validation to be done during the weekly runs. Master file updating and error corrections occur either when sufficient data have been gathered or when special retrieval is requested.

The third file added to the system was the Clinical Laboratory Determinations (LAB). In this case, daily input and output are required for those currently undergoing physical examinations. A subfile for these individuals is maintained, with reports produced for verification

of test results each day that a laboratory result is reported. These intermediate reports carry the data field validation process one step beyond the process used in the other repositories. Both unhealthy values and values outside the permissible range are denoted by special individual flags on the report printout. In addition to this comments are allowed, with the test to which the comment pertains being marked with its own flag. Master file updating occurs approximately every two months.

The Aeromedical Evaluation Summary Cover Sheet (CS) was the next repository converted. The input for this file is recorded on IBM MAG CARD II typewriters. Then, after completion, the data are sent via terminal to the IBM 360-65 for editing and validation before storing them in a temporary working subfile. After the accumulation of several hundred cases, a listing sequenced by the diagnostic codes, with any accompanying text, is printed for validation by the medical librarian. The master file is updated about once every three months.

The last file added was the Cardiac Catheterization Repository (CAT). As with the other repositories, keys are compared to the CRIFS file, data fields are validated, then the master file is updated. Corrections are included in the next update cycle.

## 2-2 Repository Definitions

### 2-2-1 Clinical Repository Information Filing System (CRIFS)

CRIFS is a small file used to validate the keys contained in transaction data. The file is considered to be definitive and, as such, its data must be accurate and current. It has no repeating groups and includes data fields, in addition to the keys, which are needed in obtaining quick look "query" type information. The repository activity status is incorporated in CRIFS through the inclusion of the following data fields: date of last examination, source of last entry, and flags indicating the existence of data in the individual repositories.

Figure 1 [1] describes this file structure.

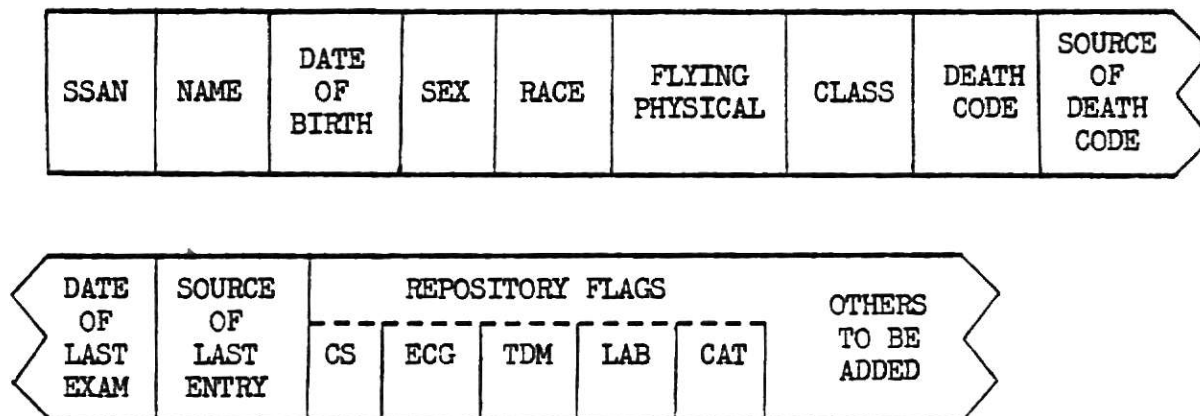


Figure 1: Clinical Repository Information Filing System (CRIFS)



## 2-2-2 Electrocardiogram Repository (ECG)

The ECG repository is represented in a record structure containing two repeating groups. The highest level of the structure consists of the two keys, SSAN and NAME, as well as a field for the vectorcardiogram (VCG) status. If the VCG status contains a "1" then a VCG for this individual is on file in the CONSULTATION SERVICE repository. Otherwise the field is a blank. Also on this level is a field, ABN-FLAG, which is set with a "1" value if this individual has ever had an abnormal ECG diagnostic code.

The first repeating group contains data pertaining to each ECG examination, with each occurrence keyed by the date of the examination. Other data obtained in the examination are recorded on this level.

The second repeating group contains the diagnostic codes as determined by the examination.

Figure 2 [1] depicts the structure of the ECG file.

## 2-2-3 Treadmill Exercise Tolerance Test (TDM)

The TDM repository is also represented in a file structure having two repeating groups. The highest level consists only of the keys SSAN and NAME with a third field indicating the number of tests for which this individual has data stored.

The first repeating group includes the individual's CASE number, examination date, and other information collected during the test.

The second repeating group contains an entry for each minute of the test completed by the patient. Up to 24 minutes of data per examination are possible, with the blood pressure and heart rate recorded for each minute, as shown in Figure 3 [1].

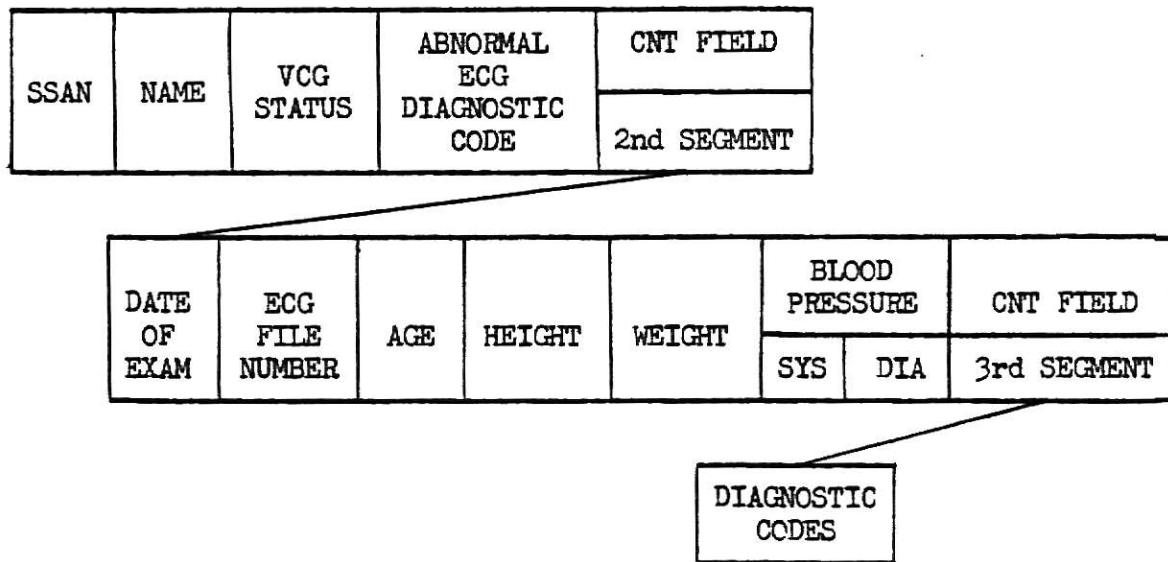


Figure 2: Electrocardiogram File Structure (ECG)

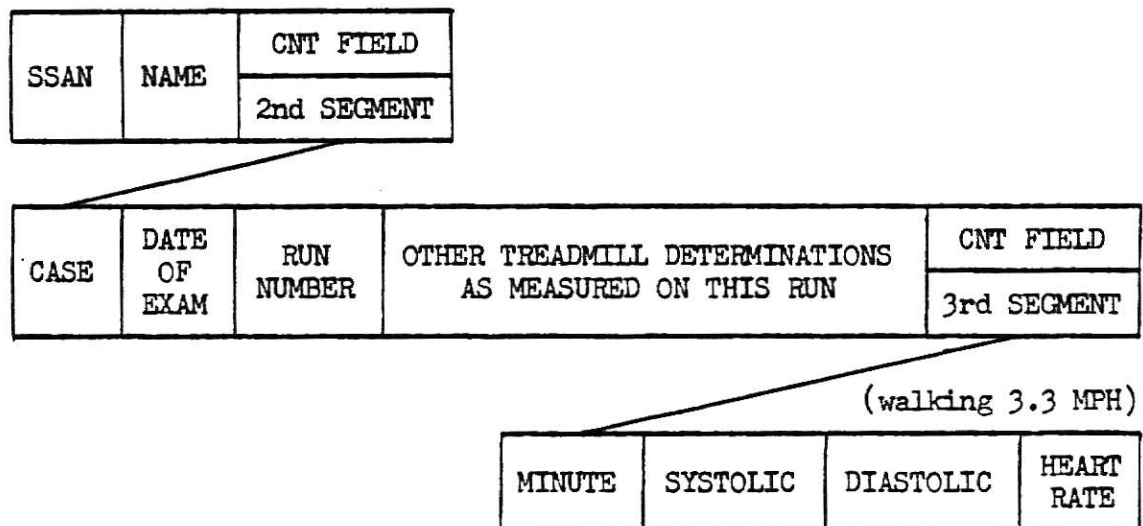


Figure 3: Treadmill File Structure (TDM)

#### 2-2-4 Clinical Laboratory Determinations (LAB)

The LAB file structure is represented in two levels. The highest level again contains the identifying keys SSAN and NAME as well as a case number. The date of the examination with the resultant laboratory determinations and messages are recorded in a repeating group. It should be noted that the laboratory determinations constitute a significant quantity of data, as these data are recorded on 12 pages of laboratory report forms.

The file structure for LAB is shown in Figure 4 [1].

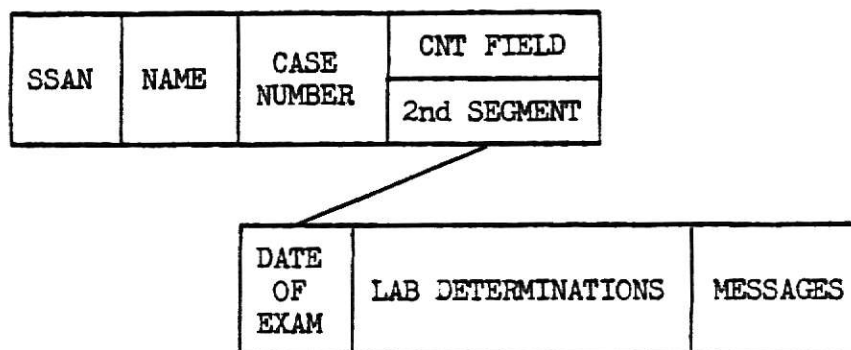


Figure 4: Clinical Laboratory Determinations (LAB)

#### 2-2-5 Aeromedical Evaluation Summary Cover Sheet (CS)

The CS file structure has three repeating groups. The record occurrence for each individual is determined by the SSAN, NAME, DOB, SEX and RACE.

The first repeating group is identified by case number and the date of the examination. A variety of other information pertaining to the individual at the time of the examination is included as shown in Figure 5.

Codes indicating the diagnosis and any accompanying text constitute the second repeating group. The physicians involved in the individual's case, as well as the physicians' particular departments, are specified in the last repeating group.

Figure 5 [1] displays the CS file structure.

#### 2-2-6 Cardiac Catheterization Repository (CAT)

The CAT file structure has, at its top level, the individual's SSAN, NAME, case number, DOE, and other data pertinent to the person's coronary risk profile.

The single repeating group is determined by the assigned sequence number and contains information from the remaining six sections of the cardiac catheterization report form.

Figure 6 [1] indicates the CAT file structure.

SSAN	NAME	DATE OF BIRTH	SEX	RACE	CNT FIELD 2nd SEGMENT	
------	------	---------------	-----	------	--------------------------	--

CASE NUMBER	DATE OF EXAM	HEIGHT	WEIGHT	MAJOR COMMAND	AERO RATING	AIR-CRAFT	FLYING HOURS	EXAM LOCATION	REFERRAL CODE
-------------	--------------	--------	--------	---------------	-------------	-----------	--------------	---------------	---------------

PURPOSE	RECOMMENDATION	AEROMEDICAL DISPOSITION	REFERRAL DIAGNOSIS	CNT FIELDS		HOME ADDRESS	MILITARY ADDRESS
				3rd SEG	4th SEG		

DIAGNOSIS CODE	DIAGNOSTIC TEXT	DEPARTMENT AND PHYSICIAN CODES
----------------	-----------------	--------------------------------

Figure 5: Aeromedical Evaluation Summary Cover Sheet (CS)

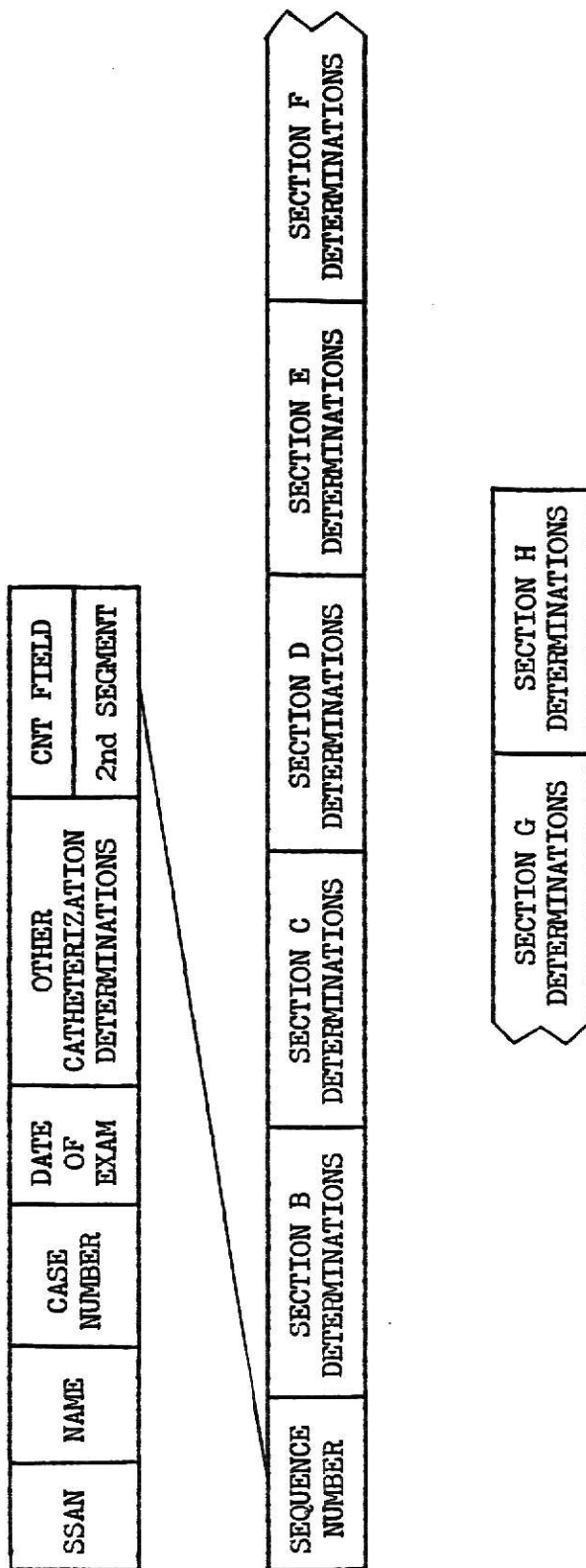


Figure 6: Cardiac Catheterization Repository (CAT)

## 2-3 Foreseeable Developments

Five repositories remain to be converted and incorporated into the current system, namely, the Vectorcardiogram (VCG), Routine Pulmonary Function Test (PULMONARY), Dental Determinations (DENTAL), Double Masters Exercise Tolerance Test (DOUBLE MASTERS), and the Tilt Table Study (TILT TABLE). In addition to these, the Hearing Conservation Registry and Waiver File, currently maintained under MARK IV, will become part of the integrated, cross-referenced system.

Future work will also include the periodic computerized validation of the CRIFS data file with the Military Personnel Center files.

In dealing with this system and its growth, one must keep in mind the magnitude of the volume of data involved. For example, the Hearing Repository currently has approximately 500,000 records on file, each one 270 bytes in length, and an estimated growth rate of 15,000 records per month. On the other hand, very few records are in existence in the Dental Repository at the present time, with each record being 1,520 bytes in length and an estimated growth rate of 40 records per month.

The overwhelming size of both the current and future repositories was considered to be a matter of great importance in the design of the data base definitions stated in the next two chapters. This problem will be dealt with explicitly in Chapter 5.

## CHAPTER 3

### DB4 DATA BASE DESCRIPTION



### 3-1 Overview

#### 3-1-1 The Relational Data Base

To a person inexperienced in the techniques of data representation, the simplest form in which to conceive data is a table, such as a multiplication table. In this two-dimensional form it is easy for one to visualize and understand the relationships existing between the data; that is, the relationships signified by the division of the data into rows and columns. The table itself represents some type of relation. DB4 is designed to manipulate the data and relationships defined in a tabular form, and, as such, is referred to as a relational data base management system.

These tables can be described mathematically and have the following properties [2]:

1. Each entry in a table represents one data item; there are no repeating groups.
2. They are column-homogeneous; that is, in any column all items are of the same kind.
3. Each column is assigned a distinct name.
4. All rows are distinct; duplicate rows are not allowed.
5. Both the rows and the columns can be viewed in any sequence at any time without affecting either the information content or the semantics of any function using the table.

Many files in current applications incorporate the use, either logically and/or physically, of repeating groups, which, from the first property above, is not allowed if the data involved are to be represented in a relational data base. Thus, the files must be converted into "flat", two-dimensional files through a process referred to as normalization. In normalization, the advantages of which are discussed in Chapter 5,

a repeating group is separated into an individual file, necessitating some data items, referred to as signatures in DB4, to appear in more than a single file for the purpose of record identification. Due to the storage techniques used in DB4, this repetition of data in the user's logical view does not imply an increase in storage requirements. Keyed items are stored separately from the data to which they belong. This separation is hidden from the user, being handled entirely by the DB4 system. The strength of retrieval techniques in DB4 is dependent upon this separation.

The DB4 data base consists of three logical spaces—control space, data space, and structure space [3]. The control space is comprised of information pertinent to the running of the system and is produced by the system. The data space is a software simulated virtual memory system composed of two parts, the data itself and a virtual directory that controls addressing to the data. Mass storage for this space is set up during the establishment of the data base definition via the execution of the Data Base Definition (DBD) processor, and both of its constituent parts will reside on the same mass storage device. Data space and control space are interconnected through the inclusion of the virtual directory as one of the segments of control space. This directory is paged in and out of core through a double buffering technique such that the number of physical accesses to the virtual directory during record retrieval is minimized.

Structure space constitutes what could be considered the lifeblood of the DB4 system due to the fact that any and all queries are dependent upon the existence of this space. Before a data base may be queried two operations, occurring independently, must be performed. First, the data

must be loaded, via the LOAD command, into an area of the data space. Then the relationships existing between the new data and any previously loaded data must be established through the use of the CONSTRUCT command. This command creates the entire contents of the structure space, which consists of storage, reserved by the DBD processor, for all values possible for all uniquely named keyed items throughout the data base, as well as an indication of the record occurrences in the data base that employ the use of these keyed fields.

Retrieval of records in a DB4 data base involves the use of a collection, which is a software simulated associative memory that contains pointers to records requested by a FIND command. Several collection names may be used by a single user, with these names specified in a person's User Interface Description (UID). The UID provides an individual user's view of the data base and also allows the user to communicate with the Data Base Monitor (DBM), which is the command language module used to load, maintain and access the data base. Through the DBM a data language is also provided wherein a non-programmer can interact directly with the data base.

All the records in the data base that fit a particular description constitute a logical file which is referred to as a data group or area. Keyed signatures which are contained in more than one data group provide the means of cross-referencing the files during queries. Data set selection through queries is now described in the following section.

### 3-1-2 Data Set Selection

Retrieval of data is totally dependent upon keyed fields, with these fields forming the logical and physical basis of relating a record from

one data group to existing records of another area.

The standard processing command used to isolate a requested data set is the `FIND` command. It has the following general form:

`FIND collection-name Boolean expression`

The collection name is one which is defined to be in the user's UID, and the Boolean expression must contain only keyed signatures or other previously established collections.

Execution of this command does not access the data space. Only the structure space is affected. After the execution of a `FIND` command, the named collection will contain pointers to all records which satisfied the query, or more specifically, all records throughout the data base to which the user is allowed access, via the individual's UID, that satisfied the query.

As an example, suppose `AGE` is a keyed signature and `DATA` a valid collection name. Then

`FIND DATA WHERE AGE GE 16 THRU 65`

will isolate all records whose `AGE` field contains a value from 16 through 65. [TO is exclusive and THRU is inclusive.] Had the following form been used

`FIND DATA WHERE AGE EQ 16 THRU 65`

then at least one record occurrence must have had a value of 16 in `AGE` or else a null set would have been returned.

Once a data set is isolated it may undergo additional processing to further refine the data set. If the above query were to be followed by

`FIND PEOPLE WHERE OCCUPATION EQ 'STUDENT'`

and then

```
FIND STUDENTS WHERE DATA AND PEOPLE
```

all students from 16 through 65 would be isolated in STUDENTS.

Other command sequences could have been used instead. The following two command series would leave the same data set as above in STUDENTS.

```
FIND STUDENTS WHERE AGE GE 16 THRU 65  
AND OCCUPATION EQ 'STUDENT'
```

```
FIND DATA WHERE AGE GE 16 THRU 65  
FIND STUDENTS WHERE OCCUPATION EQ 'STUDENT'  
FIND STUDENTS AND DATA
```

This last statement would leave the results in the first collection name while the second collection remains unchanged.

If the records containing the AGE and OCCUPATION fields were in different data groups, then in order to find those people belonging to both groups there must have been some way of relating the different record types to each other. This was done by the system, but it was dependent on the fact that the different data groups contained some shared keyed field, such as ID-NUMBER.

In some applications data may be considered to be stored in bit form. Keyed signatures which are bit-defined may also be queried by using the logical operator MV, which stands for "mask value." The general form would be

signature MV value

where "value" supplies the mask. For example, if JOB-CODE is a six bit field included in a record describing parts information, and the setting

of the low-order bit signifies that this part is used for JOB1, it is possible to retrieve all parts used in JOB1 with the following command:

```
FIND DATA WHERE JOB-CODE MV '000001'
```

with the results left, as before, in DATA. Any data sets previously isolated in DATA would have been destroyed by this command.

The above examples allowed the system to optimize the retrieval process, with the user not even having to specify the data groups from which the data were to be selected. As mentioned previously, if the data were from different areas, then the system would have looked for a shared keyed signature in the two or more areas involved.

Suppose, though, that the user does have knowledge of the data groups and wishes to confine the query to particular data groups within the data base. Taking the first example, if AGE is in the PERSON data group, OCCUPATION in the JOB data group, and both PERSON and JOB contain the keyed signature ID-NUMBER, the following sequence

```
FIND DATA WHERE AGE GE 16 THRU 65  
PROJECT DATA FROM PERSON THRU ID-NUMBER INTO JOB
```

will isolate in DATA all people from 16 through 65 along with the JOB information pertaining to each of these persons. This is equivalent, in the context of the tabular display of information, to pulling all the rows from PERSON where the desired age range exists, and adding to this table all the columns of JOB which are unique with respect to PERSON (that is, shared keyed fields will not be repeated), with the resulting table in DATA.

### 3-2 Justification of the Chosen Definition

The first step in converting the MARK IV files to a DB4 representation involved the isolation of the repeating groups, necessary due to the first property stated in Section 3-1-1. The method of normalization used for each repeating group was similar. To avoid redundancy in the following text, a synoptic discussion of the method used is now given.

Each occurrence of a repeating group in a MARK IV file was separated from the main record description and placed in its own data group. Within the newly created data groups, record occurrences for an individual were differentiated by the same data items employed in the MARK IV repeating groups, which, in most cases, was the date of the examination.

Positive record identification for data pertaining to an individual is dependent upon both the SSAN and NAME fields. Therefore, these fields occur in each data group, making the projection of one data group into another very easy throughout the entire data base.

Exceptions to these generalities will be noted as they are encountered. Appendix A lists the DB4 data group definitions, to which frequent references will be necessary throughout the remainder of this chapter.

#### 3-2-1 CRIFS

All the CRIFS information is contained in a single data group. (See Figure 1 and A-2.) Since this file is used for validation of input transactions and also for "quick-look" query information, its record size remains small but contains a variety of information. The fields included with the FLAGS field allow one to quickly determine which repositories within the data base contain at least one record occurrence for an

individual. This field could have been omitted, but then to obtain the same information, a query of the desired data group would have to have been performed and then the collection inspected to determine whether or not it was null. The inclusion of FLAGS enhances the simplicity of the query.

Space for flags to repositories to be added at a later date was not reserved. This is because the addition of data groups describing other repositories will necessitate the dumping and re-establishment of the data base via the DBD processor. At the time this becomes necessary the required flag space may be added.

The VCG field was added to eliminate the need for an additional data group in the resolution of the ECG repository, as will now be seen.

### 3-2-2 ECG

A data group for the highest level of this file was not developed. (See Figure 2 and A-3.) The VCG status information was included in CRIFS through the addition of the VCG field, and, also with CRIFS, an abnormal diagnostic code for this repository was indicated in the ECG field by a '2' value, with normal codes indicated by a '1'. This special field designation for abnormal values is currently in use in the MARK IV system. The ECG-DX-CODES data group records each diagnosis determined in the course of this particular examination.



### 3-2-3 TDM

The entire TDM file was represented as a single record in the TDM data group. (See Figure 3 and A-3 through A-4.) The SSAN and NAME fields in the highest level of the MARK IV file structure had, for purposes of record identification, to be included in the recording of the treadmill results, and as such, a separate data group for these fields would serve no function in a relational representation of the file.

A data group to record data obtained for each of the 24 possible minutes of the test would have had to include as keys the SSAN, NAME, CASE-NO, and DOE fields, as well as a field to indicate the minute, each needed to uniquely identify each record occurrence. Though this would not have increased the storage required for the keys, it would have greatly increased the structure space needed to relate all the records. Based on the assumption that most patients will complete most or all of the test, space is reserved in each record for the recording of all 24 minutes of results, rather than creating a separate data group. When the test is not completed, null values will be registered in the incompleted minute fields.

### 3-2-4 LAB

As with the TDM file, all information included in the highest level of the record structure was required for identification of all laboratory results. (See Figure 4 and A-4 through A-5.) A single data group, LAB, records all the laboratory determinations obtained during a single examination period. Twelve pages of laboratory results are contained in a single record, with these results received over the

course of a week. Since a single DOE field is indicated, it will contain the date on which the examination was started.

Messages pertaining to an individual test are possible, with the number of these messages expected to be quite low. These are recorded in the LAB-MESSAGES data group, with a message identified by SSAN, NAME, CASE-NO, DOE and TEST, indicating to which test the message pertains.

### 3-2-5 CS

This file is quite important since it contains a summary of an individual's evaluations. (See Figure 5 and A-5 through A-6.) To make absolutely certain that a record retrieved from the CS data group is applied to a positively identified person, DOB, SEX and RACE are included for this person's identification.

The CS-DIAG (cover sheet diagnostic codes and related text) data group was developed for the purpose of query requirements. A large number of currently used queries are dependent on the DX-CODE. To have this diagnostic information occur more than once in a single data group record occurrence would necessitate the use of more than one signature in an array-like representation. Thus, regardless of how many or how few diagnostic codes may be recorded in a single evaluation, each one must be recorded separately.

A separate data group, CS-PHYSICIANS, was also created to record which physicians were involved in the evaluation, but rather than being for query purposes, it was necessary in order to keep the data base representation in a normalized form.

The CAT data group contains information which is associated permanently with an individual, and, as such needs to be recorded only once. (See Figure 6 and A-6 through A-7.) The determinations recorded for each examination are recorded in the CAT-DET data group, with individual record occurrences determined by both the catheterization sequence number and the case number.

The types of queries encountered have made the recording of some data to be made in bit form. For example, under the section C-11, which pertains to the angiography completed, anywhere from zero to nine results could be recorded. On the catheterization form these are recorded by writing 01, 02, ..., 09, but a query may want to retrieve information pertaining to individuals with either a 02 or 09 recorded. As with DX-CODE, a separate data group could have been created, but the use of bits and a mask value in the query greatly simplifies both the query and storage. With six bit bytes, if the recording of '010000001000' shows that 02 and 09 were recorded for C-11, then

FIND DATA WHERE C-11 MV '010000001000'

would isolate all records with at least these two results. The user of the data base would have to be aware of this special recording of the data.

Several facts mentioned in this chapter will factor heavily in the comparisons to be made in Chapter 5. The separation of keys from their associated data is very important, affecting every concern to be investigated, namely storage requirements, recovery, expansion, the ease of query and data base definition, future redefinition of the data base, and tunability. One should also be sure to understand the physical representation of the data base in terms of the data, structure and control spaces.

## CHAPTER 4

### DMS 1100 DATA BASE DESCRIPTION

## 4-1 Overview

### 4-1-1 A Network Data Base

Another way of relating data items or records to each other, besides the tabular representation of the relational data base DB4, is to have one or more record occurrences dependent on another record for their identification and selection. This type of representation would resemble a directed graph, where edges between nodes indicate logical relationships. A data base dependent on this type of logical structuring is referred to as a network data base, and the following information describes, in particular, the data base referred to as DMS 1100 [4].

Inherent to the understanding of data manipulation in a network data base is the concept of a set and set relationships. A set is a named collection of record types. These record types have a hierarchical relationship to each other, since one must be defined as the "owner" and the others as "members". For example, a record type such as CUSTOMER could be considered to be the parent or owner of one or more occurrences of the record type TRANSACTION (see Figure 7). At the same time, in another set type, it would be possible to designate TRANSACTION as the owner of each CUSTOMER having made a given type of transaction. More information about sets, such as their definition and implementation, will be given later.

An area is a named subdivision of a data base. There are three basic types of areas. A data area contains record occurrences of one or more record types. All data stored within the data base must be contained within a data area. This is the only area type whose existence is mandatory.

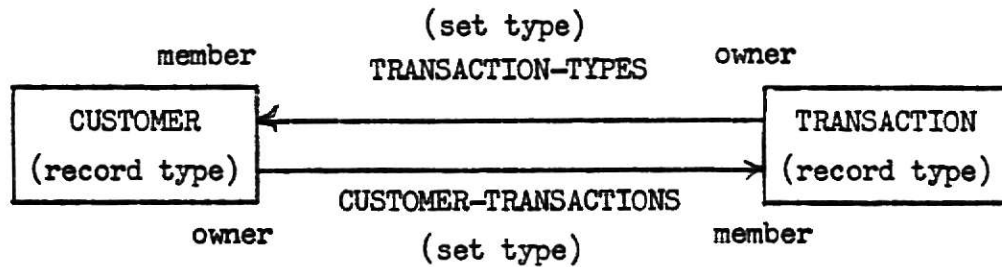


Figure 7: Set Relationships

An index area is used in conjunction with a record type whose storage and retrieval is considered to be through index sequential access. The area will contain two items: first, a list of indices, determined by the values of data items within a record occurrence which will be used to uniquely identify that record occurrence; and second, with each index is the page number on which the record occurrence that the index identifies is stored. Thus, when seeking a particular record, only the indices need to be searched to determine the record's location, rather than searching the data area in which the record is stored.

The third type of area is a pointer area. Rather than being associated with a record type, a pointer area is related to a given set type. The area will contain a number of arrays, each one associated with the owner of a set occurrence, and consisting of pointers to each member record.

The storage space for each area is divided into pages, which function as the basic unit for input/output transfers between mass storage and internal core. Once defined, the page size for each area is fixed; thus, care must be taken when determining the page size. Derivation of the page sizes will be explained later in Section 4-2.

Record placement is handled in one of four ways, dependent on the location mode specified in the record type's definition. (See Figure 8

for an example of a record definition.) First, a record type could have a location mode of DIRECT. In this case, the user must supply the area name as well as both the page number in which this record occurrence will be stored and the record's relative record number within the page.

```
RECORD NAME IS FIRST-RECORD  RECORD CODE IS 1
LOCATION MODE IS VIA FIRST-SET SET
WITHIN FIRST-AREA
O2 NAME    PIC X(18)
O2 AGE     PIC 9(3)
```

Figure 8: Example of a record definition

If the location mode of CALC is specified, the record's location will be determined by a procedure, either system or user defined. One item calculated by this procedure is a page number. Each page header has one word reserved for a calc-chain pointer. If a record is to be placed on a page through CALC, (see Figure 9), this pointer is inspected to determine if it is null. If so, the record becomes the first record on the page, with the header pointer containing its address, and the record itself containing a pointer back to the header. Subsequent additions will be linked into the chain, with the header pointing to the newest record, the newest record pointing to the next newest record, and so forth, with the first record stored pointing to the head of the chain. When a page becomes full and a new record, with the same calculated page number, needs to be added, the new record will be placed on an overflow page, with its chain pointer indicating the last previously stored record on the original page, and the head of the chain, on the original page, pointing to the new record occurrence. This allows a new record occurrence

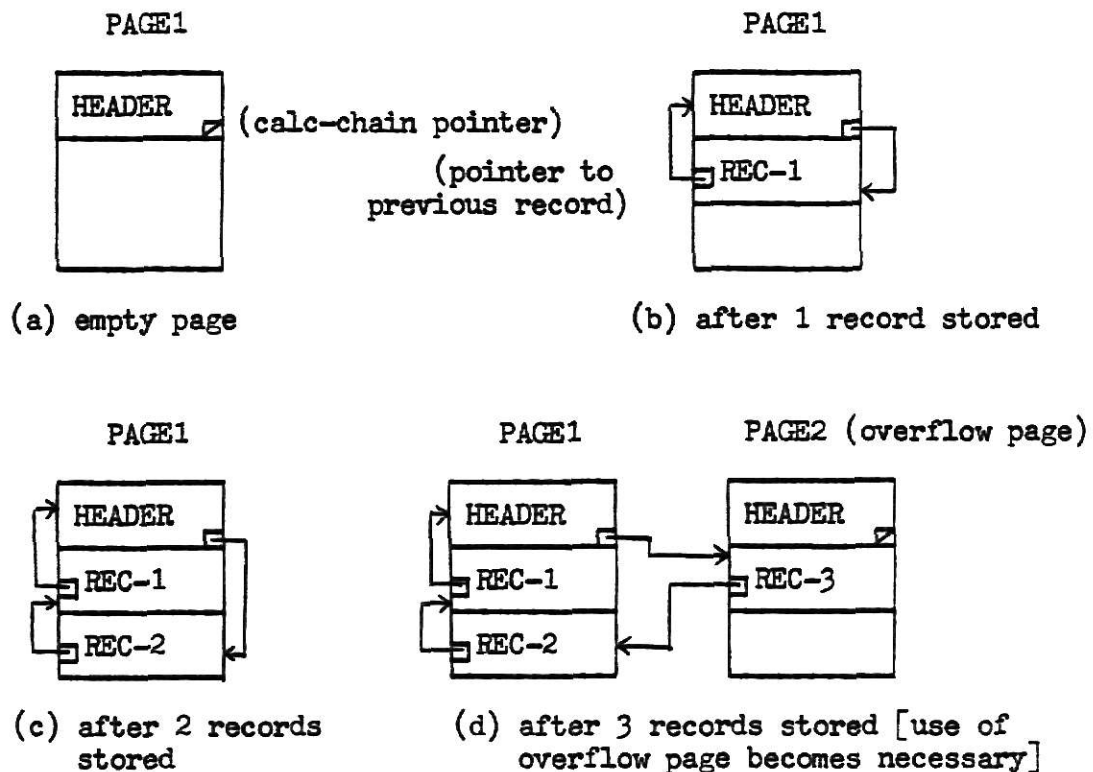


Figure 9: Use of calc-chain pointers and overflow pages

to be stored quickly, since if the header points to a record on an overflow page, new records will also go to the overflow page, rather than futilely searching for a position on the original page.

With this technique it is possible to have to follow a chain from page to page during a search to retrieve a record calculated to be on a particular page, which could become quite time consuming. To help alleviate this problem, it is possible to specify that more than one calc-chain be used per page. The CALC procedure will then determine both the page and the calc-chain number. These multiple chains shorten the chain length which would need to be traversed to retrieve a record.

A third possible location mode is INDEX SEQUENTIAL. As seen earlier, record types stored in this fashion contain data items which are to be used as indices for their retrieval. Before loading a data base, records



using this method must be sorted by their identifying data values. Then, in physical storage, they will be stored in sequence, with later additions handled by the system, and the indices stored in an INDEX area.

The last location mode is VIA set. In this instance a record is expected to be retrieved only with respect to its owner record occurrence, so when it is stored, it is placed as near to its owner as possible. Therefore, an owner will contain a pointer to its first member, with the members chained together and the last member pointing back to the owner, thus forming a ring.

The location mode for a record type is made known to the system in the record's definition at the time the data base is created. Another entry which is possible within this record definition is the OCCURS clause. This allows a data item or group of items to be subscripted and eliminates the need for separate entries within the record. In effect, it allows variable length records to be manipulated, thus producing a savings in storage. Some system overhead will be incurred when the record is compacted for storage and then expanded to its full size when retrieved for inspection or update.

From the preceeding discussion of pointer areas and the VIA set location mode, one can now better understand the implementation of sets. (See Figure 10 for an example of a set definition.) The DMS 1100 has two means of implementing sets. One method is by the CHAIN mode. In this case the owner is linked to its members through a chain ring and is considered to be "forward linked" (see Figure 11[a]). When using the set occurrence, it may be desirable to traverse the member records in reverse order, or even to traverse it in one order and then want the previously processed record. This requirement is made possible by including

SET NAME IS FIRST-SET SET CODE IS 1  
 MODE IS CHAIN  
 ORDER IS SORTED  
 OWNER IS SOME-RECORD  
 MEMBER IS FIRST-RECORD AUTOMATIC  
 ASCENDING KEY IS SOME-FIELD DUPLICATES ARE NOT  
 ALLOWED  
 SET OCCURRENCE SELECTION IS THRU LOCATION MODE  
 OF OWNER

Figure 10: An example of a set definition

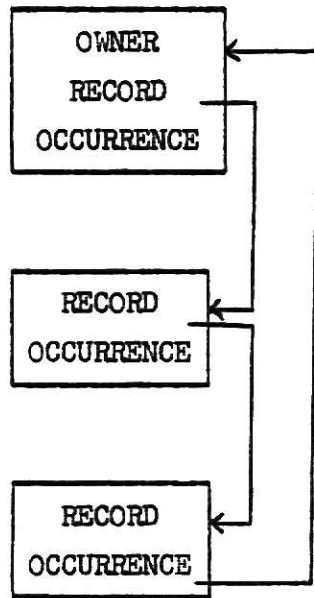


Figure 11[a]: Singly linked  
set occurrence

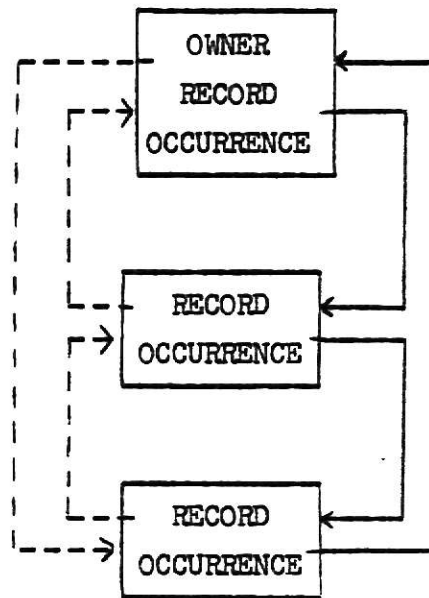


Figure 11[b]: Doubly linked  
set occurrence  
(broken lines indicate PRIOR links)

the LINKED PRIOR clause, causing each member record to contain two  
 pointers, one to the record preceeding it in the chain, and one to the  
 record following it in the chain (see Figure 11[b]).

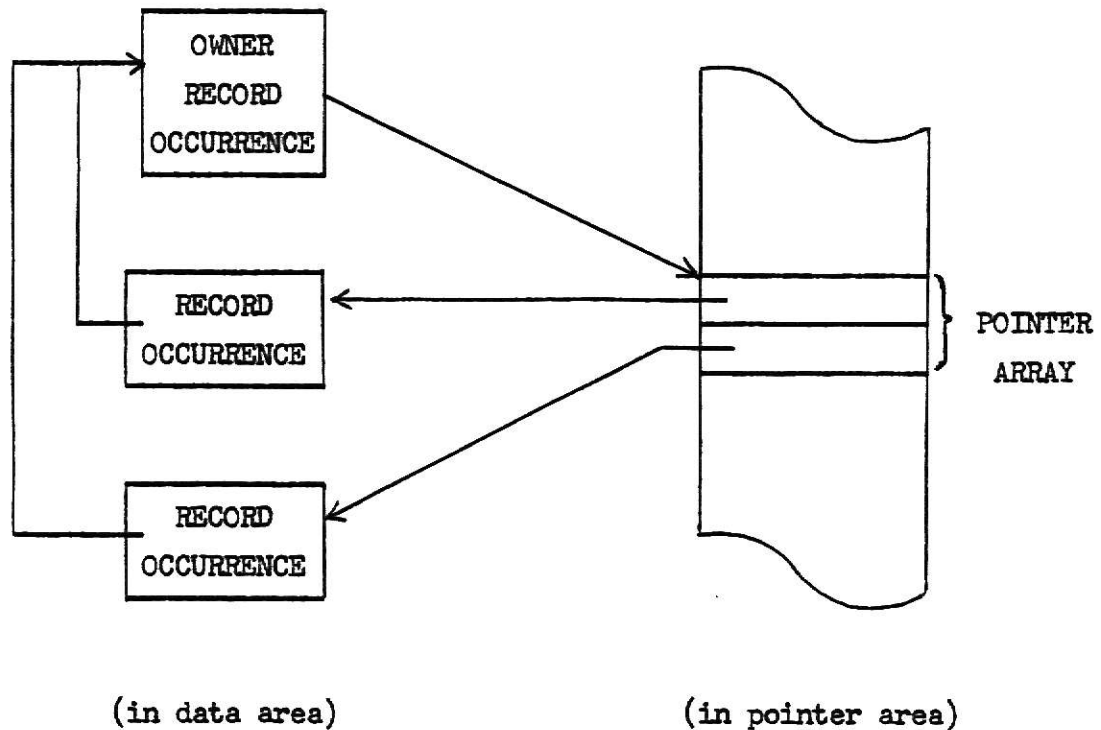


Figure 12: Set occurrence using POINTER array

The second method of implementing a set is through a POINTER array (see Figure 12). Each member occurrence of the set will have its location in a pointer array within a named pointer area. The owner of these records contains a pointer to the pointer array. Therefore, the owner's link to its members is only through its pointer array. In addition, each member will contain, in its physical storage location, a pointer to its owner.

It is usually advantageous to keep the member records of a set occurrence in some sort of order. For example, one may want new records to be placed at the front or at the end of the set, or possibly to be stored either before or after a member record occurrence which is currently being inspected. Alternately, rather than the user determining the point of insertion, the system could perform this function, placing records

according to the values of data items within the records, with candidate items possibly being a customer number or date. The set order is specified at the time the data base is initiated, within the SET definition.

This SET definition also specifies which record type is to be the owner and which record type or types will be members. The MEMBER entry indicates when an occurrence of this entry's record type is to be added to the set. Two possibilities exist. When the record is created its inclusion could be AUTOMATIC, meaning that as soon as it is created its owner is found, and the member is linked into the set. On the other hand, the creation of a record may not warrant its automatic inclusion in a set. For example, the creation of a CUSTOMER record does not mean that it should be inserted into a DELINQUENT-ACCOUNTS set. If, at a later date, this is found to be necessary, the insertion will be MANUAL, through some programmed procedure.

The MEMBER entry also determines if each member should contain a pointer to its owner, alleviating the need to traverse the rest of the set in order to return to the owner. If the set is to be sorted, the ASCENDING (DESCENDING) identifying data fields are specified, as well as an indication of what should be done with records having duplicate field values.

Following the MEMBER entries for each set is the SET OCCURRENCE SELECTION clause, which defines how the appropriate occurrence of a set will be selected. If it is known that the set relationships will be used only after an owner has been procedurally preselected, then the CURRENT OF SET will yield the proper set occurrence. If the owner must be determined and then the set accessed, both in the same query, then LOCATION MODE OF OWNER is specified. The owner will then be found, either directly

through its own identifying data values or else through its participation as a member in another set type.

In any set type, an owner record occurrence may appear only once. Otherwise, the selection of a set occurrence would not be guaranteed to be unique. Also, any record type may participate in several sets, either as an owner or a member but not both within the same set type.

Everything discussed so far is brought together to define the data base in what is called the source schema. This schema is used as input for the Data Definition Language (DDL) Translator, which produces a series of interpretative tables known as an object or absolute schema, to be used by other DMS 1100 programs, such as the Data Management Routine (DMR) and the Data Manipulation Language (DML) Preprocessor.

The DMR consists of a set of routines which access and update the data base through DML commands, based upon the schema from the DDL translator. Data base keys used by the DMR are 36 bit words, indicating an area code, a page number, and a relative record number. These keys are either provided by the user, in the event of a DIRECT location mode, or derived by the system through information provided by the user, such as an area name and values for selected data items.

The SPERRY UNIVAC 1100 Series Query Language Processor (QLP 1100) provides a means of interacting with a DMS 1100 data base, either conversationally or in a batch production mode [5]. Before using the QLP 1100, a user must COPY from the schema the areas, records, and sets required for processing, thus creating a subschema, which may be a subset of the absolute schema or the whole schema. In either case, the subschema is referenced by a user supplied name and may be saved for use in a later session.

Integrity of a DMS 1100 data base is accomplished through the combination of dump tapes and audit tapes [6]. The data base administrator determines the contents of the audit tape at the time the data base is established. One option could be to incorporate the use of a checkpoint and after looks. Each time a user modifies a page and returns it to the data base, an after look of the page, that is, its contents after modifications, will be copied to the audit tape. A checkpoint written to the tape indicates a point in time and must be written when the data base is not in use. Thus, when an area or the entire data base need to be recovered, the selected area(s) are restored from dump tapes, and after looks are applied, which were recorded from the time of the dump to the desired checkpoint time.

A second option involved the use of a recovery point, before looks, and after looks. A recovery point also indicates a point in time, but it may be written to the audit tape while the data base is being used. After looks are the same as before, and before looks are images of a page written to the audit tape just prior to the page being altered. Recovery would proceed as before, with after looks being applied up to the recovery point. Then, if the data base was in use at the time the recovery point was written, the effect of any user active at that time is reversed by applying the before looks recorded during that session and before the recovery point was written, thus "backing out" any active users.

In the event of a DMR internal error or if a system failure is encountered, a quick recovery process may be initiated to reverse the effects of any active users on the data base. In this process quick-before-looks are applied to back out the active users. These looks are

the same as before looks, with the difference being that they are recorded on a fast random access device, exist only while a user is active, and are used only in quick recovery. Any area not employing the use of quick-before-looks could be unstable or inaccurate in the event of a quick recovery being initiated during an update procedure.

#### 4-1-2 Data Set Selection

The retrieval of a record occurrence in a DMS 1100 data base is dependent on the schema designed by the user; more specifically, it is dependent on the location mode specified for a record type. A record may be selected upon the basis of user supplied data, from which a data base key is derived by the DMR to determine a record's location in the data base. Another selection process is possible through the set relationships established between record types.

The most straightforward method of retrieving a record is through the location mode of DIRECT. The user is, in effect, responsible for knowing where the record is located, with respect to which area the record is in, what page number it is on, and what the record's relative position is from the top of the page. The DMR will use this information to form a 36 bit data base key which is then used to locate and retrieve the record.

If the location mode is CALC, the only thing the user needs to know about the record's physical placement is the area in which it resides. When the record was originally stored, its location was computed by a routine, based upon the values of data items within the record which would uniquely identify it. To retrieve the record these same data item values, as well as the area name, need to be available to the routine,

which will then determine the page number and calc-chain number used to store the record. The DMR will then locate the page and follow the calc-chain until it finds the record having the same data item values as those being sought. This is necessary because the same page and calc-chain numbers could be derived for different records with similar data item values.

The INDEX SEQUENTIAL mode requires the same information as in the CALC mode, but the method of retrieval is different. In this case the data values are combined to form an index. Then, in the index area associated with the record's data area, a quick search is made among the sequentially stored indices for one matching the index formed from the user provided information. The page number associated with the index is located, and the record is retrieved as before.

Retrieval VIA set relationships is more interesting. First the owner must be determined, and then the members may be located and retrieved. The following examples will deal with queries one could use with the QLP 1100, but first an introductory note to the QLP 1100 techniques.

When establishing a subschema for the QLP 1100 the user must also specify the "paths" which may be used to traverse the data base; that is, the user must supply a list which describes through what record types entry to the data base will be made, and then, from these entry points, what sets and other record types will be used [5]. For example, if CUSTOMER is a record type, stored by CALC using ID-NO, and it is the owner of HOME using the LOCATION set, then the following path declaration would allow one to use the set.

```
PATH NAME IS LOCATION-PATH
      ROOT IS CUSTOMER
      THRU LOCATION TO HOME
```



Thus, to use the LOCATION-PATH path, the user must provide the data required for the location of CUSTOMER. A query to print the NAME within CUSTOMER along with that person's STATE which is in HOME would look like

LIST NAME STATE WHERE ID-NO = '4444' VIA LOCATION-PATH

One could also have used

LIST NAME STATE WHERE ID-NO = '4444'

In this case, the QLP 1100 would first have formed a list of the needed record types to obtain the NAME and STATE items. Then the path declarations would be inspected to find a path containing the records which could be entered and traversed using the data values given. It is possible that more than one path may satisfy the query, in which case the first path found will be used. The order of the paths is determined by the location mode of the root record, with DIRECT being first, followed by CALC, INDEX SEQUENTIAL, DIRECT CURRENCY ASSUMED, and finally, FETCH NEXT CURRENCY ALLOWED. An example will clarify this. In addition to the above information, the data base will be considered to contain the EMPLOYEE record type, which has a location mode of INDEX SEQUENTIAL using ID-NO. Through the JOB-LOCATION set this record is the owner of the WORK-SITE record type. Thus, the following path would be valid.

PATH NAME IS JOB-SITE-PATH  
ROOT IS EMPLOYEE  
THRU JOB-LOCATION TO WORK-SITE

Provided EMPLOYEE contains a NAME field and WORK-SITE contains a STATE field, the query

LIST NAME STATE WHERE ID-NO = '4444'

could be satisfied by either LOCATION-PATH or JOB-SITE-PATH, assuming a person with an ID-NO = '4444' is included in both. From the information concerning path orders, one can see that LOCATION-PATH, in which CUSTOMER is chosen by a CALC procedure, will be selected.

Further qualification of the selection criteria would involve the addition of more data item values to the WHERE clause. To return to the above example, WORK-SITE could be the owner of the EMPLOYEE records which describe those persons assigned to the particular work location. The path

```
PATH NAME IS EMPLOYEE-DISTRIBUTION
      ROOT IS WORK-SITE
      THRU SITE-EMPLOYEES TO EMPLOYEE
```

would use the SITE-EMPLOYEES set to retrieve EMPLOYEE records owned by the selected WORK-SITE record occurrence. If WORK-SITE has a location mode of CALC using CITY and STATE data item values, and EMPLOYEE also contains AGE and WEIGHT fields, then the query

```
LIST NAME WHERE CITY = 'HAMPDEN' AND ;
              STATE = 'MAINE' AND ;
              AGE < '30' AND WEIGHT < '143' ;
              VIA EMPLOYEE-DISTRIBUTION
```

would first determine the location of a WORK-SITE record, using the CITY and STATE data values. If a record occurrence is found, then the other fields are sought, causing the SITE-EMPLOYEES set to be traversed to locate the NAME, AGE and WEIGHT fields. In each EMPLOYEE record owned by the isolated WORK-SITE record occurrence in the SITE-EMPLOYEES set, the data fields AGE and WEIGHT are checked against the values specified in the query. Then the name of each employee, if any, satisfying the query is printed.

Entry to the data base without the use of a key is possible with the QLP 1100. A query necessitating this would be of a more general nature, such as

LIST NAME AGE WHERE OCCUPATION = 'DOCTOR'

where OCCUPATION is not a keyed data field. NAME could be a field in a PERSONNEL record which owns, through the set INFORMATION the DATA record, which contains the AGE and OCCUPATION fields. In addition, the PERSONNEL record type has a location mode VIA another set. A possible path could be

PATH NAME IS PERSONAL-INFO  
ROOT IS PERSONNEL  
FETCH NEXT CURRENCY ALLOWED  
THRU INFORMATION TO DATA

which would cause retrieval of the root records to be through a DMR FETCH FIRST/NEXT PERSONNEL OF AREA command. Thus all the root record occurrences and their members would be inspected to find the desired information.

One can see that this could be a very time consuming operation, dependent on the number of PERSONNEL records contained in the data base. Another important fact the user must remember when developing queries is that when processing a single query, the QLP 1100 may use a record type and set type only once. That is, if REC-1 owns REC-2 through SET1, and REC-2 owns REC-1 through SET2, a query could not traverse both SET1 and SET2 since this would cause one of the record types to be accessed twice in a single path.

#### 4-2 Justification of the Chosen Definition

A network DBMS such as DMS 1100 is noted for its capacity to interrelate different record types through the use of sets. The user is afforded a degree of control over the physical placement of records, as well as being given the capability to assist the system in its efficiency of data manipulation, through the creation of a schema which will serve as a definition for the entire data base. All the activities of DMS 1100 are dependent on this schema, using it as a type of road map to determine record locations. The user does not need to be concerned with the physical aspects of a record's location, such as what track or cylinder it resides on, because the system will perform all the I/O that is necessary. Instead, the user needs to provide only that information upon which a record's placement is dependent, such as values of particular fields. Then the location of a record for retrieval will be determined by the same method that was used to store it initially. Alternately, it can also be located through its participation in one or more sets.

Rather than approaching each repository individually, as was done in Chapter 3 with DB4, the DMS 1100 better lends itself to a discussion based on the three sections which form the schema definition, that is, the AREA, RECORD and SET sections, followed by the special considerations involved in the use of the QLP 1100.

To assist the reader, a logical diagram of the DMS 1100 data base definition is included at the end of Appendix B (see B-10). From this diagram one can see how record types are related by sets, with a legend included to explain the set characteristics.

#### 4-2-1 Area Definitions

Preceding the actual area entries (see B-2 through B-3) are two statements important to subsequent use of the data base. The AREA CONTROL statement indicates the maximum number of areas that are expected to be used. Its purpose is to indicate the number of bits to be used to indicate an area in the data base keys formed by the DMR. If omitted, 12 bits will automatically be assumed. In this schema the minimum allowable number of areas is specified, 127, reserving 7 of the 36 data base key bits for the area code, since the number of areas to be used should not exceed 127. A benefit of this is the fact that more bits may be used for page and record numbers, in the event that an exceptional number of either of these is required.

The next statement in the schema specifies what AREA LOOKS are to be taken. Updates to the data base are expected to be in a batch mode and from data stored on tape. Therefore, rather than have a recovery point written to the audit tape automatically at predefined intervals, the use of checkpoints, written only upon request, were considered to be sufficient. A large number of updates could be made at one time and then a dump tape made. Then, during a relatively low period of updates, the AFTER looks and checkpoints would be sufficient to maintain data base integrity. Another advantage of checkpoints is that only AFTER looks are required, rather than both BEFORE and AFTER looks, as with recovery points. QUICK-BEFORE-LOOKS are included in the event of a system error during an update procedure, so that QUICK RECOVERY could be initiated and the data base returned to a stable state. If the data base is being used only for queries, no looks need to be taken since there will be no changes made to the data base during a query session.

Each area definition must contain an area NAME, for user reference, and an area CODE, for use only by the system. This area CODE allows the system to reference an area by number, thus each area CODE must be unique.

The number of pages allocated to an area is very important, and is determined by the user based upon record sizes and record volume. In determining record size, a word must be reserved for each pointer that will be used in set relationships involving the record type. For example, a record which is designated as an owner must contain a pointer to its first member record, as well as a pointer to its last member record if the set is doubly linked. Chain links for CALC or INDEX SEQUENTIAL location modes, pointers for sets in which the record is a member, and storage for a one word record header must be included, as well as any data contained in the record.

The following figures show how the record size for the CRIFS-REC was determined. This record type is the most involved record used in the data base. Its record size determination is included at this point merely for exemplary purposes. Its full meaning should be understood by the reader upon completion of this chapter, and it is suggested that it be reviewed at that time.

Record header (fixed)	1 word
Owner of 5 sets with next links	5
Chain links — index sequential location mode	1
User data (59 bytes)	10
	<hr/>
Total record size requirements for CRIFS-REC	17 words

Thus, each CRIFS-REC record type will require 17 words of mass storage space.

The numbers derived in each AREA definition were based on the assumption that there would be approximately 300,000 records of each type stored. When determining the amount of storage, the user should consider both the number of records involved in the initialization of the data base as well as the number of records expected to be added during the life of the data base.

The page size establishes the number of records that may be transferred in a single I/O operation. Again, the figures in the schema definition could prove to be totally undesirable, based upon how the data will be used.

In computing page size, 10 words are reserved for a page header. If 400 CRIFS-REC records are to be stored per page, this would amount to 6800 words. Then, for CRIFS-AREA, there is one additional consideration. Since this record type is stored by an index sequential location mode, one word for each record on the page must be reserved for an index slot. This brings the total number of words to 7210. Page sizes must be in multiples of 28 words due to system requirements, so the final page size for CRIFS-AREA would be 7224 words. To store 300,000 records would require 750 pages, with 400 records per page.

Specifying PRE-INITIALIZED pages will cause each page to be brought into core during the initialization of the data base. The page header will be formed and the remainder of the page cleared before returning it to storage. If this is not done during initialization, the same process will occur when a page is first referenced for record storage. Due to the size of the data base, it was considered best to have the pages initialized as they are needed.

If a record needs to be stored on a page which is already full,

an alternate location must be found. By allowing for OVERFLOW pages, the user recognizes the possibility of future additions to the data base. As the data base becomes full, the probability of finding room on a page for a new record decreases. If overflow pages are not reserved, record placement could become difficult, causing several nearby pages to be inspected before a location is found. As seen earlier, the calc-chain pointer in a page header will point to the most recently added record. If this record is on an overflow page, then subsequent additions to the original page would begin the search for storage space on the overflow page, where room would more likely be found.

Another way to allow for data base growth is to include the EXPAND-ABLE clause in the area definition. This clause states the number of pages to which the area could later be expanded without necessitating the reloading of the enlarged area.

The areas defined in the schema are all data areas (the default specification) with the exception of one index area, named INDEX-AREA. This area is associated with the CRIFS-REC record type whose location mode is INDEX SEQUENTIAL. This mode facilitates the rapid retrieval of a record from a large number of record occurrences, based upon selected data values. When retrieval of a record is required, the user supplied values for the record's selection are combined to form an index. Then a rapid search, such as a binary search, is made of the index area to locate the index. With each index is the page number in the record's data area on which the record is stored.

Before the data base is initially loaded the records must be sorted by the data values upon which their selection is dependent. Then both the record's data and index areas are sequentially loaded. After this



no further additions to the index area are possible. New records will be stored by having the index area searched as before. Since this area was stored in sequential order, the page number of where the record would be logically stored can be determined from the existing indices. If this page has room for the new record, it is inserted in its proper sequential position, with other records moved down the page if necessary. If space does not exist, the record is placed on an overflow page and linked by a sequentially ordered chain to its proper position.

For a new record to be added to an already existing group of INDEX SEQUENTIAL records, either space on each page must have been saved during the initial loading, or overflow pages must be allowed, with the record's definition specifying that LINKS are included to associate a new record with its logical position. In either case, as mentioned before, the index cannot be added to the INDEX area. For a new data base with few record occurrences and a large potential for growth, this method would not be practical. Better results would be achieved using CALC or DIRECT.

Each record type was placed in a separate area so that the area size and expansion factors could be tailored to each record type's expected growth and volume. It is entirely possible to place different record types in the same area in order to aid the efficiency of retrieval operations. Another reason for the individual areas was the type of queries that are expected, as will be seen later.

#### 4-2-2 Record Definitions

Each record definition (see B-3 through B-6) contains the same data fields that were used in the DB4 files. This section deals only with exceptions to the DB4 definitions and with the statements required

for the schema. Additionally, the DUMMY-REC record type, as well as the DUMMY-AREA and the two sets, DUMMY-ECG and DUMMY-LAB, will be explained later in Chapter 5, where the reasons for their creation will be better understood.

Every record type definition describes the characteristics of a record, providing both information required by the system for physical access, and a complete description of the data fields used to reference data.

Each record type must be named and assigned a RECORD CODE by the user. The name will be used by the user in referencing the data base, and the RECORD CODE is used by the system to numerically reference a record type.

The LOCATION MODE indicates how a record will be stored. Since CRIFS-REC uses INDEX SEQUENTIAL, the data items identifying each record occurrence are indicated, which must be data fields contained in the record's description. The records will be stored in ASCENDING order and will be indexed by the combination of a person's SSAN and NAME. These are both used because part of a sponsor's SSAN is used for a dependent's identification. The index area to be used must be specified, along with the LINKS clause if later additions using overflow pages are to be used. Without both LINKS in the record description and OVERFLOW pages in the associated data area, future records for which no space is available on their logical page could not be stored. As in all cases where a record is stored and sorted by some data value, the system must be told what to do if storage procedures are attempted for records with duplicate identifying data values. The usage of a person's SSAN and NAME will uniquely identify a CRIFS-REC record,

so DUPLICATES ARE NOT ALLOWED. The last item required for system use indicates in what area or areas the record will be stored. The association of the area name with its area code is done by the system and is of no concern to the user.

The ECG-REC record type will be stored in its own area. Each record occurrence is considered to be a member of a CRIFS-REC record occurrence associated with the same person. Thus, the SSAN and NAME fields are not included in the record description but are implied by membership in the CRIFS-ECG set. The ECG-DX-CODEs are included in the description through the use of an OCCURS clause, causing this record to be a variable length record. When the record size and subsequent page size were derived for the ECG-REC record type, an average for the number of expected ECG-DX-CODEs was used. This same averaging consideration was applied to all other record types containing the OCCURS clause.

The remaining record types — TDM-REC, LAB-REC, CS-REC, CAT-REC — all involved the same considerations mentioned previously, with respect to their ownership by CRIFS-REC, storage within their own area, and the use of the OCCURS clause. Data field representations were not changed from those used in the DB4 files except for the exclusion of identifying data values due to the use of set relationships.

#### 4-2-3 Set Definitions

A set provides a means of relating record types to each other, allowing a group of records to be selected by the location of a single record and the existence of a set relationship. When a record is stored VIA its membership in a set, its record placement will be as near to

its owner as possible. An exception to this is when the record type's definition in the schema indicates that the record is to be placed in an area different from that of the owner. If the set has several different record types, with each one in its own area, and the set members are connected by a chain, retrieval of the record occurrences in a single set occurrence could be quite time consuming, requiring several accesses to retrieve the records from the several areas.

Five sets are used in the DMS 1100 schema described in Appendix B (see B-6 through B-7), with the CRIFS-REC record owning each of the other five record types in the schema through five individual set types. In each set CRIFS-REC is the owner and a single record type the only member.

The main purpose of the data base is to store medical information. Thus, it is important for each record stored to be accurately associated with the person to whom it pertains. During update procedures, the proper CRIFS-REC record must be chosen, with its selection made by an individual's SSAN and NAME. If a more positive identification is desired, any other information known about the person whose new record is to be stored can be compared to the data in CRIFS-REC, such as date of birth, SEX or RACE. The new record would be stored only after the proper criteria, as required by the particular programming procedure being used, have been met. The strictness of these criteria is dependent on the installation, and it is the responsibility of the update procedure to positively identify a person's CRIFS-REC before storing a new record.

An example of this can be seen in the CS-REC and the CS report form from which the record's data is obtained. This form contains SSAN and NAME fields, which will be used to find the CRIFS-REC to which

this new CS-REC will be related through the CRIFS-CS set. The form also contains DOB, SEX and RACE fields, which, due to their existence in the CRIFS-REC, will not be redundantly stored in the CS-REC. Even though these three fields are not used to select a CRIFS-REC, once an occurrence of CRIFS-REC is chosen using the SSAN and NAME, the values of these three fields can be procedurally compared to the same fields in CRIFS-REC, with any discrepancies causing the new CS-REC to be rejected until corrected.

As with both areas and records, each set type must have a NAME for user reference and a SET CODE for use only by the system. The MODE by which the set is implemented may be either POINTER or CHAIN. Record retrieval using POINTER would require a POINTER array to be referenced each time a member record occurrence is to be located, whereas with the CHAIN mode, each member contains a pointer to the next occurrence. For simplicity the CHAIN mode is used, with a record's placement in the chain dependent on the data value of a field which uniquely identifies it with respect to the other record occurrences in the set, such as the date of the examination (DOE).

Each time a new record is added to the data base, it will be immediately linked into the chain of the set occurrence to which it belongs, since AUTOMATIC is indicated in all sets in the schema. Due to the implementation of record retrieval for queries, each member of a set will also be LINKED TO OWNER. The reason for this will now be seen in the next section.

#### 4-2-4 QLP Information

The QLP 1100 allows a user to be given access to only part of an existing data base. For example, if a number of LAB-REC records are to be added, the subset of the data base which would be needed would consist of the CRIFS-AREA, INDEX-AREA and LAB-AREA areas, the CRIFS-REC and LAB-REC record types, and the CRIFS-LAB set type. This configuration of a data base subset is called a subschema. Several subschemas may exist at the same time, whereas only a single schema may exist at any point in time. The subschema used in this report (see B-9) is actually the entire absolute schema, as can be seen by the COPY ... ALL statements. If a proper subset is to be used, one must be careful to include all required parts of the absolute schema. For instance, in the above example, INDEX-AREA had to be included in order to access the CRIFS-REC records.

The subschema also specifies several paths which may be traversed. These paths will be used in the queries demonstrated in Chapter 5. The CAT-LAB path indicates that a search will begin with the CAT-REC record type. A particular value of a data field will be the object of the initial part of the query. As seen in a previous section, each record occurrence of the CAT-REC type will have to be inspected, with the FETCH NEXT CURRENCY ALLOWED statement permitting a FETCH FIRST/NEXT CAT-REC OF AREA command to be initiated by the DMR. Through the OWNER links, a CAT-REC record containing the desired queried data values can be traced to its owner, and from there, through CRIFS-LAB, it can be associated with the LAB-REC records belonging to the same person.

The many details involved in the definition of a DMS 1100 data base and the impact of these details on its subsequent use, tend to make this a more difficult system to understand as compared to the DB4 relational data base. In the next chapter the implications of this will be seen.

## CHAPTER 5

### COMPARISONS



All data base management systems strive to achieve a set of common goals — to assist a user with information storage and retrieval needs, and to allow a user to manipulate the data in order to gain knowledge from it that would otherwise be difficult to realize without an automated means such as a DBMS. In the case of this report, this "buried knowledge" pertains to a statistical analysis of the data for research purposes, and is one of the main considerations in the following comparisons.

Relational and network data bases, though they share the same purpose, are completely different in implementation and use. Thus, a complete discussion of them would be too lengthy for this report, which considered only those factors which were of the greatest concern for this specific application. A more general comparison may be found in current publications, such as [7].

As stated in Chapter 1, simplicity of the data base is essential. Factors affecting this will be brought out throughout the rest of this chapter. Also, the reader will notice the absence of detailed explanations concerning some of the points made in the following sections. This is not meant to make the discussions ambiguous, but rather to assist the reader by making it known that the data base possesses a particular capability, for which a detailed explanation may be found in the reference cited. Inclusion within the text of these explanations would not aid the understanding of the comparisons, but instead would divert the attention from the point which is being made.

#### 5-1 Basis of General Queries

In the DB4 relational data base, all queries are based upon keyed signatures. As previously stated in Chapter 3, the keyed signatures

are stored in the structure space, separated from the data to which they are a part of. When isolating a data set through the use of queries, only the structure space, consisting of pointers, is affected, with access to the data space made when output is requested.

The term "data base key" in DMS 1100 has a different connotation. In this case the key is a 36 bit word with three parts — an area code, a page number and a relative record number. Therefore, it has a more physical meaning. Only in the case of a DIRECT location mode can a record contain a field labelled as a data base key. Instead, specified fields within the record description will be defined to be used as the basis from which a record location may be calculated. In the case of a general query, such as isolating all records with a particular value for a named data field, each record must be inspected which, if a large number of record occurrences are present, can be a very time consuming procedure.

## 5-2 Queries

In this section examples of current queries used in the MARK IV system are given, followed by the DB4 and DMS 1100 queries needed to isolate the same data set. It should be noted that both systems support a REPORT feature for output formatting [3,5].

It has been found that 90% of the current queries deal with a single repository. An example of this concerns the CS repository, where a person's complete CS record is to be printed if keywords, such as HEADACHE or MIGRAINE, are contained in the diagnostic text. In DB4 one could use:

```
FIND DATA WHERE KEYWORD EQ 'HEADACHE' OR  
KEYWORD EQ 'MIGRAINE'  
PROJECT DATA FROM CS-DIAG THRU SSAN,NAME INTO CS  
REPORT DATA REPORT-AAA
```

This would cause the desired data set to be isolated in the collection name DATA and then printed. The PROJECT takes all cases where the keyword exists and locates the rest of the CS report belonging to that person.

With DMS 1100 the query could take the form:

```
LIST USING FORMAT FORMAT-ONE WHERE KEYWORD EQ 'HEADACHE' ;  
OR KEYWORD EQ 'MIGRAINE' VIA CS-CRIFS PATH
```

Using the data base description in Appendix B, the above query would isolate the same data set as the previous query, with FORMAT-ONE being a previously defined formatting specification [5].

If desired, any data set isolated by a query may be sorted by any field or fields before printing. Both data base management systems provide this feature.

A more interesting query arises when a single query involves two repositories. This is a case where the user is given power, through the DBMS, which is not available in conventional sequential file management systems. The following query is in two parts. First, sequence by NAME and print the full cover sheet for those people having a DX-CODE of 994 after 1 January 1971. Then, using the SSAN, find each person's WAIVER record, sequence again by NAME, and print the WAIVER file. The WAIVER file is not currently incorporated into the cross-referenced repository system, but for the purposes of this query, one will assume that it is. In DB4 it will be considered to be a data group with both

SSAN and NAME as keyed fields. The DMS 1100 representation will be the same as the other five repository record types, with WAIVER-REC owned by CRIFS-REC through the CRIFS-WAIVER set. When the WAIVER repository is brought into the rest of the repository system, its representation would be like those stated above, to conform with the data base definitions.

The DB4 query could be as follows.

```
FIND DATA WHERE DX-CODE EQ 994 AND DOE GT 710101
PROJECT DATA FROM CS-DIAG THRU SSAN, NAME INTO CS
ORDER DATA O NAME
REPORT DATA REPORT-BBB
PROJECT DATA FROM CS THRU SSAN INTO WAIVER
ORDER DATA O NAME
REPORT DATA REPORT-CCC
```

The DOE in the FIND statement takes the form YMMDD, making comparisons using the data field quite easy. The ORDER statement is to order all data records from all areas that are isolated in DATA, as can be seen from the use of the universal area name (O). After the first printing, the DATA collection name is still intact, and can undergo further processing. It is projected into the WAIVER file through the SSAN data field. The second ordering is not really necessary since the records in DATA are still ordered. Instead, one could have gone directly from the second PROJECT to the second REPORT with the same results.

To retrieve the above data set in DMS 1100 will require two separate queries which, due to the implementation of DMS 1100, must be totally independent of each other. The impact of this is that a previously isolated data set may not be used in a subsequent query unless it is

regenerated and used as an intermediate result.

The basic form of a call to a previously defined REPORT definition in DMS 1100 is

```
GENERATE REPORT-NAME WHERE ...
```

in which REPORT-NAME is the report's name and the WHERE clause is the same as for any other query, except that a VIA clause may not be specified. The generated report will be returned to the user unless statements are inserted before the WHERE clause specifying some other output device. Thus, the DMS 1100 query could read

```
GENERATE REPORT-DDD WHERE DX-CODE EQ '994' ;  
AND DOE GT '710101'
```

Neither of the two data fields provide an immediate means of entering the data base, so a path with FETCH NEXT CURRENCY ALLOWED will be sought which will contain both the above two fields as well as those fields contained in the REPORT-DDD report definition. From the SUBSCHEMA definition in Appendix B, the CS-CRIFS path will be chosen and the query satisfied.

The second part of the query, retrieving the WAIVER file, would require the following PATH definition:

```
PATH NAME IS CS-WAIVER  
ROOT IS CS-REC  
FETCH NEXT CURRENCY ALLOWED  
THRU CRIFS-CS TO CRIFS-REC  
THRU CRIFS-WAIVER TO WAIVER-REC
```

With REPORT-EEE containing data fields from the WAIVER file that are not present in the other repositories, the query

GENERATE REPORT-EEE WHERE DX-CODE EQ '994' ;  
AND DOE GT '710101'

will cause the CS-WAIVER path to be traversed and the query executed. As noted before, the use of the two queries in DMS 1100 to generate the desired CS and WAIVER reports necessitates the data records to be isolated twice, since a previous query's results may not be processed in another statement. The ordering of the above reports by NAME may be specified in the REPORT definition [3,5].

The previous two queries discussed, involving first one and then two repositories, were fairly easy to isolate in both the DB4 and DMS 1100 data bases. The next query is the most involved to be considered. It begins in the CAT repository where patients with a DX-CODE of 06 or 09 and an ECG-REASON of 05 (contained in section B-9 of the report form) are to be isolated. After printing a report containing information from the CAT repository, the ECG repository is to be checked for a DX-CODE of 720. If this is the case, another report of the isolated records is made, again containing information from the CAT repository. The final request in the query involves the generation of seven lists of the isolated record occurrences sequenced in different orders. The main problem with this query is that for the individuals isolated a LAB record for them is to be retrieved and one of the seven lists is to be sequenced by the sum of several values within the LAB record. Both data bases will compute and print subtotals in a report [3,5], but neither can sequence the records by the subtotalled values. If this is desired it must be done through a programmed procedure and not through the DBMS query language.

In the following DB4 query two collections are used, DATA1 and

DATA2, in order to retrieve the data sets.

```
FIND DATA1 WHERE (DX-CODE EQ 01 OR DX-CODE EQ 09)
                AND B-9 MV '00000000000010000'
REPORT DATA1 REPORT-FFF
FIND DATA2 WHERE DX-CODE EQ 720
PROJECT DATA2 FROM ECG-DX-CODES THRU SSAN, NAME INTO ECG-EX
FIND DATA1 AND DATA2
REPORT DATA1 REPORT-GGG
PROJECT DATA1 FROM CAT THRU SSAN, NAME INTO LAB
{
    reports generated
}
```

As before, the sequencing may be specified within the report.

Since section B-9 is stored in bit form, the mask value used must correspond to the way in which the data values are represented. Rather than the user projecting a collection from one area to another, the DB4 system can provide this function, based upon shared keyed signatures. The rest of the statements function as previously described, with DATA1 being reused each time a report is generated.

In DMS 1100 a new query must be generated for each desired report. For the first part of the query, the following could be used:

```
GENERATE REPORT-HHH WHERE (DX-CODE EQ '01' OR DX-CODE EQ '09') ;
                        AND B-9(5) EQ '1'
```

From the paths in the schema in Appendix B one can see that the CAT-ECG path would be selected since data fields in both the CAT-REC and ECG-REC record types are required.

Subsequent reports requiring the above isolated data set may be generated by resubmitting the command with different report names. As mentioned previously, reports sequenced by subtotals from a LAB-REC record

must be handled by a programmed procedure, but it can be shown how one could traverse the DMS 1100 data base to isolate both the above data set as well as the LAB-REC records which would be included.

From the diagram in Appendix B, one can see how the above data were retrieved. The search originated in CAT-REC, and through the owner pointer of the CRIFS-CAT set, moved to CRIFS-REC. Then using the CRIFS-ECG set, the ECG-REC records were reached. If from this point in the query the LAB-REC record type also needs to be accessed, the owner pointer of the CRIFS-ECG set cannot be followed back to CRIFS-REC in order to use the CRIFS-LAB set, since each record or set type may be traversed only once in a single path. Thus a somewhat artificial means was devised to go from the ECG-REC record type to the LAB-REC record type in this kind of situation. A DUMMY-REC record type was created which owns, through two different sets, the ECG-REC and LAB-REC record types. From its record definition in the schema one can see that its only data fields are SSAN and NAME. When a CRIFS-REC record is created for a person, a DUMMY-REC record must also be created. Then, whenever a new ECG-REC or LAB-REC record is introduced to the data base, it will be automatically added to the DUMMY-ECG or DUMMY-LAB set, respectively. Through this method, more than three records connected through sets involving the CRIFS-REC record type may be traversed in a single path, but, obviously, it complicates the representation of the data base and increases the amount of storage which must be used to store the additional record as well as to record the set relationships. Also, unexpected queries in the future which are as involved as this one may not be answerable with the DMS 1100 system, unless the schema is changed and the whole data base reprocessed to reflect the changes necessary to



retrieve the information. This is a case where the effort and cost required to satisfy a query may not be justified by the information finally retrieved.

In contrast to the above considerations with DMS 1100, the DB4 system bases its queries only upon keyed signatures, with no restrictions on the order in which data groups must be accessed. Also, the results of a previous query may be used, whereas in DMS 1100 no intermediate data sets may be saved to undergo further processing in a separate query. This can be a very costly factor, causing a great deal of computer time to be spent retrieving the same data set several times when, for example, different report forms of the same data are desired.

The same hierarchical structure possible in the DMS 1100 system may be achieved in DB4 through the use of a single command. An example of the power of this would be better illustrated using an example unrelated to the repository data. Instead, assume the existence of a data base containing CUSTOMER and PURCHASE data groups. CUSTOMER is identified by an ACCOUNT number, and each PURCHASE is identified by the ACCOUNT number of the person to which it applies as well as an ITEM number, indicating what was purchased. Using the collection name DATA, further assume that it already contains the CUSTOMER and PURCHASE records for one day of activity in a store. Now a report is to be printed showing the day's activity. The following commands

```
      SETHI DATA CUSTOMER PURCHASE  
      REPORT DATA REPORT-III
```

allow this to be done. The SETHI statement will group CUSTOMER and PURCHASE records together that both have the same value in a shared

signature, in this case ACCOUNT. After this command, DATA logically contains CUSTOMER-PURCHASE groups of records, each group pertaining to a single customer. In DMS 1100 this same hierarchy would be represented by the CUSTOMER record type owning the PURCHASE record type.

### 5-3 Storage Requirements

The figures in this section were derived from the data base descriptions in Appendices A and B. Appendix C displays these figures for each data base, assuming an average of 400,000 record occurrences for each record type. In cases where a DMS 1100 OCCURS clause was used to store data which was represented in a data group in DB4, storage for an equal amount of the data in both data bases was computed. The figures are not truly representative of the actual number of records to be stored in the data base since they assume a fully loaded data base, but they do accurately state the record sizes and can act as a basis for comparing the two data base representations, taking into consideration such system requirements as pointers.

The DB4 data space is used to store the actual data of the data base. One will note that two of the data groups have a record size of 0 (see C-2). This is because every field of these groups is keyed, and keyed signatures are stored in the structure space.

The DBD processor will reserve space in the structure space for every possible value of a keyed signature. This can be overridden by the UNIQUE OCCURRENCES clause if it is known approximately how many unique values are to be expected for a signature. For example, every data group contains an SSAN field and, without the UNIQUE OCCURRENCES

clause, storage would have been reserved for each SSAN field in each data group, when obviously no more than 400,000 values, one for each person, can be expected.

Associated with each signature in the structure space will be pointers indicating in which records that signature value occurs. Without an actual implementation, the space for these pointers, as well as the space required for the control space, cannot be determined, but the following point can be made. If all the signatures in the data base were keyed, then the amount of space required to store the keyed signatures and the overhead structure would be considerably greater than simply storing the data. This data explosion would be on the order of a factor of three [3]. The figures show the data space to be  $7.47 * 10^8$  bytes in size, while in comparison, the signature storage requires  $1.76 * 10^7$  bytes. Clearly, no data explosion will take place, and the space required for the structure space can be expected to be small in proportion to the data space.

The record and page sizes for the DMS 1100 definition are displayed on pages C-4 through C-6. The figures are in six byte words and, for comparison's sake, the totals (see C-7) are in bytes. The following figures are the result of Appendix C.

DB4	size requirements	$7.65 * 10^8$	bytes
DMS 1100	size requirements	$9.49 * 10^8$	bytes

As stated above, the DB4 size requirement includes neither the structure space pointers nor the control space. Allowing for these, the method of DB4 implementation will require about the same amount of storage as in DMS 1100. Though DB4 will require less storage for the data space, the advantage is offset by the structure and control spaces. The DMS 1100

computations are based on the size of the data base after it is fully expanded. Its initial size is about 25% smaller.

#### 5-4 System Overhead

Due to the different implementation of each DBMS, the system costs incurred during processing will vary from system to system. This section reiterates some previously mentioned factors affecting each system.

The DB4 query deals entirely with the structure space until the isolated record occurrences are to be output. This allows the DB4 system to work with a fairly concise representation of the entire data base. Also, previously isolated data sets may be used in later queries, alleviating the need to repeat an identical command.

New data must first be loaded into the data base, and then constructed to generate the structure space. The amount of time required for this generation is dependent on the number of new records, the number of keyed signatures, and the distribution of the signatures throughout the data base. If a time limit is exceeded during the execution of a CONSTRUCT command, an orderly shutdown takes place. Completion of the structuring merely requires that the command be resubmitted [3].

In using a DB4 data base the large core option may be specified. This will cause the DBM to extend its buffer pool significantly, thus improving the efficiency and turnaround time as well as reducing processing costs [3].

Record retrieval in DMS 1100 depends upon locating a record and then, if desired, following pointers to find member record occurrences. For standard record processing this is quite satisfactory, but it causes

queries to have the potential to be very time consuming. As seen before, the general queries that are expected to be asked of the data base will require each record of an area to be inspected. Upon finding a record from which the rest of the data base may be traversed, a path may be followed through several set and record types unless a data value in one of the records causes the path occurrence to be disqualified from inclusion in the data set being sought. Thus, a great deal of false path logic is very likely to be incurred during the execution of a query command.

The OCCURS clause results in a storage savings, with some overhead involved in compacting the record for storage and expanding it during retrieval. Also, as the data base begins to fill, the likelihood of having to find alternate record locations on overflow pages increases, causing the system to spend more time to locate records.

The DMS 1100 system is responsible for maintaining the integrity of the data base through the writing of LOOKS to an audit tape. Also, QUICK-BEFORE-LOOKS must be saved during a session using the data base.

The effect of these disadvantages can be controlled by carefully designing the schema, especially in the case of the addition of new record occurrences. If those designing the schema have a detailed knowledge of how the data will be used and the expected growth rate of the data base, then the schema can be tailored to their needs, resulting in a higher degree of efficiency.

#### 5-5 Data Base Integrity

Data base integrity may be defined to be the protection of the data and its relationships from system failures and could include the requirement

that data values be verified, assuring that they are within a reasonable range [2]. An installation's data may be its most valuable and important asset, thus it is paramount that it be accurate and protected.

The current procedures used with the MARK IV system included in-depth verification processes. These processes should be retained and modified for use with any new DBMS, especially since neither DB4 nor DMS 1100 provide a means of doing this automatically.

Updates to the data base are expected to be in a batch mode from data on tapes. Dumps of the data base to tape are necessary in case the data become severely damaged or destroyed. With DB4, a data base may be restored by keeping dump tapes and also the update tapes which were incorporated into the data base after the most recent dump. In the event that the data base would need to be reloaded, the dump tapes would restore the data base to a stable condition. Then any subsequent update tapes would be reprocessed to bring the data base up-to-date. The frequency of making dump tapes depends on the desires of the user, but it can be seen that dumps should be made after large updates to avoid having to reprocess large amounts of data in the event of a subsequent data base failure.

Errors in the structure space usually are localized to the structure space for a single keyed signature [3]. The query facility may be used to locate the malfunctioning signature. Then a DESTRUCT command issued for the bad signature will destroy its structure but preserve the data. This would be followed by a CONSTRUCT to regenerate the structure space for the signature.

If errors are found throughout the structure space, an S option DBD run may be executed which will destroy the structure space for the

entire data base. This, followed by a `CONSTRUCT` command, would reset the entire structure space. If this still does not solve the problem, then the data base must be restored from the most recent dump tapes.

An error in the data space could be due either to irregularities in the data records themselves or damaged areas in the virtual directory that controls access to the data space. In the first case, the errors could be corrected by locating the bad records and updating them. If the damage is more widespread but is known to occur above a particular relative record number within an area, the `TRUNCATE` command would remove all records from the named area above and including the relative record number [3]. The removed records would then have to be restored from their original tapes and constructed.

If the virtual directory is damaged, or if errors are widespread throughout a data area, the `VBUILD` command will regenerate the virtual directory for the named area. If the problem still is not solved, the bad area can be completely wiped out (through the `RAZE` command), reloaded from a data tape and then constructed. At this point, if the problem is still present, the whole data base must be restored from dump tapes.

Errors in the control space could be due either to a damaged virtual directory (discussed above) or to a problem in the data base definition. For the latter case, an `S` option `DBD` run should be done, followed by the `VCORE` command, which will repair portions of the `DBD` associated with virtual processing that are not repairable with the `S` option `DBD` run. Then the `CONSTRUCT` command should be run. If the above sequence does not correct the problem, then dump tapes must be restored.

DMS 1100 recovery procedures involving the use of `LOOKS` were extensively discussed in Chapter 4. These procedures have an advantage

over DB4 in that a complete reload of the data base is not as likely to be necessary. Also, calc-chains and sets may be verified by a system-provided procedure [7]. If errors are found, such as bad pointers or loops, an error message is printed describing the error and its location. The user must then determine the necessary correction and use the PATCH utility to make the correction.

## 5-6 Flexibility

A data base which is rigid and unchangeable would not prove to be very useful, since, almost without exception, the information and data needs of an installation will change over a period of time.

The most prominent concern in this area could be allowing for the growth of the data base. In DB4 the DBD must indicate the maximum size which is expected. If this is grossly underestimated or overestimated, each area needs to be dumped to tape, which may be done through a utility function. A new DBD, indicating the changes in size, must be processed, then the data reloaded and constructed. Data independence will be preserved, but all the UIDs must be reprocessed to reflect the changes.

With DMS 1100 the EXPANDABLE clause allows the data base size to be increased after it is initially created. The source schema will need to be changed and processed, as well as any existing subschemas. Careful consideration of the original schema must be made if it is to be expanded, to make sure the new object schema will be compatible [4]. If the data base growth necessitates an increase beyond that which is stated in the EXPANDABLE clauses, then major reprocessing must occur, causing all the keys to be changed.



Statistics concerning storage utilization within the data base are available in both DB4 and DMS 1100. In DB4, the DECAL command will provide a numerical analysis of each area or selective areas. From these data one can determine how full the data base is becoming, as well as the number of unique occurrences for each keyed signature. This latter information may indicate that an S option DBD run should be executed to change the UNIQUE OCCURRENCES clauses. A considerable storage savings could be realized if the original UNIQUE OCCURRENCES numbers were not well estimated. This same kind of run may be used to change data groupings within a record description, to change a keyed signature to an unkeyed signature (or vice versa), or to change the choice of signature names [3].

A more extensive set of statistics on storage utilization is available in DMS 1100, which includes the average number of used and unused words per page within each area. A more detailed description, listing these figures for each page, is also available upon user request. This kind of information would be very helpful to determine if and when the data base should be expanded. It is produced after page compaction is requested, during which deleted records will be removed from the data base and the remaining records on each page compacted, leaving all available space in a page in a single block. Page compaction may be specified for a single area, several areas, or the entire data base. It can be a very time consuming operation and should be done only when required.

A DBMS allows several users to view the same data in different logical organizations. This is due to the fact that a DBMS is able to take a logical description, known to the user, and map it into a physical description, known to the DBMS. From this physical description a record's

physical location may be determined, based upon user supplied data values.

This separation of logical and physical views through a DBMS is what gives a data base the capacity to be modified and changed as the needs of those using the data base change. When the need arises for a data base to be changed in some way, it will be either the DBD (DB4) or the schema definition (DMS 1100) which must be inspected and modified.

In DMS 1100 a large number of sets may exist, causing the relationships between record types to potentially be quite complex. Any changes to the data base must be carefully considered, both to avoid any disruption in the existing data base, as well as to assure that any new additions will be properly incorporated to allow paths to exist for query purposes.

The normalized form of data representation in DB4 provides protection from future changes in the data base [2]. No physical relationships between the data groups are assumed to exist, with logical relationships established dynamically upon the execution of a query. This allows the normalized data base to have a conceptually cleaner logical representation than that which is possible in a network data base, giving it an aesthetic simplicity. The lack of user control over record placement does not degrade the system's performance in query procedures due to the fact that only a subset of the data base, namely the structure space, is involved in the selection of a data set. Records are not physically accessed until required for output. Thus, the user wishing to initiate changes to the data base needs to focus mainly on only the new data and its normalized representation.

The conclusions drawn by the author may be found in Chapter 6.

## CHAPTER 6

### CONCLUSIONS

The choice of a data base is dependent upon a user's needs for information management. The factors considered in this report are by no means exhaustive, but it treated what were considered to be the most important aspects affecting the representation and query use of the USAFSAM medical repository data files.

The two data base management systems investigated in this report were found to be comparable in most respects. As seen in Chapter 5 and Appendix C, their size requirements were similar. The fact that the information required for data access and manipulation is stored separately from the data in DB4 was seen to be an advantage. Rather than the necessary pointers being embedded within the records themselves, they are contained in the structure space, causing queries encompassing a wide range of the data base to have accesses to the mass storage devices restricted to only those areas where the structure space resides. Retrieval of the actual data is not necessary until the data are desired for output. With DMS 1100, queries involving a large portion of the data base result in a sizeable number of accesses to occur in order to trace the logical relationships between records.

The integrity facilities available in DB4 were seen to be more appropriate for this application. Since updates are to be in a batch mode from data recorded on tapes, a dump tape of an updated area could be made immediately after an update session in DB4. As seen in Chapter 5, errors in the virtual directories or the structure space can be corrected by the DBMS through the execution of specific commands. Only errors occurring in the data space must be corrected by the user. If updates to the data base were to be made by individual users, the LOOKS in DMS 1100 would ensure data base integrity, making it possible to reverse

the actions of a single user, which is not automatically possible in DB4. Another major drawback to recovery in DB4 is the fact that a total reloading and reconstructing of the data base is required for quite a number of errors, which in a data base for these repositories would be very time consuming, considering its size.

The major difference found between the relational and network data bases was the method of querying. Were queries to be based on retrieving information about a particular individual, both data base types would be appropriate, but instead, they involve the inspection of data fields to try to determine, for research purposes, if any correlations exist between the fields. While DB4 can handle these types of queries quite easily, DMS 1100 must inspect each record individually during the search, which would be a very lengthy operation in most cases, involving a great deal of false path logic.

Therefore, in the author's opinion, the utilization of the contents of a medical data base for researchers would best be accomplished through the implementation of a relational data base system. With properly maintained integrity procedures and rigorous verification procedures to ensure the correctness of the data upon entry to the data base, the data base would be found to be a very powerful tool, possessing both the capacity to meet the current information management needs as well as the ability to adapt to future modifications and enhancements to the system.

## APPENDIX A

### DB4 DATA BASE DEFINITION

DATA BASE NAME IS REPOSITORIES

UNIQUE OCCURRENCES ARE   SSAN, 400000  
                               NAME, 400000  
                               CASE-NO, 400000  
                               B-9, 262144  
                               DOB, 36525  
                               DOE, 36525  
                               DX-CODE, 1000  
                               C-11, 512  
                               TEST, 237  
                               KEYWORD, 200  
                               CTH-SEQN, 10  
                               RACE, 6  
                               RUN-NO, 4  
                               SEX, 2

[keyed signatures having a  
 restricted number of values]

DG CRIFS, 400000

02 SSAN           X(9)   KEY  
 02 NAME           X(27) KEY  
 02 DOB            X(6)   KEY  
 02 SEX            X(1)   KEY  
 02 RACE           X(1)   KEY  
 02 FLY-PHYS       X(1)  
 02 CLASS           X(1)  
 02 DTH-CODE       X(1)  
 02 SODC           X(1)  
 02 DOLE           X(4)  
 02 SOLE           X(1)  
 02 FLAGS  
     03 CS         X(1)  
     03 ECG        X(1)  
     03 TDM        X(1)  
     03 LAB        X(1)  
     03 CAT        X(1)  
     03 VCG        X(1)

[examination classification]  
 [service classification]  
 [death codes]  
 [source of death code]  
 [date of last exam]  
 [source of last entry]

DG ECG-EX, 399000

[ECG examinations]

02 SSAN X(9) KEY

02 NAME X(18) KEY

02 DOE X(6) KEY

02 FILE-NO X(6) [ECG file number]

02 AGE 9(2) [age at time of examination]

02 HT 9(2) [height at time of examination]

02 WT 9(3) [weight at time of examination]

02 BP X(6) [blood pressure]

DG ECG-DX-CODES, 2000000

[diagnostic codes resulting from  
ECG examination]

02 SSAN X(9) KEY

02 NAME X(18) KEY

02 DOE X(6) KEY

02 DX-CODE X(3) KEY [identifying diagnostic code]

DG TDM, 400000

02 SSAN X(9) KEY

02 NAME X(18) KEY

02 CASE-NO 9(5) KEY [case number]

02 DOE X(6) KEY

02 GRADE X(2)

02 RUN-NO 9(1) [test run number]

02 PR-TDM 9(1) [indicates a prior TDM test]

02 AGE 9(2)

02 ACT-STUS 9(1) [indicates activity status]

⋮ [environmental conditions at  
time of examination]\*

\* In cases where a large number of data fields within the repositories are not essential to the understanding of the data base definition, the field descriptions have been omitted, with a bracketted note describing their contents.



TDM DATA GROUP — CONT.

02	WALKING-RESULTS		[3.3 MPH]
03	A		[first minute]
	04	BP X(6)	[blood pressure]
	04	HR 9(3)	[heart rate]
03	B		[second minute]
	04	BP X(6)	
	04	HR 9(3)	
	:		[results of minutes 3 - 24]
	:		[recovery data]
	:		[findings]
DG	LAB, 400000		
02	SSAN	X(9) KEY	
02	NAME	X(18) KEY	
02	CASE-NO	9(5) KEY	
02	DOE	X(6) KEY	
02	HEMOGRAM		[remaining contents of record
	03	AA 9(2)	consists of test results for
	03	AB 9(2)V(1)	237 tests]
	:		
02	GLUC-TOL-URINE		[glucose tolerance - urine results]
03	K		
	04	A 9(4)	
	04	B 9(4)	
	:		
03	R		
	04	A 9(3)	
	04	B 9(3)	

DG LAB-MESSAGES, 2000000

[messages associated with tests]

02 SSAN X(9) KEY  
02 NAME X(18) KEY  
02 CASE-NO 9(5) KEY  
02 DOE X(6) KEY  
02 TEST X(2) KEY  
02 MESSAGE X(20)

DG CS, 396000

02 SSAN X(9) KEY  
02 NAME X(18) KEY  
02 DOB X(6) KEY  
02 SEX X(1) KEY  
02 RACE X(1) KEY  
02 CASE-NO 9(5) KEY  
02 DOE X(6) KEY  
:

[other data from CS report form]

DG CS-PHYSICIANS, 800000

[physicians involved with CS report]

02 SSAN X(9) KEY  
02 NAME X(18) KEY  
02 DOB X(6) KEY  
02 SEX X(1) KEY  
02 RACE X(1) KEY  
02 CASE-NO 9(5) KEY  
02 DOE X(6) KEY  
02 PHYSICIAN  
03 PHYS-CODE X(3)  
03 DEPT-CODE X(3)

[physician's identification code]

[department code of physician]

DG CS-DIAG, 2000000

[diagnostic codes from cover sheet]

02 SSAN X(9) KEY  
02 NAME X(18) KEY  
02 DOB X(6) KEY  
02 SEX X(1) KEY  
02 RACE X(1) KEY  
02 CASE-NO 9(5) KEY  
02 DOE X(6) KEY  
02 DX-CODE X(3) KEY  
02 DX-TEXT  
03 KEYWORD X(10) KEY  
03 TEXT X(20)

[identifying diagnostic keyword  
for queries]

DG CAT, 402000

02 SSAN X(9) KEY  
02 NAME X(18) KEY  
02 CASE-NO 9(5) KEY  
02 DOE X(6) KEY  
02 GRADE X(2)  
02 HT 9(2)  
02 WT 9(3)  
02 CTH-SEQN 9(5) KEY

[catheterization sequence number]

:

[data pertaining to catheterization]

:

[data pertaining to coronary risk  
profile of patient]

DG	CAT-DET,	800000		[determinations from catheterization]
02	SSAN	X(9)	KEY	
02	NAME	X(18)	KEY	
02	CASE-NO	9(5)	KEY	
02	DOE	X(6)	KEY	
02	CTH-SEQN	9(5)	KEY	
02	SEC-B			[results from section B of report form]
	03 B-7	I(1)		
	03 B-8	I(2)		
	03 B-9	I(3)	KEY	[*]
02	SEC-C			[results from section C of report form]
	03 C-10	I(2)		
	03 C-11	I(2)	KEY	
	:			[remaining test results]

DG	CAT-DIAG,	2000000		[diagnostic codes from catheterization]
02	SSAN	X(9)	KEY	
02	NAME	X(18)	KEY	
02	CASE-NO	9(5)	KEY	
02	DOE	X(6)	KEY	
02	CTH-SEQN	9(5)	KEY	
02	DX-CODE	X(2)	KEY	

\* Data in these fields are stored in bit form for query purposes.

## APPENDIX B

### DMS 1100 DATA BASE DEFINITION

IDENTIFICATION DIVISION  
SCHEMA NAME IS REPOSITORIES  
DATA DIVISION  
AREA SECTION  
AREA CONTROL IS 127 AREAS  
AREA LOOKS INCLUDE QUICK-BEFORE-LOOKS, AFTER-LOOKS  
AREA NAME IS CRIFS-AREA AREA CODE IS 1  
    ALLOCATE 750 PAGES  
    1 OVERFLOW PAGE EVERY 24 DATA PAGES  
    EXPANDABLE TO 1000 PAGES  
    PAGES ARE 7224 WORDS  
AREA NAME IS INDEX-AREA AREA CODE IS 2  
    MODE IS INDEX AREA  
    ALLOCATE 300 PAGES  
    EXPANDABLE TO 400 PAGES  
    PAGES ARE 6524 WORDS  
AREA NAME IS ECG-AREA AREA CODE IS 3  
    ALLOCATE 430 PAGES  
    1 OVERFLOW PAGE EVERY 24 DATA PAGES  
    EXPANDABLE TO 570 PAGES  
    PAGES ARE 8428 WORDS  
AREA NAME IS TDM-AREA AREA CODE IS 4  
    ALLOCATE 3000 PAGES  
    1 OVERFLOW PAGE EVERY 24 DATA PAGES  
    EXPANDABLE TO 4000 PAGES  
    PAGES ARE 7728 WORDS  
AREA NAME IS LAB-AREA AREA CODE IS 5  
    ALLOCATE 6000 PAGES  
    1 OVERFLOW PAGE EVERY 24 DATA PAGES  
    EXPANDABLE TO 8000 PAGES  
    PAGES ARE 6860 WORDS  
AREA NAME IS CS-AREA AREA CODE IS 6  
    ALLOCATE 2500 PAGES  
    3 OVERFLOW PAGES EVERY 40 DATA PAGES  
    EXPANDABLE TO 3300 PAGES  
    PAGES ARE 6972 WORDS

AREA NAME IS CAT-AREA AREA CODE IS 7

ALLOCATE 2500 PAGES

1 OVERFLOW PAGE EVERY 24 DATA PAGES

EXPANDABLE TO 3350 PAGES

PAGES ARE 9156 WORDS

AREA NAME IS DUMMY-AREA AREA CODE IS 8

ALLOCATE 430 PAGES

1 OVERFLOW PAGE EVERY 24 DATA PAGES

EXPANDABLE TO 570 PAGES

PAGES ARE 6328 WORDS

RECORD SECTION

RECORD NAME IS CRIFS-REC RECORD CODE IS 1

LOCATION MODE IS INDEX SEQUENTIAL

USING ASCENDING KEY IS SSAN, NAME

INDEX AREA IS INDEX-AREA

LINKS ARE NEXT

DUPLICATES ARE NOT ALLOWED

WITHIN CRIFS-AREA

02 SSAN PIC X(9)

02 NAME PIC X(27)

02 DOB PIC X(6)

02 SEX PIC X(1)

02 RACE PIC X(1)

:

[other data fields identical  
to those used in DB4]

RECORD NAME IS ECG-REC RECORD CODE IS 2

LOCATION MODE IS VIA CRIFS-ECG SET

WITHIN ECG-AREA

02 DOE PIC 9(6)

02 FILE-NO PIC X(6)

02 AGE PIC 99

02 HT PIC 99

02 WT PIC 99

02 BP PIC 9(6)

02 DX-CNT PIC 9(2) USAGE IS COMP

02 DX-CODE OCCURS 0 TO 10 TIMES DEPENDING ON DX-CNT

03 ECG-DX-CODE PIC X(3)

RECORD NAME IS TDM-REC RECORD CODE IS 3

LOCATION MODE IS VIA CRIFS-TDM SET

WITHIN TDM-AREA

02 CASE-NO PIC 9(5)

02 DOE PIC 9(6)

02 GRADE PIC X(2)

02 RUN-NO PIC 9(1)

02 PR-TDM PIC X(1)

02 AGE PIC 9(2)

02 ACT-STUS PIC 9(1)

:

[environment conditions at time  
of examination]

02 TDM-CNT PIC 9(2) USAGE IS COMP

02 TDM-RESULTS OCCURS 1 TO 24 TIMES DEPENDING ON TDM-CNT

03 BP PIC 9(6)

03 HR PIC 9(3)

:

[recovery data]

:

[findings]

RECORD NAME IS LAB-REC RECORD CODE IS 4

LOCATION MODE IS VIA CRIFS-LAB SET

WITHIN LAB-AREA

02 CASE-NO PIC 9(5)

02 DOE PIC 9(6)

:

[test results]

02 MES-CNT PIC 9(2) USAGE IS COMP

02 LAB-MESSAGES OCCURS 0 TO 20 TIMES DEPENDING ON MES-CNT

03 TEST PIC X(2)

03 MESSAGE PIC X(10)



RECORD NAME IS CS-REC RECORD CODE IS 5

LOCATION MODE IS VIA CRIFS-CS SET

WITHIN CS-AREA

02 DOE PIC 9(6)

02 CASE-NO PIC 9(5)

: [other data from CS report form]

02 DIAG-CNT PIC 9(2) USAGE IS COMP

02 CS-DIAG OCCURS 0 TO 10 TIMES DEPENDING ON DIAG-CNT

03 DX-CODE PIC X(3)

03 DX-TEXT

04 KEYWORD PIC X(10)

04 TEXT PIC X(15)

02 PHYS-CNT PIC 9(2) USAGE IS COMP

02 PHYSICIANS OCCURS 1 TO 5 TIMES DEPENDING ON PHYS-CNT

03 PHYS-CODE PIC X(3)

03 DEPT-CODE PIC X(3)

RECORD NAME IS CAT-REC RECORD CODE IS 6

LOCATION MODE IS VIA CRIFS-CAT SET

WITHIN CAT-AREA

02 CASE-NO PIC 9(5)

02 DOE PIC 9(6)

02 GRADE PIC X(2)

02 HT PIC 9(2)

02 WT PIC 9(3)

: [data pertaining to catheterization]

: [data pertaining to coronary risk  
profile of patient]

02 CTH-SEQN PIC 9(5) USAGE IS COMP

02 CAT-DET OCCURS 0 TO 10 TIMES DEPENDING ON CTH-SEQN

03 SEC-B

04 SAM-REFERRAL OCCURS 6 TIMES

05 B-7 PIC X(1)

04 CLINICAL-REASONS OCCURS 10 TIMES  
     05 B-8 PIC X(1)  
 04 ECG-REASONS OCCURS 18 TIMES  
     05 B-9 PIC X(1)  
 03 SEC-C  
     04 CAT-PROCEDURES OCCURS 12 TIMES  
         05 C-10 PIC X(1)  
     04 ANGIOGRAPHY-COMPLETED OCCURS 9 TIMES  
         05 C-11 PIC X(1)  
         :  
                                     [remaining test results]  
 03 DIAG-CNT PIC 9(2) USAGE IS COMP  
 03 SEC-H  
     04 DIAG OCCURS 0 TO 20 TIMES DEPENDING ON DIAG-CNT  
     05 DX-CODE PIC 99

RECORD NAME IS DUMMY-REC RECORD CODE IS 7

LOCATION MODE IS CALC DMSCALC

IN AREA-NAME

USING SSAN, NAME

DUPLICATES ARE NOT ALLOWED

WITHIN DUMMY-AREA

02 SSAN PIC X(9)

02 NAME PIC X(27)

SET SECTION

SET NAME IS CRIFS-ECG

SET CODE IS 1

MODE IS CHAIN

ORDER IS SORTED

OWNER IS CRIFS-REC

MEMBER IS ECG-REC AUTOMATIC LINKED TO OWNER

ASCENDING KEY IS DOE

DUPLICATES ARE NOT ALLOWED

SET OCCURRENCE SELECTION IS THRU LOCATION MODE OF OWNER

SET NAME IS CRIFS-TDM

SET CODE IS 2

MODE IS CHAIN

ORDER IS SORTED

OWNER IS CRIFS-REC

MEMBER IS TDM-REC AUTOMATIC LINKED TO OWNER

ASCENDING KEY IS CASE-NO, DOE

DUPLICATES ARE NOT ALLOWED

SET OCCURRENCE SELECTION IS THRU LOCATION MODE OF OWNER

SET NAME IS CRIFS-LAB

SET CODE IS 3

MODE IS CHAIN

ORDER IS SORTED

OWNER IS CRIFS-REC

MEMBER IS LAB-REC AUTOMATIC LINKED TO OWNER

ASCENDING KEY IS DOE

DUPLICATES ARE NOT ALLOWED

SET OCCURRENCE SELECTION IS THRU LOCATION MODE OF OWNER

SET NAME IS CRIFS-CS

SET CODE IS 4

MODE IS CHAIN

ORDER IS SORTED

OWNER IS CRIFS-REC

MEMBER IS CS-REC AUTOMATIC LINKED TO OWNER

ASCENDING KEY IS CASE-NO, DOE

DUPLICATES ARE NOT ALLOWED

SET OCCURRENCE SELECTION IS THRU LOCATION MODE OF OWNER

SET NAME IS CRIFS-CAT

SET CODE IS 5

MODE IS CHAIN

ORDER IS SORTED

OWNER IS CRIFS-REC

MEMBER IS CAT-REC AUTOMATIC LINKED TO OWNER

ASCENDING KEY IS DOE

DUPLICATES ARE NOT ALLOWED

SET OCCURRENCE SELECTION IS THRU LOCATION MODE OF OWNER

SET NAME IS DUMMY-ECG

SET CODE IS 6

MODE IS CHAIN

ORDER IS SORTED

OWNER IS DUMMY-REC

MEMBER IS ECG-REC AUTOMATIC LINKED TO OWNER

ASCENDING KEY IS DOE

DUPLICATES ARE NOT ALLOWED

SET OCCURRENCE SELECTION IS THRU LOCATION MODE OF OWNER

SET NAME IS DUMMY-LAB

SET CODE IS 7

MODE IS CHAIN

ORDER IS SORTED

OWNER IS DUMMY-REC

MEMBER IS LAB-REC AUTOMATIC LINKED TO OWNER

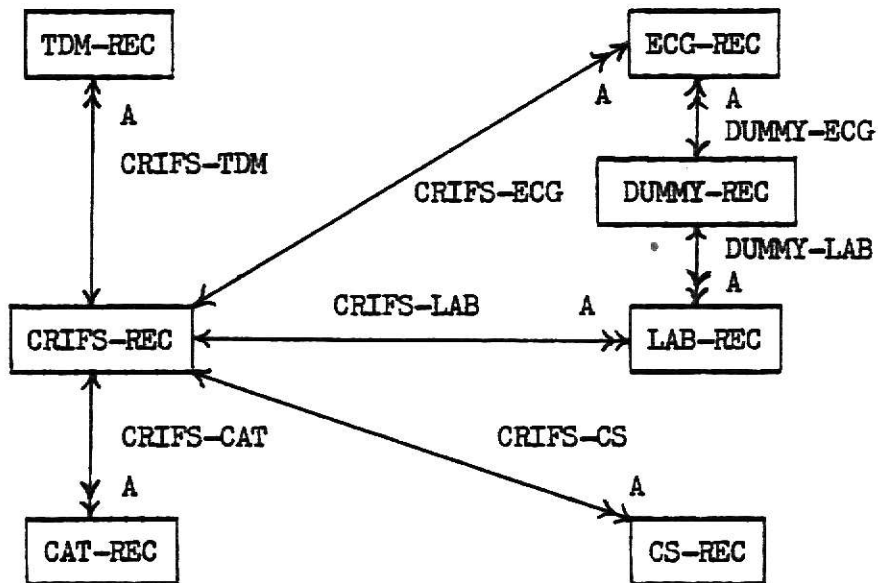
ASCENDING KEY IS DOE

DUPLICATES ARE NOT ALLOWED

SET OCCURRENCE SELECTION IS THRU LOCATION MODE OF OWNER

SUBSCHEMA NAME IS QUERY-USE OF SCHEMA REPOSITORIES FOR COBOL QLP  
COPY AREA ALL  
COPY RECORD ALL  
COPY SET ALL  
PATH NAME IS CS-CRIFS  
    ROOT IS CS-REC  
    FETCH NEXT CURRENCY ALLOWED  
    THRU CRIFS-CS TO CRIFS-REC  
PATH NAME IS CAT-ECG  
    ROOT IS CAT-REC  
    FETCH NEXT CURRENCY ALLOWED  
    THRU CRIFS-CAT TO CRIFS-REC  
    THRU CRIFS-ECG TO ECG-REC  
PATH NAME IS CAT-CRIFS  
    ROOT IS CAT-REC  
    FETCH NEXT CURRENCY ALLOWED  
    THRU CRIFS-CAT TO CRIFS-REC  
PATH NAME IS CAT-LAB  
    ROOT IS CAT-REC  
    FETCH NEXT CURRENCY ALLOWED  
    THRU CRIFS-CAT TO CRIFS-REC  
    THRU CRIFS-ECG TO ECG-REC  
    THRU DUMMY-ECG TO DUMMY-REC  
    THRU DUMMY-LAB TO LAB-REC

# LOGICAL DMS 1100 SCHEMA CONFIGURATION



## LEGEND



record type



set relationship

CRIFS-XXX

set name

A

automatic member



multiple member record occurrences possible



owner pointer in each member

## APPENDIX C

### DERIVATION OF SIZE REQUIREMENTS

# DB4 DATA SPACE — MASS STORAGE REQUIREMENTS

DATA GROUP	RECORD SIZE	NUMBER OF BYTES REQUIRED (* 10 <sup>6</sup> )
LAB	728	291.200
TDM	403	161.200
CAT-DET	146	116.800
CS	156	61.776
LAB-MESSAGES	20	40.000
CS-DIAG	20	40.000
CAT	43	17.286
ECG-EX	19	7.581
CRIFS	15	6.000
CS-PHYSICIANS	6	4.800
ECG-DX-CODES	0	0.000
CAT-DIAG	0	0.000

Total data space requirements      7.47 \* 10<sup>8</sup> bytes



# DB4 STRUCTURE SPACE — SIGNATURE SPACE REQUIREMENTS

SIGNATURE	NUMBER OF UNIQUE OCCURRENCES	NUMBER OF BYTES REQUIRED
NAME	400,000	10,800,000
SSAN	400,000	3,600,000
CASE-NO	400,000	2,000,000
B-9	262,144	786,432
DOB	36,525	219,150
DOE	36,525	219,150
DX-CODE	1,000	3,000
KEY-WORD	200	2,000
C-11	512	1,024
TEST	237	<u>474</u>
Total keyed signature requirements		$1.76 * 10^7$

# DMS 1100 RECORD AND PAGE SIZES (IN WORDS)

## RECORD SIZE

### CRIFS - REC

Header	1
Owner pointers of 5 sets	5
Index sequential pointer	1
Data (59 bytes)	<u>10</u>
	17

### ECG - REC

Header	1
Next set pointers	2
Owner pointers	2
Data (442 bytes)	<u>7</u>
	12

### TDM - REC

Header	1
Next set pointer	1
Owner pointer	1
Data (442 bytes)	<u>74</u>
	77

## PAGE SIZE

### CRIFS - AREA

Page header	10
400 recs/page	6800
Index slots	<u>400</u>
	7210
Page size	7724

### ECG - AREA

Page header	10
700 recs/page	<u>8400</u>
	8410
Page size	8428

### TDM - AREA

Page header	10
100 recs/page	<u>7700</u>
	7710
Page size	7728

# DMS 1100 RECORD AND PAGE SIZES (CONT.)

## RECORD SIZE

<u>LAB - REC</u>	
Header	1
Next set pointers	2
Owner pointers	2
Data (790 bytes)	<u>132</u>
	137

<u>CS - REC</u>	
Header	1
Next set pointer	1
Owner pointer	1
Data (326 bytes)	<u>55</u>
	58

<u>CAT - REC</u>	
Header	1
Next set pointer	1
Owner pointer	1
Data (435 bytes)	<u>73</u>
	76

## PAGE SIZE

<u>LAB - AREA</u>	
Page header	10
50 recs/page	<u>6850</u>
	6860
Page size	6860

<u>CS - AREA</u>	
Page header	10
120 recs/page	<u>6960</u>
	6970
Page size	6972

<u>CAT - AREA</u>	
Page header	10
120 recs/page	<u>9120</u>
	9130
Page size	9156

# DMS 1100 RECORD AND PAGE SIZES (CONT.)

RECORD SIZE		PAGE SIZE	
<u>DUMMY - REC</u>		<u>DUMMY - AREA</u>	
Header	1	Page header	10
Owner pointers of 2 sets	2	700 recs/page	<u>6300</u>
Data (36 bytes)	<u>6</u>		6310
	9	Page size	6328
<u>INDEX - AREA</u>			
		Page header	10
		1000 entries per page	<u>6500</u>
			6510
		Page size	6524

# DMS 1100 STORAGE REQUIREMENTS (IN BYTES)

AREA NAME	NUMBER OF PAGES (MAXIMUM)	PAGE SIZE	NUMBER OF BYTES REQUIRED (* 10 <sup>7</sup> )
CRIFS-AREA	1000	7724	4.634
ECG-AREA	570	8428	2.882
TDM-AREA	4000	7728	18.55
LAB-AREA	8000	6860	32.93
CS-AREA	3300	6972	13.80
CAT-AREA	3350	9156	18.40
DUMMY-AREA	570	6328	2.164
INDEX-AREA	400	6524	<u>1.566</u>

Total mass storage requirements    9.49 \* 10<sup>8</sup>

## BIBLIOGRAPHY

## BIBLIOGRAPHY

1. "A Computerized System for Processing Medical Repository Data".  
Report SAM-TR-77-21. Brooks Air Force Base, Texas, August 1977.
2. Martin, James. Computer Data-Base Organization. Englewood Cliffs,  
N.J.: Prentice-Hall, Inc., 1975.
3. UCC Data Management System. University Computing Company, Dallas,  
Texas, 1976.
4. Data Management System (DMS 1100) Schema Definition. Sperry Rand  
Corporation, Revision 2, 1975.
5. Query Language Processor (QLP 1100). Sperry Rand Corporation,  
Revision 1, 1977.
6. Data Management System (DMS 1100) System Support Functions. Sperry  
Rand Corporation, Revision 2, 1974.
7. Michaels, Ann S.; Mittman, Benjamin; and Carlson, C. Robert. "A  
Comparison of the Relational and CODASYL Approaches to Data-  
Base Management." Computing Surveys 8 (March 1976): 125-151.

A COMPARISON OF RELATIONAL AND NETWORK DATA BASE  
REPRESENTATIONS OF A MEDICAL REPOSITORY SYSTEM

BY

PAULA S. BOSWELL

B.S., NORTHWEST MISSOURI STATE UNIVERSITY

MARYVILLE, MISSOURI

1976

-----

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1978



## ABSTRACT

The United States Air Force School of Aerospace Medicine (USAFSAM) is currently using the MARK IV file management system for the storage and retrieval of medical repository data. The number of records existing in these repositories allows their use in clinical studies to gain statistical validity, but the present system is not being fully utilized due to the difficulty of information retrieval methods. This report involves the representation of the existing medical repositories in two different types of data bases, namely a relational data base (the UCC Data Management System) and a network data base (the Data Management System 1100). The comparison of advantages and disadvantages of each system focused mainly upon the query facilities available, with additional consideration given to storage requirements, procedures available to ensure data base integrity, system overhead incurred, and the ability of the data base to adapt to the changing information needs of the user.