METHODS FOR THE DESIGN ANALYSIS OF MEDICAL DATA BASE SYSTEMS

by

Gio C M Wiederhold

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

MEDICAL INFORMATION SCIENCE

in the

GRADUATE DIVISION

(San Francisco)

of the

UNIVERSITY OF CALIFORNIA

Approved: . . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . Committee in Charge

Deposited in the Library, San Francisco Medical Center:

. . . . . .

Date Librarian

Degree Conferred: SEP 22 1976

METHODS FOR THE DESIGN OF MEDICAL DATA BASE SYSTEMS

Gio C. M. Wiederhold

ACKNOWLEDGEMENTS

This research was done within the Program of Medical Information Science at the University of California at San Francisco. Throughout my studies I received valuable support from Dr. John A. Starkweather, Dr. Ronald R. Henley, and Dr. Marsden S. Blois. Dr. John Dervin from the Family Practice Center in Santa Rosa and Dr. Edward Feigenbaum from Stanford University contributed their special knowledge to my thesis committee. The faculty and staff of the MIS program provided essential advice and assistance.

I was fortunate in having had the opportunity to analyze the problems in automated ambulatory medical record systems with the help of Dr. Ronald Henley, Dr. John Dervin, Dr. Michael Jenkin, Ms. Ingeborg Kuhn, Dr. Emmanual Mesel, Dr. Diane Ramsey-Klee, and Dr. Jonathan Rodnick. This study was supported by the National Center for Health Services Research under the guidance of Mr. William Anthony. Mr. Gerald F. Miller, under support of a Manpower Training Grant, spent more than a year analyzing, collecting, and categorizing data from the Family Practice Center in Santa Rosa, and without his help the methodology advocated in this thesis could not have been applied.

A textbook, which provides much of the technical background for this thesis, is now being published, and for this work I owe recognition to many colleagues. I was fortunate that during my studies I was able to teach the subject of data base design both at UC, San Francisco and at UC, Berkeley and obtain the feedback necessary to develop the material for this thesis. A fellowship from the National Library of Medicine provided the opportunity to complete the methodology and write this thesis.

My wife, Voy, helped me throughout these projects, with all the tasks required and desired, from typing and editing to admonishing and budgeting. Without her support most of this work would not have been attempted and nothing would have been completed.

ABSTRACT

A methodology for the design of data bases is presented and applied to a medical record system. Many failures of data base systems in medicine have been caused by excessive concern about functional capability and insufficient consideration of quantitative performance factors. This thesis advances an approach where the quantitative requirements of the applications for the data base govern the design decisions. This methodology consists of two phases: the design of the data base structure and the analysis of implementation alternatives.

A number of relation types are defined which allow the description of the data requirements of each of the multiple uses to be made of a data base. Rules which govern the decomposition and synthesis of these relations are used to integrate the data requirements of multiple applications into a unified data base. The frequency of use of the applications provides criteria to trade retrieval performance versus redundancy and update complexity. When the desired data base structure has been formulated, a second phase of the design methodology evaluates the system performance using six basic file organization methods, or hybrids of these methods. Previous work by many researchers is placed into a consistent framework to allow a consistent analysis of file organization alternatives. By considering the relations in order according to type and magnitude of expected load, the design choices can be made expeditiously. Refinement of the design is simplified since the design methodology documents the origin of the features of the data base structure.

The use of the methodology to design a proposed ambulatory medical record system for a family practice center demonstrates that design choices can make a critical difference: A data base system design which satisfies the requirements is constructed, and this design uses only few more resources than a minimal storage system which cannot perform the services adequately: This illustration shows that this quantitative approach to data base design is feasible, and that a quantitative design procedure can be used to reject unsatisfactory approaches without the cost of experimentation, or worse, failure after implementation:

TABLE OF CONTENTS

INTRODUCTION 1

CHAPTER 1 - DATA BASES AND THEIR PROBLEMS 10

Prior Work 11
A Model of the System Development Process 14
Functional Capability of a System 16
Quantification of a System 18

CHAPTER 2 - A METHODOLOGY FOR DATA BASE AND FILE DESIGN 22

The Manipulation of Data Models and Data Base Models 22
Properties of the Five Relation Types 29
Data Base Model 30
Description of Relations 32
File Design 33
Hybrid File Organization 44
Load Considerations 46
Cost-Benefit Considerations 47

CHAPTER 3 - AN APPLICATION OF DATA BASE PRINCIPLES 48

Files for the Family System 48
Data Base Structure 52
First-Order Normalization 56
Lexicons 59
Referenced Entity Files 59
Service Files 60
File Minimization 61
Data Models 64
Data Base Model 68
Summary 71

CHAPTER 4 - PERFORMANCE 73

Initial Performance Design Considerations 76
Usage and Load 77
Storage of Nominal Data 87
Choice of File Organization 90
Ring or Hierarchical Files 103
Other Files 119
Service Files 125
Summary 125

CHAPTER 5 - CONCLUSION 127

Unresolved Issues 128

RELATED WORK 131

BIBLIOGRAPHY 132

METHODS FOR THE DESIGN OF MEDICAL DATA BASE SYSTEMS

Gio C. M. Wiederhold

INTRODUCTION

The acquisition, storage, and analysis of large quantities of data is a principal application of computer systems. In medicine large collections of patient data are an important tool in study of disease and effects of treatment, and it is expected that in the delivery of health care such automated record systems will have an important function in the management of multi-modal health care and the selection of health service priorities. A significant amount of effort has already been expended in the development of such medical record data banks [2], and increased funding in this area is advocated [FPFH75,pp.188-199].

However, only a few such data banks are today in routine operation, so that much of the effort has been wasted. One reason for the failures has been the expectation that everything a computer can do is done better by computer. How well a computer can do a complex set of tasks has not been evaluated prior to implementation, and could only be guessed by experts using their intuition.

This dissertation presents a new methodology, which permits the formal application of knowledge about computer file system behavior to the problems of the design of complex data bases. The application of the methodology guides the design process to an optimal file allocation and provides quantitative predictions of system behavior. The acceptability of the performance of the

proposed system can now be assessed prior to implementation. The value of desired functional capabilities can be traded against performance margins. This work, hence, addresses a problem area which is central to the application of computers in large scale practical problems.

Problems have been encountered in the use of computers since their inception. While initially (1950-1956) the reliability of hardware was the most critical aspect of computer operations, the development of languages for programming was the concern during the next period (1956-1962). The development of program libraries and operating systems characterizes a third phase (1962-1968). Only as these components of computer systems became adequately understood did it become feasible to process large volumes of data. The concurrent introduction of low cost direct-access-storage devices has increased the expectations of system designers and promotors, and large information management systems were and are being developed. Users of computer systems are now faced with yet another source of computer system problems.

The integration of diverse files into a data base increases the potential for the generation of new information. This capability was expected to cause a major revolution in the operations of commercial, industrial, medical, and academic enterprises [Kochen67]. Applications of computer technology which use complex files and data base systems which integrate these file have been a major source of frustration and dissatisfaction [Ackoff67, Hoos71, Fichten72, Lucas75]. The problems described have included

unrealistically long lead times,
unfullfiled expectations, and
problems in long-term operations

A survey of commercial data base systems [Datapro75] reveals that there are about 1000 non-trivial data banks in operation supported by commercial systems; there are probably at least that many which are supported by various homemade systems; Steel74 has estimated that there will be 25,000 data base system applications operational by 1985; Not all commercial systems manage to attract or keep customers; For instance, of seven data base systems surveyed in 1969 [Byrnes69], only four remained available in 1972 [Steig72]; and only two of those were in regular use in 1975 [Tanner75]; Steel74 stresses the immense loss when data banks do not achieve the level of operation commensurate with the investment made in development and data collection;

In the area of Ambulatory Medical Record (AMR) data banks, nearly 200 systems or system proposals have been identified [2:AppA], but on further inspection only about two dozen of these had become and remained operational; Even among the 17 sites selected, because of their excellence as study subjects, several have since discontinued use of the AMR system seen; The financial loss due to discontinued systems has been high; The development of the Kaiser-Permanente System [Davis70] involved several million dollars; Even small institutions have expended significant amounts; for instance, a local four man speciality group practice has invested $70,000 in hardware and $50,000 in programming without ever achieving any operational benefits [Rydell76]; A serious corollary is the loss of enthusiasm and trust of the organizations and individuals who sponsor the projects, an intangible cost which can never be replaced; Specific problems

that hindered acceptance were

Higher than expected operational cost [2:vol;2,CDA,pg;28&CDH,pg;32]

Additional investment requirements to achieve a satisfactory operational level [2:pg;129]

Incomplete services which severely limited the potential system benefits [2:vol;2,CDY,pg;22]

Inadequate performance [2:vol;2,CDH,pg;31]

At the same time there is a considerable pressure to extend the record keeping functions in medicine, which prompts regularly new attempts to automate the ambulatory medical record; Evidence of this fact is the extremely high demand to furnish copies of the report on the "Analysis of Automated Ambulatory Medical Record Systems (AAMRS)" [2];

A major problem faced in the design and implementation of medical data banks is the choice of the data elements to be collected and their organization; An increasingly wide variety of techniques is available to collect, store, or retrieve data; More recently a variety of conceptually differing data base structures have been put forward as solutions to the problems of storage and retrieval management; Rapid development of computer hardware changes the economics of data systems; On-line access to ever larger data banks becomes feasible and the number of design choices available to the developer increases; Selection of the appropriate data organization is an important factor in the eventual success or failure of data banks, but few guidelines are currently available to assist the developer in his choice;

Although many potentially useful results have been obtained in the area of the file and data base system design, the information relevant to design is scattered, is based on inconsistent assumptions, and uses isolated models. The interaction between the data base structure and the file organization is rarely made obvious. The actual practice of data system design depends on the availability of experience and good judgement derived from experience. Unfortunately, the design, implementation, and establishment of a reliable data base system requires many years, so that the experience is not easily gathered. When such experience has been gathered, technological or individual advances can make re-application of the experience difficult.

Many aspects of the design and operation of computer-based information systems warrant analysis and development in order to improve their success rate and effectiveness. This dissertation will address one specific problem area: the design of applied data base systems. A new methodology for the design of computer data bank support systems will be presented. The framework is broad and encompasses choices of data base structure, file implementation, and hardware support. In this thesis an engineering approach is developed and demonstrated which allows the recognition and elimination of aspects of the design problem which will have a negligible effect on the productivity of the system being designed. The intent of the methodology is to allow decision-making during the design of data bank systems to be based, as much as possible, on objective factors and thus to lessen the need for comprehensive, well assimilated, and recent experience.

The approach taken consists of the definition of data models which allow the description of the relationships among structured data, rules for the creation of a data base model which combines overlapping data models, and a unified evaluation of file

structures which can be used to implement the data base model. Given as input the specification of the expected applications of the data bank, the methodology provides the basis for decision-making during the design process, and eventually a prediction of system adequacy and performance.

The importance of a methodological framework to cope with the management of tasks generated by advances in technology may not be immediately obvious. The realization of the need depends on the abstraction of the problems encountered in the practice of system design. A somewhat far-fetched analog is the Problem-Oriented Medical Record organization as advocated by Weed [Weed69]. No new basic science is part of the POMR, but application of its concepts provide guidelines and confidence that medical information is handled in a responsible manner. Use of such a methodology will not change the behavior of experts in their field, but will provide them with a means to share their knowledge. Neither doe the application of the methodology make the design process "idiot-proof"; care and insight remain valuable attributes of system designers. Use of the methodology will, however, improve the level of design performance. The use of quantitative design goals and the generation of explainable and repeatable results can provide the focus of user involvement with the system design process.

Research Related to This Dissertation:

The requirements, development, and demonstration of the methodology which is advanced in this thesis has drawn on three significant efforts, in which this author was involved in the years preceding this work. The planning, implementation, and operation of ambulatory medical information systems have been surveyed by Henley, Wiederhold, Dervin, Jenkin, Kuhn, Mesel,

Ramsey-Klee, and Rodnick in AN ANALYSIS OF AUTOMATED AMBULATORY MEDICAL RECORD SYSTEMS [2]; A number of the problems seen in the systems encountered were traced to inadequacies in the design process; It is a significant measure of the extent of problems found that the survey located more than 200 systems or system development efforts; of which only about two dozen had reached operational stability; Even that stability was tenuous since of 17 sites visited, three have ceased operation since then (Yale University; the user of the Insurance Technology System site-visited in [2]; and NAS Brunswick); All of the systems visited had apparently been implemented without a significant quantitative design analysis; Inadequate system effectiveness was seen in the majority (10 out of 17) of the systems [2:pg;142]; In many instances the level of effectiveness could have been predicted from pre-implementation system analysis and might have been overcome by alternate design choices;

The knowledge about the design of data base systems has been summarized by Wiederhold in DATA BASE DESIGN; (in press) McGraw-Hill Book Company; 1977 [1]; (Manuscript available at the Section of Medical Information Science; UCSF); This volume contains a systematized collection of formulas which allow calculation of the performance of file organizations and data base systems; The approach used allows the resolution of design decision problems with sufficient accuracy for pre-implementation design; while avoiding the exponentially increasing complexity of pure mathematical methods; Update as well as retrieval is considered when files are evaluated; A data base model which retains semantic data relationships; so that the performance of intended services can be modelled; is included; The design process requires as input a description of the data requirements of the applications which are being contemplated;

Miller; in THE FAMILY SYSTEM (Lovelace Computing; 1976) [3]; describes in great detail the requirements for the operation of an Automated Ambulatory Medical Record at the Family Practice Clinic of the County of Sonoma in Santa Rosa; This work provides the major part of the input specifications to the design example which is used in this thesis to illustrate the proposed methodology for the design of medical data bases;

Organization of this Dissertation:

This thesis advances and illustrates a methodology for the design of medical data bases; The work begins with a review of the current status and prior work in relevant areas; A gap which exists between the two levels of data systems abstraction, the file structure and the data base structure, is evident; A model for the design and implementation process is presented which illustrates the potential for failure in system design; A unifying hypothesis regarding the causes of the problems seen can then be brought forth; The observations are documented by the findings which are detailed in [2];

A design methodology which is designed to avoid the problems defined by the hypothesis is then presented; A comprehensive quantitative approach forms the foundation for this method; The detailed derivation of the factors required to use the proposed framework was developed by this author in [1]; and these results are utilized within this thesis; A summary of the results is included in this thesis to show the completeness of the basis for the application of the data base model which is an intermediary result in the proposed methodology;

To demonstrate the methodology, a proposed Automated Ambulatory Medical Record System for the Family Practice Center at the Sonoma County Community Hospital in Santa Rosa, California is used. The applications for the Family System as described in [3] are taken, their data requirements are reduced to their essentials, and the data structure is presented as a consistent data base model. This model is then evaluated for optimal implementation. A summary and directions for further work to support this area of technology transfer to medicine conclude this dissertation.

CHAPTER 1
DATA BASES AND THEIR PROBLEMS

"MIS* is a mirage."
J. Dearden, 1972

An essential resource for an information system is a data base, that is, an organized collection of data. A data base may be implemented using a general data base management system, or may be structured specific to the application. In either approach the data will be stored in one or more files; the assignment of data items to files is a major part of the data base design problem.

The process of design is the choice of implementation alternatives, so that in order to design a system the designer needs a knowledge of the possible alternatives and rules to choose among them. The rules are controlled by multiple objectives: the chosen alternative has to be functionally adequate to the task, and when multiple alternatives are adequate, then reliability, performance, and cost become the dominant criteria. Aspects of this process in data base design have received regular attention in the literature, and the field is attracting the attention of many researchers.

* Management Information Systems

Prior Work:

Previous work in this area can be placed into two distinct categories: file design and data base structure selection. This division is natural and valid, but the two areas interact strongly in the process of the design of a complete system. The need to compare file organization methods in common terms was first voiced by Bachman66. Chapin69A and F described the sequential file, the indexed-sequential file, the direct file, and a hierarchical file organization using a common terminology. Senko68 and Lum70 have developed simulation programs for the access methods which were then available at IBM, so that design alternatives within these constraints could be quantitatively evaluated. Comparisons were presented by Brewer68, Angell69, McGee69, Bloom69, and Behymer74 for the indexed-sequential and direct file organizations; Collmeyer70 analyzed retrieval for single and multi-level indexed files and for the direct file organization. Shemer71 and London72 compare indexed-sequential, indexed, and direct access file organizations. These comparisons are mainly expository in nature, and while they provide an understanding of the alternatives, they do not give a quantitative performance predication useful for pre-implementation.

Quantitative analyses have been performed on many particular implementations and proposals. These are in general particular to some operating environment, and their aggregate does not present a suitable basis for objective design decisions when the environment is different. References to a number of such analyses published in the literature may be found in [1]; many more have been made by enterprises during the process of system development.

Hsiao70 developed a formal notation for the description of alternate file organization methods and applied this to indexed, indexed-sequential, and a highly redundant multi-list file organizations. Retrieval algorithms were applied using this notation, so that their performance could be evaluated in terms of an abstract model. Severance74 evaluated sequential, indexed, and direct storage organization, but assumed storage devices with homogenous access characteristics, i.e. core storage. The heterogenity of access is stressed in Merten70, and the algorithms analyzed there recognize the distinction between a fetch of a record using an arbitrary key and the getting of a record which is next according to some defined ordering. The use of explicit and precise access formulations limits the analysis which is presented to relatively small files, since the number of combinatorial choices of record placement grows exponentially. Yao74 presents the first study which allows the modelling of most file design alternatives, and his model also considers the cost of file updating. Much of the previous and other current work is limited in that it attempts to optimize solely retrieval performance. Given no constraints on space or complexity, very fast retrieval can be obtained by the use of redundant data or multi-path access structures [Chiang73]. Yao's thesis also describes an automatic optimization process for indexed file structures.

Parallel to these developments, but largely independently, the structural relationships in a data base containing multiple files were being scrutinized. A notation for a network of interrelated files was developed to describe multi-ring files, and this notation has been presented in Bachman72 to describe relationships among files in general. Codd70 and Codd72A use a set-theoretic approach to give structure to files and define operations to exploit inter-file relationships in data retrieval. These two views, termed "network" and "relational", were further developed

independently. Commercial systems based on Bachman's concepts have been put into operation [Jardine74], and their structure has been analyzed by Taylor71 and Nijssen75.

The mathematical nature of Codd's relational model engendered quick acceptance in the academic world and it has been described and analyzed widely. Of major interest here are Heath71, Date71, Delobel71 and 73, Armstrong74, Wang75, and Forsyth75.

These two approaches have been compared, and transformations from the network model to the relational model and vice versa have been presented by Bernstein75, Cook75, Kay75, Tsichritzis75, Stonebraker75, Zimmerman75, Date76, Nijssen76, and Senko76. The problem faced by these efforts is that the comparisons have to be made between concepts which have a different origin. The network models are abstractions of data base implementations whereas the relational models are procedural descriptions of mathematical concepts.

The medical information science community has begun to take note of these issues; the network approach has been analyzed by Baker72, whereas Chang75, Manacher75, and Wasserman75B propose relational approaches. Codd75 cites a relational application in a hospital setting. Classical approaches [Greenes69 and Davis70] use hierarchical trees, which can be described as a substructure of the network model. Wiederhold75 describes a clinical information system in terms of relations which support a conceptual two-level hierarchy.

The System Development Process:

In order to explain the problems and failures of data base system which were perceived above, a simple model of the system development process will be presented; Then a hypothesis can be stated and applied to the model; It is hoped that this hypothesis will provide a satisfactory explanation for some of the problems presented above;

A MODEL OF THE SYSTEM DEVELOPMENT PROCESS

In order to measure success or failure in an ongoing process, the sequence of steps that make up the process must be understood; This holds true whether a process performed by a computer program is to be analyzed or a developmental process performed by an institution is to be debugged; Such a model was used to analyze AAMRS's [2:pp;30-40] and has provided a framework for the survey; The model used below will stress the computer system development aspects;

The System Development Process:

1 CONCEPTION
   A need for information is recognized, or benefits from information availability are demonstrated at a similar institution;

2 DEFINITION
   The reports or outputs needed to convey the information are defined;

3 REQUIREMENTS

The data and the computation required to generate the outputs are defined;

4 SOURCES

The source of the data to provide input is identified;

5 FEASIBILITY

Advice is obtained on whether a computer can transform the input into output;

6 CONSTRAINTS

The means and constraints for implementation are surveyed; Resources at an institution will include financial capability, personnel, and available computers;

7 ESTIMATION

A computer expert is called in to provide an estimate of cost and time required for implementation; The functions to be performed are explained to the expert, as well as the constraints at the institution; Experience is used to provide an estimate; Depending on the degree of expertise and the desired confidence factor, the estimated costs are doubled or tripled;

8 FILE DEFINITION

Computer files are defined that contain the data required for the functions to be performed;

9 IMPLEMENTATION

Implementation is begun using the best available resources;

The hypothesis, which will be brought forward to explain weaknesses in this process, stated concisely is:

> The decisions to proceed from step to step are made on a qualitative basis rather than on a quantitative basis;

This statement can be reworded in many ways; Another expression of the alternative issues is:

> The emphasis is on function, not on effectiveness;

It is true that if the function cannot be performed, the effectiveness of its execution becomes meaningless, but ineffective execution of valid functions is wasteful and frustrating;

Functional Capability of a System:

In the initial phases of development of a technology, the capability to provide needed functions has to be established; It had to be proven that airplanes were feasible before airlines could become valid enterprises; Not all technically possible functions lead to valid enterprises; balooning, for instance, has never taken off in a commercial sense;

In applications of data bases the required building blocks include computers, languages to program them, file systems to store and retrieve data, and data base systems to organize and interrelate the files; The function of these elements has been thoroughly analyzed;

In computing the problem of function has been formally treated by Turing [Hopcroft69], and it can be shown that any computer system available today has, for finite problems, the power of the hypothetical Turing machine and hence is functionally adequate to solve any computing problem.

On the language level it has been shown that while context-free languages have limitations in the description of semantically complex processes, these limitations do not apply to context-sensitive languages. Since for pragmatic data base problems no necessity exists to constrain the choice of language, there is again no fundamental functional restriction of language which could inhibit the capability of a computer system program to describe the desired operations.

The function of a file is to store and retrieve data, and failure to retrieve stored data is indicative of an error.

At the data base system level Codd [Codd72B] has argued for functional completeness in data base systems and shown that his relational approach fulfills the requirements. Since then translation rules between relational data base models and hierarchical and network models have been defined [Bernstein75; Cook75; Kay75; Stonebraker75; Tsichritzis75; Zimmerman75; Chen76; Date76; and Senko76]. It is clear that any data base system which has the capability to execute arbitrary procedures is functionally complete, although there do exist information retrieval systems which neither have this capability nor have adequate built-in data manipulation capabilities.

The issue of functional capability is hence essentially moot, the fact that data base problems do not lie in the area of language

or function was already expressed by Schwartz72; This was also observed during the AAMRS site visits; Whenever the issue of a function which seemed desirable but was not available was raised, an answer invariably as: "This function could be provided with a little programming effort" was given;

Quantification of a System:

More data and analysis is required in order to determine how well a functional capability can be implemented; During the AAMRS study, the quantitative questions of system performance were difficult to survey; The designers felt, in general, that their systems provided fast response; although the actual response times varied a great deal; In some systems, for instance; the time required to retrieve additional data from an on-line medical record during a patient visit was so slow that the facility was rarely used [2:vol;2;CDH;pg;23]; In other instances the rapid initiation of printing was cited; the fact that it took several minutes to print the required record abstract was not considered to be a problem [2:vol;2;CDN;pg;7];

During the planning stage of the systems surveyed, questions of cost effectiveness had rarely been asked [2:pg;204], and even in operational system few attempts were made to quantify the benefits obtained and costs incurred [2:pg;182-188];

The hypothesis that function, not effectiveness, is the controlling factor in the decision making process can be applied to all the steps of the model described above;

During the first step; the conception; a functional approach will depend on the citation of examples of the impact of the availability of information on a patient's treatment; To obtain a quantification; an assessment of the effect of the information on the entire patient population is needed;

During the second step, the definition of the system, a functional approach will prepare sample reports or output formats to define the required data; In order to obtain a basis of load estimation for the proposed system, the required output is generalized and the volume is estimated; To exclude repetitive normal results the range of parameters for which output is to be produced is specified; Where volume is excessive, criteria for exception reports or warning messages are defined;

This comparison can be continued through the statement of system requirements, the identification of the source to the evaluation of system feasibility; In this fifth step a quantitative methodology becomes especially important; Nearly always when a sample task is presented to a technical expert for an evaluation of procedural feasibility, the answer will be "yes"; The fact that the effort involved may greatly exceed any possible benefit is not answerable without a broad-based quantitative background of data availability and quality;

The effects of the imposition of institutional constraints, in terms of scope of system; limitations in data collection or information usage; are also difficult to assess, especially in terms of long range system viability, unless these effects are quantified;

During the seventh step; the generation of an implementation estimate, a consistent methodology becomes essential; While

expert estimates are often based on experience with previous systems, it is rare that the past circumstances are sufficiently similar to permit an accurate assessment. The availability of a quantitative formal methodology also allows the investigation of alternatives of load and design, and study of the sensitivity to load variations.

The remaining two steps, file definition and implementation, which in a functional approach are developed by programming staff as the services are being designed, proceed under rigid guidelines, obtained during the development of quantitative measures, when the formal design methodology is used.

The list given above emphasizes the weakness of the functional approach to the extreme. For most systems which have been planned, some areas have been studied quantitatively. Studies of isolated topics, however, tend to suffer from inadequate input data, although observations from predecessor or similar systems may be used.

It may seem surprising that many system are designed and implemented without quantitative analysis covering all of the nine steps listed; although these steps, which when presented methodically as above, are without doubt important factors in the eventual success of a system. The examples and references cited earlier, as well as the experience recounted by other workers in this field, convince me that the problems listed are real.

The fact that we all tend to acquire new knowledge by example may be one of the reasons for the lack of comprehensive treatment of the system during the steps of the design process. Since several years are required to design, implement, and prove non-trivial systems, it is difficult to gain a broad range of experience through experimentation. The fact that even in commercial system design important factors are often forgotten until it is too late has led the two data processing institutions to prepare checklists for systems implementation: AFIPS [Patrick74] and CODASYL [CODASYL76]. In the area of medical record systems a list of "Ten Commandments" [Barnett71] is intended to prevent some of the failures perceived by one of the main promulgators of computers in medicine.

Checklists, however, can only help in assuring that all points are covered. An extensive checklist can in fact be extremely frustrating because it does not tell how all the facts and details can be brought together into a meaningful design process. A framework for the design of data base systems which is intended to help in overcoming these problems is presented in the next chapter.

CHAPTER 2
A METHODOLOGY FOR DATA BASE AND FILE DESIGN

"In der Kunst is nur das Beste gut genug"
W. J. Goethe, 1787

In order to achieve the goal of reliable prediction of performance of a data base oriented system, a framework for the design process has to be established. This framework has to be sufficiently powerful to allow the establishment of quantitative performance measures in addition to proving the functional adequacy of the proposed system. Since the form of the data base and its performance are interrelated, it is necessary to be able to manipulate the data base being designed. The methodology which has been developed [1] contains both a framework which allows the manipulation of data bases as well as a framework for the quantitative evaluation of file organization methods which can be used to implement such data bases. The use of such frameworks can help to transform the data base design process from an art to a science.

THE MANIPULATION OF DATA MODELS AND DATA BASE MODELS

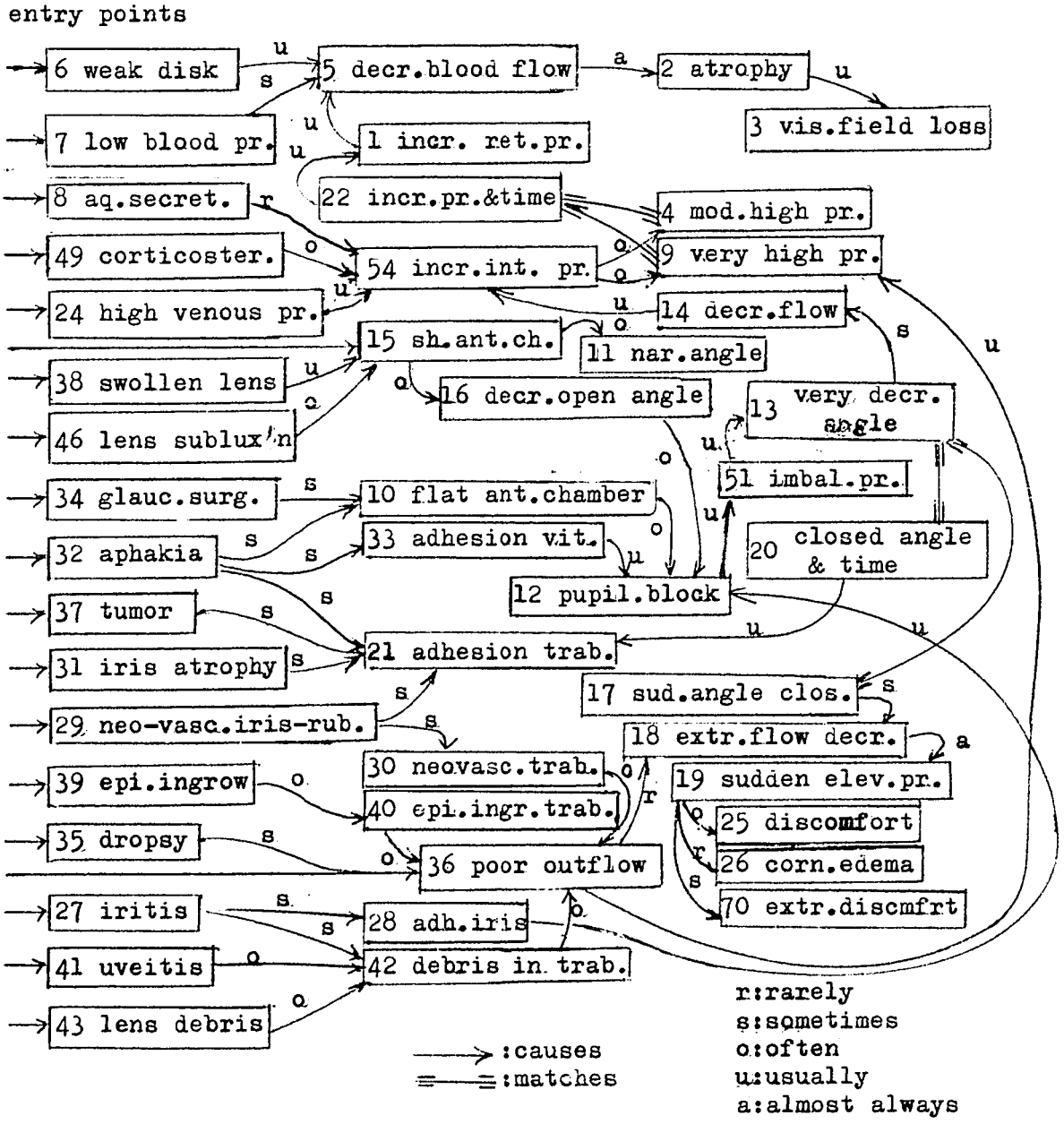
Various models have been used in recent years to gain a better understanding of the structure of the data base problem. The models represent a wide range of abstraction.

Relational Models:

The initial point for the formalization of the design process are data models [1:Ch:7:1]. A data model is a description of the data as seen from the point of view of one service or one data collection point. In order to describe the initial state, an unnormalized relational format [Codd70] can be used. Codd's relational model prescribes a normalization or decomposition process which is wholly mathematical in nature. A mathematically based representation of data models provides complete independence of usage patterns, and hence enhances the functional power at the expense of performance oriented criteria. For a quantitative analysis it is important to retain semantic relationships which are part of a data model, since these will later prescribe access paths used by the services. The application of the relational model to clinical data bases has been discussed by Chang75, Manacher75, and Wasserman75B. Manacher shows some of the semantics which exist between relations in a clinical data base.

Semantic Nets:

Very fine semantic structures have been used in data bases used in Artificial Intelligence work [Levien67], and models to support such data bases [Abrial74] have been described. These semantic nets represent problem-specific data in a compact and essentially non-redundant form. A graphic presentation of a data base developed for decision-making in the area of glaucoma [Weiss74] is shown in Figure 2-1. Many semantically differing relationships use similar structures, but a new model is required whenever there are application differences.



A partial model of glaucoma (from Weiss74)

A Causal Network

Figure 2-1

Data Base Models for System Implementation:

For the purpose of system design a balance has to be struck:

1) The model has to retain sufficient meaning so that the intent and uses of the eventual applications, as they affect the eventual file design, can be retained;

2) The model should not be so fine that application details which do not affect the file design are represented;

The relation types which have been developed in [1:Ch.7.2] to achieve this objective are five:

1) An Entity Relation is the primary and independent description of a set of data objects. There will be one entry or tuple for each data object.

   An example of an Entity relation in an AAMRS is the Patient file.

2) A Referenced Entity Relation is the description of an attribute-complex common to multiple tuples in one or more relations. While the data values of referenced entity tuples belong wholly to the owning Entity relation, the creation of Referenced Entity relation reduces redundancy and defines common linkages. Referenced Entity tuples will also obey certain maintenance rules which are specific to this relation type.

   An example of a Referenced Entity relation is the description of Problems presented by the patients. Multiple patients will present the same problem, and hence reference the same tuple of the Referenced Entity relation.

3) A Lexicon is a collection of referenced tuples which have only two attributes, where these attributes have a symmetric one-to-one relation. Lexicons provide linkage mechanisms among Entity or Referenced Entity relations which are identical in content but are viewed according to different data models.

   An example of a Lexicon in an AAMRS is the set of (patientnames, patientnumbers).

4) A Nest Relation collects data subsets which are specific to one Entity tuple. Whenever a data attribute or group of attributes can occur multiple times within a tuple, the subunits have to be identified individually. Such data are managed best by collecting all repeating subgroups into a subsidiary hierarchical level, as implemented by the Nest relation type.

   An example of a Nest is the set of visits made by a patient to a clinic. A Nest relation will collect all visits by all patients.

5) An Associative Relation collects data entries belonging to the intersection of multiple Entity relations. Here each tuple depends on more than one owning tuple for its existence.

   An example from the medical records area could be a disease registry, where each entry would be owned by a patient and a disease tuple. An inventory of a ward-oriented hospital pharmacy would form an association of drug and ward entries.

Normalization and Relation Types:

Codd, in the process of defining relations in a manner which allows that the properties of relational models can be rigorously proven, has established a number of normalization processes. These processes generate relations with set-theoretic properties, but these relations can also be identified with the structural semantic classes defined above. Some examples will show the semantics associated with Codd's transformations.

A transformation of unnormalized relations to relations in first-normal-form [Codd70] yields nest-relations. Another means of coping with nests is possible if a limit can be assigned to the number of repetitions. The nest can then be treated as a finite sequence of attributes within the owning tuple. Such a denested sequence contains undefined attribute values wherever the repetition count is less than the limit. Both choices are shown in Figure 2-2.

A transformation to second-normal-form [Codd72B] can create Lexicons or Referenced Entity relations. It can also reduce an Associative Relation into its parent Entity relation and a remainder Associative Relation, which may be empty.

Transformations to the final stage, third- or Codd-normal-form [also Codd72A] create additional Referenced Entity relations. Tuples of relations in Codd-normal-form are unique, and data are stored with minimal redundancy.

DATA

employee	age	children	spouse	experience	supervises
Hare	34	Mary, 16	Linda	9	Joe, 2
		Paul, 13			Mike, 3

PROGRAMMING DESCRIPTION

```
DECLARE 1 employee,
          2 name CHARACTER VARYING,
          2 age DECIMAL,
          2 children(no_of_children),
             3 name CHARACTER VARYING,
             3 age DECIMAL,
          2 spouse CHARACTER VARYING,
          2 experience DECIMAL,
          2 supervises(no_of_supervisees),
             3 name CHARACTER VARYING,
             3 years_supervised DECIMAL;
```

FIRST NORMAL FORM RELATION
achieved by denesting children and by removal of supervisees to a Nest Relation

employee:RELATION

name	age	child1	age1	child2	age2	child3	age3	...	spouse
Hare	34	Paul	13	Mary	16	null	null		Linda

...

supervision:RELATION

super	experience	subordinate	years_supervised
Hare	9	Mike	2
Hare	9	Joe	3

...

Relation in First Normal Form

Figure 2-2

PROPERTIES OF THE FIVE RELATION TYPES

The tuples of all these five relation types obey well-defined rules, which augment those established by Codd. Each tuple of a relation is identified by one unique ruling part. Other potential ruling parts are supplied through lexicons.

Ruling Part Definitions:

The ruling part of a Nest relation is a catenation of the parent ruling part and a local attribute to achieve the required uniqueness of the ruling part.

The ruling part of an Associative Relation is the catenation of the ruling parts of two or more Entity relations. Both parts may be from the same owner relation: i.e. a relation "marriage" which associates two tuples from a "population" relation. Ruling parts of Referenced Entity relations are always non-ruling parts (dependent parts) of other relations.

Potentiality for Sequencing:

A definition of a collating order on the ruling part establishes a single logically serial order for the tuples of a relation. Such a serial order can be exploited in the manipulation of the relation since it can define access to tuples without knowledge of the ruling part value through the use of relative identifiers as NEXT, PRIOR, FIRST, nTH, or LAST.

Reference Rule:

No tuples may be removed from a Referenced Entity relation or a Lexicon while any reference exists. Tuples in these relations can, however, exist without active references.

Nest Rule:

Tuples in a Nest relation have to be removed when the parent tuple is removed.

Association Rule:

Tuples in an Associative relation have to be removed when any of the owning tuples is removed.

DATA BASE MODEL

The relation types described above are defined by the analysis of a specific data model, which pertains to one service and expresses one view of the data base. When data are to be shared among multiple data models, then a data base model has to be created. A data base model has to be able to express multiple views and support multiple services.

Transformations:

The relation model specifies a number of transformations which can be employed in order to combine relations or create new relations for others. These operations include the conventional set operations:

Union
Intersection
Difference

and are augmented by operations which are specific to relations

Join,
Projection, and
Division.

These operations are defined in [1:Ch.7.3] following conventional standards. The set transformation Union is used to combine similar relation types and extend the range (the entries in the ruling part) or the scope (the number of attributes). The Join operation can associate dissimilar relations, as controlled by a relationship among a common attribute. The Projection operation can extract Lexicons or Referenced Entity relations, whereas Division can develop the parent relation of a Nest relation.

A Join operation can create a new associative relation out of Entity relations, but requires an algorithmic specification of the condition for catenation of tuple pairs. The simplest condition is given by an Equi-join, that is a Join with the condition that the values of the attributes used to combine tuples are equal. In any case, the associative relation stores some information which is not contained in the owning relations.

Validation of the Data Base Model Framework:

The model has been applied to a large variety of data base support systems and can easily explain the relationships which are supported by these systems [1:Ch;9]. In Chapter 3 of this thesis a specific AAMRS application is investigated and the transformations which have been developed are applied to integrate the files which were conceived for individual services and their data models.

DESCRIPTION OF RELATIONS

In order to manipulate relations, attributes have to be named. With these names, a number of descriptive characteristics may be associated. Important characteristics of an attribute are the domain, the encoding, and the privacy constraints. The domain describes the range of values allowable [1:Ch;8;2], the encoding the representation of the data [1:Ch;14;1,14;2], and privacy constraints describe ownership and potential readership of the data in the data base [1:Ch;12]. Other characteristics of attributes are detailed in [1:Ch;8].

A collection of descriptions of attribute characteristics is termed a "schema". A schema provides a powerful means of communication between the user of a data base and the implementor [Wiederhold75].

The schema of the final data base model can also provide the linkage between the data base and the application programs which carry out the computations on the stored data. The use of schemas in data base systems in general and AAMRS's in particular [2:pg;238-241] is becoming increasingly frequent.

The utility of schemas as a repository of the description of the data base extends over many phases of the design process:

Communication among the affected groups of individuals during the data base definition process is provided by the formal definition of the attributes and their characteristics;

Identical attributes in distinct relations indicate relationships which may be exploited in the construction of the data model;

When the values of the attributes have identical domains, then application programs can exploit these relationships;

FILE DESIGN

When a satisfactory data base model has been established, then it becomes possible to consider implementation of a data base system to support this data base; Two approaches are possible here:

1) A commercial data base management system may be chosen to support the files and their interrelations;

2) A data base system may be written for the application which utilizes available file systems;

In either case the file organization choices have to be evaluated for adequate performance; If a data base system is chosen, then it is also necessary to verify that the interrelationships specified by the data base model can be supported;

In order to provide a model for file system evaluation, [1:Ch;3] defines six basic file organization types; These six types are selected from the very large number of possible choices by using several criteria; One objective of the selection is that each file organization type has distinctively different features and a different range of applicability; a second objective is that all basic file organization components are adequately covered; and the final selection criterium is that the methods chosen are closely representative of available commercial techniques so that their analysis can be validated; The six methods chosen are presented in order of increased binding, which implies in this context that the structural constraints, used to provide faster access for specific retrieval types, become more severe;

The six methods are defined as follows:

1; Pile File

An unordered collection of variable length data records;

2; Sequential File

An ordered collection of fixed length data records, and an unordered log file for additions;

3; Indexed-Sequential File

An ordered collection of fixed length records, an auxiliary structure to provide access to these records, according to the ordering attribute, and an ordered linkage structure to provide access to additions;

4; Multiply-Indexed File

A collection of records where access is provided according to multiple attributes using multiple index structures;

5; Direct File

A collection of fixed length records where the physical placement is determined by a single retrieval attribute;

6; Multi-Ring File

A collection of multiple record types where access is provided according to multiple attributes using multiple linkages;

For each of these file organization types performance parameters have to be established which provide the quantitative criteria for selection; These parameters will, of course, depend critically on the hardware devices selected for the system implementation; In order to select candidate parameters [1:Ch;2;1] surveys the various available hardware types;

Four basic parameters can be used to describe the behavior of all these devices [1:Ch;2;2], namely

1; Seek Time

The period to bring a physical device into position for reading;

2; Latency

The delay before the desired data are ready so that actual data transmission can occur;

3; Transfer Rate

The speed with which data can be transferred;

4; Bulk Transfer Rate

The speed with which a large volume of data can be transferred; This rate has to account for gaps and wasted areas in the recorded data space, as well as for any intermediate physical motion required by a device;

These parameters depend not only on the physical characteristics of the device; but also on the techniques used to store data on the hardware: Such factors are the blocking strategy [1:Ch:2:2:4], buffering [1:Ch:2:3:4], and block placement [1:Ch:2:3:5 and 2:3:6]: Some boundary conditions are not easily expressed in terms of these parameters [1:Ch:2:4], but the file design process can proceed as long as the validity of the parameters is verified for any uncommon devices or extreme condition: The most common devices seen in current data base systems: disks, drums, and tapes are relatively easily characterized by these four parameters:

Using these hardware parameters the parameters required for file organization performance can be developed: Two factors must be jointly minimized for the lowest cost file design:

Storage space required for the data:

Time to perform the functions required by the application:

The functions themselves can be decomposed into four primitives, each with its own characteristic time requirements:

Time required to fetch a record 'out of the blue', that is, by associative matching to a given search key:

Time required to get the next record according to a given ordering:

Time required to add or delete a record from the file:

Time required to update the contents of a record in the file:

Another two functions use the available primitives in a manner which is file organization dependent rather than application dependent, and these warrant hence a separate evaluation:

Time required to read the entire file;

Time required to reorganize the file;

Using the notation of Table 2-1 the performance of the six file organization types will be summarized below; Most of the performance formula have been obtained by inspection of the structure of the file and an understanding of the access process; The branching probabilities in the process have been based on a uniform distribution of record insertion in the files;

Table 2-1

Symbols Used in Performance Formulas

A	average space required for attribute name
a,	number of different attributes in a file
a	average number of attributes occurring in a record
B	blocksize
b	blockcount
C	Cost factors
c	computational overhead per record - used only where the effect may not be not negligible
D	space required for goal attribute
d	number of records that have been invalidated
F	subscript denoting a fetch for a specific record
I	subscript denoting insertion of a record
m	number of available slots for records
N	subscript denoting getting the next serial record
n	number of records in a file
o	number of records that overflow the primary file before reorganization
o'	current overflow
P	space required for a pointer
p	probability
r	rotational latency time
SI	storage space for index
s	average seek time
T	the time required for various operations
t,	transfer rate from a storage unit to processing memory
t'	bulk transfer rate
U	subscript denoting an update of a record
V	average space for value part of an attribute
X	subscript denoting reading the entire file
x	number of levels in an index structure
Y	subscript denoting a reorganization of a file
y	fanout ratio

The results can then be summarized as follows:

Pile File:

$$R = a'(A+V+2)$$

$$T_F = \frac{1}{2} n \frac{R}{t'}$$

$$T_N = T_F$$

$$T_I = s + 3r + B/t$$

$$T_U = T_F + 2r + T_I$$

$$T_X = 2\,T_F$$

$$T_Y = (n+o)\frac{R}{t'} + (n+o-d)\frac{R}{t'}$$

Sequential File:

$$R = aV$$

$$T_F = \log_2 (n \frac{R}{B})(s+r+B/t+c) + T_{Fo}$$ (Binary Search)

$$T_{Fo} = \frac{1}{2} o' \frac{R}{t'}$$

$$T_N = r + \frac{B}{t}$$

$$T_I = \frac{R}{B}(s+r+B/t) + \frac{T_Y}{o}$$

$$T_U = T_I$$

$$T_X = Sort(o) + (n+o)\frac{R}{t'}$$

$$T_Y = Sort(o) + 2(n+o)\frac{R}{t'}$$

Indexed-Sequential File:

$$R = a\,V + P$$

$$y = \left\lfloor \frac{B}{V+P} \right\rfloor$$

$$i_1 = \left\lceil \frac{n}{\lfloor B/R \rfloor} \right\rceil$$

$$SI = \left(\lceil i_1/y \rceil + \lceil \lceil i_1/y \rceil / y \rceil \right) B$$

$$R_{total} = R + \frac{o}{n} R + \frac{SI}{n}$$

$$Pov = Ptf = \sum_{i=1}^{o'} P1f^i = \frac{1-P1f^{o'+1}}{1-P1f} - 1 \qquad \text{[Knuth74]}$$

$$T_F = c + s + (2 + Pov(1+\frac{1}{2}Pov))(r+B/t)$$

$$T_N = (\frac{R}{B} + 2Pov(1 - \frac{R}{B}))(r + B/t)$$

$$T_I = T_F + 5r + B/t$$

$$T_U = 2T_F + 7r + B/t$$

$$T_X = (n + o')\,(\frac{R}{t'})$$

$$T_Y = n\,(\frac{R}{t'}) + o'\,(r+B/t) + (n+o'-d)\,(\frac{R}{t'} + \frac{SI}{t'})$$

Multiply Indexed File:

$$R_{total} = a' \ (A+2V+P+2)$$

$$x = \left\lceil \log_y \left(n \frac{a'}{a}\right) \right\rceil$$

$$T_F = (1+x)(s+r+B/t)$$

$$T_N = s+r+B/t$$

$$T_I = (1+a')(s+3r+B/t) + a' \frac{(V+P)}{B} (c+s+2r+2B/t)$$

$$T_U = T_F + 2r + 2a_U \ (s+3r+B/t)$$

$$T_X = n \frac{R}{B} \left(s+r+\frac{B}{t'}\right)$$

$$b_i = (n+o') \frac{a'}{a} \frac{V+P}{B} + a' \frac{SI}{B}$$

$$T_Y = 2 \ a \ b_i \ (s+r+B/t)$$

Direct File:

$$R = \frac{m+o}{n} \ a \ (V+P)$$

$$p = \frac{1}{2} \frac{n}{m} \qquad \text{(separate overflow area)}$$

$$p = \frac{1}{2} \frac{n}{m-n} \qquad \text{(open addressing, B=R)}$$

The value of p for bucketsizes $B/R > 1$ derived by Knuth73 is complex and hence is presented here as Figure 2-3.

$$T_F = c + s+r+B/t + p\ t_{overflow}$$

the values for $t_{overflow}$ are

$s+r+B/t$	for separate overflow areas
$r+B/t$	for separate overflow areas on the same cylinder
$2r+B/t$	for linear searching, sequential blocks
$2B/t$	for linear searching, alternate blocks;

and c depends on the complexity of the hashing algorithm and the bucketsize.

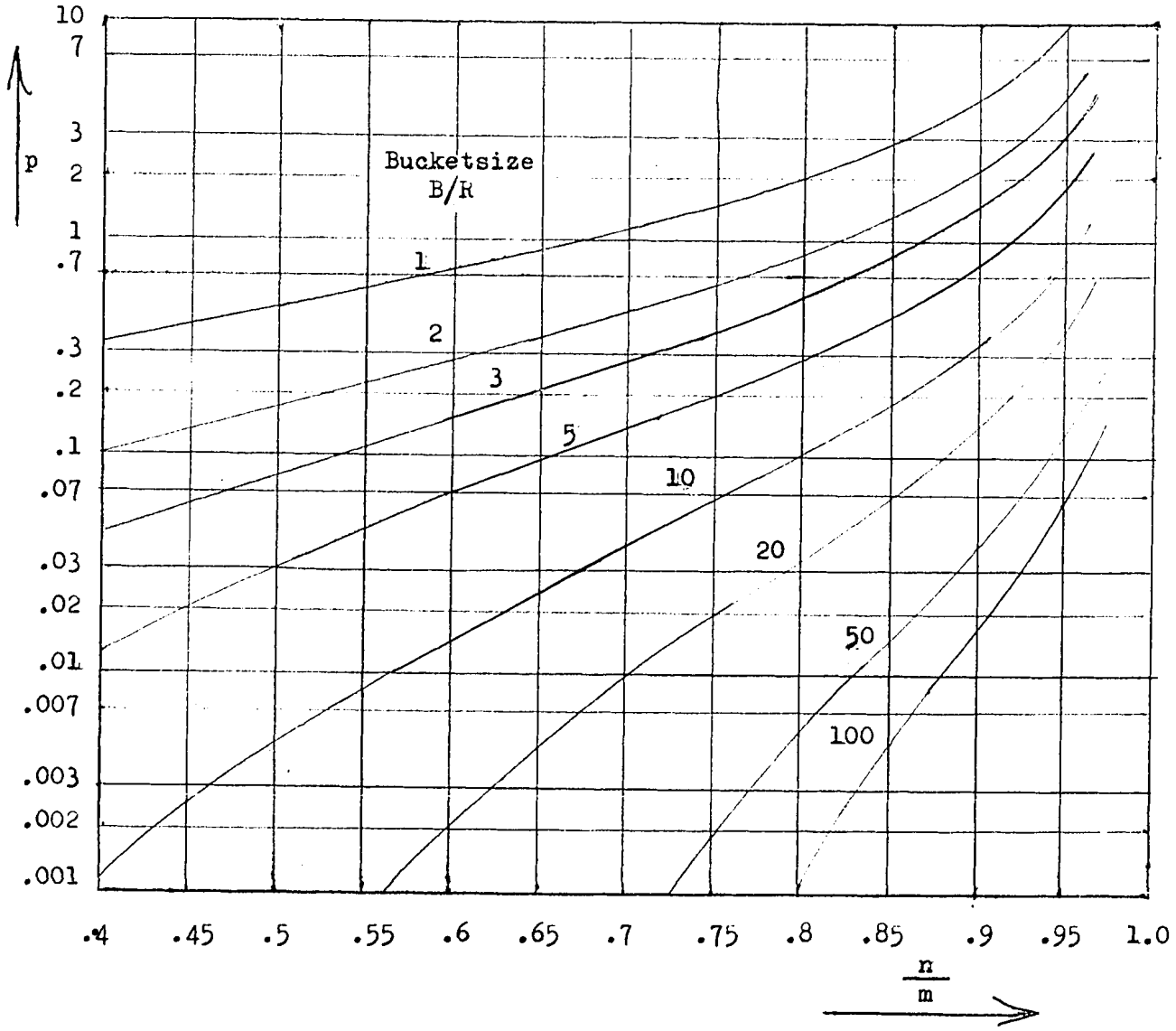
T_N : If the key for the next record is not known, there exists no practical method to retrieve that record.

$$T_I = \frac{2n}{m} T_F + s+3r+B/t$$

$$T_U = T_F + 2r$$

$$T_X = (m+o) \frac{R+W}{t'}$$

$$T_Y = c + 2T_X + Sort(n)$$



Overflow Probability for Linear Search

Figure 2-3

Multi-Ring File:

$$R = a'(V+P)$$

$$T_F = a_F \left\lceil \frac{1}{2} e^{\log_{a_F}(n)} \right\rceil (s+r+btt)$$

$$T_N = s+r+btt$$

$$T_I = a'_{link}(T_F+2r)+s+r+btt+2r - a'_{unordered}\frac{y}{2}(s+r+btt)$$

$$T_U = a'_U (2y(s+r+btt)+2r+2r) +s+r+btt+2r - a'_{unordered}\frac{y}{2}(s+r+btt)$$

$$T_X = n(1 + \frac{1}{y})(s+r+btt)$$

$$T_Y = n\, T_U$$

These formulas should not be applied blindly; an understanding of the conditions and assumptions made in their derivation has to be a part of the design process in order to assure their validity.

Summary:

A survey of the performance of these alternatives [1:Ch.4.0] shows that each file organization method has its own characteristic pattern. File organization methods which can be characterized as being loosely bound tend to perform poorly on retrieval and perform well on update, whereas the more tightly bound organizations provide better retrieval performance at the cost of increase update complexity and effort. This conclusion is summarized in Table 2-2 below.

Table 2-2

Grades of Performance for the Six Basic File Methods

	Space		Update		Retrieval		
	Structured Data		Record size		Fact	Subset Summary	Exhaustive Summary
	No	Yes	Equal	Greater			
Pile	A	E	A	A	E	D	B
Sequential	F	A	D	F	F	D	A
Indexed-Sequential	F	B	B	D	B	D	B
Multi-Indexed	C	D	C	C	A	B	D
Direct	F	B	B	F	B	F	F
Multi-ring	C	B	D	D	B	A	B

LEGEND:
A = Excellent, well suited to this purpose
B = Good
C = Adequate
D = Requires some extra effort
E = Possible with extreme effort
F = Not reasonable for this purpose

HYBRID FILE ORGANIZATION

Other file structures can be analyzed in terms of the basic methods expressed above. Some features of file structures may be imposed by hardware or software system limitations. An example of these are the simple file structures provided in mini-computer systems under FORTRAN [1:Ch.4.1] or MUMPS [Wasserman75A; 1:Ch.4.5].

In mini-computer systems a hybrid of a pile and a sequential file may be found: a file with variable length (hence compact) records with ordered fields and records in a sequential order, as determined by the chronological update sequence. Consideration of this hybrid shows that the space and fetch requirements are related to those of the pile file. The get-next and update requirements according to chronological sequence are those of the sequential file organization. By placing restrictions on blocking, the binary search or probing techniques used for sequential files can be used to improve fetch performance.

Novel and fairly complex structures can be assembled from the basic file organization methods. If a high level of performance is required for a specific well understood data model, then positive aspects from several basic organization types can be combined to achieve the desired effect [1:Ch.4.6, 4.7]. Such designs bind the data base system more rigidly to the expected usage patterns and hence should be carefully evaluated before being chosen. An example of a new hybrid technique, sequential and direct, is presented and analyzed in [1:Ch.6.1]. This particular hybrid us useful for data which ages in a regular fashion; the application shown involves hospital billing data.

In Chapter 4 of this thesis the performance of an AAMRS will be analyzed using the methods and results presented above. Since the development cost of a single instance of an AAMRS will be an important factor, the use of complex approaches will be avoided. Reference to the basic file organization methods descrived above allows rapid insight into a wide variety of hybrid file organization methods.

LOAD CONSIDERATIONS

In order to select which file organization methods provide adequate performance for the data base system which is being planned, the load which the applications place on the system has to be estimated [1:Ch:5]. In an AAMRS much of the load can be estimated on the basis of the patient volume and the expected services. The load component due to the use of the data base as an information system for scientific purposes is difficult to estimate, since it depends on the goals of associated clinicians and on the level of satisfaction which they obtain from the use of the system. Since much of the use of an AAMRS as an information system will not coincide with its use for clinic services, the scientific load can initially be ignored.

The product of the load and the response time for all the required services provides the basic measure of system adequacy. When a system is heavily loaded, multiprocessing can increase the utilization level of the hardware system at some cost in system complexity. A classification of a variety of storage system architectures and their effect on the degree of overlap to be obtained can be found in [1:Ch:5:4]. This includes a mathematical approach to the estimation of benefits of multiple parallel devices which produces results well-matched by published performance experiments [1:Ch:5:4:4]. This allows further formalization of an area where up to now only rules-of-thumb were available [Baskett76].

The load can vary considerably during a day, so that the results of such an analysis have to be applied carefully. The period of operation for most on-line systems is to be evaluated for the working day, and even within the working day a load variation of a factor of two or three is common. The results obtained from an integration of data base activity should hence not occupy more than 10 to 30% of the available time.

Queues can build up whenever the load on a system exceeds the immediate capacity. An analysis on transient queues shows that queue behavior consists of three phases. These phases are

queue increase to maximum load rate,
queue increases until excess load disappears,
queue collapses to zero while there is no excess load

Given an assumption that the excess load is due to a sinusoidal cyclical load pattern [1:Ch.5.4.2] provides the convenient result that these three phases are approximately equal in size for systems where there is a reasonable margin between average load (lambdabar) and capability (mu) (lambdabar <.9 mu).

COST-BENEFIT CONSIDERATIONS

Eventually a cost-benefit or cost-effectiveness evaluation [1:Ch.5.5] is required to establish whether the proposed data base system is desirable. In the medical area many benefits are of the intangible type, so that the analysis will be of the cost-effectiveness type. In [2:AppC-1] this approach is based on the assumption that if the decision maker has decided that the specified information is worth the specified cost, then this ratio establishes the level of effectiveness. If the performance falls short of the expectations or if costs exceed the allotted amount, then the desired level of cost-effectiveness has not been reached.

CHAPTER 3
AN APPLICATION OF DATA BASE DESIGN PRINCIPLES

"The proof of the pudding is in the eating;"
Traditional Proverb

The application in the area of ambulatory recordkeeping which will be used for demonstration of the methodology presented in Appendix 1 [1] is the Family Practice Center of the County of Sonoma, at the Community Hospital in Santa Rosa; This facility models a modern six physician group practice; It is staffed by several supervisory physicians, residents in Family Practice in a program affiliated with the University of California at San Francisco, and nurse-practitioners from the University of California at Davis; In order to establish the feasibility of an automated record system, the operations of this clinic have been extensively studied, and the functions of the required support system have been documented in detail [3]; This study was done using many of the principles developed in [1] and [2]; The proposed automated medical record system has been called "The Family System", and it is this application which will be analyzed; Since the proposed system is moderately large, much use is made of tables during the exposition of the design process;

FILES FOR THE FAMILY SYSTEM

The Family System is envisaged to have 29 data files and several auxiliary index files [3]; The data files are summarized in Table 3-1; The summary provides

An estimate of the size of the file: n;

The number and total length of all fixed fields: af, Rf;

The number and both average and maximum total length of all variable fields: av, Rv(avg, max);

Table 3-1

Files of the Family System

File No:	Ref: [3]	Name	Size n(min,avg,max)	Record af/Rf	Format av/Rv(avg,max)
1	9:2	Family Master	15000	1/4	3/(93,156)
1:1	"	Guarantors	(1,1:2,6)	1/4	-
2	9:3	Guarantor	18000	9/23	3/(93,156)
2:1	"	Family Members	(1,1:67,15)	-	3/(12,32)
3	9:4	Patient Master	30000	10/18	4/(173,956)
3a	"	Optional fields	"	-	4/(94,157)
3b	"	Temp: Billing data	"	4/16	-
3:1	"	Entitlements	(0,0:1,12)	3/11	-
3:2	"	Insurance Plans	(0:1,9)	9/18	1/(80,115)
4	9:5	Medical Data Base	7000	61/51	1/(80,800)
4:1	"	Pap Smears	(0,2,12)	2/3	-
4:2	"	Drug Allergies	(0,0:2,20)	1/3	-
4:3	"	Other Allergies	(0,0:2,10)	-	1/(80,800)
5	9:2	Problem List	96000	2/5	-
5:1	"	Problems	(1,3,20)	5/10	-
5:1:1	"	Actions	(1,3,100)	5/9	2/(97,832)
6	9:7	Preventive Care	5000	56/93	-
7	9:8	Medical Audit	2000	2/6	-
7:1	"	Audit Columns	(0,10,100)	27/38	-
8	9:9	Active Therapy	15000	2/6	-
8:1	"	Active Medications	(0,4,40)	12/16	1/(80,800)
8:2	"	Active Non-drug Therapy	(0,2,20)	5/10	1/(80,800)
9	9:10	Past Therapy	60000	2/6	-
9:1	"	Past Medication	(0,4,40)	12/16	1/(80,800)
9:2	"	Past Non-drug Therapy	(0,2,20)	5/10	1/(80,800)
10	9:11	Flow Sheets	5000	2/6	-
10:1	"	Lab Results	(0,5,200)	16/28	-
10:2	"	Visit Columns	(0,10,100)	27/38	-
11	9:12	Visit Records	60000/year	16/28	-
11:1	"	Problems Seen	(1,3,20)	7/9	-
11:1:1	"	Services Rendered	(0,2,20)	4/17	-
11:1:2	"	Tests Ordered	(0,4,40)	6/11	1/(80,800)
11:1:3	9:12	Problem Note	(0,1,3)	-	1/(80,800)
12	9:13	Appointments	1080	7/11	-
12:1	"	Patients Per Slot	(0,4,12)	3/6	1/(17,32)
13	9:14	Day Sheet	6	6/12	
13:1	"	Patients Per Office	(0,30,60)	2/5	1/(17,32)
14	9:15	User List	50	5/18	2/(10,802)
15	9:16	Transaction Log	500/day	11/28	-
15:1	"	Old and New Values	(1,5,24)	1/1	2/(10,802)
16	9:17	Fiscal Audit	3000/month	11/28	-
17	9:18	Messages	50	4/9	1/(80,800)
18	9:19	System Messages	20	4/9	1/(80,800)
19	9:20	Patient Name Index	30000	8/17	1/(17,32)
20	9:21	Family Number Index	15000	2/4	-
20:1	"	Family Member	(1,2,15)	1/1	-
21	9:22	Family Name Index	15000	1/3	1/(17,32)
22	9:25	Third Party Payors	100	1/3	1/(17,32)
23	"	Surveillance Criteria	50	1/3	1/(80,800)
24	"	Problem Classification	250	1/3	1/(17,32)
25	"	Diagnostic Procedures	250	1/3	1/(17,32)
26	"	Medication Names	700	1/3	1/(17,32)
27	"	Therapy Names	300	1/3	1/(17,32)
28	"	Service Names	500	1/3	1/(17,32)
29	"	Scheduled Actions	20	1/3	1/20
30	9:4	Family Physicians	200	1/3	1/(93,156)
31	"	Hospital Service Code	700	1/3	1/3
32	9:20	Therapy Classification	300	1/3	1/2

For each nest file, the repetition count, the number of fields, and the size: n(min, avg, max), an, Rn;

The nests themselves are denoted using a hierarchical numbering scheme: entity relation file, level 1 nest file, second and lowest level nest file: p, p:q, p:q:r

The third file (Patient) has multiple record subtypes, containing optional and transient data; These files are denoted 3a and 3b;

The length for various element types defined for the Family System is estimated as shown in Table 3-2; Groups of elements are taken in terms of multiple integer bytes; The table does not include any structural non-essential data elements; The estimates are based on data provided for the Family System [3], from the AAMRS Study [2], and from statistics presented in [1]; The expected size of the files was verified by Dr; John Dervin of the Family Practice Center, and is shown in Table 3-3;

Table 3-2

Length of Data Element Types

Data-type	V(min)	V(avg)	V(max)	Reference
Name	6 chars	17	37 bytes	[3:p:80 and 1:p:1128]
Address	25	45	65 bytes	[3:p:81]
Telephone (individual)	2	31	59 bytes	[3:p:81]
Telephone (business)	18	18	18	
Note	2	80	800 bytes	[2:p:99 and 2:vol:2 CDB p:21]
Date	2	2	2 bytes	[3:p:81]
Time	2	2	2 bytes	[3:p:81]
Sex	2 bits			
Flags	1 bit			
Response	2 bits			
Digits	4 bits			

Table 3-3

Load Projection for 1980 Family System Operation

50000	visits/year (incl. emergency)
30000	medical records
15000	family member records
3000	surveillance records
20000	guarantors
5000	flowsheets
5000	preventive care
7000	medical data base
10000	preventive care
15000	active therapy (meds and other)

This list of files, as summarized here or as described in [3], does not convey the concept of a data base system. While the preparatory work is more thorough than is commonly found in the design of medical record systems, an essential design element, namely the relationships among the files, is not defined. A data base is a collection of interrelated files; the relationships among the files provide the semantics which underlie the data base. In [3] these are not yet made explicit. If an implementation were to begin at this point, the individual application programs would define the interrelationships of the files and create from the files a data base in an ad-hoc fashion. The degree to which problems, inefficiencies, and inconsistencies will occur will depend on the foresight and experience of the implementors. The transformations and semantic notions developed in [1:Ch.7] will be applied in the remainder of this chapter in order to develop a simple and consistent data base model which represents the data in the proposed file structure.

DATA BASE STRUCTURE

The files presented by the Family System are organized to satisfy the perceived functional needs of the medical record applications. This means that all data attributes are assigned to specific files using conventional programming design procedures. The files, however, exhibit semantic relationships among each other through the use of shared attribute domains. The files themselves are furthermore complex in the sense that they are not in first-normal-form. In order to present the data base model in a form which provides guidance to the design process, the files for the Family System will be normalized.

In order to derive the interfile relationships, the principal attributes of all files are listed alphabetically in Table 3-4. With each attribute the domain, the data type, and the file usage is indicated. Attributes which have a matching domain, but are named differently, are computationally comparable, but functionally distinct.

Table 3-4

Principal Attributes

Attribute	Domain	Type	Files
actioncode	unique	1 digit code	5:2, 11:1
allergy_note_number	unique	id	4:3
amount_of_last_payment	$	6 digits	2
appointment_schedule	-	various	12
appointment_status	unique	1 digit code	12:1
billing_number	unique	2 digit code	3:1, 11:1:1
complications	prob_id	2 digit code	10:2
contract_amount	$	6 digits	2
contract_period	days	5 digits	2
control_limited_by	prob_id	2 digit code	10:2
date_of_action	date	date	5:2
date_of_appointment	date	date	12
date_of_birth	date	date	3
date_of_column_on_flowsheet	date	date	10:1, 10:2
date_of_fiscal_transaction	date	date	16
date_of_last_payment	date	date	2
date_of_message	date	date	17, 18
date_of_next_appointment	date	date	10:2
date_of_patient_entry	date	date	3
date_of_problem_onset	date	date	5:1
date_of_transaction	date	date	15
date_of_visit	date	date	11
disposition_code	unique	1 char; code	11:1
drug_allergy	drug code	5 digit code	4:2
diagnostic_test_descr	unique	name	25
family_address	address	address	1
family_member_number	p-id2	id	2
family_name	name	name	1,21
family_number	p-id1	id	1, 2, 20, 21
family_physician_number	ph-id	id	3, 30
family_telephone_numbers	telephone	telephone	1
family_physician_data	unique	name, addr:,phone	30 30
financial_class	unique	1 char; code	3
flow_sheet_detail	-	various	10:2
flow_sheet_type	unique	id	10
follow_up_code	unique	1 char; code	11:1, 19
follow_up_until	unique	date	3:1
form_of_tender	unique	1 digit code	16
guarantor_address	address	address	2
guarantor_name	name	name	2
guarantor_number	unique	id	1, 2
guarantor_telephone_number	telephone	telephone	2
hospital_service_code	unique	3 char; code	3b, 31
insurance_deductable	$	3 digits	3:2
insurance_identification	unique	5 digit id	3:2, 22
insurance_policy_limit	$	5 digits	3:2
lab_results	-	various	10:1
language_spoken	unique	1 char; code	3

<Continued>

Attribute	Domain	Type	Files
marital_status	unique	1 digit code	3
medical_data_base	unique	responses	4
medi_cal_number	unique	id	3
medicare_number	unique	id	3
medication_code	drug	5 digit code	8:1, 9:1, 26
medication_descr	-	name	26
medication_detail	-	various	8:1, 9:1
message	unique	note	17, 18
message_from	user-id	id	17, 18
message_to	user-id	id	17, 18
new_problem_number	prob-id	id	5:2
non_drug_therapy	unique	5 digit code	8:1, 9:1, 27, 32
note_on_test	unique	note	11:1:2
note_on_visit	unique	note	11:1:3
note_on_patient	unique	note	3
note_on_allergy	unique	note	4:3
note_on_medical_data base	unique	note	4
note_on_action	unique	note	5:1:1
note_on_active_medication	unique	note	8:1
note_on_active_non_drug_therapy	unique	note	8:2
note_on_past_medication	unique	note	9:1
note_on_past_non_drug_therapy	unique	note	9:2
notify_address	address	address	3
notify_name	name	name	3
notify_relationship	relationship	1 char code	3
notify_telephone	telephone	telephone	3
nurse	user-id	id	10:2, 12, 13
occupation	unique	3 digit code	4
office	unique	1 char code	10:2, 11, 12, 13, 15, 16
ordered_by	user-id	5 digit code	8:2, 9:2
other_allergy	unique	note	4:3
pap_smear_date	unique	date	4:1, 6
pap_smear_result	unique	response	4:1, 6
patient_address	address	address	3
patient_name	name	name	2, 3, 12:1, 13:1, 19
patient_number	pid1||pid2	8 digits	3, 4, 5, 6, 7, 8, 9, 10, 11, 12:1, 13:1, 16, 19, 20:1
patient_telephone	telephone	telephone	3
patient_type	unique	1 char code	3
patient_visit_status	unique	1 digit code	13:1
payment_collected	$	5 digit	11
payment_amount	$	6 digit	17
prescribed_by	user-id	5 digit code	8:1, 9:1
preventive_care_data	-	various	6
previous_chart_number	unique	7 digits	3
problem_classification	unique	5 digit code	5:2, 11:1, 19, 24
problem_number	prob-id	2 digit code	5:1, 8:1, 8:2, 9:1, 9:2, 11:1, 24
problem_note_number	unique	id	11:1:3
problem_type	unique	1 digit code	5:1
provider_number	user-id	id	5:2, 10:2, 11
practitioner	user-id	id	12, 13
receptionist	user-id	id	13
relationship_to_guarantor	relation	1 digit code	2

<Continued>

Attribute	Domain	Type	Files
scheduled_system_actions	-	various	20
service_code	unique	7 digit code	3:1, 11:1:1, 16, 28, 31
service_data	-	various	11:1:1, 16
service_description	unique	name	28
sex	unique	response	3, 4, 19
surveillance_code	unique	2 digit code	19, 23
surveillance_data	-	various	23
taking_medication	unique	1 digit code	10:2
test_code	unique	5 digit id	11:2, 25
test_data	-	various	11:2
time_of_arrival	time	time	36
therapy_class	unique	2 digit code	19, 32
therapy_description	unique	name	27
therapy_detail	-	various	8:2, 9:2
third_party_data	-	various	22
transaction_data	-	various	15, 15:1
transaction_type	unique	1 char; code	15, 16
user_description	-	various	14
user_number	user-id	id	14, 15
visit_number	unique	3 digit	11
visit_recorded_by	user-id	5 digit	11
visit_times	unique	time	11
visit_type	unique	1 char; code	11
weight	unique	3 digits	4, 6

Not included in the table are data elements which are necessary for housekeeping duties (such as number of elements in a nest), or free-text entries which do not actively participate in the structure of the Family System. Similar medical data elements within a file have been grouped to allow a compact presentation. A few new attributes, as allergy-note-number, have been introduced to assure that all records can be identified with a unique ruling part.

FIRST-ORDER NORMALIZATION

A first-order normalization of the 32 Family System Files described in Table 3-1 extracts the nested structures and places them into distinct files. There are 25 nests and two auxiliary files so that the Family System in first-normal form comprises 57 files. In practice some of these nest files can be avoided by designating a fixed number of fields for the nest in the parent entity file as shown in Figure 2-2. The degree to which nests can be omitted depends on the efficiency of the file compression support. Prime candidates for denesting are the following files:

1:1	Guarantors	- small and low n(max) of repeating entries
4:1	Pap Smears	- " " " " " " "
4:2	Drug Allergies	- " " " " " " "
10:0	Flowsheet	- small flag and few entries
13:1	Day Sheet Patients	- few records and high density
20:1	Family member	- small size of repeating field

A sample calculation of the denesting tradeoff for the first of these files (1:1 Guarantors) is shown on the following page.

Example 1

Nest Files versus Denested Attributes

	Nested	De-nested
Space: Recordsize without compression	R_n = n(avg)(af;Rf+P)	R_d = n(max)(af;Rf)
per parent	2 * (1*4+2) = 12 bytes	6 * 1 * 4 = 24 bytes
total (for 15000 records)	180;000 bytes	360,000 bytes

Access Frequency (read only):

once per visit	Lv	200/day
once per bill	Lb	100/day
total	Lt = ~~300/day~~	

Time per Access (given a direct linkage and a 2314-type disk):

$$T_n = n_{avg}\, T_{fetch} = n_{arg}\left(s+r+\frac{R_n}{t}\right)$$

$$= 2\,(0;060+0;037+\frac{12}{312000})$$

$$= 0;194 \text{ sec;}$$

$$T_d = n_{max}\,\frac{R_d}{t}$$

$$= 6(\frac{24}{312000})$$

$$= 0;000461 \text{ sec;}$$

Time per day:

TD_n = 0;194 * Lt = 52;2 sec; TD_d = 0;000462 * Lt = 0;1386 sec;

Cost Tradeoff:

Denested - Nested Implementation

Space (360000-180000) * \$600 10^{-6} year = \$108/year

Time(0;1386-52;2) = 52;06 seconds/day

Value @ system cost of \$50000/year
= \$200/day = \$12/hour ◘ \$;20/minute
= -52;06/60*0;20*300 days = -\$52;06/year

Total cost increase for denesting = \$55;94/year

This example shows that without compression the improved performance with denesting is not adequate to overcome the increased storage cost. The goal of compression is to use less space for the n(max)-n(avg)=4 empty fields expected per denested record. The crossover point, when denesting becomes cost-beneficial, is reached when compression can reduce the space for these fields as follows:

Benefit of Time saving is \$52.06/year since T_d is negligible.

Equivalent Space Cost $\quad \$52.06/(600\ 10^{-6}) = 86{,}767$ bytes.

giving a compressed nest size of 87000/15000 = 5.8 bytes to encode four empty fields. Proportionate space will be allowable to compress 5, 3, 2, or 1 empty fields. Several of the compression techniques described in [1:Ch.14.3] can achieve this goal.

This result will be extrapolated to assume that denesting can be carried out for the five nest files listed earlier. This will lead to a modification of the parameters of the parent relations as follows:

File No.	Name	Size n(min,avg,max)	Record af/Rf	Format av/Rv(avg,max)
1	Family Master	15000	1/4	3/(93,156)
1.1	Guarantors	(1,1.2,6)	1/4	-
new	Family Master	15000	1/4	4/(98/180)
3	Patient Master	30000	10/18	4/(173,956)
10	Flow Sheets	5000	2/6	-
new	Family Master	30000	10/18	6/(174,962)
4	Medical Data Base	7000	61/51	1/(80,800)
4.1	Pap Smears	(0,2,12)	2/3	-
4.2	Drug Allergies	(0,0.2,20)	1/3	-
new	Medical Data Base	7000	61/51	4/(87,896)
13	Day Sheet	6	6/12	
13.1	Patients Per Office	(0,30,60)	2/5	1/(17,32)
new	Day Sheet	6	6/12	3/(660,2220*)
20	Family Number Index	15000	2/4	-
20.1	Family Member	(1,2,15)	1/1	-
new	Family Number Index	15000	2/4	1/(2,15)

* The law of large number [1:Ch.6.2] makes this large recordsize extremely improbable.

LEXICONS

There are several files in the Family System which implement the concept of a lexicon, i.e., a one-to-one translation of a code to a description, or a several-to-one code translation. From the point of view of the essential structure of the data base model, these files can be omitted; in practice, they derive their justification from the savings in space and a reduction of redundancy relative to updating.

The following eight files are lexicons as defined in [1]:

15	Patient Name Index
20	Family Number Index
20.1	Family Member *
21	Family Name Index
25	Diagnostic Procedures
26	Medication Names
27	Therapy Names
28	Service Names

* to be eliminated by denesting

In the implementation of the Family System these files remain important. Their simple structure will allow an efficient implementation, as presented in Chapter 4.

REFERENCED ENTITY FILES

A file type which is of secondary importance from a structural viewpoint, but is important to assure consistency and minimize redundancy is the referenced entity relation. Such a relation carries descriptive information which otherwise would have to be repeated in the owner files. If the relation is referenced by multiple relations, it will exert a binding influence on the data base. However, since the content of these relations is quite static, unbinding could be implemented easily by duplication.

The following seven files are referenced entity files, as defined in [1:Ch.7]:

		Referenced by
14	User List	5.2, 8.2, 9.2, 10.2, 11, 12, 13, 15, 17, 18
22	Third Party Payors	3.2
23	Surveillance Criteria	19
24	Problem Classification	5.2, 11.1, 19
30	Family Physicians	3
31	Hospital Service Code	3b
32	Therapy Classification	19

SERVICE FILES

Several files proposed for the Family System are not part of the data base proper, but are viewed as system support facilities. They provide reliability and communication capabilities for the Family System and hence carry information in redundant or transient form. The files which fall into this category are:

3b	Temporary Billing Data
7 and 7.1	Medical Audit
15 and 15.1	Transaction Log
16	Fiscal Audit
17	Messages
18	System Messages
29	Scheduled Actions

These files, since they are at the interface of the Family System and the users, will have to be designed individually.

FILE MINIMIZATION

The remaining files can now be reviewed for structural inter-relationships and redundancies. Linkage keys which have been identified in the 34 primary files are given as the ruling part for these files in Table 3-5. This table also indicates which of these files have a NULL dependent part (1:1, 5, 8, 9, 10), these files were apparently defined in [3] for their utility in providing a linkage, and are hence not an essential part of the data base model.

There remain a number of files with identical ruling parts. These files represent different functional needs and were hence defined distinctly. In the model of the data base, however, these files are best combined. The original files are then represented by dependent part segments in the model.

Table 3-5

Primary Files

File No:	Name	Ruling Part (:> Dep: part)
1	Family Master	family_number
1:1	Guarantors *	family_number, guarantor_number :> NULL;
2	Guarantor	family_number, guarantor_number
2:1	Family Member	family_number, family_member_number
3	Patient Master	patient_number
3a	Optional Fields	patient_number
3:1	Entitlements	patient_number, service_code
3:2	Insurance Plans	patient_number, plan_number
4	Medical Data Base	patient_number
4:1	Pap Smears *	patient_number, date
4:2	Drug Allergies *	patient_number, medication_code
4:3	Other Allergies	patient_number, allergies_note_number
5	Problem List	patient_number :> NULL;
5:1	Problems	patient_number, problem_number
5:1:1	Actions	patient_number, problem_number, action_code, date
6	Preventive Care	patient_number
8	Active Therapy	patient_number
8:1	Active Medication	patient_number, medication_code
8:2	Active Non-Drug Therapy	patient_number, non_drug_therapy
9	Past Therapy	patient_number :> NULL;
9:1	Past Medication	patient_number, medication_code, date
9:2	Past Non-Drug Therapy	patient_number, non_drug_therapy, date
10	Flow Sheets *	patient_number, flow_sheet_type :> NULL;
10:1	Lab Results	patient_number, date
10:2	Visit Columns	patient_number, date
11	Visit Record	patient_number, date
11:1	Problems Seen	patient_number, date, problem_number
11:1:1	Services Rendered	patient_number, date, problem_number, service_code
11:1:2	Tests Ordered	patient_number, date, problem_number, test_code
11:1:3	Problem Notes	patient_number, date, problem_number, problem_note_number
12	Appointment	date, office
12:1	Patients per Slot	date, office, patient_number
13	Day Sheet	office
13:1	Patients per Office *	office, patient_number

* These files will be eliminated if denesting is carried out as indicated earlier.

In file 2:1 (Family Member) the catenation of 'family_number' and 'family_member_number' forms the 'patient_number', so that this file can also be represented as a segment dependent on the ruling part of 'patient_number':

The new relations created in this manner are listed in Table 3-6:

Table 3-6

Catenated Files

New Number	Name	Ruling Part	:>	Dependent Segments (Former Number)
300	Patient:	patient_number :>		Patient Master (3), includes Flowsheet (10) Family Member (2;1), Optional Fields (3a), Medical Data Base (4), Preventive Care (6);
400	Visits:	patient_number, date :>		Visit Records (11), Lab Results (10;1), Visit Columns (10;2), Pap Smears (4;1);
800	Medication:	patient_number, medication_code, date :>		Active Medication (8;1), Past Medication (9;1);
850	Non-drug Therapy:	patient_number, medication_code, date :>		Active Non-drug therapy (8;2), Past Non-drug therapy (9;2);

Four of the files classified as lexicons also have matching ruling and dependent formats. These are:

25	Diagnostic procedures	250	1/3	1/(17,32)
26	Medication names	700	1/3	1/(17,32)
27	Therapy names	300	1/3	1/(17,32)
28	Service names	500	1/3	1/(17,32)

since the ruling parts can be made distinct by catenation of a letter code (P, D, M, T, S), these files can also be combined. An advantage when combining lexicons is the code simplification as well as the increased efficiency of space utilization obtained by pooling [1:Ch;6;1]. The resulting file will be referred to as "900;Terms".

For these files new descriptive parameters have to be calculated. The parameters of dependent segments which have a one-to-one relationship (Fr=n) with the ruling part can simply be summed.

Segments which occur with a lower frequency ($Fr \neq n$) are added to the variable length part of the varying segments as follows:

$$n_{new} = \sum n_{Fr=n} \; ; \quad af_{new} = \sum af_{Fr=n} \; ; \quad Rf_{new} = \sum Rf_{Fr=n}$$

$$av_{new} = \sum a_{v_{Fr=n}} + \sum a_{v_{Fr \neq n}}$$

$$Rv(avg)_{new} = \sum Rv(avg)_{Fr=n} + \sum (Rf+Rv(avg))_{Fr \neq n} \quad \frac{n(avg)_{Fr \neq n}}{n_{Fr=n}}$$

$$Rv(max)_{new} = \sum Rv(max)_{Fr=n} + \sum (Rf+Rv(Max))_{Fr \neq n}$$

so that,

File Number	Name	n	af/Rf	av/Rv(avg,max)
300	Patient	30000	11/19	126/(295,2057)
400	Visits	60000	-	45/(44,69)
800	Medication	300000	12/16	1/(80,800)
850	Non-drug Therapy	150000	5/10	1/(80,800)
900	Terms	1750	1/4	1/(17,32)

DATA MODELS

The remaining 31 relations will be combined in various ways to form the data models for the applications of the Family System. Four of these relations are lexicons and hence not essential to the model. A number of such data models are sketched in Figures 3-1 to 3-3. These sketches retain the segment names of the catenated relations, so that they relate to the descriptions in [3].

Figure 3-1

Making an Appointment

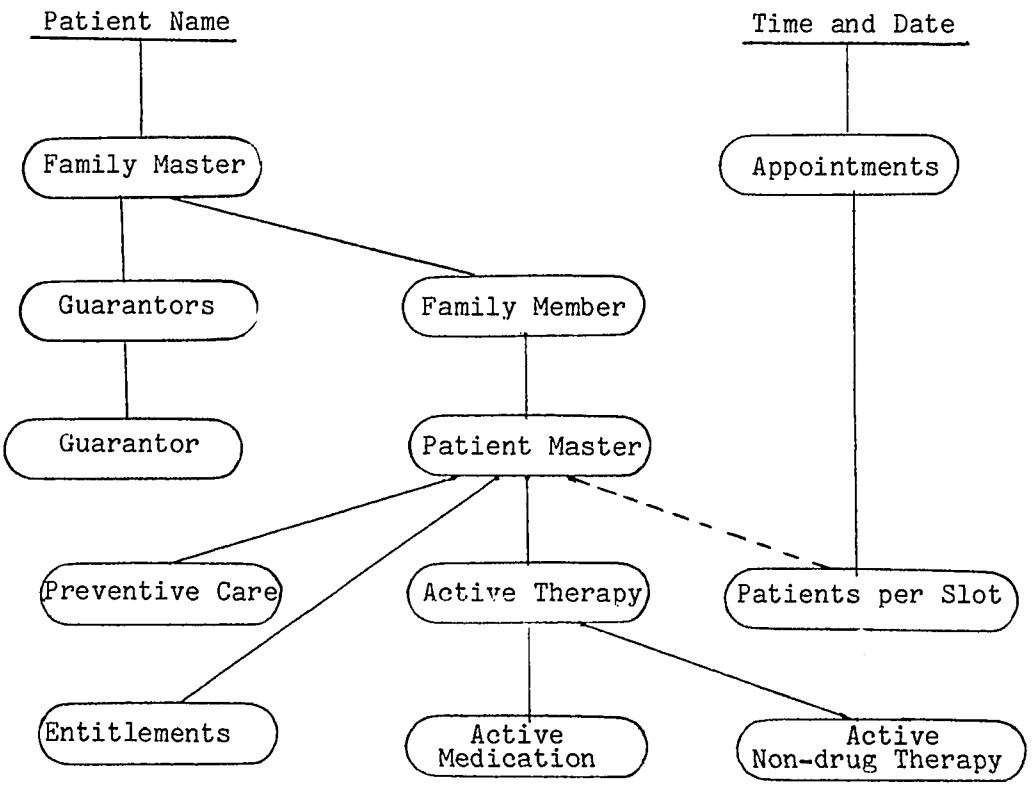


Figure 3-2

Preventive Care Visit

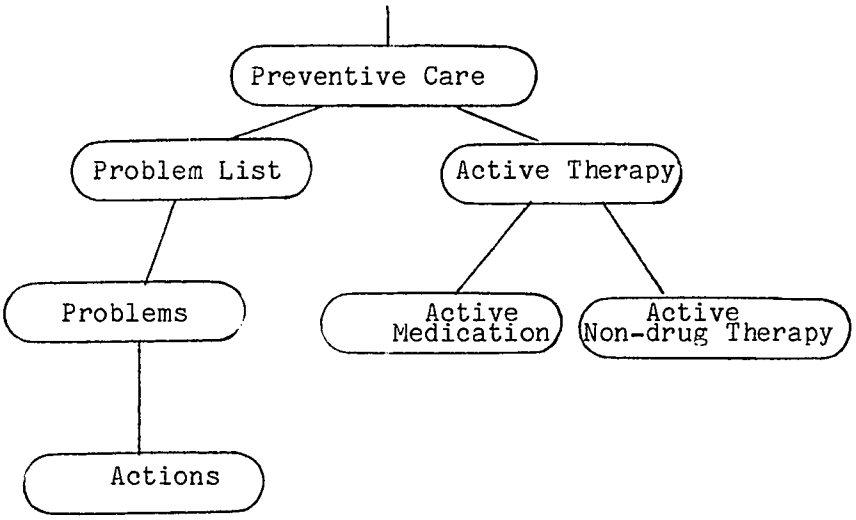
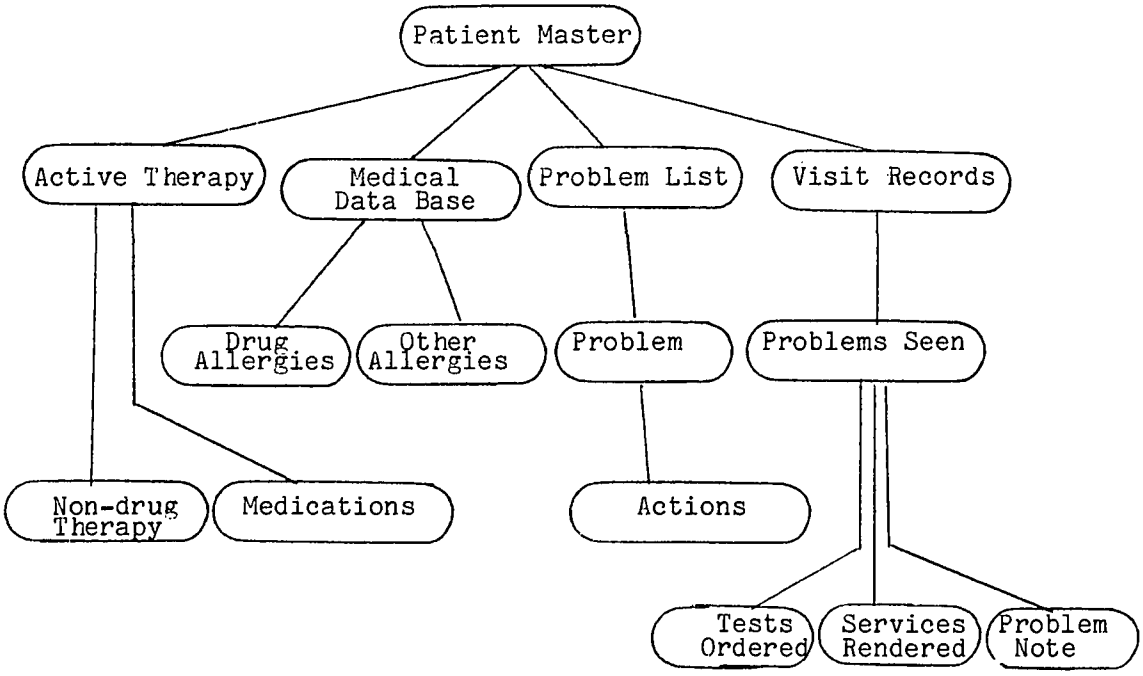


Figure 3-3

Acute Care Visit



It is seen that now the inter-file linkages pay an crucial role. Important linkage domains found in Table 3-4 are

Patient name
Patient number
Provider name
Office
Family number
Problem number
Service code
Therapy code
Dollars
Medication code

Some redundancy is evident. In order to simplify updating, it may be desirable to define certain attributes as primary, and to update the redundant copies of these attributes asynchronously. Candidate attributes for such a division are the objective medical data as obtained during a patient visit. The primary relation for such data would be the Patient Visit file. The Problem List File, Flow-Sheets, Preventive Care records, etc. could be updated overnight. Redundancy for protection of data is maintained through the service files as the Transaction Log.

Associations:

An important primitive function for a data base is the ability to associate data from file with data from another file. Associations may be created dynamically through use of the Join operation, or may be bound permanently. A permanent association can contain unlimited dependent-part information; dynamic association only carries information derived from the joining conditions.

In the Family System, as originally defined, the following three files represent associations:

11:1 Problems Seen	=	11 Visits	&	3 Patient Master
12:1 Patients per Slot	=	12 Appointments	&	3 Patient Master
13:2 Patients per Office	=	13 Pay Sheet	&	3 Patient Master

In each of these associations the dependent data element are strongly related to either one of the owning relations. For example problem status at a visit is a data element which determines problem status independent of the visit. An analysis of the dependent data elements can assign similarly ownership of all other dependent elements. The association described by the files describes hence in tabular form the join conditions for the association.

11:1 : This problem was seen during this visit.
12:1 : This patient has an appointment at this time.
13:1 : This patient appeared at this office and time.

By not binding the association to both owners, increased flexibility of implementation is retained. Only network type data bases can support associations permanently. By treating these three files as nests, any hierarchical data base can support the Family System.

DATA BASE MODEL

The data base model combines the data models of the individual applications.

The construction of the data base model provides the insight required to design an optimal and consistent implementation for a data base system which supports these applications. Table 3-7 lists the essential relations and their roles in the data base model. This model is now fully non-redundant can be presented graphically as shown in Figure 3-4.

Inspection of the data base model reveals that the essential data base can be implemented by a hierarchical system. No stored associative relations are required to store the data base model. Any associations needed will be generated by join operations at execution time. Three trees are required, and the main tree requires multiple entry points and multiple nests on a level. It is possible to survey available Data Base Management Systems in order to ascertain which of the available systems offer such choices. Appendix C of [1] provides an index to possible system choices.

Table 3-7

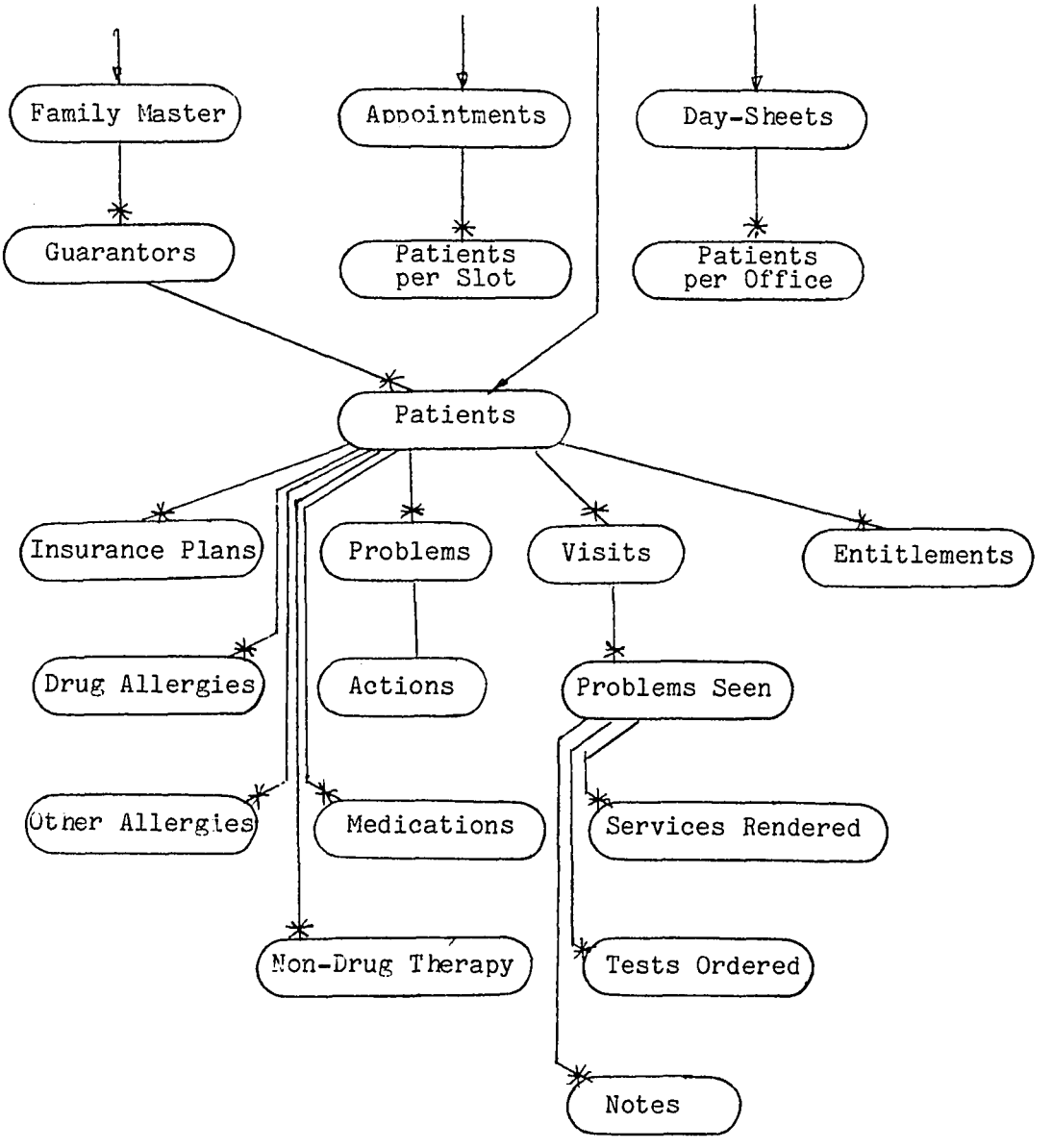
Relations in Data Base Model

File Number	Name	Ruling Part ar/Rr	Structural Function
1	Family Master	1/4	Entry, entity relation
2	Guarantors	2/8	Entry, nested relation on 1
300	Patient	1/8	Entry, nested relation on 2
3:1	Entitlements	2/8	Nested relation on 300
3:2	Insurance Plans	2/7	" " "
400	Visits	2/6	Entry " " 300
4:2	Drug Allergies	2/7	Nested " " 300 *
4:3	Other Allergies	2/5	" " " "
5:1	Problems	2/6	" " " "
5:1:1	Actions	3/8	" " " 5:1
800	Medications	3/9	" " " 300
850	Non-drug Therapy	3/9	" " " 300
11:1	Problems Seen	3/8	Entry, nested " 400
11:1:1	Services Rendered	4/10	Entry " " 11:1
11:1:2	Tests Ordered	4/10	" " " "
11:1:3	Problem Note	4/9	" " " "
12	Appointments	2/3	Entry, entity relation
12:1	Patients per Slot	3/7	Nested relations on 12
13	Day Sheet	1/1	Entry, entity relation
13:1	Patients per Office	1/5	Nested relation on 13 *

* can be denested

Figure 3-4

Family System Data Base Model



SUMMARY

The files proposed by the design of the Family System were transformed into a manipulatable model. This model was then inspected for function, redundancy, semantic relationships, and consistency. A number of transformations were performed to clarify and simplify the data base model. This model is now suitable for a performance-oriented design effort. Table 3-8 summarizes the transformations performed.

The design of the Family System, as developed from an analysis of the service requirements, described 58 distinct files.

The data base model now consists of

- 20 primary entity files (2 of these can be denested)
- 7 referenced entity files
- 4 lexicons
- 9 service files

Relationships among the entity files are documented in Table 3-7.

Table 3-8

Transformations on the Family System Model

Original Files:		Role in Data Base Model
1	Family Master	Primary file
1:1	Guarantors	Denested into 1
2	Guarantor	Primary file
2:1	Family Members	Catenated to 300 Patient
3	Patient Master	Catenated to 300 Patient
3a	Optional fields	Catenated to 300 Patient
3b	Temp; Billing data	Service file
3:1	Entitlements	Primary file
3:2	Insurance Plans	Primary file
4	Medical Data Base	Catenated to 300 Patient
4:1	Pap Smears	Catenated to 400 Visits
4:2	Drug Allergies	Denestable, Primary file
4:3	Other Allergies	Primary file
5	Problem List	null
5:1	Problems	Primary file
5.1:1	Actions	"
6	Preventive Care	Catenated to 400 Visits
7	Medical Audit	Service file
7:1	Audit Columns	Service file
8	Active Therapy	null
8:1	Active Medications	Catenated to 800 Medication
8:2	Active Non-drug Therapy	Catenated to 850 Non-drug Therapy
9	Past Therapy	null
9:1	Past Medication	Catenated to 800 Medication
9:2	Past Non-drug Therapy	Catenated to 850 Non-drug Therapy
10	Flow Sheets	Denested into 3 Patient
10:1	Lab Results	Catenated to 400 Visits
10:2	Visit Columns	Catenated to 400 Visits
11	Visit Records	Catenated to 400 Visits
11:1	Problems Seen	Primary file
11:1:1	Services Rendered	"
11:1:2	Tests Ordered	"
11:1:3	Problem Note	"
12	Appointments	"
12:1	Patients Per Slot	"
13	Day Sheet	Primary file
13:1	Patients Per Office	Denestable into 13, primary file
14	User List	Referenced Entity file
15	Transaction Log	Service file
15:1	Old and New Values	"
16	Fiscal Audit	"
17	Messages	"
18	System Messages	"
19	Patient Name Index	Lexicon
20	Family Number Index	"
20:1	Family Member	Denested into 20
21	Family Name Index	Lexicon
22	Third Party Payors	Referenced Entity file
23	Surveillance Criteria	" " "
25	Diagnostic Procedures	Catenated to 900 Terms
26	Medication Names	" " "
27	Therapy Names	" " "
28	Service Names	" " "
29	Scheduled Actions	Service file
30	Family Physicians	Referenced Entity file
31	Hospital Service Code	" " "
32	Therapy Classification	" " "
New:		
300	Patient	Primary file
400	Visits	"
800	Medications	"
850	Non-drug Therapy	"
900	Terms	Lexicon

CHAPTER 4
PERFORMANCE

"De haring vissery is de basis aller commercien."
Traditional Dutch Proverb

In order to control the performance of the data base system which is to support the Family System, several design techniques can be employed. These can be categorized as follows:

File Organization Choices: pile, sequential, indexed-sequential, indexed, direct or ring;

File Partitioning Choices: One record per tuple, or one record per segment;

Use of auxiliary access files;

In practice all three choices will be employed. In order to establish a baseline, the performance of a pile of unpartitioned records without auxiliary access files will be presented initially. Table 4-1 summarizes the basic relations and their parameters, as derived from the data presented in Table 3-1. Since all files are evaluated independently, the size of the ruling part has been added to the recordsize.

The primary files summarized in Table 4-1 do not include the lexicons and referenced entity files. The space required for these is presented in Table 4-3 using the assumptions that they will be implemented as direct files.

Table 4-1

File Parameters

File Number	Name	Size n	Record af/Rf	Format av/Rv(avg,max)
1	Family Master	15000	1/4	4/(98,180)
2	Guarantors	18000	9/23	3/(93,156)
300	Patients	30000	11/19	126/(295,2057)
3:1	Entitlements	3000	5/19	-
3:2	Insurance Plans	30000	11/19	1/(80,115)
400	Visits	60000	16/28	45/(44,69)
4:2	Drug Allergies	6000	3/10	-
4:3	Other Allergies	6000	2/5	1/(80,800)
5:1	Problems	90000	7/16	-
5:1:1	Actions	270000	8/17	2/(97,832)
800	Medications	300000	12/16	1/(80,800)
850	Non-drug Therapy	150000	5/10	1/(80,800)
11:1	Problems Seen	180000	10/17	-
11:1:1	Services Rendered	120000	8/27	-
11:1:2	Tests Ordered	240000	10/21	1/(80,800)
11:1:3	Problem Note	180000	4/9	1/(80,800)
12	Appointments	1080	7/11	-
12:1	Patients per Slot	4360	6/13	1/(17,32)
13	Day Sheet	6	6/12	-
13:1	Patients per Office	180	3/10	1/(17,32)
	Total (records)	1,703,626		

Table 4-2

File Sizes

File Number	Name	Pile File R	Pile File Size	Sequential File R	Sequential File Size
1	Family Master	102	1530000	180	2700000
2	Guarantors	117	2106000	179	3222000
300	Patient	315	9450000	2076	62280000
3:1	Entitlements	20	60000	19	57000
3:2	Insurance Plans	106	3180000	141	4230000
400	Visits	73	4380000	97	5820000
4:2	Drug Allergies	11	66000	10	60000
4:3	Other Allergies	86	316000	805	4830000
5:1	Problems	17	1530000	16	1440000
5:1:1	Actions	115	31050000	849	229230000
800	Medications	97	29100000	816	244800000
850	Non-drug Therapy	91	13650000	810	121500000
11:1	Problems Seen	18	3240000	17	3060000
11:1:1	Services Rendered	28	3360000	27	3240000
11:1:2	Tests Ordered	102	24480000	821	197040000
11:1:3	Problem Note	90	16200000	809	145620000
12	Appointments	12	12900	11	11880
12:1	Patients per Slot	31	135160	45	196200
13	Day Sheet	13	78	12	72
13:1	Patients per Office	28	5040	42	7560
	Totals (bytes)		144,051,238		1,029,344,712

Table 4-3

Lexicons and Referenced Entity files

File No:	Filename	n	Direct a/R	Filesize
19	Patient Name Index	30000	9/49	1470000
20	Family Number Index	15000	2/4	60000
20:1	Family Number	30000	3/5	150000
21	Family Name Index	15000	2/4	60000
900	Terms	1750	2/35	61250
14	User List	50	7/109	5450
22	Third Party Payors	160	2/35	3500
23	Surveillance Criteria	50	2/803	40150
24	Problem Classification	250	2/35	8750
30	Family Physicians	200	2/159	31800
31	Hospital Service Code	700	2/6	4200
32	Therapy Classification	300	2/5	1500
	Total (bytes)			1,896,600

Only the Patient Name Index is substantial in size, and will be analyzed in a later section as an example of direct file design.

Additional storage will be required for the service files estimated as follows:

Table 4-4

Service Files

File No:	Filename	Records n	Direct a/Rf(or v)	Filesize
3b	Temp: Bill Data	3000	4/16	48000
7	Medical Audit	2000	2/6	12000
7:1	Audit Columns	20000	27/38	760000
15	Transaction Log	500	11/28	14000
15:1	Old and New Data	2500	3/11(v)	27500
16	Fiscal Audit	3000	11/28	84000
17	Messages	50	5/89(v)	4450
18	System Messages	20	5/89(v)	1780
29	Scheduled Action	20	2/23	460
	Total (bytes)			952,190

In addition, there will be storage required for system and application programs, workspaces for sorting and reorganization of files, etc.; At this point 2,000,000 bytes will be allocated to this function.

INITIAL PERFORMANCE DESIGN CONSIDERATIONS

From Table 4-1 and 4-2 the following observations can be made:

The aggregate file size is such that simple sequential file organization methods use an excessive amount of storage space.

Much space in the sequential file organization is due to the maximum length of the notes (800 characters).

The files in either case are of such a size that high capacity file storage devices will be necessary.

The last observation leads us to the use of file device parameters in the further analysis which describe disk devices with a capacity of about 50M bytes/device. Without further reduction of storage needs four to twenty-one of these devices will be required. Higher density devices (100M or 200M byte/disks) require more powerful processors than the ones considered for the Family System [3:pp.187-202]. The characteristics used in the performance analyses below will be

seek time	s	: 35 milliseconds
latency	r	: 17 milliseconds
transfer rate	t	: 312 bytes/millisecond
interblock gap	G	: 193 bytes

If devices with different characteristics are eventually chosen, then the final design should be re-verified.

Other aspects of the storage requirements will now be addressed.

USAGE AND LOAD

In order to design a system which is not only functionally satisfactory but also adequate in terms of its performance, the expected load generated during usage of the system has to be considered. The problems arising from inadequate attention to this aspect of data base system design have been stated in Chapter 2 of this thesis.

Since the proposed Family System is intended largely as a service in a production environment (the product is health care), it is possible to base the estimation of system load largely on the volume of services to be provided. There is also an aspect of information services to the Family System and as indicated in [1:Ch.5.1], the usage of the Family System for information purposes will depend on the quality of services to clinic, research and educational management. The load due to this latter type of usage can be expected to be considerably less and not coincide with times of high clinic activity. This aspect will hence not now be evaluated.

The services that are to be provided are selected from [2:Sec.4C], and usage qualities are provided by [3:Ch.2] and Table 3-1.

Table 4-5

User Services

a) Patient Profile : once per visit, prior to encounter;

Read from files : 19, 20, 300, 3:1, 3:2, 400, 4:2, 4:3, 5:1, 5:1:1, 800, 850, 11:1:1, 11:1:2, 11:1:3, 13, 14, 900, 23, 24, 30

Update files : 15:1, 13:1

b) Encounter Reports : once per visit, after encounter;

Read from files : 300, 5:1, 7, 13, 14, 16, 900, 30, 31, 32

Update files : 3:1, 3b, 400, 5:1, 5:1:1, 800, 850, 11:1, 11:1:1, 11:1:2, 11:1:3, 13:1, 15, 15:1

c) Encounter Documents for Surveillance, Preventive Care, Fiscal and Medical Audit : once per visit, delayed as scheduled;

Read from files : 1, 2, 3b, 300, 3:1, 3:2, 400, 4:2, 4:3, 5:1, 5:1:1, 7, 800, 850, 11:1, 11:1:1, 11:1:2, 11:1:3, 13, 13:1, 900, 23, 24, 29, 31, 32

Update files : 300, 400, 7:1, 16, 15, 15:1

d) Medical Management to Control Services, Selection for Schedules, Surveillance, Flow Sheet, Audit, Allergy Warnings: several times per day (10% of visits);

Read from files : 19, 20, 300

Update files : 300, 4:2, 4:3, 12, 7, 17, 900, 23, 24, 30, 15, 15:1, 32

e) Medical Data Base, Enter New Patients or Update Patients: for new patients (10/day avg; and their updates 20/day);

Read from files : 1, 2, 19, 20

Update files : 1, 2, 19, 20, 21, 300, 3:2, 4:2, 4:3, 15, 15:1

f) Flowsheets for 5000 Patients : once per visit for 20% of visits prior to encounter;

Read from files : 300, 400

(Data are updated as part of b);)

g) Laboratory Results (orders are entered as part of b);) : one per visit average, when ready;

Read from files : 19, 300, 400, 2:3

Update files : 400, 11:1:2, 15, 15:1

h) Scheduling and Work List : daily and by call (250/day) including cancellations;

Read entire files : 12, 12:1, 13, 13:1, 19, 20, 300, 21

Update files : 12:1, 13, 13:1

i) Utilization Reports : weekly;

Read entire files : 12, 12:1, 400, 800, 850, 11:1, 11:1:1, 11:1:2, 24, 900

j) Practice Profile : weekly;

Read entire files : 12, 12:1, 1, 2, 300, 3:1, 3:2, 400; 4:2; 4:3, 5:1, 5:1:1, 800, 850, 32, 900

k) Disease Profile : weekly

Read entire file : 12, 12.1, 5:1, 11:1, 11:1:3, 24, 32, 900

l) Visit Reminders : daily, for scheduled visits made more than one week ago (100/day);

Read from files : 12, 12:1, 1, 2, 300, 3:1, 5:1, 30

m) Immediate Billing : once per visit;

Read from files : 2, 300, 3b, 900, 22, 30, 31

Update files : 15, 15:1, 16

n) Report to Outside Agencies : monthly. The work can be distributed over the 4:35 weekends per month;

Read entire files : 1, 2, 300, 3:2, 400, 5:1, 5:1:1, 800, 850, 11:1, 11:1:1, 11:1:2, 12, 12:1, 900, 29, 30, 24, 32

o) Bill Preparation : rotate through file, equivalent to visits/day;

Read entire file : 2 in parts;

Read from files : 3b, 22, 31

Update files : 2, 16

p) System Management : irregular, as required, avg: once/week;

Update file : 12, 14, 16, 22, 29, 31, 15; 15:1

q) System Maintenance and Audit : weekly;

Read entire files : 14, 15, 15:1, 16, 18, 29

r) Messages : 100/day

Read from files : 17, 18

Not considered for implementation now are some of the services seen during the AAMRS study [2]:

Pharmacy Labels
Automatic Referral Letters
Statistical Analysis of Data
Graphical Presentation of Data

The usage due to the services specified can be applied to the Family System files as shown in Table 4-6: This table is obtained by transposition of the data from Table 4-5:

R indicates file read
U indicates file read and updated
X indicates exhaustive file reading
Y indicates reorganization:

The monthly report generation, since it can be distributed over the weekends, has been taken as a weekly load of 12/52=0:23 of the monthly load:

Table 4-6

File Usage Due to Services

File Number	Name		Daily R svc	Daily R freq	Daily X svc	Daily X freq	Daily U svc	Daily U freq	Weekly X svc	Weekly X freq	Weekly Y svc	Weekly Y freq
PRIMARY FILES:												
1	Family Master		c	200	h	1	e	30	j	1		
			e	30					n	:23		
			h	250								
			l	100								
		total		580		1		30		1:23		-
2	Guarantors		c	200	o	:2	e	30	j	1		
			e	30					n	:23		
			l	100								
			m	200								
			n	200								
			o	200								
		total		930		:2		30		1:23		
300	Patients		a	200					j	1		
			b	200					n	:23		
			c	200			c	200				
			d	20			d	20				
			e	30			e	30				
			f	40								
			g	200								
			h	250	h	1						
			l	100								
			m	200								
		total		1440		1		250		1:23		-
3:1	Entitlements 10% update ratio		a	200			b	20	j	1		
			b	200								
			c	200								
			h	250	h	1						
			l	100								
		total		950		1		20		1		-
3:2	Insurance Plans		a	200			e	30	j	1		
			c	200					n	:23		
			e	30								
		total		430		-		30		1:23		-
400	Visits 25% update ratio for flowsheets, etc.		a	200			b	200	i	1		
			b	200			c	50	j	1		
			c	200			g	200	n	:23		
			f	40								
			g	200								
		total		840		-		450		2:23		-

<Continued>

File Number	Name		Daily R svc	Daily R freq	Daily X svc	Daily X freq	Daily U svc	Daily U freq	Weekly X svc	Weekly X freq	Weekly Y svc	Weekly Y freq
4:2	Drug Allergies		a c d e	200 200 20 30			d e	20 30	j	1		
		total		450	-			50		1	-	
4:3	Other Allergies (as 4:2)	total		450	-			50		1	-	
5:1	Problems		a b c l	200 200 200 100			b	200	j k n	1 1 :23		
		total		700	-			200		2:23		
5:1:1	Actions		a b c	200 200 200			b	200	j n	1 :23		
		total		600	-			200		2:23		
800	Medications		a b c	200 200 200			b	200	j j n	1 1 :23		
		total		600	-			200		2:23		
850	Non-drug Therapy (as 800)	total		600	-			200		2:23		
11:1	Problems Seen		a b c	200 200 200			b	200	i k n	1 1 :23		
		total		600	-			200		2:23	-	
11:1:1	Services Rendered		a b c	200 200 200			b	200	i n	1 :23		
		total		600	-			200		1:23		
11:1:2	Tests Ordered		a b e g	200 200 200 200			b g	200 200	i n	1 :23		
		total		800	-			400		1:23		
11:1:3	Problem Note		a b c	200 200 200			b	200	k	1		
		total		600	-			200		1	-	

<Continued>

File Number	Name	Daily R svc	Daily R freq	Daily X svc	Daily X freq	Daily U svc	Daily U freq	Weekly X svc	Weekly X freq	Weekly Y svc	Weekly Y freq
12	Appointments	d	20			d	20	i	1		
		h	250		1			j	1		
		l	100					k	1		
								n	.23		
								p	1		1
	total		370		1		20		4.23		1
12.1	Patients per Slot	h	250	h	1	h	250	i	1		
		l	100					j	1		
								k	1		
								n	.23		
								p	1		1
	total		350		1		250		4.23		1
13	Day Sheet	a	200	h	1	h	250				
		c	200								
		h	250								
		m	200								
	total		850		1		250		-		-
13.1	Patients per Office	a	200	h	1	a	200				
		b	200			b	200				
		c	200			h	250				
		h	250								
		m	200								
	total		1050		1		650		-		-
LEXICONS:											
19	Patient Name Index	a	200	h	1	e	30				
		d	20								
		e	30								
		r	200								
		h	250								
	total		700		1	e	30		-		-
20	Family Number Index	a	200	h	1	e	30	j	1		
		d	20					n	.23		
		h	250					p	1		
		e	30								
		l	100								
		o	200								
	total		800		1		30		2.23		-
21	Family Name Index	h	250	h	1	e	30				
		e	30								
	total		280		1		30		-		-

<Continued>

File Number	Name	Daily R svc	Daily R freq	Daily X svc	Daily X freq	Daily U svc	Daily U freq	Weekly X svc	Weekly X freq	Weekly Y svc	Weekly Y freq
900	Terms mult: ref: are req: given the freq: of item from Table 3-1 :2+:2+4+2+4+2+3+2+4=21:4	a b c d m	4280 4280 4280 428 4280			d	428	i k n	21:4 21:4 4:9		
		total	17548	-			428		47:7		-
REFERENCED ENTITY FILES:											
14	User List	a b c d e f g h l m o r	200 200 200 20 30 40 200 250 100 200 200 100					p	1	p	1
		total	1740	-		-			1		1
22	Third Party Payors	m o	200 200					p q	1 1		1
		total	400	-		-			2		1
23	Surveillance Criteria	a c d	200 200 20			d	20				
		total	420	-			20		-		-
24	Problem Classification	a c d	200 200 20			d	20	i k r	1 1 :23		
		total	420	-			20		2:23		-
30	Family Physicians	a b d l m	200 200 20 100 200			d	20	n	:23		
		total	720	-			20		:23		-
31	Hospital Service Code	b c m o	200 200 200 200					p	1	p	1
		total	800	-		-			1		1

<Continued>

File Number	Name	Daily R svc	Daily R freq	Daily X svc	Daily X freq	Daily U svc	Daily U freq	Weekly X svc	Weekly X freq	Weekly Y svc	Weekly Y freq
32	Therapy Classification	b	200			d	20	j	1		
		c	200					k	1		
		d	20					n	23		
	total		420		-		20		2:23		-
SERVICE FILES:											
3b	Temporary Billing Data	b	200			b	200				
		c	200								
		m	200								
		o	200								
	total		800		-		200		-		-
7	Medical Audit	b	200			d	20				
		c	200								
		d	20								
	total		420		-		20		-		-
7:1	Medical Audit Detail	c	200			c	200				
		d	20								
	total		220		-		200		-		-
15	Transaction Log Writing is sequential without reading					b	200	q	1	p	1
						c	200				
						d	20				
						e	30				
						f	200				
						m	200				
	total		-		-		850		1		1
15:1	Transaction Log Detail use 5 items average (from Table 3-1) for services b to m					a	200	q	1	p	1
						b	1000				
						c	1000				
						d	100				
						e	150				
						g	1000				
						m	1000				
	total		-		-		4450		1		1
16	Fiscal Audit writing is sequential without reading for files b, c, m	b	200			b	200	q	1		
						c	200				
						m	200				
						o	200				
	total		200		-		800		a		-

<Continued>

File Number	Name	Daily R svc	Daily R freq	Daily X svc	Daily X freq	Daily U svc	Daily U freq	Weekly X svc	Weekly X freq	Weekly Y svc	Weekly Y freq
17	Messages	d r	20 200			d	20				
	total		220	-			20		-		-
18	System Messages	r	200					p q	1 1	p	1
	total		200	-			-		2		1
29	Scheduled Actions	a c	200 200					n p q	:23 1 1	p	1
	total		400	-			-		2:23		1
	Grand totals		39698		(10:2)		11018		(98:07)		(9)

A review of Table 4-6 can verify the data flow in the Family System files: The verification shows that all files are used and updated: Most files are updated only during one function: In order to control the data flow where files are updated by more than one function and avoid deadlock [1:Ch:13:2], the control over data will have to be respecified in finer units: Two choices are possible:

1) control by record segmenting
2) control by time constraints

For instance, the indicators that a patient is to be placed on a flowsheet protocol, as given above as

300 Patient d/U = 20

reside in a segment of the Patient file which is not to be updated by any other function: On the other hand the updating of flowsheet entries after a visit,

400 Visits c/U = 50

is delayed in time until all other encounter activity has been recorded:

While the grand totals of the Read(R), Read-entire(X), Update(U), and Reorganize(Y) activity were recorded Table 4-6; it should be noted that the totals shown are not particularly useful for performance evaluation; For each file the time used for each action will depend on the file organization, its size, and the recordsize to be retrieved; In particular, Read-entire(X) and Reorganize(Y) are dependent on file size; From the grand total of Read(R) and Update(U), namely approximately 50,000 actions/day, it is evident, however, that each request has to be fulfilled in considerably less than a second, since there are only 28,800 seconds in a working day;

In the next section of this chapter the file design will be refined with this objective in mind;

STORAGE OF NOMINAL DATA

Omitting the notes or removing them to indirect storage, as paper or microform, could reduce storage requirements considerably; Notes appear in the following files:

Table 4-7

Stored Notes

File Number	Name	n	Notes (source)
300	Patients	30000	2(3,4)
4:3	Other Allergy	6000	1(4:3)
5:1:1	Action	270000	1(5:1:1)
800	Medication	300000	1(8:1, 9:1)
850	Non-drug Therapy	150000	1(8:2, 9:2)
11:1:2	Tests Ordered	240000	1(11:1:2)
11:1:3	Problem Notes	180000	1(11:1:3)
	Total Notes		1,206,000

In the case of variable length or compressed files the average note length is 80 characters; otherwise notes are constrained to a maximum of 800 characters, so that

Storage for variable length notes:	96,480,000 chars;
Storage for maximum length notes:	964,800,000 chars;

The equivalent storage costs at $300/Mbyte year are

variable	$ 28,964/year
maximum	$ 289,440/year

Alternative means to store notes are hence very desirable; A microform storage could be used for such a purpose and be indexed from the file system at a cost of a few characters per entry:

index to notes at 4 characters	
storage	4,824,000 chars;
cost	$1,448/year

A microform system could also be used to store other reference documents associated with the medical record of the Family Practice Center;

If notes are to be kept on the Family System, then variable length storage is desirable, and notes of short average length are preferable from the computer storage point of view; This may conflict with the educational purposes of the Family Practice Center;

In order to make the subsequent steps of the file design independent of the manner in which the notes are kept, the notes within the files will be replaced with reference pointers. These could then be used to refer to notes stored on a cheap high volume file, to a microform storage, or to a paper document.

Only in the Other Allergy File (4.3), which is relatively small, will the notes be retained since rapid reference here may be essential. With further experience a suitable encoding may also reduce the average note size.

The files affected by keeping notes separate are listed in Table 4-8. These entries replace the equivalent entries of Table 4-1.

Table 4-8

Files Shrunk by Removing Notes

File Number	Name	Size n	Record Format af/Rf	av/Rv(avg,max)
300	Patients	3000(2)	13/29	124/(135,457)
5.1.1	Actions	270000	9/21	1/(17,32)
800	Medications	300000	13/20	-
900	Non-drug Therapy	150000	6/14	-
11.1.2	Tests Ordered	240000	11/25	-
11.1.3	Problem Note	180000	5/13	-
Storage reduction		1,146,000 notes		

or (pile file org.) 87,096,000 characters (about 2 disk units)

(seq. file org.) 912,219,000 characters (about 19 disk units)

Remaining storage requirements are still

	Pile Org:	Seq: Org:	
data files	56,955,238	117,128,712	
lexicons, etc:	1,896,660	1,896,660	
service files	952,190	952,190	
programs	2,000,000	2,000,000	
total	61,804,028	121,977,502	bytes
or	< 2 disk units	< 3 disk units	

CHOICE OF FILE ORGANIZATION

The file organization chosen for a data file is the primary determinant of performance for the data base application: In [1:Ch:3] six basic file organizations are presented and evaluated: Since the Multiply-Indexed File is oriented towards complex fact retrieval [1:Ch:10:1], this organization will not be evaluated here: Some variants are shown in Chapter 4 [1]: In order to select a file organization for the major files of the Family System the performance of the basic file systems will be evaluated: One of the hybrid methods, MUMPS, will also be analyzed due to its popularity in medical data processing:

This evaluation uses throughout the same file system parameters for

blocksize	: B = 2000
blockpointer size	: P = 4
reorganization frequency	: weekly

so that the evaluation will be independent of low level system parameters:

The use of unspanned records will simplify the file access method; the wasted space per record is then

$$W = (G+1/2R+P)\ R/B + P \qquad [1:Ch;2;2;3]$$

and the bulk transfer rate is

$$t' = \frac{R}{R+W} * t$$

The records specified for the Family System [3] have a well-defined variable content; Because of this fact records will not contain attribute names in any of the files being considered, although the MUMPS files will include the descriptive subscript values for the globals [1:Ch;4;5;2]; For variable sized records, the length (Rvar) includes field separation markers;

The number of weekly additions (o) of records to the files is based on the data from the Family System [3:Ch;2], as given in Tables 3-1 and 3-3; Subsequent evaluations in this chapter find that the parameters T_N (get-next data record) and T_U (update a field of a data record) are of minor importance due to the usage patterns expected; Search for the next record is an important activity in subset searching and is associated with complex data analysis; Medical records are mainly maintained by adding to the files so that updating of old data records is infrequent;

Descriptive File Parameters:

For the various primary files to be considered, the descriptive parameters, when computed using the conditions cited above, are displayed in Table 4-9;

Table 4-9

Descriptive Primitive File Parameters

File No:	a	Rvar,Rfix	n	o/Week	w	t′
1	5	102,180	15000	29	16:648	268222
2	15	117,179	18000	35	18:947	268517
300	137	162,484	30000	58	26:518	268112
3:1	5	20	3000	115	6:070	239356
3:2	12	106,141	30000	116	17:250	268333
400	61	73,97	60000	1150	12:523	266315
4:2	3	11	6000	12	5:114	212986
4:3	3	86,805	6000	12	14:320	267464
5:1	7	17	96000	575	5:747	233176
5:1:1	10	38,53	270000	2300	8:104	257158
800	13	20	300000	2300	6:070	239356
850	6	14	150000	1150	5:428	224830
11:1	10	18	180000	3450	5:854	235432
11:1:1	8	28	120000	6900	6:954	249928
11:1:2	11	25	240000	13800	6:619	246689
11:1:3	5	13	180000	3450	5:323	212364
12	7	12	1080	1	5:218	217447
12:1	7	31,45	4360	4360	7:294	252574
13	6	13	6	7	5:323	212364
13:1	4	28,42	180	1260	6:954	249928

A Pile File Organization:

The descriptive parameters form the basis for the study of the alternative file organzation forms: For an initial upper bound a pile file organization is defined [1:Ch:3:1] with

Record fetch time	$T_F = 1/2\ nRt' + s + r$
Record get-next time	$T_N = T_F$
Record insert time	$T_I = s + 3r + btt$
Record update time	$T_U = T + T_I + 2r$
Time to read entire file	$T_X = 2\ T_F$
Reorganization Time	$T_Y = 2(n+o)\ R/t'$

The pile file organization is also of interest since it models closely the performance of a data base system which is built on 'relational principles' [1:Ch:9:1] and which has not been further augmented with ancilliary access paths:

The results for the Primary Files are shown in Table 4-10:

Table 4-10

Pile File Performance Measures

File No:	T_F	T_N	T_I	T_U	T_X	T_Y
1	2:904	:092	3:031	5:808	11:639
2	3:974	:092	4:100	7:947	15:925
300	9:115	:092	9:242	18:231	36:532
3:1	0:117	:092	0:304	0:355	0:729
3:2	5:977	:092	6:104	11:955	24:002
400	8:275	:092	8:402	16:551	33:732
4:2	0:207	:092	0:333	0:414	0:829
4:3	1:017	:092	1:143	2:033	4:074
5:1	3:552	:092	3:678	7:103	14:290
5:1:1	20:001	:092	20:127	40:002	80:683
800	12:586	:092	12:712	25:171	50:727
850	4:722	:092	4:849	9:444	19:032
11:1	6:933	:092	7:059	13:866	28:259
11:1:1	6:774	:092	6:900	13:548	28:642
11:1:2	12:213	:092	12:339	24:426	51:649
11:1:3	5:337	:092	5:464	10:675	21:755
12	0:082	:092	0:208	0:164	0:327
12:1	0:320	:092	0:446	0:639	2:349
13	0:058	:092	0:179	0:104	0:210
13:1	0:062	:092	0:188	0:124	0:531

The retrieval times for the primary files using the pile file organization are for most files much greater than allowable as established by the approximate criterion derived from Table 4-6: In fact for these primary files alone, the time for daily read

from file and file update operations is

$$T_{Total} = \sum_{p} T_{Fp} L_{Rp} + \sum_{p} T_{I} L_{Up}$$

$$= 75009 + 357 = 75,366 \text{ seconds}$$

$$\text{or } 20.94 \text{ hours} = 20 \text{ hours } 56 \text{ minutes per day}$$

It can be seen, however, that files

3.1	Entitlements
4.2	Drug Allergies
12	Appointments
13	Day Sheet
13.1	Patients per Office

are such that no further file organization can improve their performance significantly. The minimal read access time for the hardware being considered is

$$s + r + B/t = .058 \text{ sec.}$$

and the minimal update access time is

$$s + 3r + B/t = .092 \text{ sec.}$$

They will be omitted from further consideration and can either be implemented as simple pile files or according to any other convenient general organization. As pile files their aggregate time requirements will be as shown in Table 4-11. Values used are T_I or T_U as appropriate for the services being performed.

Table 4-11

Time to be Allocated to Small Files

File		T_F	L_R	T_X	L_{Xd}	L_{Xw}	$T_{I/U}$	L_U	T_Y	L_Y
3:1	Entl	:117	950	:355	1	1	:092	20	na	—
4:2	DrAl	:207	450	:414	-	1	:092	50	na	—
12	Appt	:082	370	:164	1	4:23	:208	20	:327	1
13	DaySh	:052	850	:104	1	-	:179	250	na	—
13:1	Pat/C	:062	1050	:124	1	-	:188	650	na	—

$$\text{daily time} = T_F L_R + T_X L_{Xd} + T_{I/U} L_U$$

$$= 344 + 1 + 190 = 533 \text{ sec} = 9 \text{ min/day}$$

$$\text{weekly time} = T_X L_{Xw} + T_Y L_Y = 2 \text{ sec/week}$$

Sequential, Indexed-Sequential, and Direct File Organizations:

Faster access according to one attribute is afforded by the sequential, indexed-sequential, and direct file organizations. Of major interest are here the retrieval speed T_F and the file update speed T_I. Individual record update is of interest only for the Appointment Detail file (12:1). The values obtained for the pile file organization in regard to T_X and T_Y can provide initial guidance.

In each case the computational overhead, c, is not included.

Sequential File Organization:

For access to the sequential file [1:Ch:3:2:3], the algorithm will consist of a binary search through the main file and a serial search through the transaction log file. No buffer is kept available for the transaction file so that the update response time is equal to that for a pile file. All updates are collected into transaction files for the day, and at the end of the day these files will be sorted and merged into the main sequential files. The number of daily updates is based on the daily load L_U and is related to the file growth, o. For the primary files the time required for this process, T_c, will be mainly a function of the file sizes.

The sort phases will require approximately

$$T_{sort} = L_U \log_2 L_U \ (T_F + T_I)$$

where T_F and T_I pertain to small pile files.

The merge phases will require approximately

$$T_{merge} = (n+L_U)\ R/t' \qquad \text{and} \qquad T_C = T_Y = T_{sort} + T_{merge}$$

The value of L_U can be obtained from Table 4-6. An average bulk transfer rate, $t' = 220{,}000$ bytes/second, gives $T_F = .052$, $T_I = .092$ files which contain the daily update (o/5) transaction. The

combined results are given also in Table 4-12;

$$T_F = \lceil \log_2(n\ R/B) \rceil\ (s+r+B/t) + 1/2\ o/5\ R/t'$$

$$T_N = r + B/t$$

$$T_I = s + 3r + B/t$$

Indexed-Sequential:

An indexed-sequential file organization provides faster access through a tree search of key values [1:Ch:3:2:3]; Its efficiency depends on a small number of updates per file; Some additional space, SI; is required; This space is a function of the fanout ratio, y, [1:Ch:3:3:1], blocking, and number of records, n,

$$y = \left\lceil \frac{B}{Rr+P} \right\rceil$$

$$i_1 = \left\lceil \frac{n}{\lfloor B/R \rfloor} \right\rceil$$

and $$SI = \left(\lceil i_1/y \rceil + \lceil \lceil i_1/y \rceil / y \rceil \right) B$$

for a two-level (x=2) index; File 12:1 only requires one level; The values for Rr depend on the size of the ruling part and are given in Table 3-7;

$$T_F = s + (2 + Pov\ (1+1/2Pov)(r+B/t) \quad ; \quad Pov = o/(n+o)$$

$$T_N = (\ R/B + 2\ Pov\ (1-R/B)\ )(r+B/t) \quad \text{(two buffers are available)}$$

$$T_I = T_F + 5r + B/t$$

Table 4-12

Sequential and Indexed-Sequential File Performance Measures

File No.	Sequential T_F	Sequential T_N	Sequential T_I	Sequential T_C	Indexed-Sequential T_F	Indexed-Sequential T_N	Indexed-Sequential T_I	SI
1	0.6947	.0234	.092	33.5	.0819	.0020	.1733	14000
2	0.7052	"	"	27.1	.0819	.0022	.1733	22000
300	1.0401	"	"	353.3	.0819	.0057	.1733	94000
3.2	0.8645	"	"	40.4	.0819	.0018	.1733	26000
400	1.8164	"	"	597.8	.0823	.0020	.1737	34000
4.3	0.7975	"	"	62.8	.0819	.0095	.1733	30000
5.1	0.6819	"	"	227.6	.0820	.0005	.1734	12000
5.1.1	1.9783	"	"	285.2	.0820	.0010	.1734	90000
800	0.9857	"	"	247.4	.0820	.0004	.1734	42000
850	0.8035	"	"	230.4	.0820	.0005	.1734	16000
11.1	1.2635	"	"	234.9	.0823	.0011	.1737	22000
11.1.1	2.5745	"	"	235.4	.0831	.0028	.1745	26000
11.1.2	4.1509	"	"	525.2	.0831	.0028	.1745	46000
11.1.3	1.0910	"	"	230.8	.0823	.0010	.1737	18000
12.1	1.9614	"	"	287.7	.0965	.0234	.1879	2000

Total time for daily cleanup 3620.0 or approx. 1 hour

Total space for index 494,100

Direct File Organization:

For a direct file the performance depends mainly on the extra storage allocation (m/n) and on the bucketsize [1:Ch.3.5.3]. With a fixed blocksize the blocking factor B/R is especially important. With unspanned blocking of records there is an additional loss of space so that required space is recalculated here as

$$\text{basic filesize} = \left(\frac{n}{\lceil B/R \rceil}\right) B$$

$$T_F = s + r + B/t + 1/2\ p\ (2r+B/t)$$

T_N is not available.

$$T_I = 2\ p\ T_F + s + 3r + B/t$$

p = funct(B/R,m/n) as shown in Figure 2-2.

Larger values of m/n increase the storage cost and this increase is indicated.

There are instances where serial access (T_N) is vital. Before evaluating the direct file, the remaining primary files will be reviewed in order to eliminate from analysis of direct files those for which serial access is important. The services to be considered were listed earlier in Table 4-5.

Table 4-13

Suitability for Direct Access

Service	Access Controlling Process
a) Patient Profile	Keyed on individual patient name.
b) Encounter Report	" " " " "
c) Encounter Documents	Serial on day sheet (13,1).
d) Medical Management	Keyed on individual patient.
e) Medical Data Base	" " " "
f) Flowsheets	" " " " record.
g) Lab results	" " " " and test.
h) Scheduling	Serial access on appointment files (12, 12,1)
i) Utilization Reports	Weekly serial access on visits (400).
j) Practice Profile	" " " " " "
k) Disease Profile	" " " " problems seen (11,1).
l) Visit Reminders	Serial access on appointment (12,1)
m) Immediate Billing	Keyed on individual patient.
n) Reports to Outside Agencies	Read entire Patient file.
o) Bill Preparation	Read file 3b serially.
p) System Management	Read file 12 and non-primary files serially.
q) System Maintenance	Read non-primary files.
r) Messages	Read files 17, 18 serially

A review of these requirements show that of the primary files only files Visits (400), Problems Seen (11,1), and Appointment Detail (12,1) should be eliminated as candidates for direct file organization. The remaining primary files are presented in Table 4-14.

Table 4-14

Direct File Performance Measures

File No:	B/R	Basic Filesize	m/n 1:05 T_F	1:05 T_I	1:10 T_F	1:10 T_I	1:25 T_F	1:25 T_I
1	11	2727272	.0736	.2028	.0653	.1382	.0766	.2303
2	11	3272727	.0736	.2028	.0655	.1382	.0766	.2303
300	4	15000000	.1049	.5749	.0695	.1689	.0659	.1412
3:2	14	4285714	.0695	.1689	.0631	.1214	.0594	.0984
4:3	2	6000000	.1534	1.5341	.1089	.6370	.0764	.2284
5:1	117	1641026	.0592	.0971	.0587	.0942	.0584	.0925
5:1:1	37	14594595	.0625	.1174	.0599	.1012	.0589	.0951
800	100	6000000	.0592	.0973	.0588	.0948	.0584	.0925
850	142	2112676	.0590	.0960	.0587	.0938	.0584	.0925
11:1:1	71	3380282	.0596	.0995	.0589	.0954	.0585	.0926
11:1:2	80	6000000	.0595	.0986	.0587	.0950	.0584	.0926
11:1:3	153	2352941	.0590	.0960	.0587	.0938	.0584	.0925
Total File		67,139,961		70,496,960		73,853,957		83,924,952
Increase		-		3,356,999		6,713,994		16,784,991

Summary of Sequential, Indexed-Sequential, and Direct Primary Files:

A review of Table 4-12 shows that the sequential file organization with a binary search still does not reduce the access time for most files to a comfortable limit (<< 1 second). The additional hour required to clean up the files at the end of every day is another liability, although this operation could be combined with the creation of back-up files. Given a reasonably reliable operation and adequate performance of the Transaction Log file such an effort would not be required every day for backup purposes alone.

An indexed-sequential file, although more complex, shows a more acceptable level of performance. The penalty of space for the index is modest.

Table 4-14 presents the performance of direct files. For many of the files listed the performance is yet better, even at very low (1:05) excess space ratios. Only for files with large records (B/R < 10) is the performance less than that for indexed-sequential files. The extra space requirements are, however, in any case greater than those required for indexes.

In order to compare the three file organizations, the total daily usage time, UT, will be calculated:

$$UT_t = \sum_p L_{Pr} T_{Ftp} + \sum_p L_{Xp} T_{Xtp} + \sum_p L_{Up} T_{Itp} +$$

for t = S (sequential)
IS (indexed-sequential)
D (direct)

and p identifies the various files;

The following considerations apply to this summary calculation:

For the sequential file, L_{Xp} =1 for all files due to the cleanup required, and

$T_{XSp} = T_{Xp}$ for these files

For the direct files, the files identified

with p = 300 uses m/n = 1:10

and p = 4:3 uses m/n = 1:25

All other files use m/n = 1:05

Also $T_{XDp} = m/n\ T_{Cp}$ where applicable to account for the extra space to be read in a direct file exhaustive read operation (p = 1, 2, 300):

Furthermore, an indexed-sequential organization will be used for those files where a direct organization was not found feasible according to Table 4-13, so that

$$T_{FDp} = T_{FISp},$$

$$T_{XDp} = T_{XISp}, \text{ and}$$

$$T_{IDp} = T_{IISp};$$

for files identified with p = 400, 11:1, 12

With these considerations the daily primary file usage for the three file organizations is

UT_S = 14515 + 3620 + 265 = 18400 sec/day or 5 hours 6 mins/day

UT_{IS} = 837 + 392 + 504 = 1733 sec or 29 min/day

UT_D = 696 + 429 + 389 = 1541 sec or 25 min/day

This shows even more clearly that a sequential file, even with binary search and deferred update, is not adequate. The load presented by indexed-sequential or direct file for this aspect of file services is still significant. Another nine minutes per day will be required to serve the small primary files of Table 4-11, and yet more time is needed to provide for usage of the Lexicons; Referenced Entity Files; and Service Files;

The indexed-sequential and direct file organizations show a space versus speed tradeoff. For the 12 files of Table 4-14, at the m/n ratios chosen, the additional space required for direct files is

$$SAD = \sum_{p} \frac{m-n}{n} \text{filesize}_{Dp} = 5{,}306{,}998 \text{ bytes};$$

whereas for these same files the indexed-sequential organization

requires

$$SIS = \sum_{p} SI_p = 436{,}000 \text{ bytes}$$

or 4,871,000 bytes less.

The direct file organization applied to these 12 files saves 4 min/day at $.20/minute cost for the entire system. This equals $360/year and increases storage cost by 4,871,000 bytes at $300/Mbyte year or $1461.30/year.

With these values the indexed-sequential file seems to be preferable, but not to an overriding extent. When a specific file can make a significant contribution to system performance, then a direct file organization may well be justified.

RING OR HIERARCHICAL FILES

The sequential, the indexed-sequential, and the direct file depend on fixed size records, and there is hence an additional space penalty over the pile file. This penalty was documented following Table 4-8 and amounted for all primary files to 117,129,000 bytes versus 56,955,000 bytes. It remains hence desirable to investigate file organization methods which handle variable length data more efficiently. Prime candidates are here the indexed file organization [1:Ch.3.4.3], the ring file organization [1:Ch.3.4.6], and the MUMPS file structure [1:Ch.4.5.2].

Of special interest here are the Family System files that have a natural hierarchical structure, as shown by the data model presented as Figure 3-4. Table 4-15 presents again the familiar primary files, now with the parameters which are important to ring or hierarchical file design.

Table 4-15

Hierarchical Parameters

File	Filename	Parent	Entry	Level	Filesize n	Recordsize Ravg	Recordsize a	Fanout y
1	Family Master	none	yes	x=6	15000	102	5	-
2	Guarantors	1	yes	5	18000	117	15	1:2
300	Patient	2	yes	4	30000	62	137	1:67
3:1	Entitlements	300	no	3	3000	20	5	0:1
3:2	Insurance Plans	300	no	3	30000	106	12	1
400	Visits	300	yes	3	60000	73	61	2
4:2	Drug Allergies	300	no	3	6000	11	3	:2
4:3	Other Allergies	300	no	3	6000	86	3	:2
5:1	Problems	300	no	3	96000	17	7	2:3
5:1:1	Actions	5:1	no	2	270000	38	10	2:81
800	Medications	300	no	3	300000	20	13	10
850	Non-drug Therapy	300	no	3	150000	14	6	5
11:1	Problems Seen	400	yes	2	180000	18	10	3
11:1:1	Services Rendered	11:1	no	1	120000	28	8	:67
11:1:2	Tests Ordered	11:1	no	1	240000	25	11	1:33
11:1:3	Problem Note	11:1	no	1	180000	13	5	1
12	Appointments	none	yes	x=2	1080	12	7	-
12:1	Patients per Slot	12	yes	1	4360	31	7	3
13	Day Sheet	none	yes	x=2	6	13	6	-
13:1	Patients per Office	13	yes	1	180	28	4	60

The average natural fanout, y, appears to be very low.

Block-Oriented Hierarchies:

In some file implementations the rings or hierarchies are defined by the use of block pointers [1:Ch:2:3:3]. Each subsidiary ring will occupy a single block chain in a system which does not allow sharing of blocks, and at least one block is required for every subsidiary ring.

At the top level (x), B/R has to be compared with n, and here the ring may require many blocks. The total number of blocks is then

$$b = \sum_{p} \left\lceil \frac{n_p}{\lfloor B/R_p \rfloor} \right\rceil \quad | \text{ level}_p = x$$

$$+ \sum_{p} \frac{n_p}{y_p} \quad | \text{ level}_p < x \text{ ; and } y_p > 1$$

$$+ \sum_{p} n_p \quad | \text{ level}_p < x \text{ , and } y_p < 1$$

$$= 833{,}724 \text{ blocks}$$

This is equivalent to 1667 Mbytes of file storage or 34 disk units for the primary files alone, so it is obvious that the natural hierarchy will have to be reshaped if blocks cannot be shared.

A MUMPS Implementation of the Family System:

A MUMPS system [1:Ch.4.5.2] is an example of such a block-oriented hierarchy. While the storage requirements will be excessive, the same data base structure used up to this point will be used to illustrate the corresponding structure using MUMPS globals.

The structure of MUMPS also forces the use of a distinct level for ruling parts unless denesting has taken place. The transformation of the data base model to a MUMPS tree is shown in Table 4-16. The level numbers are assigned beginning from the directory level, d. Each level of the MUMPS hierarchy is related to the number of subscripts in the global variable. Each global variable defines one tree of the hierarchical data base model.

Table 4-16

Global Definitions

File	Tuple-Part	MUMPS-Level	Global Name :Contents
-	directory	d	^F :name of Family file
1	rul;pt	d-1	^F(f) :family_number
	dep;pt	d-2	^F(f,2) :family_name
	dep;pt	d-2	^F(f,3) :family_address
	dep;pt	d-2	^F(f,4) :family_telephones
	linkage	d-2	^F(f,5) :guarantor#1 denested
	linkage	d-2	^F(f,g) :guarantor#(g-4) denested
2	dep;pt	d-3	^F(f,g,4) :contract_period
	dep;pt	d-3	^F(f,g,a) :attribute a(4 to 15) of Guarantor record
	linkage	d-3	^F(f,g,16) :reference to Patients
300	rul;pt	d-4	^F(f,g,16,m) :family_member_number m
	dep;pt	d-5	^F(f,g,16,m,a) :attribute a(2 to 137) of Patient record
	linkage	d-5	^F(f,g,16,m,138) :reference to Entitlements
	linkage	d-5	^F(f,g,16,m,139) :reference to Insurance Plans
	linkage	d-5	^F(f,g,16,m,140) :reference to Visits
	linkage	d-5	^F(f,g,16,m,141) :reference to Drug Allergies
	linkage	d-5	^F(f,g,16,m,142) :reference to Other Allergies
	linkage	d-5	^F(f,g,16,m,143) :reference to Problems
	linkage	d-5	^F(f,g,16,m,144) :reference to Medications
	linkage	d-5	^F(f,g,16,m,145) :reference to Non-drug Therapy

<Continued>

File	Tuple-Part	MUMPS-Level	Global Name :Contents
3;1	rul;pt	d-6	↑F(f,g,16,m,138,e) :Entitlement e
	dep;pt	d-7	↑F(f,g,16,m,138,e,a) :attribute a(3 to 5) of Entitlements
3;2	rul;pt	d-6	↑F(f,g,16,m,139,i) :Insurance Plan i
	dep;pt	d-7	↑F(f,g,16,m,139,i,a) :attribute a(3 to 12) of Insurance Plans
400	rul;pt	d-6	↑F(f,g,16,m,140,v) :Visit v
	dep;pt	d-7	↑F(f,g,16,m,140,v,a) :attribute a(3 to 61) of Visit
	linkage	d-7	↑F(f,g,16,m,140,v,62) :reference to Problems Seen
4;2	ref;tuple	d-6	↑F(f,g,16,m,141,d) :Drug Allergy d
4;3	ref;tuple	d-6	↑F(f,g,16,m,142,o) :Other Allergy o
5;1	rul;pt	d-6	↑F(f,g,16,m,143,p) :Problem p
	dep;pt	d-6	↑F(f,g,16,m,143,p,a) :attribute a(3 to 7) of Problem
	linkage	d-7	↑F(f,g,16,m,143,p,8) :reference to Actions
5;1; 1	rul;pt	d-8	↑F(f,g,16,m,143,p,8,act) :Action act
	dep;pt	d-9	↑F(f,g,16,m,143,p,8,act,a) :attribute a(4 to 10) of Action
800	rul;pt	d-6	↑F(f,g,16,m,144,med) :Medication med
	dep;pt	d-7	↑F(f,g,16,m,144,med,a) :attribute a(3 to 13) of Medication
850	rul;pt	d-6	↑F(f,g,16,m,145,nd) :Non-drug Therapy nd
	dep;pt	d-7	↑F(f,g,16,m,145,nd,a) :attribute a(3 to 6) of Non-drug

<Continued>

File	Tuple-Part	MUMPS-Level	Global Name :Contents
11;1	rul;pt	d-8	↑F(f,g,16,m,140,v,62,ps) :Problem Seen ps
	dep;pt	d-9	↑F(f,g,16,m,140,v,62;ps,a) :attribute a(4 to 10) of Problems Seen
	linkage	d-9	↑F(f,g,16,m,140,v,62,ps,11) :reference to Services Rendered
	linkage	d-9	↑F(f,g,16,m,140,v,62,ps,12) :reference to Tests Ordered
	linkage	d-9	↑F(f,g,16,m,140,v,62,ps,13) :reference to Problem Notes
11;1 ;1	rul;pt	d-10	↑F(f,g,16,m,140,v,62,ps,11,sr) :Service Rendered sr
	dep;pt	d-11	↑F(f,g,16,m,140,v,62,ps,11,sr,a) :attribute a(5 to 8) of service Rendered
11;1 ;2	rul;pt	d-10	↑F(f,g,16,m,140,v,62,ps,12,to) :Test Ordered to
	dep;pt	d-11	↑F(f,g,16,m,140,v,62,ps,12,to,a) :attribute a(5 to 11) of Test Ordered
11;1 ;3	tuple	d-10	↑F(f,g,16,m,140,v,62,ps,13,pn) :Problem Note pn
-	directory	d	↑A :name of Appointment file
12	rul;pt	d-1	↑A(dt) :appointment date and time
	linkage	d-2	↑A(dt,of) :Office of
	dep;pt	d-3	↑A(dt,of,a) :attributes a(4 to 7) of Office
	linkage	d-3	↑A(dt,of,8) :reference to Patients per Slot
12;1	rul;pt	d-4	↑A(dt,of,8,pt) :Patient per slot
	dep;pt	d-5	↑A(dt,of,8,pt,a) :attributes a(4 to 7) of Patient

<Continued>

File	Tuple-Part	MUMPS-Level	Global Name :Contents
—	directory	d	↑D :name of Day Sheet file
13	rul;pt	d-1	↑D(of) :Office of
	dep;pt	d-2	↑D(of,a) :attributes a(2 to 6) of Office
	linkage	d-2	↑D(of,7) :reference to patients per Office
13;1	tuple	d-3	↑D(of,7,pt) :Patients per Office

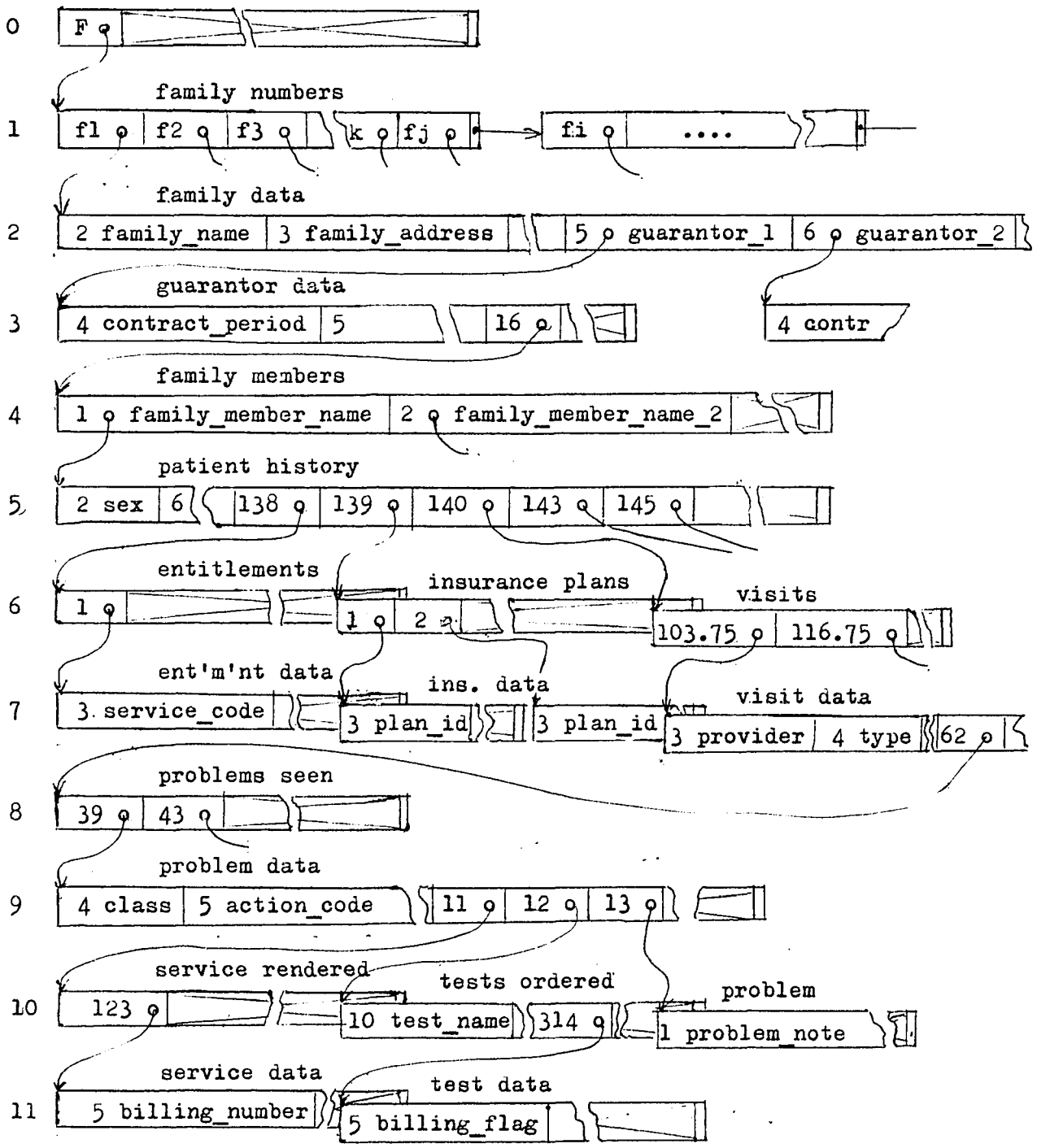
The tree is now up to 11 levels deep. The blocksize of MUMPS systems is less than the blocksize assumed up to this point, typically 512 or 768 bytes. But even these small blocks will be rarely well utilized if the original data base model is used. Figure 4-1 illustrates the layout for part of the primary hierarchy.

In a file structure with linkages the space for the linkage pointers has to be accounted for. The linkage pointers refer to subsidiary segments and the number of pointers at a level can hence be obtained as

$$a_{ptr_{level}} \geq y_{level-1}$$

In order to locate the individual fields of the globals, MUMPS uses a two-byte identification. Where the ruling part of the tuple contains only an identification field, no distinct value field will be required. The two requirements are combined in Table 4-17 in the column pid where

$$R_{pid} = 2a_{ptr} + 2a_{fld}$$



MUMPS Family System.

Figure 4-1

The segment length is then

$$R_{seg} = R_{avg} + R_{pid}$$

The keys or ruling parts of a file appear in sequence, so that a search can be made to locate the reference to the dependent part data; Multiple fields of each dependent part are also kept in sequence, each field with its distinct identification; As many blocks are chained together as are required, but at least one block is needed for each sequence; The number of blocks required is then

blocks per segment $bs = \lceil R_{seg} / B \rceil$

total blocks $b = n\ bs$

The values in the Table 4-17 are evaluated for B = 500;

One of the goals of the vairable-length segment architecture, better utilization of disk space is hence not achieved by a direct implementation of the data base model; The low density is evidenced by the fact that $R_{seg} << B$ for most of the segments shown in Table 4-17;

The value of R_{pid} for the MUMPS file organization did not affect the actual total file size significantly; The increase of segment size, as the file is now constituted, with a very low average block density will cause only very few blocks to overflow; The values of the subscript fields used to describe the entries also replaces some of the coded data fields now used to identify records and relationships;

Table 4-17

MUMPS Storage Requirements

File, part		n	a			R			blocks
			ptrs	values	fields	avg	pid	seq	
directory	d	1	1	-	1	0	4	4	1
1 Fam.									
rul;pt	d-1	1	15000	-	15000	0	60000	60000	120
linkage	d-2	15000	1;2	4;2	4;2	102	10;8	111	15000
2 Gua;									
dep;&lk;	d-3	18000	1	12	13	117	28	145	18000
300 Pat;									
rul;pt	d-4	18000	1;67	1;67	17		6;7	13;7	18000
dep;&lk;	d-5	30000	20;8	136	156;8	45	31	355	30000
3;1 Ent;									
rul;pt;	d-6	3000	1	-	1	0	4	4	3000
dep;pt;	d-7	3000	0	3	3	20	6	26	3000
3;2 Ins;									
rul;pt;	d-6	30000	1	0	1	0	4	4	30000
dep;pt;	d-7	30000	0	10	10	106	20	126	30000
400 Vis;									
rul;pt;	d-6	30000	1	0	1	0	4	4	30000
dep;&lk;	d-7	60000	1	58	59	73	120	193	60000
4;2 Drg;Alg;									
ref;tuple	d-6	6000	0	1	1	11	2	13	6000
Oth;Alg;									
ref;tuple	d-6	6000	0	1	1	86	2	88	6000
5;1 Probs;									
rul;pt;	d-6	30000	2;3	0	3	0	9;2	9;2	30000
dep;&lk;	d-7	96000	1	6	7	7	16	23	96000
5;1;1 Act;									
rul;pt;	d-8	96000	2;81	0	2;8	0	11;2	11;2	96000
dep;pt;	d-9	270000	0	7	7	38	14	52	270000
800 Meds									
rul;pt	d-6	30000	10	0	10	0	20	20	30000
dep;pt;	d-7	300000	0	11	11	20	44	64	300000
850 Non-drug									
rul;pt;	d-6	30000	5	0	5	0	20	20	30000
dep;pt;	d-7	150000	0	4	4	14	8	22	150000
11;1 Pr;S;									
rul;pt;	d-8	60000	3	0	3	0	12	12	60000
dep;&lk;	d-9	180000	3	7	10	18	26	44	180000
11;1;1 SvcRd									
rul;pt;	d-10	120000	1	0	1	0	4	4	120000
dep;pt;	d-11	120000	0	4	4	28	8	36	120000
rul;pt;	d-10	180000	1;5	0	1;5	0	6	6	180000
11;1;2 1st Ord;									
dep;pt;	d-11	240000	0	7	7	25	14	39	240000
11;1;3 PrNt									
tuple	d-10	180000	0	1	1	13	2	15	180000

<Continued>

File, part	n	a			R			blocks
		ptrs	values	fields	avg	pid	seq:	
directory d	1	1	0	1	0	4	4	same
12 Appt:								
rul:pt: d-1	1	1080	0	1080	0	4360	4360	5
linkage d-2	1080	4	0	4	0	16	16	1080
dep:pt: d-3	4360	1	4	5	12	12	24	4360
12:1 Pat/S								
rul:pt: d-4	4360	1:5	0	1:5	0	4:6	4:6	4360
dep:pt: d-5	5000	0	4	4	31	8	39	5000
directory d	1	1	0	1	0	4	4	same
13 Day S:								
rul:pt: d-1	1	6	0	6	0	12	12	1
dep:&lk: d-2	6	1	5	6	13	14	27	6
13:1 Pat/O								
rul:pt: d-3	6	0	30	30	28	60	78	6
				Total				2,345,939

23:5 disk units of 50 Mbytes

The hierarchical structure also makes the high order attributes low level ruling parts unnecessary, since these connections are explicitly expressed by pointers. Omitting these attributes, however, has an undesirable effect on structural integrity, since now the linkages become essential [1:Ch:13:3:2]. If the hierarchy is transformed to provide a higher storage density, then the space taken by descriptors and pointers will have to be reconsidered.

The access speed in MUMPS system is mainly a function of the number of levels. Given full compute and file accessing overlap, and an optimal record placement (possible when the files are not densely utilized), the access speed can be as fast as

$$T_{Fd} = s + r + \lceil 1/2\ bs \rceil (B+W)/t' \quad \text{for the top level}$$

and

$$T_{Fi-1} = T_{Fi} + \lceil 1/2\ bs \rceil (B+W)/t' \quad \text{for level } i = d \text{ to bottom-1}$$

Note that the use of t' is appropriate since the search sequence extends contiguously over block boundaries.

Table 4-18

Levels in a MUMPS File

Tree	Level	Dependent Parts	bs	T_F
1	d-2	1	150	.222
1	d-3	2	1	.223
1	d-5	300	1	.226
1	d-6	4:2, 4:3	1	.227
1	d-7	3:1, 3:2, 400, 5:1, 800, 850	1	.228
1	d-9	5:1:1, 11:1	1	.230
1	d-10	11:1:3	1	.231
1	d-11	11:1:1, 11:1:2	1	.232
2	d-3	12	5	.061
2	d-5	12:1	1	.063
3	d-2		1	.054
3	d-3		1	.055

The large file space used in this design provides fast access. A reconstitution of the data base model in order to reduce the excessive storage requirements will increase the expected fetch times. An optimal design from the performance point of view may achieve an acceptable balance of access speed and filespace; the data base model will then be quite different and should be re-evaluated to assure that it satisfies the needs of the applications which led to its construction.

Record-Oriented Hierarchies:

In a ring structures file which allows multiple record types per block, the storage can be used very efficiently. Only space for pointers, record descriptors, and record separators is needed, and as pointed out earlier, the pointers can replace some redundant ruling part attributes if essential links are permissible. The storage requirement will then be very similar to that of the pile file (Table 4-2).

The access problem to the top level (x) is solved in ring-structured file system generally by the use of direct access. The results from Table 4-14 are hence usable for files 1, 12, 13. Subsidiary rings are accessed by following individual pointers. Since the rings are short (y is small), the fetch time between

rings (i+1, i) has to be taken precisely

$$T_{Fi} = \left(\frac{1+y_i}{2}\right)(s+r+B/t) \quad | \; y > 1$$

$$+ (s+r+B/t) \quad | \; y < 1$$

and

$$T_F = T_{Fx} + \sum_{i=x-1}^{level} T_{Fi} \qquad \text{for a file at level=level}$$

Similarly, to read the entire file all rings on a given level and their ancestors have to be accessed:

On level i the required time is

$$T_{Xi} = \frac{n_1}{y_i}(1+y_i)(s+r+B/t)$$

so that

$$T_X = T_{Xx} + \sum_{i=x-1}^{level} T_{Xi} \qquad \text{for a file at level=level}$$

The ringstructure required by the Family System specification is simply hierarchical rather than of network type: Adding records in this structure is a simple case of the case presented in [1:Ch;3;6;3], since

$$a_{link}=1 \quad ; \quad \text{and} \quad a_{unordered}=0$$

Then

$$T_I = T_F + s + 5r + B/t$$

With these estimates the performance of a ring-structured file organization can be presented:

Table 4-19

Ring Structure Performance

File	Access	T_F	T_I	T_X
1	direct	.0736	.2028	.1750* from Table 4-12
2	via 1	.1111	.2375	1962.7135
300	via 2	.1620	.2884	4764.3072
3:1	via 300	.2204	.3469	6691.8457
3:2	via 300	.2204	.3469	na
400	via 300	.2469	.3760	na
4:2	via 300	.2204	.3468	na
4:3	via 300	.2204	.3468	na
5:1	via 300	.2584	.3468	na
5:1:1	via 5:1	.3696	.4961	na
800	via 300	.4833	.6097	na
850	via 300	.3372	.4636	na
11:1	via 400	.3637	.4901	na
11:1:1	via 11:1	.4221	.5485	na
11:1:2	via 11:1	.4318	.5582	na
11:1:3	via 11:1	.4221	.5485	na
12	direct	.0590	.0960	.1722* from Table 4-11
12:1	via 12	.1758	.3022	339.7305
13	direct	.0590	.0960	.1092* from Table 4-11
13:1	via 13:1	1.8405	1.9669	10.7983

* Direct files are taken with m/n = 1:05:

The total operation of the ring-structured file requires then

$$T_{day} = \sum_p L_R T_F + L_U T_I + L_X T_X$$

$$= 5122.5 + 2687.4 + 12234.7$$

$$= 20044.6 \text{ sec/day} = 5.57 \text{ hours/day} = 5 \text{ hrs. } 34 \text{ min.}$$

This is again not an adequate level of performance: A large fraction of the time (2 hrs: 23 min) is required for the Reads of Entire files: While a careful maintenance of locality could reduce this time [1:Fig:3-36], the fact that a single scan of a file will often require more than an hour can seriously inhibit the utility of the file to serve scientific use of the data base at the Family Clinic:

Other operations can be reduced by the use of additional entry point and auxiliary files, but the resulting complexity can negate the benefits obtained:

OTHER FILES

The above evaluations have concentrated on the primary files: For these the direct file and indexed-sequential file organization methods provide the best performance for the expected usage pattern: The general evaluations made above can be further refined: Some examples of file specific designs can be given using files not yet covered by the design process presented:

Lexicons and Referenced Entity files have the following usage characteristics

- Frequent fetch access
- Occasional update access
- No sequential access

Inspection of the characteristics of file organization methods shows that direct files match these requirement perfectly, so that the organization choice is simplified:

The largest lexicon in the Family System is the Patient Name Index (19). This file will be used as an example.

Detailed Design of a Direct File:

The parameters which control the design of a direct file are [1:Ch.3.5].

the number of records, n
the average recordsize, R
the space available for records, mR
the size of the buckets, B

The known parameters have the values

$$n = 30000 \quad , \quad R = 49$$

The request rate for the Patient Name Index is found from Table 4-6 as $L_R = 700$. From [1:Ch.3.4.3] it can be seen that the time required per access is

$$T_F = \left(1 + \frac{1}{2} \frac{n}{m-n} \right)(s+r+B/t) \qquad \text{for } B = R$$

or $$T_F = (1+p)(s+r+B/t) \qquad \text{with } p = \text{funct}(m/n, B/R)$$

The equivalent cost is: $T_R L_R \text{cost}_{\text{system}}$

The disk storage is: $m\,(R+W)\,\text{cost}_{\text{disk}}$ where $W = G\,R/B$ [1:Ch.2.2.4]

The core storage cost for large blocks is:

$$T_F L_R (B-R)\,\text{cost}_{\text{core}}$$

Tables 4-18a,b,c,d show these costs for a number of values of B/R and m/n and typical values of s (.035 sec), r (.017 sec), and a high speed disk t(620000/sec).

cost$_{system}$ = \$0.20/min

cost$_{disk}$ = \$300/Mbyte year

cost$_{core}$ = \$200000/Mbyte year

G = 193 bytes

c = 1 second

Table 4-18a

Fetch Cost

cost/day = (s+r+B/t)*p*700*.20/60

B/R	s+r+B/t	m/n = 1	1.05	1.1	1.25	1.5	1.75	2
1	.052	$910.0	$1.335	$ .728	$ .364	$ .243	$ .206	$ .180
5	.052	182.0	.340	.231	.158	.135	.129	.125
10	.053	92.75	.218	.173	.136	.127	.125	.124
20	.054	47.250	.176	.146	.130	.127	.126	.126
50	.056	19.600	.148	.137	.131	.131	.131	.131
100	.060	10.500	.146	.143	.140	.140	.140	.140
150	.064	7.467	.151	.149	.149	.149	.149	.149
200	.068	5.950	.159	.159	.159	.159	.159	.159

Table 4-18b

Storage Cost

cost/day = n*m/n*(R+G/(B/R)) \$300/300/10**6

B/R	m/n = 1	1.05	1.1	1.25	1.5	1.75	2
1	$ 7.260	$7.623	$7.986	$9.075	$10.890	$12.705	$14.52
5	2.628	2.759	2.891	3.285	3.942	4.599	5.256
10	2.049	2.151	2.254	2.561	3.074	3.586	4.098
20	1.760	1.848	1.936	2.199	2.639	3.079	3.519
50	1.585	1.664	1.744	1.982	2.379	2.775	3.172
100	1.528	1.604	1.681	1.910	2.292	2.674	3.056
150	1.509	1.584	1.659	1.886	2.263	2.640	3.017
200	1.499	1.574	1.649	1.874	2.248	2.623	2.998

Table 4-18c

Incremental Core Memory

cost/day = (1+(s+r+B/t)*p)(B-R) 700*.000667/8/60/60

B/R	B-R	m/n = 1	1.05	1.1	1.25	1.5	1.75	2
1	0	$ 0	$ 0	$ 0	$ 0			
5	196	.251	.004	.003	....			
10	441	.291	.008	.008	....			
20	931	.321	.016	.016	....	constant		
50	2401	.366	.041	.041	....			
100	4851	.433	.084	.083	....			
150	7301	.497	.126	.126	....			
200	9751	.561	.169	.169	....			

Table 4-18d

Total Cost/Day

Total cost = fetch cost + storage cost + incremental core cost

B/R	m/n = 1	1.05	1.1	1.25	1.5	1.75	2
1	$917.26	$8.958	$8.711	$9.430	$11.133	$12.911	$14.700
5	184.88	3.103	3.125	3.446	4.080	4.731	5.384
10	95.090	2.377	2.435	2.705	3.209	3.719	4.230
20	49.331	2.040	2.098	2.345	2.782	3.221	3.661
50	21.551	1.853	1.922	2.154	2.551	2.947	3.344
100	12.461	[1.834]	1.905	2.133	2.515	2.897	3.279
150	9.473	1.861	1.934	2.161	2.538	2.915	3.292
200	7.655	1.902	1.977	2.302	2.575	2.951	3.326

It can be seen from the tables that

a) The fetch cost decreases with increasing m/n and increases slightly with increasing blocksize; When m/n > 1;1 and B/R > 10 the effects become minimal;

b) The storage cost is significant; The minimum is reached at high B/R and low m/n; The effect of increasing blocksize diminishes as B/R > 50;

c) The core memory cost for the small amount of time that the buffer is active is low; It is minimally affected by increased m/n for m/n >= 1;05;

d) If slower devices are used, then the loss in access speed can only be slightly compensated by increased storage since the selected optimum is already close to the minimum (Table 4-18a), and a different blocking factor has no effect;

The minimum total cost is found at

B/R = 100 or blocksize b = 4900 bytes
m/n = 1;05 or storage = 30000 * 1;05 * 49 = 1;54 Mbytes

This is a fairly large blocksize, and it may be desirable to use a smaller blocksize in order to simplify memory management; The cost of a smaller blocksize, as shown in the column for m/n = 1;05 in Table 4-18d does not increase rapidly until B/R < 10 or B < 490 characters;

These results can be extrapolated easily to the other direct files; The usage of the Terms file (900) is very high; The

optimum for this file is expected to occur at a similar point, since as noted in paragraph (d) above the fetch minimum is close to the optimum found.

At the optimum (m/n=1.05), the overflow probability p= 0.06 [Figure 2-3] per request leads to

$$T_{Fopt} = (1+p)(s+r+B/t)(1.06)(.0596) = 0.0632$$

The corresponding update time is

$$T_I = 2\frac{n}{m}T_F + (s+3r+B/t) \quad [1:Ch.3.4.3]$$

$$= 0.2140$$

If these values apply to all Lexicons and Referenced Entity files, then the daily times will be

$$Tday_F = 1532 \text{ sec.}; \quad Tday_I = 128 \text{ sec.}$$

Files 19, 20, and 21 have to be read in their entirety once daily. Given their basic sizes from Table 4-3, m/n=1.05 and t′=220000/sec., this will require

$$Tday_X = \frac{(1470000+60000+60000)\ m/n}{t'} = 8 \text{ seconds.}$$

so that if these files are optimally configured require

$$Tday = Tday_F + Tday_I + Tday_X = 1668 \text{ sec. or } 28 \text{ min. daily}$$

is used to access all Lexicons and Referenced Entity files.

SERVICE FILES

The demand on most of the service files is modest; The greatest demand documented in Table 4-6 is on the Transaction Log; If a separate channel is available; then its activity can be largely overlapped [1:Ch;5;4]; Otherwise the parameters for sequential writing will apply; A high security Transaction Log may not buffer output; Then every transaction requires (s+r+R/t);

The aggregate daily load is then

Tday = 850(s+r+28/t)+4450(s+r+11/t) = 276 sec; = 4 min; 36 sec;

SUMMARY

Several file methods were investigated as to their applicability for the primary files; While for particular files the pile file design is adequate; for the more important files the indexed-sequential and direct files are both better and adequate; The indexed-sequential organization is more flexible and generally applicable;

Ring-structured files can save storage space but do not have adequate performance; Hierarchical files, as implemented in MUMPS, have extremely large storage requirements; These could be overcome by restructuring of the hierarchy; This would mean returning to the transformations of the data base model described in Chapter 3 of this thesis and has not been done; The values which were obtained have always assumed average conditions; The margins have been kept sufficiently large so that the combined effect of a very busy day, and an even busier period during such a day can be managed;

The total average time required to provide the data base activities of the Family System is

Primary Files	UT	29 min;
Lexicons	IS	28 min;
Service Files		4 min; 36 sec;
		62 min;/day

This provides adequate capacity for periods of high utilization and computational requirements; so that overlap of computation with file activity is not necessary;

CHAPTER 5
CONCLUSION

The preceeding two chapters have demonstrated how the semantic and quantitative design methods developed in Appendix 1 of this thesis provide a straightforward and rational methodology in file design. Only a few of the immense number of file design choices were investigated, but the choice of the basic methods is such that the tree of possible designs can be rapidly pruned. While the design can undergo further refinement stages, it is now possible for an implementor to evaluate the quantitative effect of changes in the logical data model, of new data models, and of changes in the load parameters.

It has been shown that the choice of design parameters does have important consequences on the feasibility of the system studied, and that the wrong choices can cripple the resulting system. The frequency of execution of primitive operations in the Family System is such that the effect of a unit fetch execution time of 1.0 second versus .2 seconds is important, although either retrieval speed will be called "nearly instantaneous" when viewed independently.

Several simple file choices can satisfy the requirements of the Family System. This choice can be exploited to increase the hardware system vendor alternatives or simplify implementation. It is also shown that the margins of performance are adequate so that complex operating systems techniques are not required to extract the maximum of hardware capability. This is a case where criteria are only to be satisfied rather than optimized.

A change in specifications; be they storage device parameters, data volume; or access frequency require recalcuation of these tables to assure continued satisfying of the objectives; While this involves a computational effort, it avoids the guesing games or disappointments common when system specifications change; If the computation is automated, then a sensitivity analysis is possible through a methodical variation of the parameters;

The objective, stated in Chapter 2 of this thesis, to transform the design of data base systems from a procedure based on experience to a procedure using a formal quantification of performance has hence been accomplished; The data and services specified for an automated ambulatory medical record system; the Family System; have been transformed to measures which provided comparability of data base design choices; The degree to which any given storage system will be utilized is a final single measure of design adequacy;

Adherence to the design process described here cannot guarantee success of the system to be implemented, but can remove some of the reasons for failure; There remain; of course; many further issues for which no satisfactory answers have yet been provided;

UNRESOLVED ISSUES

The Design Process:

As elaborated in [1:Ch;5;0] the design process presented follows the traditional engineering method: one or more design choices are proposed and the one which fullfills the requirements best is

chosen. By programming the performance formulas developed above, it becomes possible to investigate very many alternatives and arrive at an optimum. All alternative designs, however, have to be specified. There is no generator, corresponding to hypothesis generators in some Artificial Intelligence system, which provides automatically all possible combinations so that one could be sure that the very best design has not been ignored. Because of this deficit the process remains philosophically unsatisfying. Will it ever be possible to begin with the requirements, proceed through a logical sequence of steps, and obtain the desired design as output?

Man-Machine Interfaces:

The productivity and social value of a system is greatly influenced by the manner in which it is viewed by its immediate users. The lack of quantitative measures of the various factors which make a system interface pleasant causes this aspect of system design to be neglected when quantitative methods are stressed. In information systems [1:Ch.5:0] usage and hence the system load itself is controlled by the quality of the interface so that even the internal system design model is incomplete unless this area can be quantized.

Measures of Information Productivity:

While information theory provides results in regard to the absolute information content of retrieved data [1:Ch.5:5], the theory does not provide adequate measures to evaluate the value of the retrieved data to the user. The value of data to a user

should be measured in terms of the effect it can have on the user's environment, and it matters little if the data quantity is a single fact or the result of a complex analysis of a large data bank. Unfortunately, much computer output belabors the obvious. In order to design systems for maximal effectiveness it is necessary to put a premium on the no-so-obvious, on those results which can evoke actions or improve understanding. Such measures could be used to drive data reduction processes which in turn can decrease the volume and increase the relevance of computer produced results. Inquiry into user satisfaction [2:pp.173-181] provides some indices, but are not adequate for prospective design purposes.

Finale:

While the issues raised above, as well as the many other problems which are found in the area of exploitation of technology for medicine are important, it should not discourage the use of a thorough approach to data base design. It is my hope that a quantitative methodology will become the accepted standard to data base design in medical applications.

RELATED WORK

[1] Wiederhold, Gio: DATABASE DESIGN, McGraw-Hill Book Company, New York, (Computer Science Series), in print, to be available April 1977.

[2] Henley, R. R., G. Wiederhold, J. V. Dervin, M. A. Jenkin, I. M. Kuhn, E. Mesel, D. Ramsey-Klee, J. E. Rodnick: AN ANALYSIS OF AUTOMATED AMBULATORY MEDICAL RECORD SYSTEMS, Technical Report #13, Vols. 1 and 2, Laboratory of Medical Information Science, Office of Medical Information Systems, University of California Medical Center, San Francisco, CA, June 1975.

[3] Miller, Gerald F.: THE FAMILY SYSTEM, Lovelace Computing, Coulterville, CA, June, 1975.

Both [1] and [2] contain extensive bibliographies. References for the thesis are given in the bibliography below.

BIBLIOGRAPHY

Abrial74
Jean-Raymond Abrial: "Data Semantics"; in DATA BASE MANAGEMENT, (Proc: 1974 IFIP TC-2 Conf:; Cargese); J: W: Klimbie and K: L: Koffeman (editors), North Holland:

Ackoff67
R: L: Ackoff: "Management Misinformation Systems", MANAGEMENT SCIENCE; vol:14#4; Dec: 1967; pp:147-156:

Angell69
Thomas Angell and Theron M: Randell: "Generalized Data Management Systems"; IEEE Computer Group News; Nov:1969; pp:5-12:

Armstrong74
W: E: Armstrong: "Dependency Structures of Data Base Relationships", Proc: 1974 IFIP Congress, North Holland, pp:998-1006:

Bachman66
Charles W: Bachman: "On a Generalized Language for Organization and Manipulation"; CACM; vol:9#3, Mar:1966, pp:225-226:

Bachman72
Charles W: Bachman: "The Evolution of Storage Structures", CACM; vol:15#7; Jul:1972; pp:628-634:

Baker72
M: G: J: Baker, A: K: Betts, G: K: Collton, and M: J: Mitchell: DATA BASE MANAGEMENT SYSTEMS, THEIR USE IN HOSPITAL DATA PROCESSING, Her Majesty's Stationery Office, London 1972:

Barnett71
G: Octo Barnett: "The Use of Computers in Clinical Data Management"; Proc: of AMA Symposium on Computers in Medicine, 1971:

Baskett76
Forest Baskett and Alan Jay Smith: "Interference in Multiprocessor Computer Systems with Interleaved Memory", CACM, vol:19#6, June 1976, pp:327-334:

Behymer74
James A: Behymer, Robert A: Ogilive; and Alan G: Merten: "Analysis of Indexed Sequential and Direct Access File Organizations", in DATA DESCRIPTION, ACCESS, AND CONTROL, (Proc: 1974 SIGMOD Workship); ACM; 1974:

Bernstein75
P: A: Bernstein, J: R: Swenson; and D: C: Tsichritzis: "A Unified Approach to Functional Dependencies and Relations", in INT'L CONFERENCE ON MANAGEMENT OF DATA, (Proc: 1975 SIGMOD Conf:), ACM NY; 1975:

Bloom69
Burton H: Bloom: "Some Techniques and Trade-offs Affecting Large Data Base Retrieval Times", Proc: 1969 ACM National Conference, pp:83-95:

Brewer68
Susan Brewer: "Data Base or Data Maze - An Exploration of Entry Points", Proc: 1968 National ACM Conf:, pp: 623-630:

Byrnes69
Carolyn J: Byrnes and Donald B: Steig: "File Management Systems - A Current Summary", DATAMATION, Nov: 1969, pp:138-142:

Chang75
Shi-Kuo Chang: "A Preliminary Report on a Relational Data Base System for Medical Decision Making", Report of the Depts: of Information Engineering and Ophthamology, Univ: of Illinois, Chicago Circle, undated:

Chapin69A
Ned Chapin: "A Comparison of File Organization Techniques", Proc: 1969 ACM National Conference, pp:273-283:

Chapin69F
Ned Chapin: "*Common File Organization Techniques Compared*", Proc: 1969 FJCC, AFIPS, vol:35, pp:418-422:

Chen76
Peter P: S: Chen: "The Entity-Relationship Model - Toward a Unified View of Data", ACM TRANS: ON DATA BASE SYSTEMS, vol:1#1, Mar: 1976:

Chiang73
Tung Chin Chiang: "Canonical Structure in Attribute-Based File Organization", Univ: of Calif: Berkeley, EECS, PhD Thesis, 1973:

CODASYL76
SELECTION AND ACQUISITION OF A DATA BASE MANAGEMENT SYSTEM, CODASYL Systems Committee, available from ACM NY, 1976:

Codd70
E: F: Codd: "A Relational Model of Data for Large Shared Data Banks", CACM, vol:13#6, June 1970:

Codd72A
E: F: Codd: "Further Normalization of the Data Base Relational Model", in DATA BASE SYSTEMS, Prentice Hall, 1972:

Codd72B
E: F: Codd and C: J: Date: "Interactive Support for Non-Programmers - The Relational and Network Approaches", in DATA BASE SYSTEMS, Randall Rustin (editor), Prentice-Hall, 1972:

Codd75
E: F: Codd (editor): "Implementation of Relational Data Base Management Systems", FDT (Publ: ACM SIGMOD), vol:7#2, Sept:1975:

Collmeyer70
A: J: Collmeyer and J: E: Shemer: "Analysis of Retrieval Performance for Selected File Organzation Techniques", Proc: 1970 FJCC, AFIPS, vol:37, pp:201-210:

Cook75
T: J: Cook: "A Data Base Management System Design Philosophy", in INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA, (Proc: 1975 SIGMOD CONF:), ACM NY, 1975, 245 pages:

Datapro75
DATAPRO, Datapro Research Corp, Delran, NY.

Date71
C. J. Date and P. Hopewell: "File Definition and Logical Data Independence", in DATA DESCRIPTION ACCESS AND CONTROL (Proc. 1971 SIGFIDET Conf.), ACM 1971, pp.117-138

Date76
C. J. Date: "An Architecture for High-Level Language Database Extensions", in SIGMOD '76 PROCEEDINGS, James B. Rothnie (editor), ACM NY, June 1976, pp.101-122.

Davis70
Lou Davis: "Prototype for Future Computer Medical Records", COMPUTERS AND BIOMEDICAL RESEARCH, vol.3#5, Oct.1970, pp.539-554.

Dearden72
J. Dearden: "MIS is a Mirage", HARVARD BUSINESS REVIEW, Jan.-Feb.1972.

Delobel71
C. Delobel: "Aspects Theoretiques sur le Structure de l'Information dans une Base de Donnees", REVUE FRANCAISE D'INFORMATIQUES ET DE RECHERCHE OPERATIONELLE, No. B-3, Sept.1971.

Delobel73
C. Delobel and R. Casey: "Decomposition of a Data Base and the Theory of Boolean Switching Functions", IBM JR&D, vol.17#5, Sept.1973.

Fichten72
J. P. Fichten: "The Weyerhaeuser Information System, a Progress Report", Proc. 1972 FJCC, AFIPS vol.41, pp.1017-1024.

Forsyth75
J. Forsyth and R. Fadous: "Data Base Integrity as Provided by a Particular Data Base Management System", in INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA (Proc. 1975 SIGMOD Conference), avail. from ACM, 1975.

FPFH75
FORWARD PLAN FOR HEALTH, U.S. Department of Health, Education, and Welfare, (OS)76-50024, August 1975.

Greenes69
R. A. Greenes, A. N. Pappalardo, C. M. Marble, and G. O. Barnett: "A System for Clinical Data Management", Proc. 1969 FJCC, AFIPS, vol.35, pp.297-305.

Heath71
I. J. Heath: "Unacceptable File Operations in a Relational Data Base", in DATA DESCRIPTION ACCESS AND CONTROL (Proc. 1971 SIGFIDET Conf.), avail. from ACM, NY, 1971, pp.19-34.

Hopcroft69
J. E. Hopcroft and J. D. Ullman: FORMAL LANGUAGES AND THEIR RELATIONSHIP TO AUTOMATA, Addison-Wesley, 1969.

Hoos71
Ida R. Hoos: "Information Systems and Public Planning", MANAGEMENT SCIENCE, vol.17#10, June 1971, pp.658-671.

Hsiao70
David K. Hsiao and Frank D. Harary: "A Formal System for Information Retrieval from Files", CACM, vol.13#2, Feb.1970, pp.67-73.

Jardine74
D. J. Jardine (editor), DATA BASE MANAGEMENT SYSTEMS, North Holland and Am. Elseviers, 1974.

Kay75
M. H. Kay: "An Assessment of the CODASYL DDL for Use with a Relational Subschema", in DATA BASE DESCRIPTION - AN IN-DEPTH TECHNICAL EVALUATION OF CODASYL DDL, (Proc. 1975 IFIP TC-2 Conf., Namur), B. C. M. Douque and G. M. Nijssen (editors), North-Holland.

Knuth73
Donald E. Knuth: THE ART OF COMPUTER PROGRAMMING, VOL.3 - SORTING AND SEARCHING, Addison-Wesley, 2nd Ed., 1974.

Knuth74
Donald E. Knuth: THE ART OF COMPUTER PROGRAMMING, VOL.1 - FUNDAMENTAL ALGORITHMS, Addison-Wesley, 2nd ed., 1974.

Kochen67
Manfred Kochen (editor): THE GROWTH OF KNOWLEDGE, Wiley, 1967.

Levien67
R. E. Levien and M. E. Maron: "A Computer System for Inference Execution and Data Retrievel", CACM, vol.10#11, Nov. 1967, pp.715-721.

London72
Keith London: TECHNIQUES FOR DIRECT ACCESS, Auerbach Publishers, 1972.

Lucas75
Henry C. Lucas jr.: WHY INFORMATION SYSTEMS FAIL, Columbia Univ. Press, NY, 1975, 116 pages.

Lum70
V. Y. Lum: "Multi-Attribute Retrieval with Combined Indexes", CACM, vol.13#11, Nov.1970, pp.660-665.

Manacher75
G. K. Manacher: "On the Feasibility of Implementing a Large Relational Data Base with Optimal Performance on a Mini-Computer", in VERY LARGE DATA BASES, (Proc. International Conf. on VLDB), Douglas S. Kerr (editor), ACM NY, pp. 175-201.

McGee69
William C. McGee: "Generalized File Processing", Ann. Rev. in Automatic Data Processing (Pergamon Press), vol.5#13, pp.77-149.

Merten70
Alan G. Merten: "Some Quantitative Techniques for File Organization", PhD Thesis, Technical Report #15, Univ. of Wisconsin Computing Center, June 1970.

Nijssen75
G. M. Nijssen: "DDL Illustrated with Data Structure Diagrams", in A TECHNICAL IN-DEPTH EVALUATION OF THE DDL (Douque and Nijssen, editors), North Holland, 1975.

Nijssen76
G. M. Nijssen: Data Structuring in DDL and Relational Data Model", in MODELLING IN DATA BASE MANAGEMENT SYSTEMS, (Proc. 1976 IFIP TC-2. Conference), E. J. Neuhold (editor).

Patrick74
Robert L. Patrick: SECURITY SYSTEM REVIEW MANUAL, AFIPS Press, Montvale, N.J.; Oct. 1974.

Rydell76
R. Rydell, Western Heart Associates, San Jose, private communication.

Schussel75
George Schussel: "Data Bases" in INFORMATION SYSTEMS HANDBOOK, F. W. McFarlan and Richard Nolan (editors), Dow Jones-Irwin Inc., Homewood Ill., 1975.

Schwartz72
Jacob T. Schwartz: "Abstract and Concrete Problems in the Theory of Files", in DATA BASE SYSTEMS, Prentice Hall, 1972, pp.1-21.

Severance74
D. G. Severance: "Identifier Search Mechanisms, a Survey and a Generalized Model", ACM Computing Surveys, vol.6#3, Sept.1974.

Senko68
Michael E. Senko, V. Y. Lum, and P. J. Owens: "A File Organization Evaluation Model (FOREM)", Proc. 1968 IFIP Congress, pp.C19-C28.

Senko75
Michael E. Senko: "Information Systems - Records, Relations, Sets, Entities, and Things", INFORMATION SYSTEMS, vol.1#1, Pergamon Press, Jan. 1975, pp.3-13.

Senko76
Michael E. Senko: "DIAM as a Detailed Example of the ANSI SPARC Architecture", in MODELLING IN DATA BASE MANAGEMENT SYSTEMS, (Proc. 1976 IFIP TC-2 Conf., Freudenstadt, E. J. Neuhold, editor).

Shemer71
J. E. Shemer: "Access Methods: A Brief Overview for the 1971 ACM SIGFIDET Workshop", in DATA DESCRIPTION ACCESS AND CONTROL, (Proc. 1971 SIGFIDET Conf.), E. F. Codd and A. L. Dean, editors.

Steel74
Thomas B. Steel jr.: "Data Base Systems - Implications for Commerce and Industry", in DATA BASE MANAGEMENT SYSTEMS, Donald A. Jardine (editor), North-Holland and American Elseviers, 1974, 279 pages.

Steig72
D. B. Steig: "File Management Systems Revisited", DATAMATION, Oct. 1972, pp.48-51.

Stonebraker75
Michael Stonebraker and Gerald Held: "Networks, Hierarchies, and Relations in Data Base Management Systems", Proc. of the ACM Pacific-75 Conf., SF, Apr. 1975, pp.1-9.

Tanner75
Daniel J. Tanner: "User Ratings of Software Packages", DATAMATION, vol.21#12, Dec. 1975, pp.138-154.

Taylor71
R. W. Taylor: "Generalized Data Base Management System Data Structures and Their Mapping to Physical Storage", PhD thesis, Univ. of Michigan, 1971.

Tsichritzis75
Dionysios C. Tsichritzis: "A Network Framework for Relational Implementation", in DATA BASE DESCRIPTION - AN IN-DEPTH TECHNICAL EVALUATION OF CODASYL DDL, (Proc. 1975 IFIP TC-2 Conf., Namur), B. C. M. Douque and G. M. Nijssen (editors), North-Holland.

Wang75
C. P. Wang and H. Wedekind: "Segment Synthesis in Logical Data Base Design", IBM JR&D, vol.19#1, Jan., 1975, pp.71-77.

Wasserman75A
Anthony I. Wasserman, David D. Sheretz, and Charles L. Rogerson: "MUMPS Globals and Their Implementation", MUMPS Dev. Comm. Report, Doc.No. 2/1, May 1975, National Bureau of Standards, Washington, DC.

Wasserman75B
Anthony I. Wasserman: "Clinical Information and Relational Data Base Organization", UCSF Lab. of Med. Inf. Sci. Tech Report #17, Sept. 1975.

Weed69
L. L. Weed: MEDICAL RECORDS, MEDICAL EDUCATION, AND PATIENT CARE, THE PROBLEM-ORIENTED RECORD AS A BASIC TOOL, Yearbook Medical Publishers, Chicago 1969.

Weiss74
Sholom M. Weiss: A SYSTEM FOR MODEL-BASED COMPUTER-AIDED DIAGNOSIS AND THERAPY, PhD Thesis, Rutgers University, June 1974.

Wiederhold75
Gio Wiederhold, J. F. Fries, and S. Weyl: "Structured Organization of Clinical Data Bases", Proc. 1975 National Computer Conference, AFIPS Press, Montvale, N.J. 1975.

Yao74
Shi Bing Yao: "Evaluation and Optimization of File Organizations through Analytic Modelling", PhD thesis, Univ. of Michigan, 1974.

Zimmerman75
Klaus Zimmerman: "Different Views of a Data Base - Coexistence between Network Model and Relational Model", in VERY LARGE DATA BASES, (Proc. International Conf. on VLDB), Douglas S. Kerr (editor) ACM NY, 1975, 592 pages.