A DEMONSTRATION MODEL OF A DISTRIBUTED AND INTERPRETIVE DATA BASE

by

Ernest J-H Chang

Research Report CS-74-16

Department of Computer Science

University of Waterloo
Waterloo, Ontario, Canada

June 1974

A DEMONSTRATION MODEL OF A DISTRIBUTED AND INTERPRETIVE DATA BASE

BY

ERNEST J-H CHANG

DEPARTMENT OF APPLIED ANALYSIS AND COMPUTER SCIENCE
FACULTY OF METHEMATICS
JUNE 1974

NOTE

This report is based on a thesis submitted to the Faculty of Mathematics for the M. Math degree.

ABSTRACT

Information systems built around a central computer impose many restrictions on individual users, incur large overhead costs, and have limited data handling capabilities.

A data base which is distributed over a computer network, and maintains a separation between logical and physical data structures, is at once simpler and more flexible. Local nodes process a small part of the total data, and can also be tailored to suit the particular views of users through the use of logical data maps.

The design of such a distributed data base is discussed. A pilot model, demonstrating the concept of physical and logical data descriptor maps, is implemented, and simulates the processes of a single node.

TABLE OF CONTENTS

Acknowledgements

Abstract

1. Introduction 1

2. Background of Data Bases 2

2.1 Historical Evolution 2

2.2 The Data Base Concept 3

2.3 The CODASYL Direction 5

2.4 Relational Data Bases 6

2.5 Other developments 7

2.6 Distributed Data Bases. 8

2.7 Illustrative Application Area 8

2.8 Kaiser-Permanente 10

2.9 The Danderyd System. 12

3. Proposed Data Base Schema 14

3.1 Introduction 14

3.2 Decentralization. 15

3.3 Interpretive Access to Data 16

3.4 Separation of Logical and Physical Representations 17

3.5 System-to-System Communication 19

3.6 Logical Data Base 20

3.7 Possible Implementation

3.7.a The Network 20

3.7.b The Nodes 21

3.7.c The Descriptor Maps. 21

3.7.d Local Programs of Node. 22

3.7.e File Communication Protocol 23

3.7.f Pilot Model 24

4. Design of Pilot Model 26

4.1 Data Access File Manager 26

4.2 File Restrictions 26

4.3 Restrictions on Descriptor Maps. 27

4.4 Synonyms 28

4.5 Retrieval Methods 28

5. Implementation Details of the Pilot Model. . . . 30

5.1 Computer System 30

5.2 Simulated Environment 30

5.3 Descriptor Files. 31

5.4 Display Capability 33

5.5 Retrieval Capability 34

5.6 Data Entry Capability 35

5.7 Overhead and Efficiency 35

6. Extensions and Summary 37

6.1 Extensions. 37

6.2 Summary. 38

Bibliography. 40

Appendix A Descriptor Maps

Section 1 Physical Storage Structure Map

File MEDIDENT

File MEDVISIT

File MEDBLOOD

Section 2 Logical Data Access Map MTESTZZ1

File MEDIDENT

File MEDVISIT

File MEDBLOOD

Section 3 Logical Data Access Map MTESTZZ2

File MEDIDENT

File MEDVISIT

File MEDBLOOD

Appendix B Data Records

Section 1 Physical Representation of Records

File MEDIDENT

File MEDVISIT

File MEDBLOOD

Section 2 Logical Records (version 1)

File MEDIDENT

File MEDVISIT

File MEDBLOOD

Section 3 Logical Records (version 2)

File MEDIDENT

File MEDVISIT

File MEDBLOOD

Appendix C Retrieval Runs

Section 1 Logical Records of MTESTZZ1

Entire Record Displayed

Simple Boolean Key

Section 2 Logical Records of MTESTZZ1

Partial Record Displayed

Compound Boolean Keys

Section 3 Logical Records of MTESTZZ2

Entire Record Displayed

Compound Boolean Keys

Section 4 Logical Records of MTESTZZ2

Entire Record Displayed

Compound Boolean Keys

Appendix D Data Entry

Section 1 Creating a New File

Section 2 Appending New Records

Section 3 Records Displayed

Chapter 1. Introduction

The problem of making data created at many different locations available simultaneously, rapidly, and efficiently, has seen much attention in recent years. The developement of the data base concept is in direct response to this problem. A data base has a large collection of data items, as well as the access mechanisms yielding efficient retrieval of this data. In Chapter 2, we will examine some of the traditional approaches taken in the design of data bases, and point out some of the problems inherent in these philosophies.

Chapter 3 is a demonstration model for a data base schema with the characteristics of being distributed, interpretive, and having a clear separation between physical and logical data structures.

An implementation of this data base schema will be discussed in the rest of Chapter 3.

In Chapter 4, we introduce a pilot model of this data base, which will illustrate some of the concepts presented, and Chapter 5 will outline the implementation of this model. In Chapter 6, extensions are suggested.

Sample data and retrieval runs will be presented in the Appendices.

Chapter 2 Background of Data Bases

2.1 Historical Evolution

The advent of magnetic tape recording medium marked a quantum jump in the amount and complexity of data that could be processed. Compared to card files, tapes offered far more convenience and processing capability. For example, the computer could be used to sort large files for the first time. Nevertheless, early tape file systems were often characterized by redundancy. Many files were required, for convenience, to carry data that already existed, perhaps in distributed form, on other files. Each file was created and updated by its own special set of programs. To bring together information found on many separate files often required the time-consuming processes of repeated selecting, sorting, and merging.

As organizations grew and the number of programs and files increased, all aimed at the similiar functions of insertion, deletion, and update, the inefficiencies became intolerable. Generalized File Systems such as MARK-IV (1972) were introduced. Typically, MARK-IV processes files in batch mode, through a non-procedural language using pre-defined descriptions of the record formats. Processes were constructed by filling out a tabulated sheet, subsequently key-punched.

At the same time, growth of data, and of perceived information needs, proceeded in another direction, perhaps

best described as an organizational one. Given the many different locations in which data concerning a particular entity or corporate body could originate, and the large numbers of different representations in such data, how was it possible to integrate all such information into a useful 'Data Bank' from which relevant information became easily and efficiently accessible? This kind of information need, and the organization of data systems to meet these demands, marked a radical departure from the rather simple file systems that were current, and gave rise to the notion of Data Bases with retrieval programs or 'managers'.

2.2 The Data Base Concept

A Data Base must provide efficient access to ALL of the information found on the files of an organization. At the same time, it must shield the user from the mundane details of the structure and encoding of this data. Access to data should occur at a higher conceptual level than in Generalized File Systems, which typically only extends the file processing capabilities to multiple files. The Data Base Manager must be able to collate, re-organize, and integrate the data to provide generalized access, while keeping this process transparent to the user. In other words, the Data Base should give a user powerful access paths to a collection of information, virtually organized according to his need.

These data management concepts have received much

attention in the literature, reviewed by the CODASYL Committee in the publications 'A Survey of Generalized Data Base Management Systems' (CODASYL 1969) and 'Feature Analysis of Generalized Data Base Management Systems' (CODASYL 1971). Many of these represent intermediates between Generalized File Systems with pre-defined , static data relations, and the broader concept of data bases. Typical of these systems is the following philosophy: Given the nature of the data from source documents, and the retrieval operations required of the system, the internal data structures for the system are designed. These tend to embody fixed relations between the data, oriented to the pre-defined retrieval paths. Implicitly, a single central computer is assumed. Examples given in the CODASYL report are MARK-IV, IDS, and TDMS, among others.

Such schemes inherently have the following disadvantages:

-centralized systems require compromising the needs of every single user to some overall commonality.

-the required pre-determined logical paths through the data creates an undesirable rigidity. For example, having decided on a particular IDS Master-Chain to Chain structure, the system can only process requests along the path given (IDS 1971).

-application programs using the data system also become frozen to the data structures chosen, and a small change in

one file may create major upheavals in the whole system.

The following major developments highlight research in the past few years, which have been aimed at some of these problems.

2.3 The CODASYL Direction

The CODASYL Committee Data Base Task Group (DBTG) in April 1971 (CODASYL 1971) specified a Data Definition Language (DDL). This would facilitate the description of logical data relations, from the level of data elements, themselves defined in the host-language COBOL, to the data base itself. The data base thus modelled is known as a Data Base Schema. User programs could structure their own views of data base subsets by defining Sub-Schemas in the DDL. An ancillary Data Management Language (DML) provided the user program with access to the data base. The DDL is really addressed to the modelling of logical features of the data structures and the data base. As pointed out by Sibley (1973), no physical specifications of any storage structures are made. Hence, this represents an incomplete scheme, and leaves open the questions of implementation and data portability.

Addressing the second problem, the CODASYL Stored Data Definition and Translation (SDDT) Task Group in 1972 (SDDT 1972) proposed a Stored Data Definition Language (SDDL). Much of this work is credited to the prior research of Smith (1971) and Taylor (1971). Conceptual (or logical) data

structures are defined separately from their storage structure and encoding, and the mapping between the two is an integral part of the SDDL. The language is rich enough to encompass the definition of a data base and of data structures found in many other data management systems. The corresponding complexity of languages arising from this approach is also clearly seen in a SIGFIDET paper by Smith (1972).

2.4 Relational Data Bases

Those computer scientists frustrated by the inflexibility of having the logical access paths to data permanently wedded to the physical data structures, turned to a different model of data bases. This model is based on the Relational Algebra, and is best seen in the work of Codd (1970) and Bracchi (1972). Starting from the idea that data entities bear certain relationships to other entities, and that, given these relations, other more complex relations can be derived in an algebraic sense, they attempted to develop an information system along these lines. Such data bases would be extremely flexible in the range of requests and types of retrieval possible, and the user would be completely shielded from the physical structure of the data. Only data entities, their relations, and the set of relational operators would be seen by the user.

The relational data base further allows the user to define relations as he sees them, and thus enables the

organization of the data to be modelled according to his view of the problem. This capability of course creates a good deal of inefficiency, but on the other hand gives a great deal of flexibility. To date only rather simple realizations of relational data bases have been demonstrated.

2.5 Other developments

There are two other proposals to consider. One is the Data Independent Access Method proposed by Altman (1972), Astrahan (1972), and Senko (1973). and the other relates to the Data Dictionary/Directory concept of Uhrowczik (1973).

The Data Independent Access Method (DIAM) is based on a model of relations between objects, which clearly delineates the abstractions we understand as concepts and as 'real entities' from their representations, whether verbal or graphic. The Entity Set Model describes the logical, i.e., conceptual, nature of data entities in terms of their relations to one another. The String Model and the Encoding Model are physical realizations of the conceptual data structuring given by the Entity Set Model. They use the idea of connector 'strings' between different entities to yield access paths, and impose ordering on data elements. The actual physical representation, given by the Encoding Model, is through a pointer-based interpretive scheme.

The Data Dictionary/Directory (DD/D) schema proposed by Uhrowczik (1973), advances the obvious and powerful idea of

separating the physical and logical representations of data, through mappings between virtual logical records and actual physical records. Keeping such data dictionaries in a single repository, an orgainzation can maintain centralized control over all vital data structures, while also being available to any number of application areas. Uhrowczik further extends this notion to include the construction of virtual data bases from given subsets of the total DD/D.

These two extremely interesting systems advance the state of the art in the proper direction of attempting integration of dissimiliar data, while creating flexibility in its retrieval.

2.6 Distributed Data Bases

Recent interest in computer communication and computer networks has stimulated some preliminary thinking on file systems and data bases which are physically distributed among the nodes of such a network (Booth 1972, Farber 1972). These are as yet primitive, and in the early stages of conceptual design, serving primarily to focus attention on a promising field.

2.7 Illustrative Application Area

A field with one of the most demanding requirements for mass information storage and retrieval is medicine. Typical of the problems in this area, 150,000 different terms are used for signs, symptoms and observations of 3800 different

diseases, referenced by 24,000 synonyms in the medical literature (Gordon 1970). In a medical information system, data concerning any particular person's medical history can come from the Registrar of Births, many physicians, dentists, clinics, hospitals, laboratories, nursing stations, and finally, the Coroner. Such data would exist in many different forms and different locations. While some work has been done in collating this kind of data post-hoc for research (see Acheson,1967 on Medical Record Linkage), this is by no means similiar to the directions required of a medical data base using diverse data.

One of the goals of a medical information system would be efficient file handling. This would give rapid access, on demand, to any particular aspect of a patient's history deemed relevant to the user. Conversely, another would be in limiting such access to only those users privileged to see the data. On a different plane, a medical information system should, in a data base capacity, be able to <u>abstract</u> from all patients' data information relevant to other purposes, for example, disease characteristics, population health indices, or medical diagnosis. The data base would therefore be expected to co-ordinate data from different locations, with varying structures, into a comprehensively and rapidly accessible pool of up-to-date information.

It is axiomatic that a medical data base should serve as large a population base, over as large a geographic area,

as possible. Since there are usually limitations to the practical size of any single system, inter-system communication is a very desirable feature. Finally, an information system giving 'critical' information must have the highest reliability, both in terms of accuracy and reliability.

Two major experiments have been conducted over the past ten years in large-scale medical information systems. One of these has been the Kaiser-Permanente Medical Data System in San Francisco (Van Brunt,1969,1970), and the other has been the Danderyd system (Abrahamsson,1970,1971) in Stockholm. Both are based on large centralized computers using integrated data formats.

2.8 Kaiser-Permanente

The largest private pre-paid Health Insurance plan in the United States, Kaiser began development of its Medical Data System (MDS) in 1965. Though a good deal of the data recorded was pertinent only to research applications, the data record for each patient nevertheless attempted to be an integrated medical record of all patient visits, collected from the many interactions of the patient with the system. The overall system, as described by Van Brunt (1970), used an IBM 360/50, three data cells and multiple disk drives to implement Direct Access file retrieval to about 150,000 patient records.

This record, the PCMR (Patient Computer Medical Record)

was interpretive (Davis 1970), in that all data fields had a length descriptor, a type descriptor, and a pointer to the next relevant field. A generalized tree was mapped into the record, so that data segments in the record were abstractly represented by sub-trees, and pointers from level to level were maintained for the virtual tree. All possible data entities for the system were pre-defined, and, within the record, a data item was represented by the Entity Name:Entity Value pair, similiar to the model of Mealy (1967). A data entity Dictionary was maintained for the whole system, describing the allowable range of values and data types for particular entities when entered from particular sources.

This data structure was extremely powerful and used concepts clearly ahead of its time. It provided a flexible data storage capability, which directly attacked the problem of the complexity of medical data, while at the same time lending itself to a generalized data retrieval mechanism. Nonetheless, the Kaiser-Permanente MDS was really a generalized file system, and not a data base. The disadvantages of the system were plain: the enormous overhead generated by the complexity of the system, the difficulties resulting from keeping an Entity Dictionary for many thousands of items, and curiously, the very restricted set of data items that the originator at source was permitted to enter. In fact, data entry from the physician

was primarily limited to the marking of pre-printed optical sense forms.

2.9 The Danderyd System

This is the only other significant medical information system attempted on a large scale. In 1967, it was proposed to build an on-line data retrieval system to serve the population of Stockholm, some 1.5 million (Abrahamsson,1970). The system would use twin Univac 494s, and maintain records on direct access storage devices, integrated amound the 'critical' medical information of all persons present in the Central Registry. Updates to the system were to be done in batch mode, and retrieval would be on-line using a variety of terminals.

This project was implemented using the MIDAS (Medical Information System Danderyd Stockholm) software package. File description tables implemented the data structures outside the retrieval programs, which thus became independent of file structure changes. Records could have fixed and variable-length segments, and the record was structured in three levels, viz., subrecords, segment types, and elements.

MIDAS included a FORTRAN-like retrieval language oriented specifically to the file system and to on-line functions.

This ambitious project aimed to be truly a 'data bank', using a Central Population Registry as the base of an on-

line medical retrieval system. It illustrates the use of data descriptors to free programs from the chains of physical data strucutres, an important innovation. At the same time, the discussion in Abrahamsson (1970,1971) also shows how such a centralized approach will always create the difficult problem of what data items are to be included, and what excluded, from the centralized record.

Chapter 3. Proposed Data Base Schema

3.1 Introduction

Many of the generalized file systems discussed above present the user with restrictive system constraints, since the amount of data he keeps in his system, the structure of that data, and the processing thereof, are subject to the limitations of the overall centralized system. Such systems also present large overheads, incurred by the collection of data from many separate sources to a single location, ensuring the conformity and correctness of all such data, and in the retrieval of any single piece of information from a large centralized set of files. It may be a truism to say that the complexity of a system directly compounds the likelihood of error and failure of the system, but this nevertheless is often ignored, and blithely more 'eggs-in-one-basket' systems continue to be created.

To deal with the overall complexity of large centralized systems, correspondingly large computers have also been required, while paradoxically, the actual processing of any single piece of data is perhaps trivial. Thus, where minicomputers are quite sufficient for the real data manipulation, goliaths are actually used. In a market environment where the costs of minicomputers are decreasing as their computing power is increasing, this is not a point to be taken lightly.

3.2 Decentralization

Clearly one possible way to bypass the actual-to-required computing mismatch present in many large systems due to overall complexity, is to use a distributed network of smaller and less expensive computers. In such a network of minicomputers, each would have a much smaller load of data structures and data files to process. Data entry can take place and be verified locally, and potentially, each minicomputer system can be quite simple and straightforward, working on its own small set of files.

Furthermore, the failure of one node of the network does not necessarily negate the activity of the rest of the network, and expected problems of hardware and software failures in any one piece of equipment does not have to cripple an entire system. In other words, the overall reliability of the entire system should be much improved compared to that of a single large system. In some application areas this is of vital importance, as in medical information systems.

However, if the distributed network is to be equivalent in a logical sense to a single central system, there must be logical access to data stored at one node from all other nodes. In a centralized system, the uniformity of the data structures ensures that all data collected maintains a consistent and known semantic meaning throughout the system. if we impose a similiar restriction on all the nodes of a

network, we are only keeping one of the problems of centralization. The processor at each node must be aware of all possible data structures, and can only create files of pre-determined and system-approved formats and data structures.

3.3 Interpretive Access to Data

We propose that logical communication between nodes occur using an interpretive access to data structures. Each node would be free to create data structures and records suited to the local applications. Furthermore, each node would be able to describe the structure of such data so that the information therein would be available to another node with no prior knowledge of that structure. Since communication really occurs at the semantic level, it is only logical that we model requests for particular data items in terms of the NAMEs of the data items, and that, as a corollary, the interpretive schema on the data records is also by name.

In short, data structures and records created at a node are to contain descriptors of the NAMEs of the fields and their data characteristics. A Data Access Manager would, at the node, interpret the data records according to the structural maps, and retrieve a NAMEd piece of data requested by another node.

This method of logical communication does require some system-wide consistency in terms of the meanings given to

particular names, and also incurs problems created by synonyms. Yet we have elevated the restrictions necessary for the system to impose, from the level of physical structures, to the level of semantics, which is perhaps a far more appropriate place for compromises.

3.4 Separation of Logical and Physical Representations

It follows from the interpretive method of access to data, that files created locally can be optimized for the particular features of the local application. We take this one step further, to say that the needs of a single user may also be better met in this scheme, by using the notion of logical records.

Different users certainly have different logical viewpoints, and see the same collection of data in many ways, depending on their needs for information, and according to the perspective of their particular application areas. In some collections of information, there are also data items which should be privileged, in the sense of being available only to certain individuals or classes of individuals, and not to anyone else.

We extend the idea of descriptors of data records to two types of data maps, one being a map of the physical record itself, and the NAMEs given to the most primary FIELDs of the record by its originator. This constitutes a Physical Data Descriptor Map. The next class of maps is an ordering of the FIELD names, to constitute a logical record,

formed by some subset of the FIELDs of the physical record. This constitutes a Logical Data Access Map. Clearly there can be only a single Physical Data Descriptor Map, and any number of Logical Data Access Maps, for any given data record type. A user would always see a logical record, as defined by the applicable Logical Data Access Map.

This separation of physical and logical representations yields at once the powerful concept of being able to shape the data representations to the needs of the user. Furthermore, it gives a very convenient way of protecting privileged information. At the same time, changes in logical structure need not always incur changes in the physical structures, involving costly and time-consuming re-programming. More important, changes to physical structure need not imply changing, recoding, and re-compiling all programs using those structures, but only the changing of an appropriate Physical Data Descriptor Map.

At this conceptual level, the distinction between logical and physical records is similiar to that made by the SDDT Task Group (SDDT 1972). However, we emphasize here not a definition language, but a data base design, utilizing the principle described.

The overhead involved in using this interpretive schema can be kept to a minimum by retaining the maps in the processor while a given file is being actively used. This overhead is clearly offset by the many advantages gained.

3.5 System-to-system Communication

Information systems designed around a centralized computer are based on a narrow and confined view of the world. The fact that each system optimizes its design to meet local objectives usually makes the interface between different systems extremely difficult, if not virtually impossible.

Yet it is often highly desirable to have this facility of inter-system communication. In medical information systems, for example, 'critical' information on an individual may be scattered over many systems. The characteristics of a rare disease may be found only by examining the data from many areas. Comprehensive population data are difficult to obtain unless the system is large enough to cover the population of interest.

For these, and many other reasons, isolation is an undesirable characteristic of an information system.

Distributed Data Bases offer a reasonable resolution of this dilemma. Communication is a feature built into the system, as is local optimization. Moreover, any centralized system usually has a practical limit to the amount of data it can process, and therefore a finite patient population. However, a distributed network, by being able to share computing loads, can make the total processing and data-handling capability of the system far greater.

3.6 Logical Data Base

We have presented some principles of design for a data base schema which are aimed at many of the serious disadvantages present in centralized systems. Given a distributed network of minicomputers with an interpretive access to data, using for this interpretation descriptor maps defining a name space on the physical record, and a logical path through the name space, we have a data base, physically distributed but logically unified and equivalent to a single centralized large data base.

3.7 Possible Implementation

Given the principles of a data base as proposed, we outline here a possible implementation.

3.7.a The Network

We connect a number of minicomputers, such as PDP-11/45s, in, arbitrarily, a Ring network (Farber,1970,1972). Each node will be capable of sending and accepting packets, which will be either requests or retrieved data. Each packet will have a Destination Identifier. Requests will have the Destination Identifier of the node which originated the request, so that the request can be broadcast to all nodes. A node which receives a packet addressed to it does not pass the packet forward. If the packet is retrieved data, the node then has what it asked for. If it is the original request, it means the

request has gone all around the network.

A node receiving a request with a Destination Identifier not its own will process the request, forward the original request packet, and any data retrieved, with the Destinatio ntifier of the requ A packet of retrieved data not addressed to it is simply shipped forward.

3.7.b The Nodes

Each node, being a PDP-11/45, will have the capability of time-sharing the resources of the node with the local users, as well as with the network. Each node will have some basic language processors in the operating system, and a basic file system which ideally can handle variable length records using Indexed Sequential and Direct Access file mode. The Catalog structure for the file system should be such that Read-sharing of files, at the least, between different local users is easily done. We note that at this time no PDP-11/45 has such a file system.

3.7.c The Descriptor Maps

Each file in the node will have a Physical Data Descriptor Map, which is a mapping of FIELD-names to physical storage. In addition, for each file there should be a set of Logical Data Access Maps, one for each class of users. A request packet will also identify the request class of the originator.

When a request enters the node from the network, the Data Access Manager of the node will initiate processing the request against all the files of the node. To do this, the pertinent Logical Data Access Map for the user class is located from a Direct Access Device, and read into core. If the requested FIELD name is present in the Logical Data Access Map for that file, then the Physical Data Descriptor Map for the file is read into fast memory. The file is then processed using these two maps for the request.

We note that at the moment it is easiset to plan for one record type per file.

When the file finishes processing, the rest of the files are examined in the same way, and any other pertinent data is shipped forward to the next node.

3.7.d Local Programs of the Node

Files, and their descriptor maps, will be created locally. Thus local programs which change the physical structure of a file will also have to update the proper Physical Data Descriptor Maps. We intend the files of a node to be used in two fashions:

(a) locally, non-interpretively, and, in the usual way, with data structures defined by the programming language, and

(b) in data base mode.

Data Base Mode costs the node the overhead of finding

the Descriptor Maps, and interpretively accessing the fields of the data. The possibility is raised here that all files should be accessed only through the Descriptor Maps, which creates an environment similiar to that discussed by Uhrowczik (1973).

The distinction between local and data base mode allows us to specify that all new records, updates, insertions, and deletions must be done locally. Data Base access is purely for retrieval of non-local information. In systems such as Medical Information Systems, it is not only sensible, but probably mandatory, for legal reasons of authority and responsibility, that local nodes alone can change the data records stored therein. This separation simplifies the data base operation greatly. Interpretive access to data files for update, deletion and insertion would increase the complexity of the Data Access Managers greatly. This is a topic we leave for further discussion.

3.7.e File Communication Protocol

A program at a node may request a data item from the network, or conversely, may be asked for a data item by another node. In either case, clearly a sequence of pre-defined steps will be required, which is the Communication Protocol. Much of it has already been alluded to in previous sections.

In the first place, a program will need to know, if we maintain the distinction between local and data base mode,

that a particular data item may be in the network. It must then know the NAME of the data item, and the range of values required. Then, a calling sequence, similiar to an external procedure call, must be established with the Data Access Manager of the local node, in order to send the request through the network. The Data Access Manager, upon receiving the NAME and value specification of the request, can construct a packet as discussed, and also keep track of the requesting program's identity and status. When the network sends the requested data back, the Data Access Manager, through the calling sequence, forwards it to the local program originating the whole process.

If multiple values are returned, a workspace must be set aside, probably in the local program, to hold all the data until processed. In the case of an overflow, secondary storage will probably be required.

When a node receives a request, the Data Access Manager processes it against the relevant files, and sends addressed packets of retrieved data back into the network stream.

This protocol, as described, is of course not complete, and should be viewed in the light of a suggested feasible approach, rather than as a detailed design.

3.7.f Pilot Model

We do not intend here to implement in full the network described in this section. Rather, we view this as a proposal for a data base schema, and go on to determine how

we may demonstrate the concepts outlined in this schema. We introduce a pilot model, which will emphasize the Data Access Manager and an implementation of the Interpretive Data Access method.

Chapter 4 Design of Pilot Model

4.1 Data Access File Manager

In attempting to show the feasibility of the concepts introduced in Chapter 3, we implement a pilot scheme. This model involves not a network of independent and connectable nodes, but a single Data Access Manager, which might act as the Data Access Manager of any of the nodes in a network .

This Manager, instead of being accessed in network mode, is instead used interactively to simulate such a function. User queries simulate on-line queries, and the display of information retrieved simulates the return of such data to the originating node.

Furthermore, instead of attempting the implementation of such a Data Access Manager on a minicomputer, we use some of the facilities built into a larger computer system, specifically, using the language PL/1 under CMS (IBM 1973).

This model Data Access Manager will, however, be able to define a Name Space on a given file, and an Access Path through this Name Space, by interpreting and imposing a logical data map (the Data Access Map) on the data record as defined by the physical data map (the Physical Data Descriptor Map).

4.2 File Restrictions

We make only three files known to the Data Access Manager. Since we impose our interpretive access method upon

a pre-defined file system, our Pilot Model has available to it the File Access methods of the parent Operating System. For this demonstration, we restrict the File Access Methods used to Sequential alone, since Indexed Sequential is not currently available under CMS software.

We further restrict the files to contain only a single record type each. Each record will be of fixed length and fixed format in this model.

4.3 Restrictions on Descriptor Maps

The basic entity in a record is a FIELD, identified by a Field-name. Associated with each Field is a Value. The Phsyical Data Descriptor Map is a direct mapping of Field-names into the physical record through a description, for each Field, of its Offset, Length, and Value Type (for example, Character or Fixed Decimal).

The Logical Data Access Map will be restricted to only one kind of logical structuring. Field-names can be sequentially ordered into GROUPS. Field-names and Group-names can be further ordered into other Groups. We choose a COBOL-like syntax for this logical ordering. For example,

```
01 NAME
   02 LNAME     *
   02 FNAME     *     where * indicates the name to be a
                            field-name rather than
                            a Group-name.
```

We demonstrate the concept of flexible logical structuring by using two different sets of Logical Data Access Maps, and showing the records as they appear given the two different logical orientations. The construction of the Descriptor Maps are, in this model, done by hand, rather than through a programmed facility. We leave this aspect of the problem to further extensions.

We insist that a Field-name or Group-name has the same semantic meaning in all the files known to the Data Access Manager.

For the sake of simplicity, we have restricted Field-value types to only Character and Fixed Decimal. We only demonstrate that the Data Access Manager is capable of handling mixed data types, and leave more complex situations to a fuller implementation.

4.4 Synonyms

It should be apparent that by defining a Group-name which contains a single Field-name in a Logical Data Access Map, we have just defined a synonym for the Field-name.

4.5 Retrieval Methods

We have restricted retrieval to the very simple operations of specifying Boolean combinations of up to five Field-names, with associated ranges of values. The AND operator is given precedence over the logical OR. Records matched are either displayed in full, or the user may

specify particular Group-names or Field-names of interest.

Files whose Logical Data Access Maps do not contain the Field-names in a given request, are of course not searched.

The retrieval operations are only intended to show that it is possible for the Data Access Manager to view three different files logically as one set of data entities, through the name space imposed on the physical records.

Chapter 5 Implementation Details of the Pilot Model

5.1 Computer System

The Data Access Manager was implemented on an IBM 370/145 using the facilities of the VM-CMS system (IBM 1973). The language used was PL/1-F. This particular combination was dictated by the need to have an interactive high-level language, capable of handling complex data structures, and facilitating systems programming.

5.2 Simulated Environment

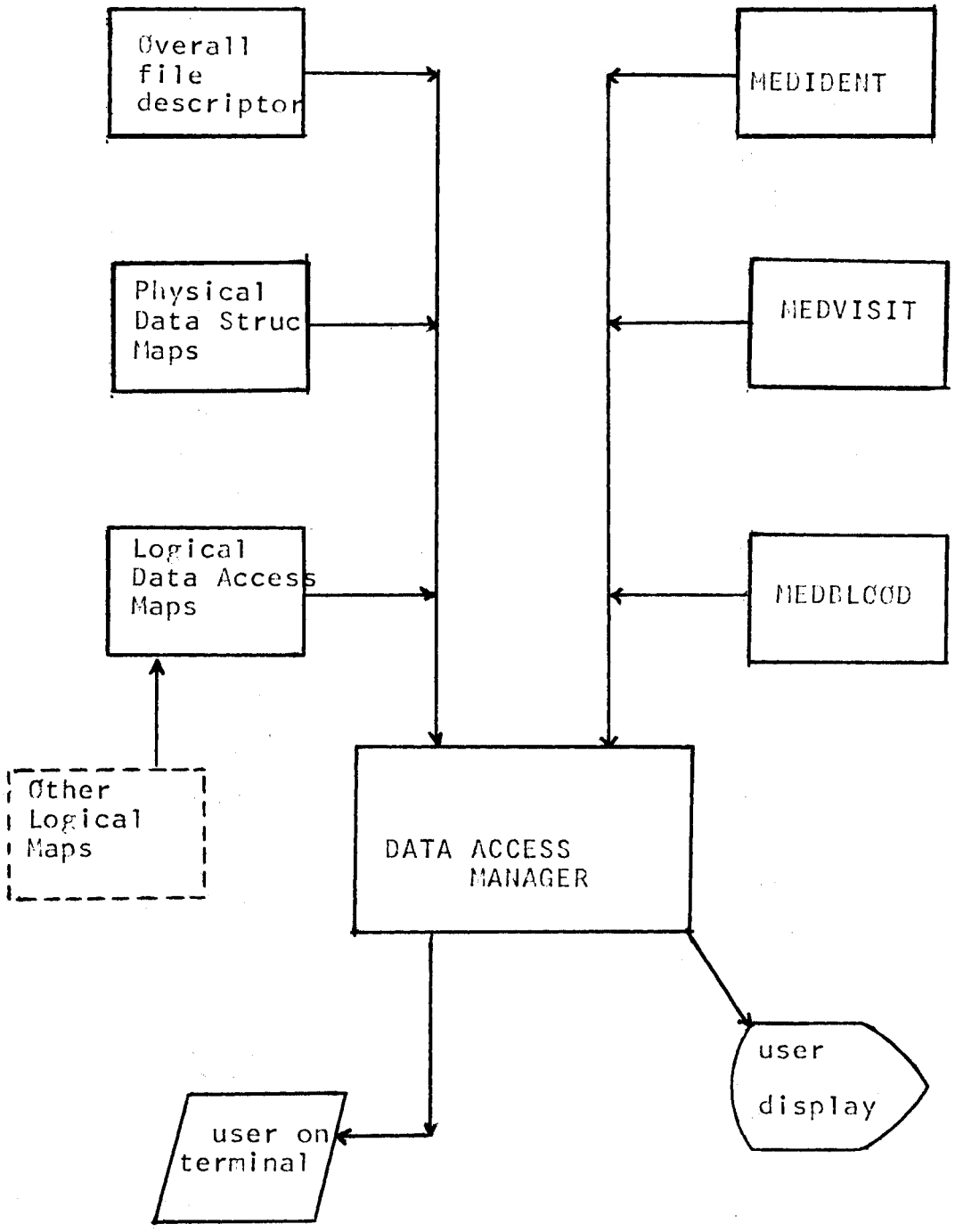
With the particular orientation of the author to medical information sytems, an environment suited to that application area was chosen. Given a small medical clinic, we simulate a small part of its data operations by creating three files which might be used in such a setting:

MEDIDENT a Patient Identity File
MEDVISIT a Patient Visit File, and
MEDBLOOD a file of Laboratory Reports for the test Complete Blood Count (CBC).

We point out again that each file contains only one record type. further, each record type is of fixed format, and fixed length. The field-names and field-characteristics for the records were chosen arbitrarily, but with an eye to realism.

The Access Method available to each file is Sequential,

Configuration of Pilot Model



as described in Chapter 4.

5.3 Descriptor Files

Three sets of descriptors are needed to communicate the characteristics of the files to the Data Access Manager of our simulated network node. The first contains the number of entries in the Physical Data Descriptor Map and in the Logical Data Access Map, for each of the files. The second is the set of Physical Data Descriptor Maps for the files, and the third is the set of Logical Data Access Maps.

We have chosen to place the first set of descriptors in a separate file, which contains the names of the known data fields, e.g., MEDIDENT, the Field-names constituting the primary organization-keys of the files, the file-access modes (for example, SEQUENTIAL), and the number of entries for the descriptor maps of each file. This file is known to the system as MTESTXX1.

The set of Physical Data Descriptor Maps are stored in a single record with the descriptors in contiguity, in the file known as MTESTYY1. This data remains invariant for the files. It is of course possible that the Physical Data Descriptor Maps for each file be prefaced in front of the actual data in the file itself, but in this particular implementation, for convenience, we have chosen to do it the other way around. No great advantages are to be gained in this model from either method.

The set of Logical Data Access Maps are also stored in

the same manner, in the file known as MTESTZZ1. Since the Logical Data Access Maps provide for logical paths into the data, and thus potentially logically different records, we have created a second set of Logical Data Access Maps, in the file MTESTZZ2.

The Data Access Manager, upon being initiated, first reads the Master File-Descriptor file (MTESTXX1), and allocates the required amount of storage for the other Descriptor Maps. Then, the Physical Data Descriptor Maps for the files are read in and made core-resident, which will enhance processing efficiency. Finally, the Logical Data Access Maps are also read in. There are two sets of Logical Data Access Maps. This model simulates a static situation, and, in a particular run, only one set of Logical Data Access Maps Maps is available. A run simulates communication with a particular other node, hence limiting the complexity of the data interpretation. Since the file MTESTXX1 sets the amount of storage required for the descriptor maps, when the second set of Logical Data Access Maps is used, the program must be initialized properly by using the file MTESTXX2, which contains the appropriate number of entries for the second set of Logical Data Access Maps.

Appendix A lists the structures of the maps pertaining to the three data files MEDIDENT, MEDVISIT, and MEDBLOOD. Section 1 shows the physical data descriptor maps, Section 2 the first set of logical data access maps, from MTESTZZ1,

and Section 3 the logical maps from MTESTZZ2.

5.4 Display Capability

The display of retrieved data by the system to the user simulates the communication of information from a node to the requesting node in a network. For semantic reasons, the data that is displayed is prefaced by the relevant Field-names.

The system can display to the user, in addition to data, the structures of either the Physical Data Descriptor Maps or the Logical Data Access Maps for any data file.

Furthermore, when a record is located in a retrieval process, it may either be displayed in full, according to the logical view given by the pertinent Logical Data Access Map, or in part. This latter capability is effected by prompting the user to enter the Group-name or Field-name that is to be displayed. Such names are of course recognized only if present in the Logical Data Access Map. It cannot be overemphasized that what a user sees is a logical record. The physical representation of that record is unavailable to him.

Appendix B contains three sets of listings of the data records of the three files. The first is a listing according to the actual Physical Data Descriptor Maps. In other words, the logical to physical Mapping is one:one. This listing is obtained by a special modificatin to the program. The second set lists the records according to the Logical Data Access

Maps of MTESTZZ1, and the third set according to the logical mappings imposed by the Logical Data Access Maps of MTESTZZ2.

5.5 Retrieval Capability

Up to this point we have described a system which views data according to a logical map, and accesses fields of the data by a phsyical descriptor map. This is in the environment of a multi-file system, and gives the Data Access Manager the capability of viewing all files logically as a single name-to-access-path space.

To demonstrate the utility of this, we have imposed onto these primitive operations a simple set of retrieval operators. The user may specify a combinaton of Field-names, up to 5, associated by the Boolean operators '&' and '/' (Logical AND and Logical OR) where '&' binds more tightly. In other words, AB/C is equivalent to (A&B)/C.

In addition, for each FIELD, the user specifies a lower and upper limit of values acceptable. If no restrictions on value are desired, '**' may be entered.

Either NN records or '**' may be retrieved, where '**' means ALL records satisfying the retrieval keys. Upon encountering a record satisfying the retrieval conditions, the program either displays the whole record or allows the display of specific FIELDs or GROUPs.

Appendix C shows two sets of retrieval runs, one using the Logical Data Access Maps of MTESTZZ1, and one using the

Logical Data Access Maps of MTESTZZ2.

When a set of retrieval keys has been entered, the system checks that a particular file is logically capable of meeting the request by matching the request keys against the Logical Data Access Map for the file. Files for which the request keys clearly cannot match are not searched.

5.6 Data Entry Capability

The Physical Data Descriptor Map for a record specifies the name as well as the characteristics of each FIELD in the record. It is simple to take advantage of this descriptor map to allow dynamic data entry by displaying the FIELDs in the proper order, and interactively building up the record through data entered by the user, for any number of records. Also, by using an OLD and a NEW version of the file, it is possible to append new data. A typical run is shown in Appendix D.

5.7 Overhead and Efficiency

In gaining the flexibility and power of interpretive data access, and a unified name-space, we have to pay the expense of having additional data descriptors, and accessing all data fields through the descriptor maps.

In the pilot model, the descriptor maps themselves use CHARACTER fields. For a Physical Data Descriptor Map, each entry describing a Data FIELD requires 36 characters. Each entry for a FIELD-name or GROUP-name in a Logical Data

Access Map requires 11 characters of description.

The records we have chosen have approximately 20 FIELDs in each, and for each of the files, there is the overhead then of approximately 800 characters per Physical Data Descriptor Map. Each Logical Data Access Map probably provides a logical view of at least one-half of the data entities present in a record, and thus incurs at least another 300 characters per Logical Data Access Map per file. This kind of load is trivial in comparison to the amount of data usually contained in a single file.

The major added cost in using this schema would come not so much from any change in file access methods, but in the interpretive access to the fields of a record once it has been found. This would consist basically of finding the FIELD-name in a core-resident map, and then using the location descriptors to find the FIELD itself in the record. The advantages of logical-interpretive data access should easily offset this cost in a network environment. We have postulated that, where data created locally was most often used in the local environment, decentralization of the data base gives many benefits. Within this kind of network model, the medical information data base should show very favorable characteristics.

Chapter 6 Extensions and Summary

6.1 Extensions

We have described the design and operation of a pilot model for a data base schema using the principles of interpretive access to data. In Chapter 3, a far more extensive implementation was discussed, involving minicomputers, network distribution, and interpretive data access. Towards achieving this implementation, and a more realistic and useful system, the pilot model will require extensions in the following directions:

a- implement the basic model for the Data Access Manager on minicomputers, viz., PDP-11/45s.

b- establish communication links between two or more similiar minicomputers, in order to create a distributed data base.

c- extend and modify the display mode, so that it will be possible for a program to request by a procedure CALL, the value of a FIELD(s) from the Data Base, and use the value returned as a normal internal datum.

d- extend file access methods to random access, e.g., ISAM (Indexed Sequential Access Method) or DIRECT.

e- extend the capability of the Descriptor Maps to handle variable length records with variable field occurrences.

f- extend the Logical Data Access Map concept to enable specific users or classes of users to see, for specific purposes, different logical views of the data.

g- to re-assess the distinction drawn between Local and Data Base mode. Perhaps ALL programs should access data only by procedure calls for specific FIELD names, thus eliminating the need to know which data items are local, and which available through the network.

h- Last and not least, the application area of interest, medical information systems, will require more intensive systems engineering for implementation.

6.2 Summary

Data Base models proposed to date often have the disadvantageous characteristic of being centralized, which leads to large systems overhead for even simple processing, and tends to be overly restrictive in the latitude allowed any single user. While it is recognized that different users have different needs, and different logical views of data, the usual strategies for data representations have been addressed to system efficiencies rather than to user requirements. These flaws are seen in two large medical information systems, reviewed briefly.

We have proposed a data base schema which is to be

emphasizes local features in the structuring of data records, thus tending to favor the user. In addition, we have subscribed to the principle that different users should and must have different logical views of the same data, a notion which lends itself to the protection of privileged information. This design clearly facilitates system-to-system communication.

A possible implementation of this data base scheme using a network of minicomputers was discussed.

The concepts in the proposal were demonstrated using a pilot model, which served to simulate the functioning of a single node of the network by the interaction of a user at a terminal. The major features in this model show the feasibility of separating the physical and logical representations of data, and the interpretive access to data files that this implies. This interpretive-logical access is implemented by defining Name:Storage Descriptor maps, and imposing on this name-space Name:Name Logical Data Access maps. The Data Access Manager in the pilot model then has available to it, not many separate files, but a single data space on which can be defined numerous access paths. A simulated medical environment is used. The extension of this model to the full network implementation should encounter no conceptual barriers.

Bibliography

Abrahamsson,S. Danderyd hospital computer system. 2. Total regional system for medical care. Comput Biomed Res 3:30-46,1970.

Abrahamsson,S. Danderyd hospital computer system. 3. Basic software design. Comput Biomed Res 4:126-140,1971.

Acheson,ED. Medical record linkage. Oxford University Press, London-New York-Toronto, 1967.

Altman,ED. Specifications in a data independent accessing model. Proc ACM SIGFIDET workshop on data description and access, Denver, 303-382,1972.

Astrahan,MM. Concepts of a data independent accessing model. Proc ACM SIGFIDET workshop on data description and access, Denver, 349-362,1972.

Booth,GM. The use of distributed data bases. Proc of the 1st international conference on computer communications, Washington, D.C., 1972. Published as S. Winkler (ed.). Computer Communication--Impacts and Implications, 364-370,1972.

Bracchi,G. A language for a relational data base management system. Proc 6th annual Princeton conference on information sciences and systems, Princeton, 1972.

CODASYL Data Base Task Group report. ACM, New York,1971.

CODASYL Systems Committee. A survey of generalized data base management systems. ACM, New York,1969.

CODASYL Systems Committee. Feature analysis of generalized data base management systems. ACM, New York,1971.

Codd,EF. A relational model of data for large, shared data banks. Comm ACM 13:377-387,1970.

Davis,LS. Prototype for future computer medical records. Conference on MIS, San Francisco, 167-182,1970.

Farber,DJ. Data ring oriented computer networks. Proc Courant Institute symposium on computer networks, New York, 1970. Published as R.Rustin (ed.). Computer Networks, Prentice Hall, 79-94,1972.

Farber,DJ. The structure of a distributed computer system- the distributed file system. Proc of the 1st international conference on computer communication, Washington, D.C., 1972. Published as S. Winller (ed.). Computer Communication - Impacts and Implications, IEEE, 364-370,1972.

Gordon,BL. Terminology and content of the medical record. Conference on MIS, San Francisco, 53-62,1970.

IBM Virtual machine facility /370: command language guide for general users. Order no. GC20-1804-1. 1973.

IDS Reference manual, Honeywell Information Systems, Inc. Order no. BR69, Phoenix, Arizona, 1971.

MARK-IV File Management System reference manual. Informatics Inc. Document no. SP-72-860-1C. Canoga Park, California. 1972.

Mealy,GH. Another look at data. AFIPS conference proc, Fall Joint Computer Conference, 31:525-534,1967.

SDDT Task Group. An approach to stored data definition and translation. Proc ACM SIGFIDET workshop on data description and access, Denver,1972.

Senko,ME. Data structures and accessing in data-base systems. IBM Systems Journal 12:30-93,1973.

Sibley,EH. A data definition and mapping language. Comm ACM, 16:750-759,1973.

Smith,DP. An approach to data description and conversion. Ph.D. dissertation. The Moore School of Electrical Engineering, U. of Pennsylvania, 1971.

Smith,DP. A method for data translation using the stored-data definition and translation task group languages. Proc ACM SIGFIDET workshop on data description and access, Denver,1972.

Taylor,RW. Generalized data base management system data

structures and their mapping to physical storage. Ph.D. dissertation, U of Michigan,1971.

Uhrowczik,PP. Data dictionary/directories. IBM Systems Journal 12:332-352,1973.

Van Brunt,EE. A pilot data system for a medical center. Proc IEEE 57:1934-1940,1969.

Van Brunt,EE. Current status of a medical information system. Meth Inform Med 9:149-160,1970.

Appendix A Descriptor Maps

Section 1 Physical Storage Structure Map
File MEDIDENT

```
prego
FILEDEF FILEIN DISK MTESTXX1 TEST A1 ( PERM RECFM V LRECL 250
FILEDEF FILEST DISK MTESTYY1 TEST A1 ( PERM RECFM V LRECL 2500
FILEDEF FILEAC DISK MTESTZZ1 TEST A1 ( PERM RECFM V LRECL 1000
FILEDEF FILE1 DISK MTESTID1 TEST A1 ( PERM RECFM F LRECL 117
FILEDEF FILE2 DISK MTESTVS1 TEST A1 ( PERM RECFM F LRECL 218
FILEDEF FILE3 DISK MTESTBL1 TEST A1 ( PERM RECFM F LRECL 99
R; T=0.22/0.49 13:54:55

load moda (stat@rt nomap)
EXECUTION BEGINS...

THE FOLLOWING FILES ARE KNOWN TO THE SYSTEM:
  MEDIDENT  SEQUENTIAL KEY = LNAME
  MEDVISIT  SEQUENTIAL KEY = SELFMR
  MEDBLOOD  SEQUENTIAL KEY = SELFMR
WOULD YOU LIKE TO LOOK AT THE MAPS?

yes
SPECIFY FILENAME

medident


STORAGE OR ACCESS MAP?

st


NAME-OF-FIELD:  LNAME
OFFSET: 0        LENGTH: 10
TYPE: FNAME


NAME-OF-FIELD:  FNAME
OFFSET: 10       LENGTH: 8
TYPE: CHAR


NAME-OF-FIELD:  INITIAL
OFFSET: 18       LENGTH: 1
TYPE: CHAR


NAME-OF-FIELD:  SEX
OFFSET: 19       LENGTH: 1
TYPE: CHAR
```

Appendix A Descriptor Maps

Section 1 Physical Data Descriptor Map
File MEDIDENT

NAME-OF-FIELD: BMONTH
OFFSET: 20 LENGTH: 2
TYPE: CHAR

NAME-OF-FIELD: BYEAR
OFFSET: 22 LENGTH: 2
TYPE: CHAR

NAME-OF-FIELD: SELFMR
OFFSET: 24 LENGTH: 8
TYPE: CHAR

NAME-OF-FIELD: DOCTOR
OFFSET: 32 LENGTH: 10
TYPE: CHAR

NAME-OF-FIELD: SINNO
OFFSET: 42 LENGTH: 8
TYPE: CHAR

NAME-OF-FIELD: MLNAME
OFFSET: 50 LENGTH: 10
TYPE: CHAR

NAME-OF-FIELD: BLOODTP
OFFSET: 60 LENGTH: 2
TYPE: CHAR

NAME-OF-FIELD: ADDRST
OFFSET: 62 LENGTH: 15
TYPE: CHAR

NAME-OF-FIELD: ADDRCITY
OFFSET: 77 LENGTH: 10
TYPE: CHAR

NAME-OF-FIELD: ADDPOSTZ
OFFSET: 87 LENGTH: 6
TYPE: CHAR

Appendix A Descriptor Maps

Section 1 Physical Storage Structure Map
File MEDIDENT

NAME-OF-FIELD: ADDTELE
OFFSET: 93 LENGTH: 8
TYPE: CHAR

NAME-OF-FIELD: MARITAL
OFFSET: 101 LENGTH: 1
TYPE: CHAR

NAME-OF-FIELD: SPOUSEMR
OFFSET: 102 LENGTH: 8
TYPE: CHAR

NAME-OF-FIELD: COVERTYP
OFFSET: 110 LENGTH: 1
TYPE: CHAR

NAME-OF-FIELD: COVERDAT
OFFSET: 111 LENGTH: 6
TYPE: CHAR

Appendix A Descriptor Maps

Section 1 Physical Storage Structure Map
File MEDVISIT

```
MORE?
yes
SPECIFY FILENAME

medvisit

STORAGE OR ACCESS MAP?
st


NAME-OF-FIELD:  SELFMR
OFFSET: 0        LENGTH: 8
TYPE: CHAR


NAME-OF-FIELD:  VISDAY
OFFSET: 8        LENGTH: 2
TYPE: CHAR


NAME-OF-FIELD:  VISMON
OFFSET: 10       LENGTH: 2
TYPE: CHAR


NAME-OF-FIELD:  VISYR
OFFSET: 12       LENGTH: 2
TYPE: CHAR


NAME-OF-FIELD:  VTIME
OFFSET: 14       LENGTH: 4
TYPE: CHAR


NAME-OF-FIELD:  VISTYPE
OFFSET: 18       LENGTH: 1
TYPE: CHAR


NAME-OF-FIELD:  LOCATION
OFFSET: 19       LENGTH: 8
TYPE: CHAR
```

Appendix A Descriptor Maps

Section 1 Physical Storage Structure Map
File MEDVISIT

NAME-OF-FIELD: DOCTOR
OFFSET: 27 LENGTH: 10
TYPE: CHAR

NAME-OF-FIELD: DIAGN1
OFFSET: 37 LENGTH: 15
TYPE: CHAR

NAME-OF-FIELD: DIAGN2
OFFSET: 52 LENGTH: 15
TYPE: CHAR

NAME-OF-FIELD: DIAGN3
OFFSET: 67 LENGTH: 15
TYPE: CHAR

NAME-OF-FIELD: RX1
OFFSET: 82 LENGTH: 15
TYPE: CHAR

NAME-OF-FIELD: RX2
OFFSET: 97 LENGTH: 15
TYPE: CHAR

NAME-OF-FIELD: RX3
OFFSET: 112 LENGTH: 15
TYPE: CHAR

NAME-OF-FIELD: RX4
OFFSET: 127 LENGTH: 15
TYPE: CHAR

NAME-OF-FIELD: REFER1
OFFSET: 142 LENGTH: 10
TYPE: CHAR

NAME-OF-FIELD: SPECLT1
OFFSET: 152 LENGTH: 8
TYPE: CHAR

Appendix A Descriptor Maps

Section 1 Physical Storage Structure Map
File MEDVISIT

NAME-OF-FIELD: REFER2
OFFSET: 160 LENGTH: 10
TYPE: CHAR

NAME-OF-FIELD: SPECLT2
OFFSET: 170 LENGTH: 8
TYPE: CHAR

NAME-OF-FIELD: NOTE1
OFFSET: 178 LENGTH: 20
TYPE: CHAR

NAME-OF-FIELD: NOTE2
OFFSET: 198 LENGTH: 20
TYPE: CHAR

Appendix A Descriptor Maps

Section 1 Physical Storage Structure Map
File MEDBLOOD

```
MORE?
yes


SPECIFY FILENAME
medblood
STORAGE OR ACCESS MAP?
st


NAME-OF-FIELD:  SELFNR
OFFSET: 0        LENGTH: 8
TYPE: CHAR

NAME-OF-FIELD:  TESTDAY
OFFSET: 8        LENGTH: 2
TYPE: CHAR

NAME-OF-FIELD:  TESTMON
OFFSET: 10       LENGTH: 2
TYPE: CHAR

NAME-OF-FIELD:  TESTYR
OFFSET: 12       LENGTH: 2
TYPE: CHAR

NAME-OF-FIELD:  DOCTOR
OFFSET: 14       LENGTH: 10
TYPE: CHAR

NAME-OF-FIELD:  LABREQN
OFFSET: 24       LENGTH: 8
TYPE: CHAR

NAME-OF-FIELD:  LABTECH
OFFSET: 32       LENGTH: 10
TYPE: CHAR
```

Appendix A Descriptor Maps

Section 1 Physical Storage Structure Map
File MEDBLOOD

```
NAME-OF-FIELD:  SPECOK
OFFSET: 42        LENGTH: 1
TYPE: CHAR


NAME-OF-FIELD:  HB (GM%)
OFFSET: 43        LENGTH: 2
TYPE: FIXED
PRECISION 3 ,1


NAME-OF-FIELD:  ESR
OFFSET: 45        LENGTH: 2
TYPE: FIXED
PRECISION 3 ,0


NAME-OF-FIELD:  WBC
OFFSET: 47        LENGTH: 4
TYPE: FIXED
PRECISION 6 ,0


NAME-OF-FIELD:  HCT (%)
OFFSET: 51        LENGTH: 2
TYPE: FIXED
PRECISION 3 ,1


NAME-OF-FIELD:  PLATELET
OFFSET: 53        LENGTH: 4
TYPE: FIXED
PRECISION 6 ,0


NAME-OF-FIELD:  RBC
OFFSET: 57        LENGTH: 4
TYPE: FIXED
PRECISION 7 ,0


NAME-OF-FIELD:  MCV
OFFSET: 61        LENGTH: 2
TYPE: FIXED
PRECISION 3 ,0
```

Appendix A Descriptor Maps

Section 1 Physical Storage Structure Map
File MEDBLOOD

```
NAME-OF-FIELD:  MCH
OFFSET: 63        LENGTH: 2
TYPE: FIXED
PRECISION 3 ,0


NAME-OF-FIELD:  MCHC
OFFSET: 65        LENGTH: 2
TYPE: FIXED
PRECISION 3 ,0


NAME-OF-FIELD:  SEGMENTD
OFFSET: 67        LENGTH: 2
TYPE: FIXED
PRECISION 3 ,0


NAME-OF-FIELD:  BANDS
OFFSET: 69        LENGTH: 2
TYPE: FIXED
PRECISION 3 ,0


NAME-OF-FIELD:  LYMPHS
OFFSET: 71        LENGTH: 2
TYPE: FIXED
PRECISION 3 ,0


NAME-OF-FIELD:  MONOS
OFFSET: 73        LENGTH: 2
TYPE: FIXED
PRECISION 3 ,0


NAME-OF-FIELD:  EOSINOPH
OFFSET: 75        LENGTH: 2
TYPE: FIXED
PRECISION 3 ,0


NAME-OF-FIELD:  BASOPH
OFFSET: 77        LENGTH: 2
TYPE: FIXED
PRECISION 3 ,0


NAME-OF-FIELD:  DESCRIP
OFFSET: 79        LENGTH: 20
TYPE: CHAR
```

Appendix A Descriptor Maps

Section 2 Logical Data Access Map
File MEDIDENT version 1 (MTESTZZ1)

```
medident
STORAGE OR ACCESS MAP?
access


 IDENTITY
  NAME
   FNAME    *
   INITIAL  *
   LNAME    *
  SEX       *
  BIRTH
   BMONTH   *
   BYEAR    *
  SELFMR    *
  MARITAL   *
  TAGS
   SINNO    *
   SELFMR   *
   BLOODTP  *
   MLNAME   *
 ADDRESS
  ADDRST    *
   ADDRCITY*
    ADDPOSTZ*
 PHONE
  ADDTELE   *
 COVERAGE
  COVERTYP*
  COVERDAT*
 DOCTOR     *
MORE?
```

Appendix A Descriptor Maps

Section 1 Logical Data Access Map
File MEDVISIT version 1 (MTESTZZ1)

```
                    medvisit
STORAGE OR ACCESS MAP?
access


 IDENTITY
  SELFMR   *
 VDATE
  VISDAY   *
  VISMON   *
  VISYR    *
 VISIT
  VISTYPE  *
  LOCATION*
  DOCTOR   *
 DIAGNOSX
  DIAGN1   *
  DIAGN2   *
  DIAGN3   *
 VRXS
  RX1      *
  RX2      *
  RX3      *
  RX4      *
 REFERRAL
  RDOC1
   REFER1   *
   SPECLT1  *
  RDOC2
   REFER2   *
   SPECLT2  *
 NOTES
  NOTE1    *
  NOTE2    *
MORE?
```

Appendix A Descriptor Maps

Section 1 Logical Data Access Map
File MEDBLOOD version 1 (MTESTZZ1)

```
medblod@od
STORAGE OR ACCESS MAP?
access


 IDENTITY
  SELFMR  *
 DATE
  TESTDAY *
  TESTMON *
  TESTYR  *
 DOCTOR  *
 HB (GM%)*
 HCT (%) *
 MCH     *
   (MCMCG)
 MCV     *
   (CU MIC)
 MCHC    *
   (%)
 RBC     *
   (/CU MM)
 ESR     *
   (MM/HR)
 PLATELET*
   (/CU MM)
 WBC     *
   (/CU MM)
 DIFFRNTL
  SEGMENTD*
  BANDS   *
  LYMPHS  *
  MONOS   *
  EOSINOPH*
  BASOPH  *
 DESCRIP *
MORE?
```

Appendix A Descriptor Maps

Section 3 Logical Data Access Map
File MEDIDENT version 2 (MTESTZZ2)

```
medident
STORAGE OR ACCESS MAP?
access

 IDENTITY
  SELFNR  *
  SINNO   *
  MLNAME  *
 DOCTOR  *
 BLOODTP *
MORE?
```

File MEDVISIT version 2 (MTESTZZ2)

```
read error
medvisit
STORAGE OR ACCESS MAP?
access

 DOCTOR  *
 VISTYPE *
 DIAGNOSX
  DIAGN1  *
  DIAGN2  *
  DIAGN3  *
 REFERRAL
  SPECLT1 *
  SPECLT2 *
MORE?
```

Appendix A Descriptor Maps

Section 3 Logical Data Access Map
File MEDBLOOD version 2 (MTESTZZ2)

```
yes
SPECIFY FILENAME

medblood
STORAGE OR ACCESS MAP?
access

 SELFMR   *
 DOCTOR   *
 DATE
  TESTDAY *
  TESTMON *
  TESTYR  *
 HB (GM%)*
 WBC      *
MORE?
```

Appendix B Data Records

Section 1 Physical representation of records
File MEDIDENT

```
SEARCH PARAMETERS: SPECIFY
SR: FIELDNAME = ........
doctor
SR: LOWER VALUE = ......
**
SR: UPPER VALUE = ......
**
NUMBER OF RECORDS = NN OR **
4
SR: ACTION WHEN HIT OCCURS...DISPLAY OR ANNOUNCE
special
FILE MEDIDENT BEING SEARCHED
MATCHING FIELDS ARE:
DOCTOR
LNAME     = FARRELL
FNAME     = SUSAN
INITIAL   = Q
SEX       = F
BMONTH    = 01
BYEAR     = 29
SELFNR    = 16948524
DOCTOR    = FERGUSON
SINNO     = 38862366
MLNAME    = FERGIN
BLOODTP   = A-
ADDRST    = 314 REGINA
ADDRCITY  = KITCHENER
ADDPOSTZ  = N2L2N6
ADDTELE   = 578-1685
MARITAL   = M
SPOUSENR  = 83947752
COVERTYP  = B
COVERDAT  = 010275
```

Appendix B Data Records

Section 1 Physical representation of records
File MEDVISIT

```
SEARCH PARAMETERS: SPECIFY
SR: FIELDNAME = ........
vistype
SR: LOWER VALUE = ......
**
SR: UPPER VALUE = ......
**
NUMBER OF RECORDS = NN OR **
5
SR: ACTION WHEN HIT OCCURS...DISPLAY OR ANNOUNCE
special
VISTYPE  NOT IN ACCESS PATH MAP
MEDIDENT DOES NOT MATCH FOR KEYS...IGNORED
FILE MEDVISIT BEING SEARCHED
MATCHING FIELDS ARE:
VISTYPE
SELFNR    = 12583719
VISDAY    = 12
VISMON    = 01
VISYR     = 73
VTIME     = 1620
VISTYPE   = E
LOCATION  = OFFICE
DOCTOR    = TRACHSELL
DIAGN1    = DIABETES M - O
DIAGN2    = OBESITY
DIAGN3    = HYPERTENSION
RX1       =
RX2       =
RX3       =
RX4       =
REFER1    = DAVIDSON
SPECLT1   = INTERNST
REFER2    =
SPECLT2   =
NOTE1     = URINE +3 X 3D.
NOTE2     = BP 180/140
```

Appendix B Data Records

Section 1 Physical representation of records
File MEDBLOOD

```
SEARCH PARAMETERS: SPECIFY
SR: FIELDNAME = ........
wbc
SR: LOWER VALUE = ......
**
SR: UPPER VALUE = ......
**
NUMBER OF RECORDS = NN OR **
**
SR: ACTION WHEN HIT OCCURS...DISPLAY OR ANNOUNCE
special
WBC       NOT IN ACCESS PATH MAP
MEDIDENT DOES NOT MATCH FOR KEYS...IGNORED
WBC       NOT IN ACCESS PATH MAP
MEDVISIT DOES NOT MATCH FOR KEYS...IGNORED
FILE MEDBLOOD BEING SEARCHED
MATCHING FIELDS ARE:
WBC
SELFNR    = 13957386
TESTDAY   = 10
TESTMON   = 02
TESTYR    = 73
DOCTOR    = JAGER
LABREQN   = CBC00201
LABTECH   = ABDARKUS
SPECOK    = Y
HB (GM%)  =                  13.7
ESR       =                    0
WBC       =                 7400
HCT (%)   =                   .0
PLATELET  =                    0
RBC       =                    0
MCV       =                    0
MCH       =                    0
MCHC      =                    0
SEGMENTD  =                   70
BANDS     =                    2
LYMPHS    =                   20
MONOS     =                    1
EOSINOPH  =                    4
BASOPH    =                    3
DESCRIP   = FACTOR VIII OK
```

```
no
SEARCH PARAMETERS: SPECIFY
SR: FIELDNAME = ........

lname
SR: LOWER VALUE = ......
**
SR: UPPER VALUE = ......
**
NUMBER OF RECORDS = NN OR **
**
SR: ACTION WHEN HIT OCCURS...DISPLAY OR ANNOUNCE
display


FILE MEDIDENT BEING SEARCHED
MATCHING FIELDS ARE:
LNAME

 IDENTITY
  NAME
   FNAME     = SUSAN
   INITIAL   = Q
   LNAME     = FARRELL
  SEX        = F
  BIRTH
   BMONTH    = 01
   BYEAR     = 29
  SELFMR     = 16948524
  MARITAL    = M
  TAGS
   SINNO     = 38862366
   SELFMR    = 16948524
   BLOODTP   = A-
   MLNAME    = FERGIN
 ADDRESS
  ADDRST     = 314 REGINA
   ADDRCITY  = KITCHENER
    ADDPOSTZ = N2L2N6
 PHONE
  ADDTELE    = 578-1685
 COVERAGE
  COVERTYP   = B
  COVERDAT   = 010275
 DOCTOR      = FERGUSON
```

Appendix B Data Records

Section 2 Logical records v.1 (MTESTZZ1)
File MEDVISIT

```
VISDAY   NOT IN ACCESS PATH MAP
FILE MEDIDENT BEING SEARCHED
MATCHING FIELDS ARE:
END OF FILE MEDIDENT
FILE MEDVISIT BEING SEARCHED
MATCHING FIELDS ARE:
VISDAY

 IDENTITY
  SELFMR    = 12583719
 VDATE
  VISDAY    = 12
  VISMON    = 01
  VISYR     = 73
 VISIT
  VISTYPE   = E
  LOCATION  = OFFICE
  DOCTOR    = TRACHSELL
 DIAGNOSX
  DIAGN1    = DIABETES M - O
  DIAGN2    = OBESITY
  DIAGN3    = HYPERTENSION
 VRXS
  RX1       =
  RX2       =
  RX3       =
  RX4       =
 REFERRAL
  RDOC1
   REFER1    = DAVIDSON
   SPECLT1   = INTERNST
  RDOC2
   REFER2    =
   SPECLT2   =
 NOTES
  NOTE1     = URINE +3 X 3D.
  NOTE2     = BP 180/140
```

Appendix B Data Records

Section 2 Logical records v.1 (MTESTZZ1)
File MEDBLOOD

```
IDENTITY
 SELFMR    = 20648596
DATE
 TESTDAY   = 10
 TESTMON   = 02
 TESTYR    = 73
DOCTOR     = JAGER
HB (GM%)   =                  14.8
HCT (%)    =                   .0
MCH        =                    0
  (MCMCG)
MCV        =                    0
  (CU MIC)
MCHC       =                    0
  (%)
RBC        =                    0
  (/CU MM)
ESR        =                    0
  (MM/HR)
PLATELET   =                    0
  (/CU MM)
WBC        =                 9200
  (/CU MM)
DIFFRNTL
 SEGMENTD  =                   81
 BANDS     =                    4
 LYMPHS    =                   11
 MONOS     =                    1
 EOSINOPH  =                    1
 BASOPH    =                    2
DESCRIP    = FACTOR VIII OK
```

Appendix B Data Records

Section 3 Logical records v.2 (HTESTZZ2)
File MEDIDENT

```
no
SEARCH PARAMETERS: SPECIFY
SR: FIELDNAME = ........

selfmr
SR: LOWER VALUE = ......
**
SR: UPPER VALUE = ......
**
NUMBER OF RECORDS = NN OR **
**
SR: ACTION WHEN HIT OCCURS...DISPLAY OR ANNOUNCE
disp
FILE MEDIDENT BEING SEARCHED
MATCHING FIELDS ARE:
SELFMR

 IDENTITY
  SELFMR    = 16948524
  SINNO     = 38862366
  MLNAME    = FERGIN
 DOCTOR     = FERGUSON
 BLOODTP    = A-

 IDENTITY
  SELFMR    = 83947752
  SINNO     = 59634720
  MLNAME    = FELHABER
 DOCTOR     = FEDY
 BLOODTP    = A+

 IDENTITY
  SELFMR    = 21478604
  SINNO     = 75403558
  MLNAME    = QUANZ
 DOCTOR     = FERGUSON
 BLOODTP    = B+

 IDENTITY
  SELFMR    = 12583719
  SINNO     = 50128041
  MLNAME    = QUANZ
 DOCTOR     = TRACHSELL
 BLOODTP    = A+
```

Appendix B Data Records

Section 3 Logical records v.2 (MTESTZZ2)
File MEDVISIT

```
no
SEARCH PARAMETERS: SPECIFY
SR: FIELDNAME = ........
diagn1
SR: LOWER VALUE = ......
**
SR: UPPER VALUE = ......
**

NUMBER OF RECORDS = NN OR **
**
SR: ACTION WHEN HIT OCCURS...DISPLAY OR ANNOUNCE
dispaly@@@lay
DIAGN1   NOT IN ACCESS PATH MAP
FILE MEDIDENT BEING SEARCHED
MATCHING FIELDS ARE:
END OF FILE MEDIDENT

FILE MEDVISIT BEING SEARCHED
MATCHING FIELDS ARE:
DIAGN1

 DOCTOR   = TRACHSELL
 VISTYPE  = E
 DIAGNOSX
  DIAGN1   = DIABETES M - O
  DIAGN2   = OBESITY
  DIAGN3   = HYPERTENSION
 REFERRAL
  SPECLT1  = INTERNST
  SPECLT2  =

 DOCTOR   = FERGUSON
 VISTYPE  = E
 DIAGNOSX
  DIAGN1   = SPRAINED ANKLE
  DIAGN2   =
  DIAGN3   =
 REFERRAL
  SPECLT1  =
  SPECLT2  =
```

Appendix B Data Records

Section 3 Logical records v.2 (MTESTZZ2)
File MEDBLOOD

```
SELFMR    = 57450607
DOCTOR    = JAGER
DATE
 TESTDAY  = 24
 TESTMON  = 01
 TESTYR   = 73
HB (GM%) =                14.3
WBC      =               7800

SELFMR    = 60504758
DOCTOR    = JAGER
DATE
 TESTDAY  = 10
 TESTMON  = 02
 TESTYR   = 73
HB (GM%) =                12.8
WBC      =               5400

SELFMR    = 62083788
DOCTOR    = JAGER
DATE
 TESTDAY  = 06
 TESTMON  = 01
 TESTYR   = 73
HB (GM%) =                15.0
WBC      =              18000

SELFMR    = 83947752
DOCTOR    = FEDY
DATE
 TESTDAY  = 13
 TESTMON  = 02
 TESTYR   = 73
HB (GM%) =                14.1
WBC      =              13700
END OF FILE MEDBLOOD
END OF THIS RUN...WANT TO GO ON?
```

Appendix C Retrieval Runs

Section 1 Logical records of MTESTZZ1
Entire record displayed
Simple Boolean key

```
load moda (start nomap)
EXECUTION BEGINS...
THE FOLLOWING FILES ARE KNOWN TO THE SYSTEM:
  MEDIDENT  SEQUENTIAL KEY = LNAME
  MEDVISIT  SEQUENTIAL KEY = SELFMR
  MEDBLOOD  SEQUENTIAL KEY = SELFMR
WOULD YOU LIKE TO LOOK AT THE MAPS?
no
SEARCH PARAMETERS: SPECIFY
SR: FIELDNAME = ........
selfmr
SR: LOWER VALUE = ......
35852959
SR: UPPER VALUE = ......
35852959
NUMBER OF RECORDS = NN OR **
**
SR: ACTION WHEN HIT OCCURS...DISPLAY OR ANNOUNCE
display
```

Appendix C Retrieval Runs

Section 1 Logical records of MTESTZZ1
Entire record displayed
Simple Boolean key

```
FILE MEDIDENT BEING SEARCHED
MATCHING FIELDS ARE:
SELFMR

 IDENTITY
  NAME
   FNAME     = MICHAEL
   INITIAL   = M
   LNAME     = WURR
  SEX        = M
  BIRTH
   BMONTH    = 03
   BYEAR     = 39
  SELFMR     = 35852959
  MARITAL    = M
  TAGS
   SINNO     = 28105703
   SELFMR    = 35852959
   BLOODTP   = A+
   MLNAME    = MAXWELL
 ADDRESS
  ADDRST     = 227 WINSTON
   ADDRCITY  = WATERLOO
    ADDPOSTZ = N2L2P7
 PHONE
  ADDTELE    = 822-3011
 COVERAGE
  COVERTYP   = A
  COVERDAT   = 10475
 DOCTOR      = JAGER
END OF FILE MEDIDENT
```

Appendix C Retrieval Runs

Section 1 Logical records of MTESTZZ1
Entire record displayed
Simple Boolean key

```
 FILE MEDVISIT BEING SEARCHED
 MATCHING FIELDS ARE:
 SELFMR

  IDENTITY
   SELFMR    = 35852959
  VDATE
   VISDAY    = 11
   VISMON    = 01
   VISYR     = 73
  VISIT
   VISTYPE   = A
   LOCATION  = OFFICE
   DOCTOR    = JAGER
  DIAGNOSX
   DIAGN1    = BRONCHITIS
   DIAGN2    =
   DIAGN3    =
  VRXS
   RX1       = TETRACYC 2WK
   RX2       = RTC 2WK
   RX3       =
   RX4       =
  REFERRAL
   RDOC1
    REFER1   =
    SPECLT1  =
   RDOC2
    REFER2   =
    SPECLT2  =
  NOTES
   NOTE1     = SMOKES 2PPD!
   NOTE2     =
```

Appendix C Retrieval Runs

Section 1 Logical records of MTESTZZ1
Entire record displayed
Simple Boolean key

```
  IDENTITY
   SELFMR    = 35852959
  VDATE
   VISDAY    = 25
   VISMON    = 01
   VISYR     = 73
  VISIT
   VISTYPE   = A
   LOCATION  = OFFICE
   DOCTOR    = JAGER
  DIAGNOSX
   DIAGN1    = C.O.P.D.
   DIAGN2    =
   DIAGN3    =
  VRXS
   RX1       = QUIT CIGS!
   RX2       =
   RX3       =
   RX4       =
  REFERRAL
   RDOC1
    REFER1   =
    SPECLT1  =
   RDOC2
    REFER2   =
    SPECLT2  =
  NOTES
   NOTE1     = EXERTIONAL S.O.B.
   NOTE2     = YELLOW A.M. SPUTUM
END OF FILE MEDVISIT
```

Appendix C Retrieval Runs

Section 1 Logical records of MTESTZZ1
Entire record displayed
Simple Boolean key

```
 FILE MEDBLOOD BEING SEARCHED
 MATCHING FIELDS ARE:
 SELFMR

  IDENTITY
   SELFMR    = 35852959
  DATE
   TESTDAY   = 10
   TESTMON   = 02
   TESTYR    = 73
  DOCTOR     = JAGER
  HB (GM%)   =                  13.1
  HCT (%)    =                    .0
  MCH        =                     0
    (MCMCG)
  MCV        =                     0
    (CU MIC)
  MCHC       =                     0
    (%)
  RBC        =                     0
    (/CU MM)
  ESR        =                     0
    (MM/HR)
  PLATELET   =                     0
    (/CU MM)
  WBC        =                  7100
    (/CU MM)
  DIFFRNTL
   SEGMENTD  =                    50
   BANDS     =                     2
   LYMPHS    =                    41
   MONOS     =                     2
   EOSINOPH  =                     3
   BASOPH    =                     2
  DESCRIP    = FACTOR VIII OK
 END OF FILE MEDBLOOD
 END OF THIS RUN...WANT TO GO ON?
```

Appendix C Retrieval Runs

Section 2 Logical records of MTESTZZ1
Partial record displayed
Compound Boolean keys

```
SEARCH PARAMETERS: SPECIFY
SR: FIELDNAME = ........
sex       &
SR: LOWER VALUE = ......
f
SR: UPPER VALUE = ......
f
SR: FIELDNAME = ........
byear     /
SR: LOWER VALUE = ......
44
SR: UPPER VALUE = ......
54
SR: FIELDNAME = ........
mlname
SR: LOWER VALUE = ......
diebolt
SR: UPPER VALUE = ......
diebolt
NUMBER OF RECORDS = NN OR **
**
SR: ACTION WHEN HIT OCCURS...DISPLAY OR ANNOUNCE
announce
FILE MEDIDENT BEING SEARCHED
MATCHING FIELDS ARE:
SEX       &
BYEAR     /
MLNAME
RECORD FOUND FROM FILE MEDIDENT
WHATS TO BE DONE?
S: REPLY FIELD/GROUP NAME

name
  NAME
  FNAME     = JUDI
  INITIAL   = N
  LNAME     = FENWICK
S: MORE?
```

Appendix C Retrieval Runs

Section 2 Logical records of MTESTZZ1
Partial record displayed
Compound Boolean keys

```
yes
S: REPLY FIELD/GROUP NAME

phne
PHNE      NOT IN ACCESS PATH MAP
RPFLD/GROUP NOT IN ACCESS PATH MAP
S: MORE?

yes
S: REPLY FIELD/GROUP NAME
phone
 PHONE
 ADDTELE  = 884-3022
S: MORE?

yes
S: REPLY FIELD/GROUP NAME

address
 ADDRESS
 ADDRST   = 111 CONWAY
 ADDRCITY = GUELPH
 ADDPOSTZ = N2J7B5
S: MORE?

yes
S: REPLY FIELD/GROUP NAME

marital
  MARITAL  = S
S: MORE?

no
RECORD FOUND FROM FILE MEDIDENT
WHATS TO BE DONE?
S: REPLY FIELD/GROUP NAME
next
NEXT      NOT IN ACCESS PATH MAP
RPFLD/GROUP NOT IN ACCESS PATH MAP
S: MORE?
```

Appendix C Retrieval Runs

Section 3 Logical records of MTESTZZ2
Entire record displayed
Compound Boolean keys

```
THE FOLLOWING FILES ARE KNOWN TO THE SYSTEM:
  MEDIDENT  SEQUENTIAL KEY = LNAME
  MEDVISIT  SEQUENTIAL KEY = SELFMR
  MEDBLOOD  SEQUENTIAL KEY = SELFMR
WOULD YOU LIKE TO LOOK AT THE MAPS?
no

SEARCH PARAMETERS: SPECIFY
SR: FIELDNAME = ........
doctor   /
SR: LOWER VALUE = ......
ferguson
SR: UPPER VALUE = ......
ferguson
SR: FIELDNAME = ........
bloodtp
SR: LOWER VALUE = ......
a-
SR: UPPER VALUE = ......
a-
NUMBER OF RECORDS = NN OR **
**
SR: ACTION WHEN HIT OCCURS...DISPLAY OR ANNOUNCE
d
FILE MEDIDENT BEING SEARCHED
MATCHING FIELDS ARE:
DOCTOR   /
BLOODTP

 IDENTITY
  SELFMR    = 16948524
  SINNO     = 38862366
  MLNAME    = FERGIN
 DOCTOR    = FERGUSON
 BLOODTP   = A-

 IDENTITY
  SELFMR    = 21478604
  SINNO     = 75403558
  MLNAME    = QUANZ
 DOCTOR    = FERGUSON
 BLOODTP   = B+
```

Appendix C Retrieval Runs

Section 4 Logical records of MTESTZZ2
Entire record displayed
Compound Boolean keys

```
load moda (start nomap)
EXECUTION BEGINS...
THE FOLLOWING FILES ARE KNOWN TO THE SYSTEM:
  MEDIDENT  SEQUENTIAL KEY = LNAME
  MEDVISIT  SEQUENTIAL KEY = SELFNR
  MEDBLOOD  SEQUENTIAL KEY = SELFNR
WOULD YOU LIKE TO LOOK AT THE MAPS?
no
SEARCH PARAMETERS: SPECIFY
SR: FIELDNAME = ........
speclt1  /
SR: LOWER VALUE = ......
cardiol
SR: UPPER VALUE = ......
cardiol
SR: FIELDNAME = ........
testday  &
SR: LOWER VALUE = ......
01
SR: UPPER VALUE = ......
31
SR: FIELDNAME = ........
testmon
SR: LOWER VALUE = ......
01
SR: UPPER VALUE = ......
01
NUMBER OF RECORDS = NN OR **
**
SR: ACTION WHEN HIT OCCURS...DISPLAY OR ANNOUNCE
d
SPECLT1  NOT IN ACCESS PATH MAP
TESTDAY  NOT IN ACCESS PATH MAP
TESTMON  NOT IN ACCESS PATH MAP
MEDIDENT DOES NOT MATCH FOR KEYS...IGNORED
TESTDAY  NOT IN ACCESS PATH MAP
TESTMON  NOT IN ACCESS PATH MAP
FILE MEDVISIT BEING SEARCHED
MATCHING FIELDS ARE:
SPECLT1  /
```

Appendix C Retrieval Runs

Section 4 Logical records of MTESTZZ2
Entire record displayed
Compound Boolean keys

```
 DOCTOR    = JAGER
 VISTYPE   = E
 DIAGNOSX
  DIAGN1   = ACUTE MI
  DIAGN2   = PULM EDEMA
  DIAGN3   = ARRHYTHMIA
 REFERRAL
  SPECLT1  = CARDIOL
  SPECLT2  =
END OF FILE MEDVISIT
SPECLT1  NOT IN ACCESS PATH MAP
FILE MEDBLOOD BEING SEARCHED
MATCHING FIELDS ARE:
TESTDAY  &
TESTMON

 SELFMR    = 27264531
 DOCTOR    = JAGER
 DATE
  TESTDAY  = 05
  TESTMON  = 01
  TESTYR   = 73
 HB (GM%)  =                  14.3
 WBC       =                7000

 SELFMR    = 57450607
 DOCTOR    = JAGER
 DATE
  TESTDAY  = 10
  TESTMON  = 01
  TESTYR   = 73
 HB (GM%)  =                   9.4
 WBC       =              125000

 SELFMR    = 57450607
 DOCTOR    = JAGER
 DATE
  TESTDAY  = 17
  TESTMON  = 01
  TESTYR   = 73
 HB (GM%)  =                  11.4
 WBC       =               75000
```

Appendix D Data Entry

Section 1 Creating a New File

```
dgo dmake1 dmake2 117
FILEDEF FILEIN DISK MTESTXX1 TEST A1 ( PERM RECFM V LRECL 250
FILEDEF FILEST DISK MTESTYY1 TEST A1 ( PERM RECFM V LRECL 2500
FILEDEF MODIN DISK DMAKE1 TEST A1 ( PERM RECFM F LRECL 117
FILEDEF MOUTF DISK DMAKE2 TEST A1 ( PERM RECFM F LRECL 117
LOAD DMAKE ( START )
EXECUTION BEGINS...
FILE NUMBER FOR MODIFICATION?
1
NEW FILE?
yes
LNAME
jones
FNAME
henry
INITIAL
f
SEX
m
BMONTH
12
BYEAR
56
SELFMR
12345678
DOCTOR
johnson
SINNO
23456789
MLNAME
thomas
BLOODTP
a+
ADDRST
123 moore
ADDRCITY
waterloo
ADDPOSTZ
```

Appendix D Data Entry

Section 1 Creating a New File

```
ADDTELE
885-1211
MARITAL
m
SPOUSEMR
23456783
COVERTYP
b
COVERDAT
1274
LNAME
**done
R; T=5.06/10.56 19:07:22


t dmake2 test

JONES      HENRY    FM125612345678JOHNSON    23456789THOMAS

R; T=0.05/0.13 19:07:51
```

Appendix D Data Entry

Section 2 Appending New Records

```
dgo dmake2 dmake1 117
FILEDEF FILEIN DISK MTESTXX1 TEST A1 ( PERM RECFM V LRECL 250
FILEDEF FILEST DISK MTESTYY1 TEST A1 ( PERM RECFM V LRECL 2500
FILEDEF MODIN DISK DMAKE2 TEST A1 ( PERM RECFM F LRECL 117
FILEDEF MOUTF DISK DMAKE1 TEST A1 ( PERM RECFM F LRECL 117
LOAD DMAKE ( START )
EXECUTION BEGINS...
FILE NUMBER FOR MODIFICATION?
1
NEW FILE?
no
IF INSERT ENTER *AP*
*ap*
LNAME
smith
FNAME
fred
INITIAL
d
SEX
m
BMONTH
12
BYEAR
44
SELFMR
98765432
DOCTOR
weston
SINNO
97654321
MLNAME
tyrell
BLOODTP
o-
ADDRST
321 eroom
ADDRCITY
waterloo
ADDPOSTZ
```

Appendix D Data Entry

Section 2 Appending New Records

```
ADDTELE
885-2480
MARITAL
s
SPOUSEMR

COVERTYP
t
COVERDAT
1275
LNAME
**done
R; T=5.62/11.49 19:10:45

t dmake1 test

JONES       HENRY     FM125612345678JOHNSON    23456789THOMAS
SMITH       FRED      DM124498765432WESTON     97654321TYRELL

R; T=0.06/0.15 19:11:41
```

Appendix D Data Entry

Section 3 Records displayed

```
prego
FILEDEF FILEIN DISK MTESTXX2 TEST A1 ( PERM RECFM V LRECL 250
FILEDEF FILEST DISK MTESTYY1 TEST A1 ( PERM RECFM V LRECL 2500
FILEDEF FILEAC DISK MTESTZZ2 TEST A1 ( PERM RECFM V LRECL 1000
FILEDEF FILE1 DISK MTESTID1 TEST A1 ( PERM RECFM F LRECL 117
FILEDEF FILE2 DISK MTESTVS1 TEST A1 ( PERM RECFM F LRECL 218
FILEDEF FILE3 DISK MTESTBL1 TEST A1 ( PERM RECFM F LRECL 99
R; T=0.21/0.48 19:12:39

filedef filein disk mtestxx1 test a1 (perm recfm v lrecl 250
R; T=0.03/0.04 19:12:59

filedef fileac de@isk mtestzz1 test z@a1 (perm recfm v lrecl 1000
R; T=0.03/0.04 19:13:23

filedef file1 disk dmake1 test a1 (perm recfm f lrecl 117
R; T=0.03/0.04 19:13:56

lod@ad moda (start nomap)
EXECUTION BEGINS...
THE FOLLOWING FILES ARE KNOWN TO THE SYSTEM:
  MEDIDENT  SEQUENTIAL KEY = LNAME
  MEDVISIT  SEQUENTIAL KEY = SELFNR
  MEDBLOOD  SEQUENTIAL KEY = SELFNR
WOULD YOU LIKE TO LOOK AT THE MAPS?
no
SEARCH PARAMETERS: SPECIFY
SR: FIELDNAME = ........
lname
SR: LOWER VALUE = ......
**
SR: UPPER VALUE = ......
**
NUMBER OF RECORDS = NN OR **
**
```

Appendix D Data Entry

Section 3 Records Displayed

```
SR: ACTION WHEN HIT OCCURS...DISPLAY OR ANNOUNCE
d
FILE MEDIDENT BEING SEARCHED
MATCHING FIELDS ARE:
LNAME

 IDENTITY
  NAME
   FNAME     = HENRY
   INITIAL   = F
   LNAME     = JONES
  SEX        = M
  BIRTH
   BMONTH    = 12
   BYEAR     = 56
  SELFMR     = 12345678
  MARITAL    = M
  TAGS
   SINNO     = 23456789
   SELFMR    = 12345678
   BLOODTP   = A+
   MLNAME    = THOMAS
 ADDRESS
  ADDRST     = 123 MOORE
   ADDRCITY  = WATERLOO
    ADDPOSTZ =
 PHONE
  ADDTELE    = 885-1211
 COVERAGE
  COVERTYP   = B
  COVERDAT   = 1274
 DOCTOR      = JOHNSON
```

Appendix D Data Entry

Section 3 Records Displayed

```
  IDENTITY
   NAME
    FNAME     = FRED
    INITIAL   = D
    LNAME     = SMITH
   SEX        = M
   BIRTH
    BMONTH    = 12
    BYEAR     = 44
   SELFMR     = 98765432
   MARITAL    = S
   TAGS
    SINNO     = 97654321
    SELFMR    = 98765432
    BLOODTP   = O-
    MLNAME    = TYRELL
  ADDRESS
   ADDRST     = 321 EROOM
    ADDRCITY  = WATERLOO
     ADDPOSTZ =
  PHONE
   ADDTELE    = 885-2480
  COVERAGE
   COVERTYP   = T
   COVERDAT   = 1275
  DOCTOR      = WESTON
END OF FILE MEDIDENT
LNAME     NOT IN ACCESS PATH MAP
MEDVISIT DOES NOT MATCH FOR KEYS...IGNORED
LNAME     NOT IN ACCESS PATH MAP
MEDBLOOD DOES NOT MATCH FOR KEYS...IGNORED
END OF THIS RUN...WANT TO GO ON?
no
JOB FINISHED....BYE
R; T=7.32/14.32 19:16:54
```