A RELATIONAL DATABASE SYSTEM FOR HOSPITAL ENVIRONMENT

DISSERTATION SUBMITTED TO THE JAWAHARLAL NEHRU UNIVERSITY
IN PARTIAL FULFILMENT FOR THE DEGREE OF
MASTER OF PHILOSOPHY

NARENDER KUMAR BHATIA

SCHOOL OF COMPUTER & SYSTEMS SCIENCES
JAWAHARLAL NEHRU UNIVERSITY
NEW DELHI-110067
1981

TO
MY PARENTS

CERTIFICATE

The research work embodied in this dissertation has been carried out at the School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi-110 067. This work is original and has not been submitted so far, in part or full, for any other degree or diploma of any University.

Narender K Bhatia

(NARENDER KUMAR BHATIA)
Student

P. C. Saxena

(Dr P. C. SAXENA)
Supervisor

N. P. Mukherjee

(Prof. N.P.MUKHERJEE)
Dean

SCHOOL OF COMPUTER & SYSTEMS SCIENCES
JAWAHARLAL NEHRU UNIVERSITY
NEW DELHI 110067

ACKNOWLEDGEMENTS

It is a pleasure to thank Dr P.C. Saxena for his valuable guidance, personal interest and constant encouragement. He has been a source of inspiration throughout this work.

I sincerely thank Professor N.P. Mukherjee, Professor D.K. Banerjee for their co-operation and providing me all the facilities in completing this work.

I also thank Dr A.K. Majumdar for his valuable suggestions and constructive criticisms.

I am thankful to Dr S.P. Rana, Lecturer, Computer Centre, IIT, New Delhi for useful discussions.

I would like to thank Mr A.K. Bansal and Mr O.P. Agnihotri, for their help and enlightening discussions.

My thanks are also due to the faculty and the other staff for their co-operation and help in many ways.

I am grateful to Dr A.D. Tuskar, Deputy Director, Institute for Research in Medical Statistics, ICMR, and Mr I.M.S. Lamba, Incharge, EDP Centre, IRMS, ICMR for providing necessary information and fruitful discussions.

I am also thankful to Mr S.S. Jolly, Officer Incharge, Medical Record Department & Training Centre, Safdarjang Hospital, New Delhi for providing necessary information and requirements of the hospital.

I also thank Mr C.R. Menon for typing the thesis so neatly.

I would like to express my gratitude to all my relatives, especially to my (late) father, who was very eager, but did not live through to see the final shape of my efforts.

Finally I am grateful to Jawaharlal Nehru University for providing financial assistance.

Narender K Bhatia

NARENDER KUMAR BHATIA

New Delhi,
19 November 1981.

CONTENTS

ABSTRACT

A Data Base System is designed for Hospital Administration and Management. The project was undertaken from Indian Council of Medical Research, New Delhi. We selected Safdarjang Hospital for this work. Medical Record Department (MRD) looks after the functions relating to the enquiries about the patients, their conditions and whereabouts and availability of beds. It also manages the activities by using the statistics obtained from the previous records. Since the hospital is big and the number of patients coming everyday is quite large, it has become very difficult for the authorities to respond to the enquiries promptly and to manage the statistics efficiently. To help them out with this problem through the computer, we have designed a relational data base with simple query language for the enquiries made by patients' relatives and to generate the statistics required by MRD and Institute for Research in Medical Statistics. An attempt has been made to provide data independence and good security. The system is developed in PL/I on CDC Cyber 170/720 in NOS Environment.

CHAPTER - I

INTRODUCTION

1.1 Introduction to the Project:

Hospitals throughout the world have experienced how attractive automation can be in dealing with complex problems. Manual systems are time consuming and require a variety of tedious clerical chores that may or may not result in an efficient system. One attraction of automated processing is the promise that they can eliminate many of these chores and free the staff for other services. Another attraction is the possibility of quick and efficient transactions and at the same time, the provision of an accurate, up-to-date record of all operations.

In the field of information processing whenever systematic storing and rapid processing of a large amount of information are required, the use of computers is almost a necessity. Any comprehensive hospital information system involves storing, processing and quick accessing of medical records of thousands of patients, covered by the hospital system under consideration. The design of an automated medical information system has drawn considerable attention. In recent years, computers are increasingly

being used for solving various problems in medical research, for systematic maintenance and updating of patients' record and for aiding the hospital administration in improving the efficiency of health care services.

There are two broad categories of hospital information system:

(a) Home grown systems:

These are designed by hospitals relying on a host computer to process day to day transactions. The capabilities of these systems vary from hospital to hospital depending upon size, needs and organization. Some are strictly batch processing systems while others offer either partial or full on-line access to computer files.

(b) Commercial available packages:

The first package which appeared in the market in U.S.A. around 1973. These packages are available with mini-computer, terminals and associated hardware plus the software to operate the system. These may or may not meet the requirements of a particular hospital.

Some information systems (Appendix-E) primarily deal with the systematic computerization of medical records of the patients who have been admitted to the hospital or are currently undergoing treatment. Most of these systems, use hierarchical data model in the design of data base. This model is not the most efficient when frequent updation and reorganization of records are required. In this respect, a relational approach to the design of data base provides better flexibility. The data manipulation language constructs for the relational system are simpler and more powerful than DL/I or any other similar language used with a hierarchical data base. In view of this, we will adopt the relational approach to the design of hospital data base.

The present project was undertaken from Indian Council of Medical Research, New Delhi. A relational data base is designed with simple query language to support the queries made by patients' relatives and to generate the statistics for the management. It has been implemented in PL/I on CDC Cyber 170/720 in NOS environment. All the files have been provided an IS organization. The data is kept up-to-date at all times.

The next section briefly describes the existing system.

1.2 Introduction to Existing System:

1.2.1 Introduction to Safdarjang Hospital:

Safdarjang Hospital is one of the biggest hospitals in the country. The hospital provides all types of services to its patients that are expected from a hospital of this nature. The bed strength of this hospital is 1207. About 7000 patients are discharged every year. About 70 persons are working in the Medical Record Department to maintain the records. There are 29 general wards and four casualty wards in the hospital. It has also got two emergency rooms of 25 beds each. It has got no private rooms or wards. There are 17 specialities and 32 units in the hospital. Each speciality is divided into one, two or three units. Wards are allocated to specialities.

This hospital receives patients from different states with various diseases. The patients are of two types:-

(a) New Cases:

These are the patients which are admitted in the hospital for the first time. These patients might have been referred by small hospitals or dispensaries. In most cases, they may come from O.P.D.

(b) Repeat Cases:

These patients are already receiving the services of the hospital for the same disease. There are follow-up visits for these patients.

Every patient admitted to the hospital is routed through central admitting office which assigns a unique number to each patient to identify it later. By convention this number is started from 1 or 100001 (for the sake of uniformity) in the beginning of the year. This number is increased by 1 for each new patient.

This number is called the Medical Record Department Number and is utilized by Medical Record Department (MRD) for maintaining the records. This serves as the key to manual records. By name and associated data, we may find MRD number and hence all the information.

1.2.2 System Description and Requirements:

Medical Record Department performs all functions relating to maintenance and operations of data. Fig.1.1 illustrates the inflow of patients in Safdarjang Hospital.

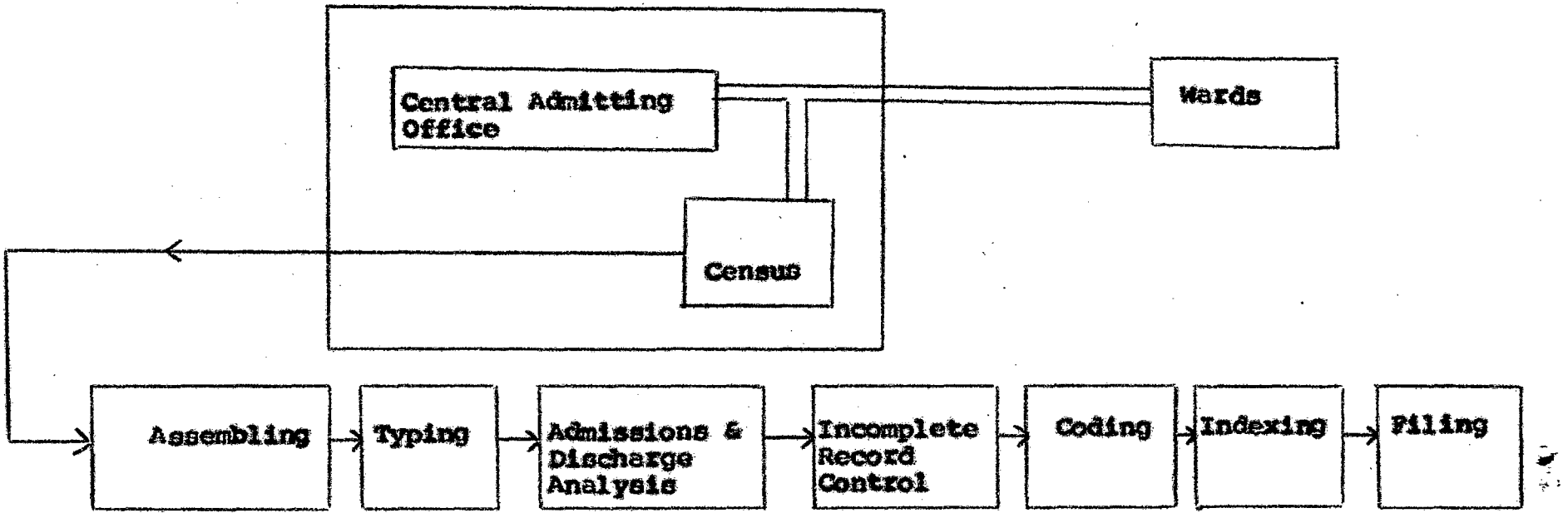


Fig. 1.1. FLOW OF INPATIENTS MEDICAL RECORDS

Two copies of the patient's identification data are prepared: one for the Medical Record Department (MRD) and one is sent to the wards. MRD also looks after the functions relating to enquiries about the patients, their conditions and whereabouts and availability of beds. It also manages the activities by using the statistics obtained from previous records.

It gives answers to the queries of the patients admitted as well as discharged.

Queries can be of the following form:

Given the name and date of admission, one should be able to retrieve the ward-bed number of the patient. Also if the ward-bed number is given, one would be able to retrieve the condition and diagnostic of the patient and so on.

The data model, containing all the relevant information, should be designed in such a way that it meets all the requirements that are discussed above. Finally the data model should be able to provide daily, weekly, monthly, annual statistics. The data model for the proposed system is discussed in Chapter III.

1.3 Thesis Overview:

The first chapter is devoted to an introduction of the project and the existing system. In the second chapter, we introduce the basic concepts of data base. In the third chapter, proposed system is discussed in detail taking all the aspects into consideration. Starting from the logical view and physical representation, all details of query language and preprocessor are presented. The final concluding chapter is devoted to discuss the problems experienced in the process of evolving the data base and also to present suggestions for future work.

In Appendices, hospital statistics with method of collection, overview of some hospital computer systems, results of programs, data base creation program and preprocessor program are presented.

CHAPTER - II

BASIC CONCEPTS OF DATA BASE

2.1 Definitions:

The smallest unit of data usually considered is the data item, also called field or attribute. A collection of data items constitute a logical record. A logical record type is a record with particular data item make-up. A file is a collection of occurrences of the same record type. The representation of collection of items which are related to one another in one way or other is called data structure.

The overall logical Data base description together with the interrelationships is referred to as schema. The term sub-schema refers to an application programmer's view of the data he uses. Many different sub-schemas can be derived from one schema.

An index may be referred as a table which with a procedure that accepts information about certain attribute or data item values as input and gives information as output which assists in quickly locating the record or records that have those attribute values.

A data base may be formally defined (Date,1977; Martin, 1977) as a collection of interrelated data stored together with controlled redundancy to serve one or more applications in an optimal fashion. Data are stored so that they are independent of programs which use them. A common and controlled approach is used in adding new data or modifying and retrieving the existing data within the data base.

One of the most attractive features of data base is that it is a collection of integrated, shareable and nonredundant data. An integrated data base brings together a variety of data and interrelates it for a variety of users, not just for one user or for a limited few as in conventional files. As such, a single user or application program might be concerned with a small subset of data base. Subsets of different users may overlap, that is, portions of the subsets may be common as shown in Fig.2.1 (Date, 1977).

A data base administrator (DBA) (Date, 1977) is a person or group of persons, responsible for the overall control of the data base system.

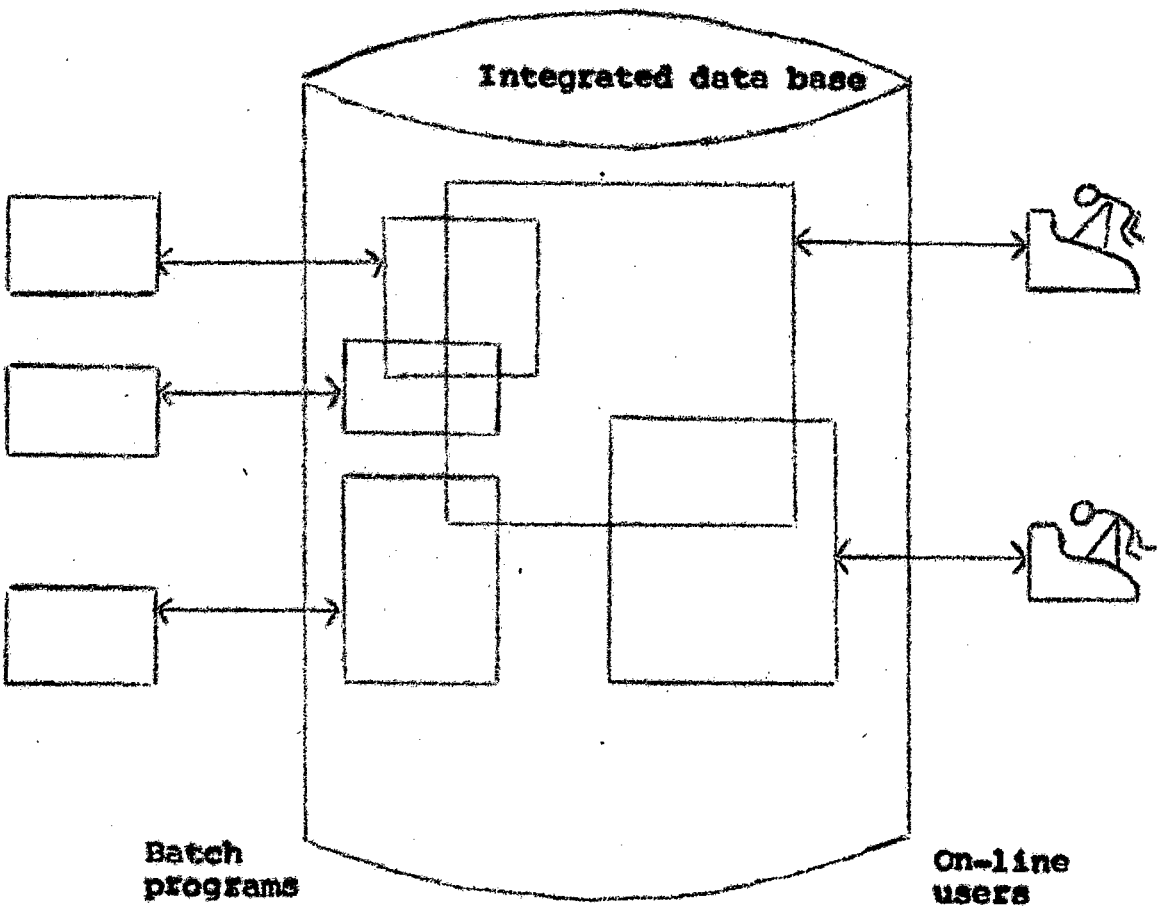


Fig. 2.1 SIMPLIFIED VIEW OF A DATA BASE SYSTEM

The data base management system (DBMS) is the interface between the application programmer and data base. It is the software that handles all accesses to the data base. It also provides the facilities for maintaining the data base. The user can interact with the system directly or via application programs written in higher level languages e.g. COBOL, PL/I, BASIC, etc. or through an English like language called query language. The DBMS is also responsible for authorization checks and validation procedures.

2.2 Security and Integrity in Data Bases:

The term security as used in DB environment implies the protection of data against unauthorized disclosure, alteration or destruction. The DBMS should not allow any operation to be performed on the data base unless the user is authorized for it. Authorization is responsibility of the DBA. The operations each user is allowed to perform and a means of identifying users to the system should be provided. Users of the data base system should identify themselves to the system (by means of an operator identifier or m/c readable identity cards or badges etc.) and authenticate this identification (by quoting a password or by answering some questions from

the system). Encryption provides an important technique for protecting sensitive data (Date, 1977; Wiederhold, 1977).

The problem of integrity is the problem of ensuring that the data in the data base is accurate at all times. There is a limit on the extent to which this objective can be achieved as the system cannot check the correctness of every individual value entered into the data base. Maintenance of integrity can be viewed as protecting the data against invalid alteration or destruction. Integrity is thus quite similar to security and similar measures as used for preserving security can be provided for maintaining integrity also.

Loss of integrity may be caused by any of the following: hardware failures (e.g. at CPU data channel or I/O devices), human error on the part of the computer operator or terminal user, programming errors within the DBMS or by the operating system used.

The following are a set of integrity constraints which can be specified by an authorized user (but to enter it in the data dictionary or not, is in the hands of the DBMS).

(i) Existence of unique primary key specified by a KEY entry.

(ii) Existence of candidates keys specified by UNIQUE (A) where A is the attribute or set of attributes forming the candidate key.

(iii) Functional dependencies.

(iv) Values occurring in a particular attribute may be required to lie between certain bounds or that there may be a very small set of permitted values for some particular attribute.

2.3 Data Base Design:

Data model is the heart of any data base system. The range of data structures supported in the model is a factor that critically affects every other component of the system, it helps in designing the corresponding data sub-language. There are two classes of models: functional and structural. Functional models are invariably used to check the consistency of ideas, whereas, structural model are used to predict the performance of a system which is already existing or is planned.

Functional models are extremely helpful in understanding a phenomenon, whereas, structural models obscure the fundamental information in the mass of data. Considering the data models, we would like the models to be fundamental so that we can understand the relationship between different kinds of data. At the functional level, we can visualize three types of models.

There are three well-known approaches to the design of data bases:

1 Hierarchical Model;

2 Network Model;

3 Relational Model.

2.3.1 Hierarchical Model:

In this model, the data is represented in the form of simple tree structure. In tree structure, one type of record is termed as root and other record types are called nodes. Nodes are dependent on the root. No dependent record can exist without its superiors. The hierarchical representation of a sample hospital data base is illustrated in Fig. 2.2.

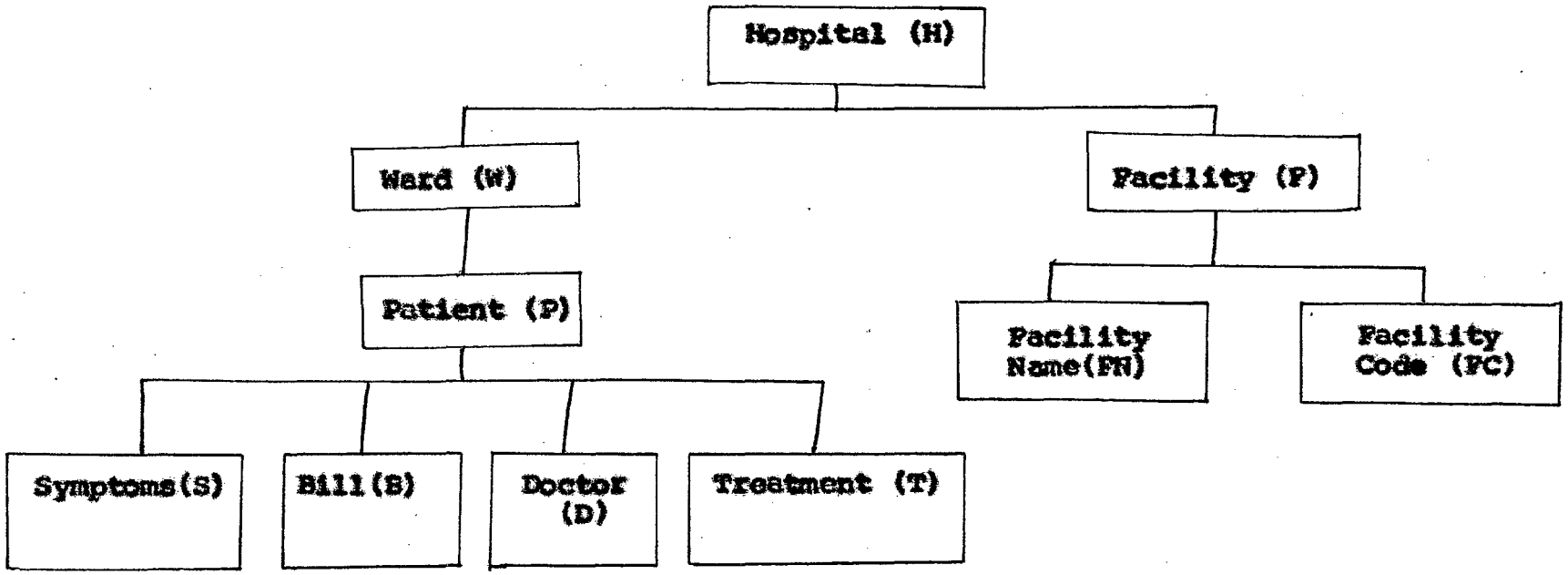


Fig. 2.2. A SAMPLE HIERARCHICAL HOSPITAL DATA BASE

H, W, F, P, FN, FC, S, B, D, T, are called segments. Multiple occurrences segments W, F, P, FN, FC, S, B, D, T have not been shown for a particular H segment. Segment P takes on its full significance only when its predecessors S, B, D, T are known. Anomalies arise in connection with the various storage operations viz. insertions, deletions, etc. For example the deletion of a W segment will result in the deletion of P, S, B, D, T segments.

The DSL operations are quite complex in this case. IBM corporation has developed a data base system for hierarchical data bases called IMS (Information Management System) (Date,1977).

The DSL used in IMS is called DL/I (Data Language-one).

The following is a list of DL/I operations.

Get Unique (GU) for direct retrieval.
Get Next (GN) for sequential retrieval.
Get next within parent (GNP) for sequential retrieval under the current parent segment.
Get hold (GHU, GHN, GHNP) - same as above but allow subsequent DLET/REPL.
Insert (ISRT) for inserting a new segment.
Delete (DLET) for deleting an existing segment.
Replace (REPL) for replacing an existing segment.

2.3.2 Network Model:

In this model, data are represented by records and links. A given record occurrence can have any number of immediate superiors as well as any number of dependents. Thus a many-to-many correspondence can be modelled more easily here, then in hierarchical representation (Date, 1977).

Associations between record occurrences are represented by connector record occurrences or links. Links are generally named in a network model but are anonymous in hierarchical approach. The anomalies in the hierarchical model regarding update, deletion etc. do not appear in this case. However, the programming involved is quite complex. The prime disadvantage of the network model is undue complexity both in the data model as well as in the DSL.

The most widely used network data base system is the Data Base Task Group (DBTG) system (Date, 1977; Sprowls, 1977).

An important concept in a DBTG system is a set, which has a certain type of record as its owner and some other types of record as its member(s). Each occurrence of a set represents a hierarchical relationship between the owner occurrences and the corresponding member occurrences. Fig. 2.3 illustrates a simple network model.

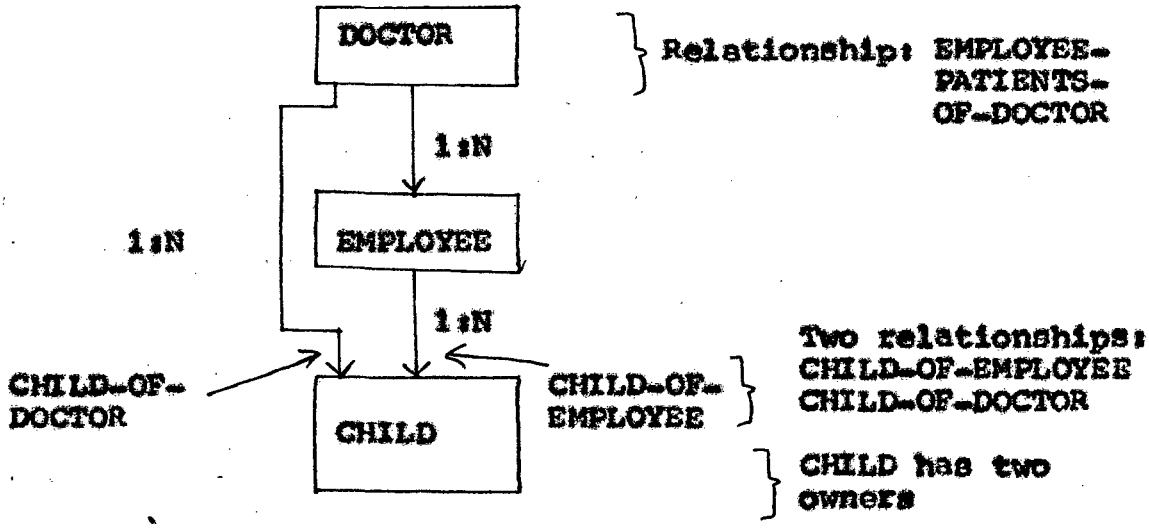


Fig. 2.3. A SIMPLE NETWORK MODEL

The DSL operations for the DBTG system are:

FIND for locating an existing record occurrence and establishing it as the current of run-unit. (The most recently accessed record occurrence, is referred to as

the "current of run-unit").

MODIFY for updating the current of run-unit.

CONNECT for inserting the current of run-unit into one or more set occurrences.

ERASE for deleting the current of run-unit.

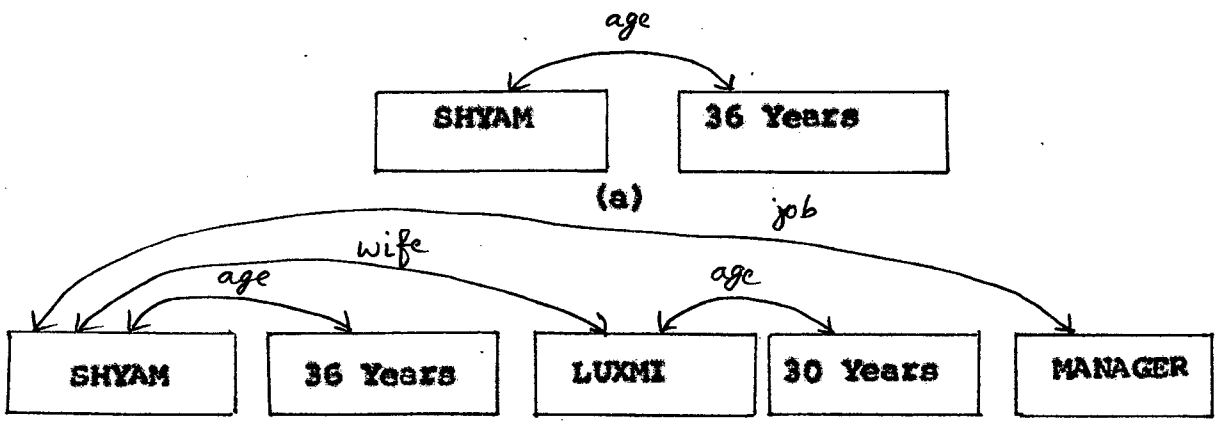
STORE for creating a new record occurrence and establishing it as the current of run-unit.

The FIND statement is logically required before each of the other statements except STORE.

2.3.3 Relational Model:

2.3.3.1 Introduction:

The relational model maintains the simplicity of so-called flat files. A flat file is one in which each record instance has a similar number of fields. The conventional COBOL-like logical file structure without repeating groups is a flat file. Fig. 2.4 (d) shows a relation.



(b)

Man	Age	wife	Wife's age	Job
SHYAM	36	LUXMI	30	MANAGER

(c)

People name	Age	Spouse name	Spouse-age	Job
SHYAM	36	LUXMI	30	MANAGER
MOHAN	28	SONA	24	DOCTOR
SOHAN	25	RITA	23	DIRECTOR
LUXMI	30	SHYAM	36	OFFICER

(d)

Fig. 2.4. RELATIONSHIPS, TUPLES AND RELATIONS (a) Binary Relationship (b) Binary Relationships (c) Tuple (d) Relation.

The relational model has attracted widespread interest among database practitioners and researchers for its conceptual simplicity and for its mathematical elegance. One database problem that is especially easy to formulate in terms of relations is that of logical schema design: given a description of an application find a "good" schema that describe the structure of the corresponding database. Currently, this logical design problem is largely in the hands of database administrators (DBA's) whose tools for performing this task are quite meager relative to the complexity of the problem.

The problem of schema design may be loosely stated as follows: transform a DBA-supplied set of functional dependencies that describe an application, possibly augmented by an initial guess of some relations, into a relational schema satisfying certain requirements. Some basic requirements are that the schema satisfies Codd's normal forms and that the supplied functional dependencies are appropriately represented in it.

The set of tuples that comprise a relation normally changes over time as entities are inserted, deleted, or modified. Thus a relation is a time-varying set of tuples. However, the structure of a relation-its name and attributes - remains quite static, by comparison. The notation for describing the structure of a relation is $R(A_1, A_2, \ldots, A_n)$ where R is the name of the relation and $\{A_1, A_2, \ldots, A_n\}$ is the set of attributes appearing in it. An $\langle$entity, attribute$\rangle$ entry in a relation is a value associated with the entity, chosen from the domain of the attribute. An entity is described by a tuple of such entries, one for each attribute in the relation.

The word relation is used in literature both to denote the structure of the relation (its intention) and to denote a set of tuples having the appropriate structure (an extension). We will use the term relation scheme to refer to an intention, that is, to refer to a structural description of a relation. The word relation will be used to refer to an extension, that is, to a set of tuples.

Relational approach has the following advantages over hierarchical and network approaches (Martin 1977, Date 1977).

(1) Easiest way to represent data for users not familiar with techniques of data processing.

(2) Associations between various relations are represented solely by means of data values. This uniformity of data representation leads to a corresponding uniformity in the operator set of the DSL.

(3) The operations of JOIN, PROJECTION, etc. can easily yield relations as desired by different users.

(4) Security checks and integrity constraints can be more easily implemented for separate relations.

(5) Update operations can be performed easily and effectively and better data independence can be attained.

(6) Relations being precise in meaning, can be manipulated with the precise mathematics of relational algebra and the relational calculus.

2.3.3.2 Definition of a Relation:

Let D_1, D_2, ..., D_n be sets, not necessarily distinct. The sets D_1, D_2, ..., D_n are called the domains

of R and each one of them represents one data item type (field type or attribute). R is a relation on these sets if it is a set of ordered n-tuples $\langle d_1, d_2, \ldots, d_n \rangle$ such that $d_1 \in D_1$, $d_2 \in D_2$, $\ldots$, $d_n \in D_n$. The relation R over a set of attributes $\{D_1, D_2, \ldots, D_n\}$ is denoted by R $(D_1\ D_2, \ldots, D_n)$. The number 'n' of the attributes determines the degree of the relation. Thus a relation of degree one is called a unary relation, and a relation of degree two is called a binary relation and similarly a relation of degree n is called n-ary relation. The rows of the relation matrix are called tuples.

A relation is a two-dimensional array with the following properties:

1 Each entry in the table is one data item, there are no repeating groups. In other words, each domain D_i must be a simple domain; that is, it does not represent another relation.

2 Each column, the domain, is assigned a distinct name, called a domain name, and is made up of values of the same data item.

3 All rows or tuples are distinct; duplicates are not allowed.

4 The rows and columns can be ordered in any sequence at any time without affecting the information content of the semantics involved.

Each relation must have a key by which a tuple can be uniquely identified and differentiated from other tuples of that relation. The key is either a simple domain (field) or a combination of domains.

2.3.3.3 Different relational operators:

In the presentation of the relational model (Codd, 1970), Codd introduced the relational algebra as a data manipulation language. The important relational algebra operators are: (i) SELECT; (ii) PROJECT; and (iii) JOIN.

The SELECT Operator constructs a new relation by taking a horizontal subset of the existing relation i.e. all tuples that satisfy a certain condition.

The PROJECT Operator forms a vertical subset of an existing relation by extracting specified columns and removing any redundant duplicate values in the set of columns extracted.

The JOIN Operator involves two relations which should have one of their attributes related to each other i.e. the

relations have a domain in common. The attributes can be related through any of the comparison operators (= or $\neq$ or $<$ or $\leq$ or $\geq$) and when an attribute from each of the two relations is equal, then the join operation is called an equijoin. The result of JOIN is a new, wider relation, in which each tuple is formed by joining together two tuples, one from each relation, such that the two tuples have the attribute values related to each other, in the common domain.

2.3.3.4 Functional and multivalued dependencies:

Let A and B be two attributes of a relation R, then B is said to be functionally dependent on A, if for a given value of a E Dom (A), there corresponds at most one value b E Dom (B) i.e. if two tuples have the same value for attribute A, then they have the same value for B. This is expressed as: f : A $\longrightarrow$ B, where f is the functional dependency (FD) (Beeri et al., 1979).

For example, a student name (STUNM) has a unique enrolment number (EN): The FD existing between them is EN $\longrightarrow$ STUNM.

Attribute B is said to be fully functionally dependent on attribute A (A must be composite), if B is functionally dependent on A and not on any subset of A.

For instance, the quantity of shipment (QTY) is determined by the supplier (SUP) and the part (P) supplied i.e. SUP, P $\longrightarrow$ QTY. Here QTY is said to be fully functionally dependent on SUP, P since it is not functionally dependent on either SUP or P alone.

It has been shown by Delobel (1978), that if in a relation R (X, Y, Z), the FD X $\longrightarrow$ Y holds then R is decomposable as R (X, Y, Z) = R (X, Y) * R (X, Z).

For instance, in the relation STUDENT (EN, STUNM, CITY), the FD between EN and STUNM is EN $\longrightarrow$ STUNM, hence relation STUDENT can be decomposed as follows:

STUDENT (EN, STUNM, CITY) = STU1 (EN, STUNM)*
STU2 (EN, CITY).

Definition of a key: (Beeri, et al., 1979)

Let R (A_1, A_2, ..., A_n) be a relation with attributes $\{A_1, A_2, \ldots, A_n\}$ and let X be a subset of $\{A_1, A_2, \ldots, A_n\}$. X is said to be a super key of R if every attribute $Ai \in \{A_1, A_2, \ldots, A_n\}$, is functionally dependent on X i.e. $X \longrightarrow A_i$, $\forall$ i = 1, ..., n.

X is a key of R if it is a minimal superkey i.e. if it is a superkey and does not properly contain any super key.

An attribute which does not belong to the key is called a non-key attribute.

(ii) Multivalued dependency (MVD) (Fagin, 1977)

MVDs are generalizations of FDs. Let X, Y, Z be disjoint sets of attributes of a relation R (X, Y, Z). An MVD $X \rightarrow\!\!\!\!\rightarrow Y$ is said to hold for R if (X, Y, Z) and (X, Y^1, Z^1) are tuples of R, then so are (X, Y^1, Z) and (X, Y, Z^1).

For instance in the following relation EMP (Employee, Child, Salary), the MVD Employee $\rightarrow\!\!\!\!\rightarrow$ Child holds.

'EMP

Employee	Child	Salary
Ram	Bobby	1700
Sham	Lovely	1900
Sham	Bittu	1900
Madhu	Lallu	2400

Fagin (1977) and Delobel (1978) have shown that if in a relation R (X, Y, Z), $X \rightarrow\!\!\!\!\rightarrow Y$ holds, then R can be expressed as the join of relations R1 (X, Y) and R2 (X, Z) i.e. R (X, Y, Z) = R1 (X Y) $*$ R2 (X, Z).

Relation EMP can be decomposed without loss of information into relations E1 and E2 owing to the MVD Employee $\longrightarrow\!\!>$ Child i.e. EMP (Employee, Child, Salary) = E1 (Employee, Child) $*$ E2 (Employee, Salary).

(iii) Embedded multivalued dependencey (EMVD) (Delobel, 1978; Fagin, 1977)

If X $\longrightarrow\!\!>$ Y holds for the projection R1 (X, Y, Z) of R (X, Y, Z, W) then the embedded MVD X $\longrightarrow\!\!>$ Y/Z holds for the relation R (X, Y, Z, W).

2.3.3.5 Properties of FDs and MVDs:

Given a set of FDs, it is possible to derive other FDs by applying these FDs. Armstrong (1974) proposed a set of rules called inference rules which allow one to infer syntactically from a given set of FDs, all other dependencies logically implied by the given set and have been listed below:

Inference rules of FDs: (Armstrong, 1974)

In the following, A, B, C, D are sets of attributes:

1 Reflexivity: if B $\subseteq$ A, then A $\longrightarrow$ B

2 Augmentation: if A $\longrightarrow$ B, then AUC $\longrightarrow$ B or AC $\longrightarrow$ B

3 Pseudotransivity: if $A \longrightarrow B$ and $BC \longrightarrow D$ then $AC \longrightarrow D$

4 Transitivity: if $A \longrightarrow B$ and $B \longrightarrow C$, then $A \longrightarrow C$

5 Additivity: if $A \longrightarrow B$ and $A \longrightarrow C$, then $A \longrightarrow BC$.

Given a set of MVDs, it is possible to derive other MVDs from these. The following are the inference rules for MVDs, proposed by Mendelzon (1979).

(In the following X, Y, Z, W are disjoint sets of attributes).

1 Complementation: if $X \twoheadrightarrow Y$ holds for a relation R (X, Y, Z), then $Y \twoheadrightarrow Z$ also holds.

2 Transitivity: if $X \twoheadrightarrow Y$ and $Y \twoheadrightarrow Z$, then $X \twoheadrightarrow Z$ holds for R (X, Y, Z).

3 Reflexivity: if $Y \subseteq X$ then $X \twoheadrightarrow Y$

4 Augmentation: if $X \twoheadrightarrow Y$ and $Z \subseteq W$, then $XW \twoheadrightarrow YZ$

5 Pseudotransitivity: if $X \twoheadrightarrow Y$, $YW \twoheadrightarrow Z$ then $XW \twoheadrightarrow Z - YW$

6 Union: if $X \rightarrow\!\!\!\!\rightarrow Y$ and $X \rightarrow\!\!\!\!\rightarrow Z$ then $X \rightarrow\!\!\!\!\rightarrow YZ$

7 Decomposition: if $X \rightarrow\!\!\!\!\rightarrow Y$, $X \rightarrow\!\!\!\!\rightarrow Z$ then
$X \rightarrow\!\!\!\!\rightarrow Y \cap Z$, $X \rightarrow\!\!\!\!\rightarrow Y - Z$ and $X \rightarrow\!\!\!\!\rightarrow Z - Y$.

2.3.3.6 Normalization of relations:

A relational model of a data base can be defined as the user's view of the data base as a collection of time varying (allowing for insertions, deletions, updation of tuples etc.) normalized relations of assorted degrees.

Normalization is concerned with the problem of choosing the most appropriate set of relations to represent a given collection of data.

In the presence of FDs and MVDs, updating, insertions etc. of tuples in relations may pose serious problems. In order to obviate such difficulties, Codd defined (1972) three levels of normalization viz. the first, second and third normal forms (1NF, 2NF, 3NF).

A normalized relation (also called a 1 NF relation) is one, whose attribute values in each tuple are atomic or non-decomposable (Date,1977; Ghosh,1977) i.e. at every row and column position, there exists precisely one value, never a set of values. An unnormalized relation can always be reduced to a normalized one with some redundancy.

A relation R is in 2NF if it is in 1NF and every nonkey attribute is fully dependent on the primary key. And R is said to be in 3NF if it is in 2NF and every nonkey attribute is nontransitively dependent on the primary key.

The concern for normalisation and the interest in using third normal form relations arises as a result of observation that in dynamic data base environment possibly undesired side effects referred to as 'anomalies' in relational jargon, may result due to insertions, deletions and updation.

An important objective of normalization is to eliminate these anomalies by splitting a relation into simpler but equivalent relations.

The definition of 3NF involves only non-key attributes, but some of the problems related to update anomalies still remain for prime attributes (Beeri et al., 1979). This led to a stronger normal form called the Boyce Codd Normal Form (BCNF) due to Boyce and Codd.

A relation R is said to be in BCNF (Date, 1977; Beeri et al, 1979; Fagin, 1977) if whenever a nontrivial FD $X \longrightarrow Y$ holds for R, then so does the FD $X \longrightarrow A$ for every

attribute A of R (X and Y are attribute sets).

When MVDs are present Fagin (1977) has introduced a new normal form called the fourth normal form (4NF).

A relation R is said to be in 4NF if, whenever a non-trivial MVD $X \rightarrow\!\!\!\!\rightarrow Y$ (X, Y being sets of attributes) hold for R, then so does the FD, $X \longrightarrow A$ (A being an attribute of R) i.e. all dependencies are the result of keys.

It has been shown by Beeri et al (1979) that given a set of FDs and MVDs, the problem of synthesizing a relation in 4NF or BCNF is a NP complete problem. The problem of deciding whether a 4NF or BCNF exists, is NP hard. Hence in view of the problems associated with update, insertions etc. the aim of the data model design should be to generate collection of relations which are in 3NF or in 4NF (if one exists).

2.3.3.7 Relational data sublanguages:

Relational algebra and relational calculus form a basis for extremely high-level language operating on relations. The various DSLs are: ALPHA based on relational calculus, SEQUEL and Query by example. ALPHA has not been practically

implemented so far, SEQUEL and Query by Example have been partly implemented (Date,1977).

DSL ALPHA is simple, is relationally complete (i.e. any relation derivable for the data model by means of an expression of the relational calculus, can be retrieved using the language), is nonprocedural and can be easily extended and supports higher level languages. DSL ALPHA supports insertions, deletions and updations of tuples. It provides, some library functions viz selecting max., min., average of values existing in a relation.

SEQUEL and Query by example are also relationally complete.

In the proposed system, we have not implemented any of these DSLs. It being a simple and specific problem of enquiries, a special purpose language with terminology specific to the application area, has been developed, which will be discussed later.

Many of the features provided by DSL$_\delta$ ALPHA, SEQUEL etc. are not needed at all in the present case.

2.4 Data Independence:

Data independence can be best understood by first defining its opposite. To the extent that the user is involved with details of storage structure (how the data are physically stored) and access strategy (how they are accessed), he is said to be data dependent. For example the user may be required to know which fields in a particular records are accessed and to write access statements which refer to these indices explicitly. In this case the user is dependent on the existence of the indices in question.

However, the user should not be interested in anything other than (a) what information is in the data base, and (b) what operations he may perform on that information. We cannot provide total data independence. Thus it should be possible to achieve data independence by providing the user with (a) a view or model of the data, and (b) a language to operate on the model, as far as possible independent of all storage and access.

2.5 Storage Structure:

2.5.1 Introduction:

The basic file organizations, also called basic access methods, available in present day computing systems

via higher level (e.g. COBOL, PL/I) and assembly languages are the well known sequential, random and indexed sequential file organizations. These provide very limited entry capability to the stored data space: direct access to records on the basis of the field selected as the one and the only access key when the file is initially created. This is a minimal facility, since in most actual environments it is essential to be able to access data on the basis of any of the data item or field or a combination of them.

In the case of our data base for the hospital system also many queries are to be answered on the basis of non-key attributes. So none of these methods is fully suitable for our model.

A number of important data or file organization concepts globally termed secondary file organizations or higher-level file organizations have emerged through the years in order to alleviate the mismatch between the complex information retrieval demands of users and the single key basic facilities provided by operating systems.

The main ones are:

1 List and multilist organizations;

2 Partially inverted and fully inverted organizations; and

3 Ring and chained free structures.

Inverted organization is chosen for our data base.

2.5.2 Basic Access Methods (File Organization):

A brief description of the various important file organizations now follows:

1 Sequential Organization:

The logical and physical sequence of records is identical. Records are read and written in sequence. New records can be added at the end. Insertion in the middle of file is not possible without rewriting the whole file.

2 Indexed Sequential (IS) Organization:

It allows both sequential and random processing. Random processing is accomplished by specifying the key field value for the desired record which is found via indices.

IS files are composed of two areas:

(i) Prime area - contains data of the file, during creation or reorganization.

(ii) Overflow area - contains records which could not fit in the prime area, as a result of additions to the file.

(iii) Index area - contains the indices - (a) track index (b) cylinder index and (c) master index, used to locate any particular record.

The track index contains the value of the highest key on each track. The cylinder index contains the highest key value on each cylinder.

When an IS file is created, all records are written into the prime area, in key sequence. The indices are generated at this point. The process of updating an existing record (replace or update-in-place) is similar to the retrieval process. Deletions from the file are also straightforward. The indices are used to find the desired record and a special mark "deletion mark" is inserted into the record to indicate that it has been deleted. The record is deleted physically, only during reorganization. Prior to such organization the deleted data remains on the storage medium insertions to an IS file must maintain the key order of the file.

3 Direct file organizations

Direct or random file organization is primarily used for random processing, although sequential reading is possible. Records in a direct access file may not have

any particular order, they may or may not have keys. The programmer has greater flexibility with direct file than with IS file, but the programming task is more complex.

Records are identified in one of the following ways: (i) by specifying the relative track and relative record number or (ii) by relative track and key (records must have keys, in those cases), or (iii) by relative block address, or (iv) by physical address which is the actual cylinder, head and track record number of the device or (v) by hashing.

Direct file organization does not require any particular correspondence between record content and file address. The programmer has to establish a way of correlating record to file location.

2.5.3 Inverted Organization:

The basic structure of inverted organization involves:

1 The original data records without any embedded pointers; and

2 The inverted dictionary or directory or index that contains certain field values followed by the list of pointers to the records characterized by such field values.

The fields placed in the directory in order to provide fast access to records are called inverted keys, indexed keys, or access keys. The Fig. 2.6 shows the basic layouts of the inverted file structure.

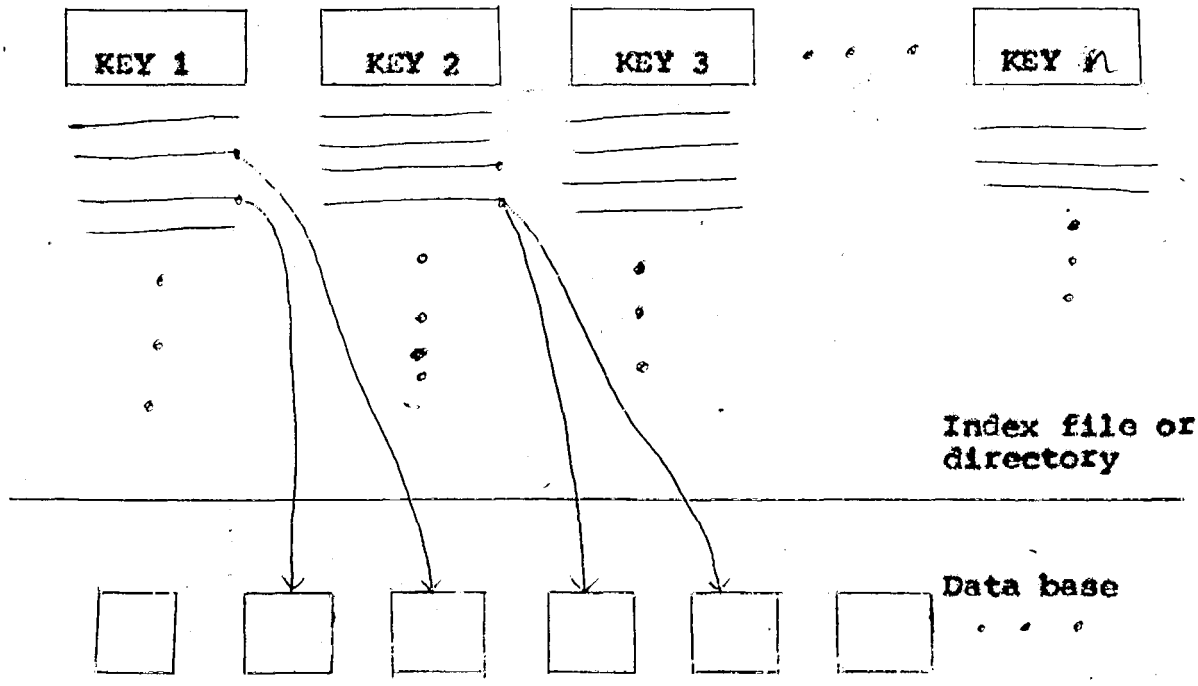


Fig. 2.6 BASIC INVERTED FILE ORGANIZATION

The degree of invertion refers to the extent to which field-values and pointers to corresponding records are placed on the directory; in other words, the extent to which extent paths are established. The higher the degree of invertion, the directory becomes larger file posing the

same search problems as the original set of records. So the directory or index itself may be inverted resulting in a second index and so on. Such an organization is called a multi-level index organization.

Making a decision about what fields to invert is very crucial and difficult. Few authors have suggested practical and specific guidelines for selecting the optimal, or close to optimal, inversion fields. A useful design rule of thumb in this regard is that if the average number of records qualified by a query based on a particular data item field is less than 10 per cent of the total number of records, then it is justifiable to invert that field. We have taken into consideration the practical aspects in designing the inverted organization for our data base.

In the next chapter, the proposed system will be discussed.

The design of the data model, the FDs existing, the relations used, the external schema, the conceptual schema, the query language and preprocessor will be discussed.

CHAPTER - XXI

THE PROPOSED SYSTEM

3.1 Introduction:

In the proposed system, we have developed a relational data base for hospital environment. The data relevant to the proposed system include the following information:

(i) patient's identification information,

(ii) diagnostic and symptoms information,

(iii) treatment information,

(iv) inverted files,

(v) statistical information relating to beds in each speciality,

(vi) statistical information relating to each speciality and each unit,

(vii) statistical data relating to total hospital.

From the above data one can obtain different types of information, e.g.

(i) from the patient's name and/or his/her father's name, one can get the ward-bed number and condition of the patient.

(ii) similarly from the patient's MRD Number or ward-bed number, one can obtain the information about the treatment to be undergone by the patient.

It has been discussed previously, that the relational approach to the design of a data model, has considerable

advantages over the network and hierarchical approaches. Hence we adopt a relational approach to the design for the proposed system. The details of our model are discussed below. Here we will be using file and relation as synonyms.

3.2 Logical views of relations:

In the logical view, we describe how the user visualizes the relations. The user can understand the relations in the following form:

FULLINFO (MRDNO, TITLE, NAME, FNAME, WARDBED, ADDRESS, LADDRESS, NKADDRESS, CGHS, AGE, SEX, CSTATUS, RELIGION, OCCUPATION, DATEADM, MONTHLY-INCOME, PHONENO, READM, PRVDGN, TMADM, SPECIALITY, UNIT) --- R1

DIAGNOSTIC(MRDNO, WARDBED, NAME, AGE, SEX, CGHS, PHYSICEXAM, PCOMPLAINT, PSTHISTORY, FDIAGNOSIS, TESTS, DISEASE-CODE) --- R2

TREATMENT(MRDNO, WARDBED, NAME, AGE, SEX, CGHS, XRAY, DELIVERY, LABEXAM, OPERATION, OTHRTRTMNT, CONDITION, RESULT, ADVICE, DEATH, AUTOPSY, CAUSE-OF-DEATH, BILL, DATE-OF-DISCHARGE, TOTAL-STAY, FETAL-DEATHS) --- R3

NAMEFILE(NAME, MRD1, MRD2, ..., MRD99) --- R4

FNAMFILE(FNAME, MRD1, MRD2, ..., MRD99) --- R5

ADDRFILE(ADDRESS, MRD1, MRD2, ..., MRD99) --- R6

In the above three relations

MRD1 --- MRDNO of first patient

MRD2 --- MRDNO of second patient and so on.

WBEDFILE(WARDBED, MRDNO) --- R7

WARDBED --- Ward number concatnated with Bed number.

DTADM(DATEADM, FSTMRD, LSTMRD, TOTAL) --- R8

FSTMRD --- First MRDNO on a given date

LSTMRD --- Last MRDNO on a given date

DATEADM -- Date of admission

BSTATUS(DATE, INFORMATION-ABOUT-BEDS-IN-EACH SPECIALITY) --- R9

SPECUNIT(DATE, INFORMATION-ABOUT-EACH-SPECIALITY AND EACH-UNIT) --- R10

HOSPITAL(DATE, INFORMATION-ABOUT-TOTAL-HOSPITAL) --- R11

Relations R1 to R8 contain the information relating to the patients and, therefore, are useful to the users. Relations R9 to R11 contain the information about the hospital status and are useful to the management.

By convention, we underline the key attribute and write the relation name outside the round braces.

3.3 Functional Dependencies and Normalization of the Proposed System:

An important type of relationship between attributes is FD which guarantees that for any two sets of attributes X & Y, X determines Y. So we will convert all the MVDs into FDs. We have three MVDs:

NAME $\longrightarrow\!\!\!\!\rightarrow$ MRDNO i.e. more than one patient can have the same name.

FNAME $\longrightarrow\!\!\!\!\rightarrow$ MRDNO i.e. more than one patient's father can have the same name.

ADDRESS $\longrightarrow\!\!\!\!\rightarrow$ MRDNO i.e. more than one patient can have the same address.

We convert NAME, FNAME and ADDRESS to NAMKEY, FNMKEY and ADDKEY respectively by concatenating two character strings (01 to 99).

We can represent the FDs of the proposed system by the following bijections:

MRDNO ⟶ TITLE, NAME, FNAME, WARDBED, ADDRESS, LADDRESS, NKADDRESS, PRVDGN, DATEADM, TMADM, CSTATUS, RELIGION, CGHS, AGE, SEX, USECODE, MONTHLY-INCOME, PHONENO, READM, SPUNIT, SPECIALITY, MALCHLD, MALADLT, PHYSICEXAM, PCOMPLAINT, PSTHISTORY, FINALDGN, DISEASE-CODE, TESTS, XRAY, DELIVERY, LABEXAM, OPERATION, OTHRTRTMNT, CONDITION, RESULT, ADVICE, DEATH, AUTOPSY, CAUSE-OF-DEATH, DATE-OF-DISCHARGE, TOTAL-STAY, FETAL-DEATHS, BILL

NAMKEY ⟶ MRDNO, NCODE

FNMKEY ⟶ MRDNO, FCODE

ADDKEY ⟶ MRDNO, ACODE

WARDBED ⟶ MRDNO, WCODE

DATEADM ⟶ FSTMRD, LSTMRD, TOTAL, DACODE

Relations R7 and R8 have repeating groups (nested segments). Fig. 3.1 illustrates the concept of repeating groups.

For relations RL7, RL8 and RL9, we can write FDs as

Key-attribute ⟶ Non-key attributes.

All the relations are normalized, so these are in 1NF. There is no nonfull FDs in any relation, hence, all are in 2NF. There are no transitive dependencies in any relation, therefore, all the relations are in 3NF. There are no MVDs that are not FDs, hence, the relations are in 4NF. In fact, we have already eliminated all the MVDs.

DATE	SPECNO	TOTAL-BED	...	...	PERCENTAGE-OF-OCCUPANCY

DATE	SPUNIT	ADMISSIONS	DISCHARGES	...	FEMALE-CHILD-DISCHARGED

(a)

DATES	TOTAL-BED	...	...		PERCENTAGE-OF-OCCUPANCY

DATESU	ADMISSIONS	DISCHARGES	...		FEMALE-CHILD-DISCHARGED

(b)

Fig. 3.1 REPEATING GROUPS IN TWO RELATIONS

(a) Before normalization
(b) After normalization

NOTE: DATES = DATE || SPECNO

DATESU = DATE || SPUNIT

3.4 Stored Relations:

In the stored relations, we study how these relations are stored on the storage. Generally the stored relations are different from the logical view. Relations R1 to R3 are stored as single relation called PERSON.

PERSON (MRDNO, TITLE, NAME, FNAME, WARDBED, ADDRESS, LADDRESS, NFADDRESS, PRVDGN, DATEADM, TMADM, CSTATUS, RELIGION, CGHS, AGE, SEX, MONTHLY-INCOME, PHONENO, READM, SPUNIT, SPECIALITY, USECODE, MALCHLD, MALADLT, PHYSICEXAM, PCOMPLAINT, PSTHISTORY, FINALDGN, DISEASE-CODE, TESTS, XRAY, DELIVERY, LABEXAM, OPERATION, OTHRTRTMENT, CONDITION, RESULT, ADVICE, DEATH, AUTOPSY, CAUSE-OF-DEATH, DATE-OF-DISCHARGE, TOTAL-STAY, FETAL-DEATHS, BILL) ---- RL1

In relation PERSON, attributes are defined as:

MRDNO	Medical Record Department Number
TITLE	DR., PROF., MR., MRS., MISS etc.
NAME	Patient's Name
FNAME	Father's name of patient
WARDBED	Ward No. concatnated with Bed-number
ADDRESS	Permanent address of patient
LADDRESS	Local address of patient
NKADDRESS	Next of kin's address
PRVDGN	Provisional Diagnosis
DATEADM	Date of Admission (YY MM DD)
TMADM	Time of Admission

CSTATUS	Civil status
CGHS	Central Government Health Scheme
READM	Readmission
PHYSICEXAM	Physical Examination
PCOMPLAINT	Present Complaint
OTHRTRTMNT	Other treatment
PSTHISTORY	Past history
USECODE	'U' for used, 'D' for deleted
SEX	Male ('M') or Female ('F').

Similarly relations R4 to R8 are stored as RL2 to RL6 respectively.

NAM(NAMKEY, MRDNO, NCODE) --- RL2

NAMKEY --- NAME concatnated with character representation of serial number (01 to 99)

NCODE --- 'U' for used, 'D' for discharged or deleted.

FNAM(FNMKEY, MRDNO, FCODE) --- RL3

FNMKEY --- FNAME concatnated with character representation of serial number (01 to 99)

FCODE --- 'U' for used and 'D' for deleted or discharged.

ADDRS(ADDKEY, MRDNO, ACODE) --- RL4

ADDRESS has four fields i.e. STREET, CITY, STATE and PIN.

```
1  ADDRESS
   2  STREET      CHAR (37),
   2  CITY        CHAR (15),
   2  STATE       CHAR (15),
   2  PIN         CHAR (6).
```

ADDKEY --- First 30 characters of STREET concatnated with character representation of serial number (01 to 99)

ACODE --- 'U' for used, 'D' for deleted or discharged.

WBED(WARDBED, MRDNO, WCODE) --- RL5

WCODE --- 'U' for used, 'D' for deleted and 'T' for transferred.

DTADM(DATEADM, FSTMRD, LSTMRD, TOTAL, DACODE) --- RL6

DATEADM --- Date of admission

FSTMRD --- First MRDNO on a given date

LSTMRD --- Last MRDNO on a given date

TOTAL --- Total patient on a given date

DACODE --- 'U' for used, 'D' for deleted or discharged.

Statistical relations R9 to R11 can be stored as:

BSTATUS(DATES, SPNAME, TOTAL-BED, TOTAL-BED-OCCUPIED, PERCENTAGE-OF-OCCUPANCY) --- RL7

DATES --- Date concatnated with speciality number (01 to 17)

SPNAME --- Speciality Name

SPUNIT(DATESU, ADMISSIONS, DISCHARGES, DAYS-OF-STAY, AVERAGE-LENGTH-OF-STAY, TOTAL-DEATHS, DEATHS-UNDER-48-HOURS, DEATH-OVER-48-HOURS, GROSS-DEATH-RATE, NET-DEATH-RATE, MALE-ADMISSION, FEMALE-ADMISSION, MALE-CHILD-ADMISSION, FEMALE-CHILD-ADMISSION, MALE-DISCHARGED, FEMALE-DISCHARGED, MALE-CHILD-DISCHARGED, FEMALE-CHILD-DISCHARGED) --- RL8

DATESU --- Date concatnated with speciality-unit (00 to 40)

HOSPITAL(DATE, BED-STRENGTH, TOTAL-PATIENTS-DISCHARGED, TOTAL-ADULTS-AND-CHILDREN-DISCHARGED, TOTAL-NEWBORN-INFANTS-DISCHARGED, DAYS-OF-CARE-TO-ADULTS-AND-CHILDREN, DAYS-OF-CARE-TO-NEWBORN-INFANTS, AVERAGE-STAY-ADULTS-AND-CHILDREN, AVERAGE-STAY-NEWBORN-INFANTS, BED-TURN-OVER-ADULTS-AND-CHILDREN, BED-TURN-OVER-NEWBORN-INFANTS, TOTAL-DEATHS, DEATH-UNDER-48-HOURS, DEATH-OVER-48-HOURS, GROSS-DEATH-DATE, NET-DEATH-RATE, FETAL-DEATHS, TOTAL-PATIENTS-ADMITTED, TOTAL-ADULTS-CHILDREN-ADMITTED, TOTAL-NEWBORN-INFANTS, MAX-DURING-WEEK, MAX-DURING-MONTH, MAX-DURING-YEAR, MIN-DURING-WEEK, MIN-DURING-MONTH, MIN-DURING-YEAR, DAYS-OF-CARE-TO-PATIENTS-IN-HOSPITAL, DAYS-OF-CARE-TO-ADULTS-AND-CHILDREN, DAYS-OF-CARE-TO-NEWBORN-INFANTS, TOTAL-DAILY-AVERAGE, TOTAL-AVERAGE-ADULTS-CHILDREN-DISCHARGED, TOTAL-AVERAGE-NEWBORN-INFANTS, AVERAGE-OCCUPANCY-ADULTS-CHILDREN, AVERAGE-OCCUPANCY-NEWBORN-INFANTS, GEOGRAPHICAL-DISTRIBUTION-OF-PATIENTS, TOTAL-NO-OF-DELIVERIES, TOTAL-NO-OF-OPERATIONS, TOTAL-NO-OF-XRAY-EXAM, TOTAL-LAB-EXAM, NEW-CASES, REPEAT-CASES, DAILY-AVERAGE-DELIVERIES, DAILY-AVERAGE-OPERATIONS, DAILY-AVERAGE-XRAY, DAILY-AVERAGE-LAB-EXAMS, DAILY-AVERAGE-NEW-CASES, DAILY-AVERAGE-REPEAT-CASES, DAILY-AVERAGE-ADULTS-CHILDREN-DISCHARGED, DAILY-AVERAGE-INFANTS-DISCHARGED, DAILY-AVERAGE-ADULTS-

CHILDREN-ADMITTED, DAILY-AVERAGE-
INFANTS-ADMITTED) --- RL9

In relations R4, R5 and R6, we have MVD of NAME, FNAME and ADDRESS with MRDNO. Normally name may be unique, therefore storing as in R4, R5 and R6 will be wastage of storage space. So we have implemented these relations viz R4, R5 and R6 in the form of RL2, RL3 and RL4 respectively. We have converted the above MVD into FD because in IS organization we require unique key and FD (after normalization) gives assurance that key will be unique. We try to convert all the relations in 3NF (if it is not possible to convert in 4NF). The concept of FD guarantees that for any two sets of attributes X and Y, X determines Y.

3.5 Architecture for the Proposed System:

We discuss the architecture for the proposed system at three levels: conceptual, external and internal.

(i) Conceptual Schema:

The conceptual schema may be thought of as a community user view. In our case, it consists of the relations RL1 to RL9 for the proposed data model. It also includes domain definition of the attributes used in those relations and conceptual/internal mapping.

The domain definitions for the proposed system are as given below:

DOMAIN	MRDNO	CHAR (6)
	TITLE	CHAR (5)
	NAME	CHAR (30)
	FNAME	CHAR (30)
	WARDBED	CHAR (5)
	ADDRESS	CHAR (73)
	LADDRESS	CHAR (70)
	NKADDRESS	CHAR (70)
	PRVDGN	CHAR (20)
	DATEADM	CHAR (6)
	TMADM	CHAR (4)
	CSTATUS	CHAR (20)
	RELIGION	CHAR (6)
	CGHS	CHAR (1)
	AGE	CHAR (4)
	SEX	CHAR (1)
	MONTHLY-INCOME	CHAR (5)
	PHONENO	CHAR (8)
	READM	CHAR (1)
	SPUNIT	CHAR (2)
	SPECIALITY	CHAR (2)
	MALCHLD	CHAR (1)
	MALADLT	CHAR (1)
	PHYSICEXAM	CHAR (78)
	PCOMPLAINT	CHAR (80)
	PSTHISTORY	CHAR (100)

FINALDGN	CHAR (20)
DISEASE-CODE	CHAR (7)
TESTS	CHAR (100)
XRAY	CHAR (60)
DELIVERY	CHAR (40)
LABEXAM	CHAR (60)
OPERATION	CHAR (70)
OTHRTRTMNT	CHAR (60)
CONDITION	CHAR (40)
RESULT	CHAR (40)
ADVICE	CHAR (40)
DEATHU48H	CHAR (1)
DEATHO48H	CHAR (1)
AUTOPSY	CHAR (60)
CAUSE-OF-DEATH	CHAR (15)
FETAL-DEATH	CHAR (15)
DATE-OF-DISCHARGE	CHAR (6)
TOTAL-STAY	CHAR (3)
BILL	CHAR (10)
NAMKEY	CHAR (32)
FNMKEY	CHAR (32)
ADDKEY	CHAR (32)
NCODE	CHAR (1)
FCODE	CHAR (1)
ACODE	CHAR (1)
WCODE	CHAR (1)
DACODE	CHAR (1)

USECODE	CHAR (1)
TOTAL	FIXED (3)
FSTMRD	CHAR (6)
LSTMRD	CHAR (6)
DATE	CHAR (6)
DATES	CHAR (8)
DATESU	CHAR (8)

The conceptual/internal mapping defines the correspondence between the data model and the internal model (or the stored data base). It also specifies how conceptual records and fields map into their stored counterparts. In this model, this is done by the keys of the records of the IS organization.

(ii) External Schema:

In the external schema, we study the external data model (the user's view of the portion of the data base used by him). It consists of the "external relations" (i.e. those known to the user) and the external/conceptual mapping together with the definitions of the domains on which relations are defined.

Relations R1 to R11 constitute the external relations. The domain definitions are the same as discussed above.

The external/conceptual mapping specifies the rule for extracting the external model from the conceptual model. In the proposed system, this function will be performed by a query language preprocessor to be discussed in section 3.7.

(iii) Internal Schema (or Storage Structure definition):

In the internal schema, we study the physical representation of the relations. The internal schema for our system consists of the physical representation of the relations as discussed in section 3.4.

The storage structure for the different files are as follows:

```
STORED FILE     PERSON (MRDNO, TITLE, NAME, FNAME, WARDBED,
                        ADDRESS, LADDRESS, NKADDRESS,
                        PRVDGN, DATEADM, TMADM, CSTATUS,
                        RELIGION, CGHS, SEX, USECODE,
                        MONTHLY-INCOME, PHONENO, READM,
                        SPUNIT, SPECIALITY, MALCHLD,
                        MALADLT, PHYSICEXAM, PCOMPLAINT,
                        PSTHISTORY, FINALDGN, DISEASE-CODE,
                        TESTS, XRAY, DELIVERY, LABEXAM,
                        OPERATION, OTHRTRTMNT, CONDITION,
                        RESULT, ADVICE, DEATHU48H,
                        DEATHO48H, AUTOPSY, CAUSE-OF-DEATH,
                        DATE-OF-DISCHARGE, TOTAL-STAY,
                        FETAL-DEATH, BILL)

                SEQUENCED ASCENDING MRDNO

                INDEXED MRDNO
```

```
STORED FILE   NAM (NAMKEY, MRDNO, NCODE)
              SEQUENCED ASCENDING NAMKEY
              INDEXED NAMKEY

STORED FILE   FNAM (FNMKEY, MRDNO, FCODE)
              SEQUENCED ASCENDING FNMKEY
              INDEXED FNMKEY

STORED FILE   ADDRS (ADDKEY, MRDNO, ACODE)
              SEQUENCED ASCENDING ADDKEY
              INDEXED ADDKEY

STORED FILE   WBED (WARDBED, MRDNO, WCODE)
              SEQUENCED ASCENDING WARDBED
              INDEXED WARDBED

STORED FILE   DTADM (DATEADM, FSTMARD, LSTMRD,
                     TOTAL, DACODE)
              SEQUENCED ASCENDING DATEADM
              INDEXED DATEADM
```

Similarly we can write as above for the relations RL7 to RL9.

In the proposed system, all the relational stored files have been provided with an IS organization (operating

under NOS of CDC Cyber 170/720).

The next section briefly describes the query language for the proposed system.

3.6 The query language for the proposed system:

There are two levels of the query language for the proposed system.

(i) Higher level or User level
(ii) Lower level or DBA level

In the user level, the user should know only what information there is in the data base and with which attributes, he can retrieve the information. He uses the queries with "GET" statement.

DBA level is more complex than the user level. This level is for the insertion, deletion and modification of records.

The existing relational calculus and relational algebra based DSLs viz. ALPHA, SEQUEL, 'Query By Example' etc. have not been used as the query language for the proposed system. These are quite sophisticated languages,

developed for intricate systems and require some sort of programming skill.

The proposed system is mainly for the medical personnel, management and users (e.g. dispatch clerks), who are not expected to have any programming experience and, hence the developed DSL, is a simple, non-procedural language.

Any valid query supported by the system begins with IF and ends with $ sign.

Authorized users only can make insertions, deletions, updations of tuples in the various relations of the system. Thus every user is provided with a security code specified by the reserved word "SECCODE" of the query language and the validity of the security code is checked against the existing set of codes stored in the system.

The following operations are acceptable:

GET: used for retrieval, specifies the attribute(s) or the relation to be retrieved.

INSERT: used for inserting relations in the data base.

MODIFY: for modifying the existing value in the relations. Only one attribute can be modified at a time.

DELETE: when a patient is discharged, all his/her data is deleted.

In all standard DSLs, any attribute name is qualified with reference to the relation to which it belongs. In the query language for the proposed system, the name of the relation is implicit and the DBMS is assumed to be powerful enough to know just by examining the attribute value, the relation to which it is associated e.g.
IF NAME = 'JOHN PERK HAYES' GET WARDBED $

In the above query, we have not specified the relation name. The DBMS will examine the attribute name and the associated files will be searched. Whenever a file is searched, its code is checked. If the code is 'D', then the results are not printed and information about deletion (discharged) is printed. If the code is 'U', this means the record is in 'use' (i.e. patient is staying in the hospital).

The valid queries derivable from the syntax given in Appendix-A following some semantic restrictions are the ones which are discussed with reference to both the levels viz. user and DBA level.

(i) User level:

In the domain part user can specify

(i) MRDNO
or
(ii) WARDBED
or
(iii) Any one or two of the following
NAME, FNAME, ADDRESS, DATEADM

In the target part, user can specify either one of the following FULLINFO, DIAGNOSTIC, TREATMENT and ALLREPORT or at most three of the following

MRDNO, WARDBED, NAME, FNAME, ADDRESS, DATEADM, LADDRESS, NKADDRESS, PHYSICEXAM, PCOMPLAINT, PSTHISTORY, FDIAGNOSIS, TESTS, DELIVERY, LABEXAM, OPERATION, OTHRTRTMNT, CONDITION, XRAY, ADVICE, RESULT.

We now discuss some of the queries with 'GET' statement.

1 Given MRDNO = ' 125724', find the address, ward-bed number and father's name of the patient.

The query language construct for the above query is:

IF MRDNO = '125724' GET ADDRESS AND WARDBED
AND FNAME $

The above query is processed as follows:

First, the query is checked with reference to the syntax. If it is found correct then the record corresponding to key MRDNO = '125724' is searched into file PERSON and the desired attributes/relations are retrieved.

2 Given ward number as 12 and bed number as 35, find the name, local address and father's name of the patient.

The query language construct for the above query is:
IF WARDBED = '12035' GET NAME AND FNAME AND
LADDRESS $

The above query is processed as follows:

File WBED is searched for WARDBED = '12035' and MRDNO is retrieved. With this MRDNO, record from file PERSON is obtained and requested information is retrieved.

3 Given the name as XYZ, find address, condition and date of admission.

The query language construct for the above query is:

IF NAME = 'XYZ' GET ADDRESS AND CONDITION AND DATEADM $

The above query is processed as follows:

NAME is concatnated with '01' and is called NAMKEY which is searched from file NAM and NAME is concatnated with '02', '03' and so on, until search is stopped. List of MRDNOs is retrieved in this way which are processed as above and requested information is received corresponding to these MRDNOs.

4 Given address = '7 Parliament Street' and date of admission as 18th December 1981 find condition and ward-bed number.

The query language construct for the above query is:

IF ADDRESS = '7 PARLIAMENT STREET' AND DATEADM = '811218' GET CONDITION AND WARDBED $

The above query is processed as follows:

List of MRDNOs is got corresponding to this ADDRESS and DATEADM from the files ADDRS and DTADM respectively

and their intersection is taken. Finally the file PERSON is searched with remaining MRDNOs to get the requested information.

We have developed the following algorithm which is written in bastard PL/I language, to process the queries with GET statement.

Algorithm:

```
Procedure GET
Input:   At most two attributes from Domain-part
         and at most three attributes from
         Target-part.
Output:  The information requested in the query is
         displayed.
Action:  The record identified by the attributes of
         the domain part of the query is fetched and
         the information requested by the attributes
         of the target part is displayed.
begin
dcl SD(4) init ('NAME', 'FNAME', 'ADDRESS',
                'DATEADM')
dcl SL(4) init ('FULLINFO', 'DIAGNOSTIC', 'TREATMENT',
                'ALLREPORT')
```

```
dcl RS(26) init ('MRDNO', 'NAME', 'FNAME', 'WARDBED',
           'DATEADM', 'ADDRESS', 'LADDRESS',
           'NKADDRESS', 'PHYSICEXAM', 'PSTHISTORY',
           'PCOMPLAINT', 'CONDITION', 'RESULT',
           'ADVICE', 'XRAY', 'DELIVERY', 'LABEXAM',
           'OPERATION', 'OTHRTRTMNT', 'TESTS',
           'FDIAGNOSIS', 'BSTATUS', 'DSTATUS',
           'WSTATUS', 'MSTATUS', 'ASTATUS')

IF KEY = 'DATE' Then
   begin
   display the corresponding status from RS(22 to 26)
   GO to LAST
   end
IF KEY = 'WARDBED' then
       begin
       Get MRDNO from WBED
       Go to SEARCH
       end
IF KEY = SD (1 to 4) then
       begin
       Get MRDNOs from corresponding file
       Go to SEARCH
       end
```

```
        else begin
             IF KEY (1) = SD (1 to 4) AND KEY (2) =
             SD (1 to 4) then
             begin
             Get MRDNOs from corresponding files
             Take common MRDNOs
             Go to SEARCH
             end
        else IF KEY = 'MRDNO' then go to SEARCH
             end
        end
SEARCH: Read a record from file PERSON with above MRDNO
        IF USECODE = 'D' then display "ALREADY DELETED"
        IF Target-attribute = SL (1 to 4) then
        Display the corresponding full file
        else begin
             Display the contents of Attribute 1, Attribute 2
             and Attribute 3
        /* Attribute 1, Attribute 2, Attribute 3 are from
           RS (1 to 21)  */
             end
LAST:  end
      STOP.
```

We discuss below some queries relating to the DBA level.

(ii) DBA level:

Data base Administrator is responsible for this level. Three types of operations are performed corresponding to this level viz.DELETE, INSERT and MODIFY.

These operations are performed by providing security codes. As far as retrieval is concerned, the information stored in the data base is not changed but in all the above operations for the DBA level the information is changed. Therefore some security is needed. Different users will have different security codes so that it can be known which user has updated the information.

We discuss below the operation for the DBA level viz. DELETE, INSERT and MODIFY.

1 DELETE:

Delete the patient whose data is given with MRDNO. The query language construct for the above query is:

```
IF SECCODE = 'BHATIA' DELETE      $
```

The data will be read from file DLT. The format of data is given in Appendix-B. Data is called the data for discharged status.

The query will be processed as:

The given data will be read from GET (DLT) LIST of PL/I language. Data will be read from the file till the end of the file. The records related to the patient are deleted by giving this data for all the patients to be discharged.

Firstly SECCODE will be checked for validity. If this is found correct, then the user will be allowed to perform the requested operation, else the request will be rejected. Since 'BHATIA' is a valid security code, the user will be allowed to delete the required records. The record is not deleted physically, but the USECODE, NCODE, FCODE, ACODE and WCODE in the files PERSON, NAM, FNAM, ADDRS and WBED respectively will be changed to 'D'. The data for discharged status will be inserted into the file PERSON.

We give below the algorithm for DELETE routine which we have developed in bastard PL/I language.

Algorithm:

Procedure DELETE

Input: File DLT which contains MRDNO and data for discharge status.

Output: Updated file PERSON.

```
        Action: It deletes all the records by putting 'D'
                code in all the files with the given MRDNO
                and inserts the data for discharge status
                into file PERSON. Many patients' record can
                be deleted with single query.
        begin
        On end file (DLT) Go to LEND
        DINF; Read a record for discharge status data from
                file DLT
            IF record is read from file PERSON with above
               MRDNO then
              begin
            IF USECODE = 'D' then "PATIENT ALREADY DELETED"
            else begin
                 Read record from files NAM, FNAM, ADDRS,
                 DTADM, WBED, PERSON having above MRDNO
                 Put Code "D" into all the files
                 Go to DINF
                 end
               end
            else display "RECORD DOES NOT EXIST IN THE SYSTEM
            CORRESPONDING TO THIS MRDNO"
LEND      ;end
STOP.
```

2 INSERT:

Insert full information of the patient whose MRDNO is given with data. The query language construct for the above query is:

IF SECCODE = 'SAXENA' INSERT FULLINFO $
followed by MRDNO and the data
(format given in Appendix-B).

The above query is processed as follows:

The data is read from the file INF till the end of file INF. Any number of patients can be inserted with the query. Firstly the SECCODE is matched against the standard security codes used by the system. If it matches, the user will be allowed to perform the requested operation, else the request will be turned down.

If instead of FULLINFO we have DIAGNOSTIC and TREATMENT, then the data will be read from DGN and TRT files respectively.

The algorithm which we have developed for INSERT routine is given below in bastard PL/I language.

Algorithm:

```
        Procedure INSERT
        Input: Given data in file INP/DGN/TRT with MRDNO
               and INSERT-ATTRIBUTE
        Output: Updated file PERSON
        Action: This procedure updates the file PERSON
                by inserting the records as given in file
                INP or modifies the information of records
                in PERSON file with the information given
                in DGN or TRT file.
        begin
        On end file (INP/DGN/TRT) Go to LAST
        IF INSERT-ATTRIBUTE = 'FULLINFO' then
           begin
Process-INP: Read a record from file INP
             Enter the record into file PERSON with given MRDNO
             Enter the record into all inverted IS files
             viz. NAM, FNAM, WBED, ADDRS, DTADM
             Go to Process-INP
             end
        else begin
             IF INSERT-ATTRIBUTE = 'DIAGNOSTIC' or 'TREATMENT'
             then begin
```

```
Process-DGN-TRT: Read record from file PERSON with given
          MRDNO
          Update the record in file PERSON with DGN or TRT
          file
          Go to Process-DGN-TRT
              end
          end
LAST:  RETURN
    end
STOP.
```

3 MODIFY:

This can be of two types:

(i) Updation of key attribute

(ii) Updation of nonkey attribute

Only one attribute can be updated at a time. These queries are answered by giving MRDNO and SECCODE. In key attribute i.e. only WARDBED in file WBED can be updated. In nonkey attributes, the following attributes can be updated one at a time

LADDRESS, NKADDRESS, PHYSICEXAM, PCOMPLAINT, PSTHISTORY, FDIAGNOSIS, TESTS, DELIVERY, LABEXAM, OPERATION, OTHRTRTMNT, CONDITION, XRAY, ADVICE, RESULT.

But NAME, FNAME, ADDRESS, DATEADM and MRDNO cannot be updated e.g. patient's name, father's name and permanent address do not change generally during the patient's stay in the hospital. Date of admission does not also change during the patient's stay. MRDNO does not change as it uniquely identifies each patient.

Now we give one example for each viz. updation of key attribute and updation of nonkey attribute.

(1) Updation of key attribute:

Shift to ward number 7, bed number 32 to patient whose MRDNO is 123456.

The query language construct for the above query is:

```
IF SECCODE = 'NARAIN' AND MRDNO = '123456' MODIFY
WARDBED = '07032'  $
```

The above query is processed as follows:

Previous WARDBED is got from the file PERSON with this MRDNO. Corresponding to the previous WARDBED, file WBED is searched and WCODE is changed to 'T' (transferred). The new record in the file WBED is written with WARDBED and MRDNO given in this query. In file PERSON, new WARDBED is replaced with the previous WARDBED.

(ii) Updation of nonkey attribute:

Change condition to 'improving' of the patient whose MRDNO is 654845.

The query language construct of the above query is:

```
IF SECCODE = 'NARAIN' AND MRDNO = '654845' MODIFY
CONDITION = 'IMPROVING'  $
```

The above query is processed as follows:

First the SECCODE is check for validity. File PERSON is searched with MRDNO = '654845' and CONDITION is replaced to 'IMPROVING' in file PERSON.

The algorithm for MODIFY routine follows, which has been developed in bastard PL/I language.

Algorithm:

```
Procedure MODIFY
Input: MRDNO, attribute-to-be-updated, new value
Output: Updated file PERSON
Action: This procedure modifies the value of
        attribute taking one at a time in the
        existing files.
```

```
begin
dcl ATTRIBUTE(16) init ('WARDBED', 'LADDRESS',
    'NKADDRESS', 'PHYSICEXAM', 'PCOMPLAINT',
    'PSTHISTORY', 'CONDITION', 'RESULT', 'ADVICE',
    'XRAY', 'DELIVERY', 'LABEXAM', 'OPERATION',
    'OTHRTRTMNT', 'TESTS', FDIAGNOSIS')
Read a record from file PERSON with given MRDNO
IF record not found then displey
"THE RECORD CORRESPONDING TO THIS MRDNO DOES NOT
EXIST IN THE SYSTEM"
else begin
     IF ATTRIBUTE = WARDBED
        begin
        Store Old WARDBED
        Rewrite the file PERSON with new WARDBED
        Write record with new WARDBED in WBED
        Put Code 'T' into WBED with old WARDBED
        end
     else begin
          Change the desired attribute
          Rewrite the file PERSON
          end
        end
     end
end
STOP.
```

3.7 The query language preprocessor:

The preprocessor consists of two main routines: (i) the syntax analysis routine (ii) semantic analysis and execution routine. The syntax of the input query is checked for validity according to its grammar as stated in Appendix-A. If first error is encountered then it gives message 'ERROR DETECTED BEFORE COLUMN NUMBER' followed by the column number of row. The syntax rules form an extremely compact definition of a large number of queries. The semantic constraints bring down the number of valid queries to the one discussed in section 3.6. The query is processed by semantic analysis and execution routine only when it is found to be syntactically correct. The length constraints were discussed in section 3.5.

The processing of the query language involves manipulation of strings to a great extent, hence the preprocessor was designed for PL/I due to the excellent string manipulation capabilities of PL/I language.

The processing of the queries for the proposed query language could be done either by means of a compiler designed explicitly for this purpose or by means of a preprocessor of some higher level language processor. The designing of a

compiler for the query language will require more language constructs viz. the inclusion of specific input/output statements etc. This may not maintain the simplicity and nonprocedurality of the query language and will thus mean greater programming overhead for the users of the system viz. the medical personnel and management personnel.

As the query language is intended to be simple, non-procedural and easily adoptable by users, a preprocessor to the standard PL/I compiler of CDC Cyber 170/720 system (running under NOS) was designed. The preprocessor converts the input query language string into a sequence of PL/I language instructions, which are processed by the PL/I compiler in the usual way.

The SYMBL routine whenever called takes one token of language at a time. The designed language is simple and free-formatted. Any number of blanks can be left between any two tokens of the language. The data is inserted via the LIST I/O statement of PL/I language which is also free-formatted.

We have developed an algorithm to process the syntax of the queries. The algorithm has been developed in bastard PL/I language. The algorithm for syntax is given below:

Algorithm:

```
Procedure SYNTAX
Input: The query
Output: Display "error" if query is invalid.
Action: This procedure calls another routine
        "Get symbol" which supplies tokens. It
        then checks the syntax of the query. If
        query is O.K. then it stores all the
        information from the query for utilizing
        in GET, DELETE, INSERT and MODIFY routines.

begin
dcl DOMAIN (8) Char (10) init ('DATE', 'MRDNO',
        'WARDBED', 'NAME', 'FNAME', 'DATEADM',
        'ADDRESS', 'SECCODE')
dcl TARGET (30) Char (10) init ('MRDNO', 'NAME',
        'ADDRESS', 'FNAME', 'DATEADM', 'WARDBED',
        'LADDRESS', 'NKADDRESS', 'PHYSICEXAM',
        'PCOMPLAINT', 'PSTHISTORY', 'FDIAGNOSIS',
        'TESTS', 'CONDITION', 'DELIVERY',
        'LABEXAM', 'OPERATION' 'OTHRTRTMNT',
        'XRAY', 'ADVICE', 'RESULT', 'FULLINFO',
        'DIAGNOSTIC', 'TREATMENT', 'ALLREPORT',
        'BSTATUS', 'DSTATUS', 'WSTATUS', 'MSTATUS',
        'ASTATUS')
```

```
        Read card
        Get symbol
        IF SYMBOL 7= 'IF' then display "error"
Check-domain:
        Get symbol
        IF SYMBOL 7 = DOMAIN (1 to 8) then display "error"
        Call Check-equal
        Call Store-data
        IF DATE checked then go to Check-status
        Get symbol
        IF SECCODE   MRDNO checked then go to MODIFY
        IF SYMBOL 7 = 'AND then go to GET 1
        IF FLAG = 1 then display "error"
        FLAG = 1
        Go to check-domain
        Get symbol
GET 1:  IF SECCODE checked then go to Delete-insert
        IF SYMBOL 7 = 'GET' then display "error"
        Get symbol
        IF SYMBOL 7 = TARGET (22 to 25) then go to NEXT
        else go to Check-dollar
NEXT:   IF SYMBOL 7 = TARGET (1 to 21) then display "error"
        Get symbol
        IF SYMBOL 7 = 'AND' then go to Check-dollar
        IF more than two 'AND' checked then display "error"
        else go to NEXT
```

```
Delete-insert: IF SYMBOL 7 = 'DELETE' then go to INSERT
       else go to check-dollar
INSERT: IF SYMBOL 7 = 'INSERT' then display "error"
       Get symbol
       IF SYMBOL 7 = TARGET (22 to 24) then display "error"
       Go to Check-dollar
MODIFY: IF SYMBOL 7 = 'MODIFY' then display "error"
       Get symbol
       IF SYMBOL 7 = TARGET (7 to 21) then display "error"
       Call check-equal
       Call Store-data
       Go to Check-dollar
Status:Get symbol
       IF SYMBOL 7 = TARGET (26 to 30) then display "error"
Check-dollar  Get symbol
       IF SYMBOL 7 = '$' then display "error"
       else display 'QUERY IS O.K.'
Procedure Check-equal
          Get symbol
          IF SYMBOL 7 = '=' then display "error"
          end Check-equal
Procedure Store-data
          Get symbol
          IF SYMBOL 7 = Quote then display "error"
          Get Symbol
     LOOP: IF Next-Char 7 = Quote then
           begin
           Increment counter
```

```
        Store data in Buffer
        Go to LOOP
end     Store-data
STOP.
```

Comment: The status attributes are defined as:

BSTATUS	-- Bed status
DSTATUS	-- Daily status
WSTATUS	-- Weekly status
MSTATUS	-- Montly status
ASTATUS	-- Annual status

The next section discusses the security and integrity in the proposed system.

3.8 Security and Integrity in the proposed system:

Security constraints are implemented by using security codes. Users of the system are provided with the security codes. For any kind of update operation (e.g. INSERT, DELETE, MODIFY) to be performed on the system, by any user, the validity of the security code of the user is checked against the standard set of security codes stored in the system. The user will be allowed to perform the requested operation only if the security code presented by him, to the system conforms with any one of the valid security codes.

Integrity constraints in the proposed system are the following:

(i) Only two unique keys MRDNO and DATE.

(ii) Only one candidate key WARDBED

(iii) FDs discussed previously.

(iv) Constraints of domain definitions.

Only one user at a time can modify existing tuples or attribute values in the various relations of the system and as long as his requested operation is not completely over, that user will have exclusive control on the system (by locking the data base e.g. setting some status byte on) and no other updation operations will be entertained during that time. The security is provided by requiring a terminal operator to identify himself by a unique password.

CHAPTER - IV

CONCLUSIONS AND SUGGESTIONS

4.1 Present status of the project:

The system has been successfully designed on the basis of relational approach. The relational data base model has been selected because of its logical simplicity and other attractive features. A simple query language has been designed to support the system and has also been checked with the computer. Programs are developed for the maintenance of the data base and also for answering different queries.

At each stage in designing, care has been taken to see that the users are not involved with the physical organization of the data and the different higher level file organization employed.

Data base is designed for supporting enquiries made by the patient's relative and for generating statistics for the menagement. The User is not involved with the details of relation name.

4.2 Future Work:

As outlined in chapter I, the various operations of the hospital are going to be automated gradually. This project work is the first attempt in that direction in India. The ideas presented in this dissertation will be of immense help to the future works on hospital information system. The proposed system can be implemented, with slight modifications, in similar environments e.g., Educational Institutions, Libraries, etc.

The system can be enhanced by implementing the statistics part. The language part is fully implemented for the statistical part.

REFERENCES

1 Armstrong, W.W. "Dependency Structures of data base relationships" Information Processing 74, North-Holland Pub. Co., Amsterdam, 1974, pp.580-83.

2 Beaman, P.D., Justice N.S. and Bernett G.O. "A Medical Information System and Data Sublanguage for Ambulatory Practices" Computer, vol.12, no.11, 1979, pp.9-18.

3 Beeri, C. and Bernstein, P.A. "Computational Problems Related to the Design of Normal Form Relational Schemas" ACM Transactions on Database Systems, vol.4, no.1, March 1979, pp.30-59.

4 Cardenas, A.F. "Analysis and Performance of Inverted Data Base Structures" Communications of ACM, vol.18, no.5, May 1975, pp.253-63.

5 Cardenas, A.F. "Data Base Management Systems" Allyn and Bacon, Inc., Boston, Massachusetts, 1979.

6 Codd, E.F. "A Relational Model of Data for Large Shared Data Banks" Communications of ACM, vol.13, no.6, June 1970, pp.377-87.

7 Codd, E.F. "Further Normalization of the Database Relational Model" In Data Base Systems Courant Inst. Computer Science Symposia 6 R. Rustin, Ed., Prentice-Hall, Englewood Cliffs New Jersey, 1972, pp.33-64.

8 Collen, M.F. "Hospital Computer Systems" John Wiley and Sons Inc., New York, 1974.

9 Date, C.J. "An Introduction To Database Systems" Addison Wesley Publishing Co. Inc., Philippines, 1977.

10 Delobel, C. "Normalization and Hierarchical Dependencies In the Relational Data Model" ACM Transactions on Database Systems, vol.3, no.3, 1978, pp.201-2.

11 Estrin, T. and Usgalis, R.C. "Information Systems for Patient Care", Computer, vol.12, no.11, 1979, pp.4-7.

12 Fagin, R. "Multivalued Dependency and a New Normal Form for Relational Database", ACM Transactions on Database Systems, vol.2, no.3, 1977, pp.262-78.

13 Ghosh, S.P. "Data Organization for Data Management" Academic Press, New York, 1977.

14 Kapp, D. and Leben, J.F. "IMS Programming Techniques: A Guide to Using DL/I" Van Nostrand Series, New York, 1977.

15 Martin, J. "Principles of Database Management", Prentice Hall of India, New Delhi, 1977.

16 Mendelzon, A.O. "Multivalued Dependencies in Relational Database" Journal of the Association for Computing Machinery, vol.26, no.1, 1979, pp.37-44.

17 Ross, R.G. "Database Systems: Design, Implementation and Management" AMACOM, New York, 1978.

18 Sprowls, C. "Management Databases" John Wiley and Sons Inc., Santa Barbara, California, 1977.

19 Ullman, J.D. "Principles of Database Systems", Computer Science Press, 1980.

20 Wiederhold, G. "Database Design", Mc-Graw Hill, New York, 1977.

APPENDIX - A

SYNTAX OF THE QUERY LANGUAGE

⟨Query-language⟩ :: = ⟨Query-input⟩

⟨Query-input⟩ :: = ⟨Query-statement⟩ | X

⟨Query-statement⟩ :: = ⟨Get-query⟩ / ⟨Insert-query⟩ /
⟨Delete-query⟩ / ⟨Modify-query⟩

⟨Get-query⟩ :: = ⟨Get-target⟩ / ⟨Get-status⟩

⟨Insert-query⟩ :: = IF ⟨SecCode⟩ INSERT ⟨Insert-target⟩ $

⟨Delete-query⟩ :: = IF ⟨Sec-Code⟩ DELETE $

⟨Modify-query⟩ :: = IF ⟨Sec-Code⟩ AND ⟨Main-key⟩ MODIFY
⟨Modify-key-word⟩ $

⟨Get-target⟩ :: = IF ⟨Domain-part⟩ GET
⟨Target-part⟩ $

⟨Get-status⟩ :: = IF ⟨Date-key⟩ GET ⟨Status⟩ $

⟨Sec-Code⟩ :: = SECCODE = ' ⟨Domain-value⟩ '

⟨Insert-target⟩ :: = FULLINFO/DIAGNOSTIC/TREATMENT

⟨Main-key⟩ :: = MRDNO = ' ⟨Domain-value⟩ '

⟨Modify-key-word⟩ :: = ⟨Modify-attribute⟩ =
' ⟨Domain-value⟩ '

⟨Domain-part⟩ :: = ⟨Wbed-key⟩ / ⟨Main-key⟩ /
/ ⟨Domain-stat⟩ / ⟨Domain-stat⟩
AND ⟨Domain-stat⟩

⟨Target-part⟩ :: = ⟨Target⟩ AND $\left\{\langle \text{Target} \rangle\right\}_{0}^{2}$ |
⟨Target-attribute⟩

⟨Date-key⟩ :: = DATE = '⟨Domain-value⟩'

⟨Status⟩ :: = BSTATUS/DSTATUS/WSTATUS/MSTATUS/
ASTATUS

⟨Domain-value⟩ :: = ⟨Alphabetic⟩ / ⟨Alphabetic⟩
⟨Domain-value⟩

⟨Modify-attribute⟩ :: = WARDBED/LADDRESS/NKADDRESS/
PHYSICEXAM/PCOMPLAINT/PSTHISTORY/
FDIAGNOSIS/TESTS/XRAY/DELIVERY/
LABEXAM/OPERATION/CONDITION/
OTHRTRTMNT/RESULT/ADVICE

⟨Wbed-key⟩ :: = WARDBED = '⟨Domain-value⟩'

⟨Main-key⟩ :: = MRDNO = '⟨Domain-value⟩'

⟨Domain-stat⟩ :: = ⟨Domain⟩ = '⟨Domain-value⟩'

⟨Target⟩ :: = MRDNO/WARDBED/NAME/FNAME/DATEADM/
ADDRESS/LADDRESS/NKADDRESS/
PHYSICEXAM/PCOMPLAINT/PSTHISTORY/
FDIAGNOSIS/TESTS/XRAY/DELIVERY/
LABEXAM/OPERATION/CONDITION/RESULT/
OTHRTRTMNT/ADVICE

⟨Target-attribute⟩ :: = FULLINFO/DIAGNOSTIC/
TREATMENT/ALLREPORT

⟨Alphabetic⟩ :: = A/B/C/D/E/F/G/H/I/J/K/L/M/N/O/
P/Q/R/S/T/U/V/W/X/Y/Z/0/1/2/3/4/
5/6/7/8/9/ --/ // blank/./*/./
-/+/:/=

⟨Domain⟩ :: = NAME/FNAME/DATEADM/ADDRESS

Comment: X in the first column of the card terminete the syntax analyzer program.

APPENDIX - B

FILE FORMAT

The file INP provides data after the INSERT query given below:

IF SECCODE = 'BHATIA' INSERT FULLINFO $

The file INP has the following format:

					PERMANENT ADDRESS			
MRDNO	TITLE	NAME	FNAME	WARDBED	STREET	CITY	STATE	PIN
'6X'	'5X'	'30X'	'30X'	'5X'	'37X'	'15X'	'15X'	'6X'

LOCAL ADDRESS	NEXTKIN-ADDRESS	PROVISIONAL DIAGNOSIS	DATEADM	TIME-ADM	CSTATUS
'73X'	'73X'	'20X'	'6X'	'4X'	'20X'

RELIGION	CGHS	AGE	SEX	MONTHLY-INCOME	PHONENO
'6X'	'X'	'4X'	'X'	'5X'	'8X'

READM	SPUNIT	SPECIALITY	USECODE	MALCHLD	MALADLT
'X'	'2X'	'2X'	'X'	'X'	'X'

PHYSICAL-EXAM	PRESENT-COMPLAINT	PAST-HISTORY
'70X'	'80X'	'100X'

FINAL-DIAGNOSIS	DISEASE-CODE	TESTS	XRAY	DELIVERY
'20X'	'10X'	'100X'	'60X'	'40X'

LAB-EXAMS	OPERATIONS	OTHER-TREATMENT
'60X'	'70X'	'60X'

CONDITION	RESULT	ADVICE	DEATHU48H	DEATH048H
'40X'	'40X'	'40X'	'X'	'X'

AUTOPSY	CAUSE-OF-DEATH	FETAL-DEATH	DATE-OF-DISCHARGE	TOTAL-STAY	BILL
'60X'	'15X'	'15X'	'6X'	'3X'	'10X'

Note: X denotes one character. nX denotes n characters where n is a natural number and depends on the constraints used in domain definition. Quote is a must on both sides of data and at least one comma or blank is necessary between two data values.

The file DGN provides data after the following INSERT query:

IF SECCODE = 'SAXENA' INSERT DIAGNOSTIC $

The file DGN has the following format:

MRDNO	PHYSICAL-EXAM	PRESENT-COMPLAINT
'6X'	'78X'	'80X'

PAST-HISTORY	FINAL-DIAGNOSIS	DISEASE-CODE
'100X'	'20X'	'10X'

The file TRT provides after the following query for INSERT operation i.e.

IF SECCODE = 'NARAIN' INSERT TREATMENT $

The file TRT Has the following format:

MRDNO	TESTS	XRAY	DELIVERY	LABEXAM	OPERATION
'6X'	'100X'	'60X'	'40X'	'60X'	'70X'

OTHER-TREATMENT	CONDITION	RESULT	ADVICE
'60X'	'40X'	'40X'	'40X'

The file DLT provides data after the DELETE query given below:

IF SECCODE = 'BHATIA' DELETE $

The file DLT has the following format:

MRDNO	DEATHU48H	DEATH048H	AUTOPSY	CAUSE-OF-DEATH
'6X'	'X'	'X'	'60X'	'15X'

FETAL-DEATH	DATE-OF-DISCHARGE	TOTAL-STAY	BILL
'15X'	'6X'	'3X'	'10X'

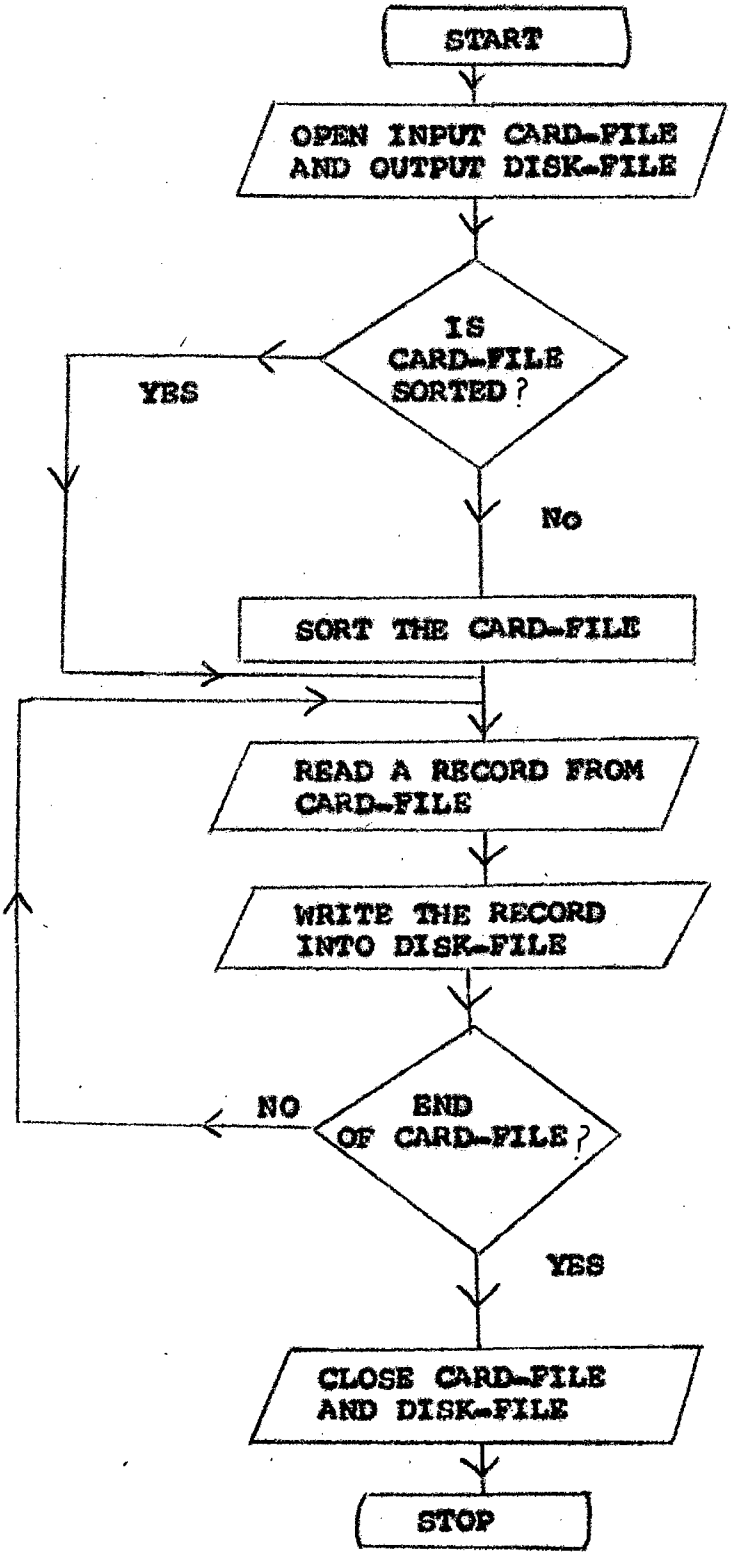


Fig.1. FLOWCHART FOR CREATING AN INDEXED SEQUENTIAL FILE

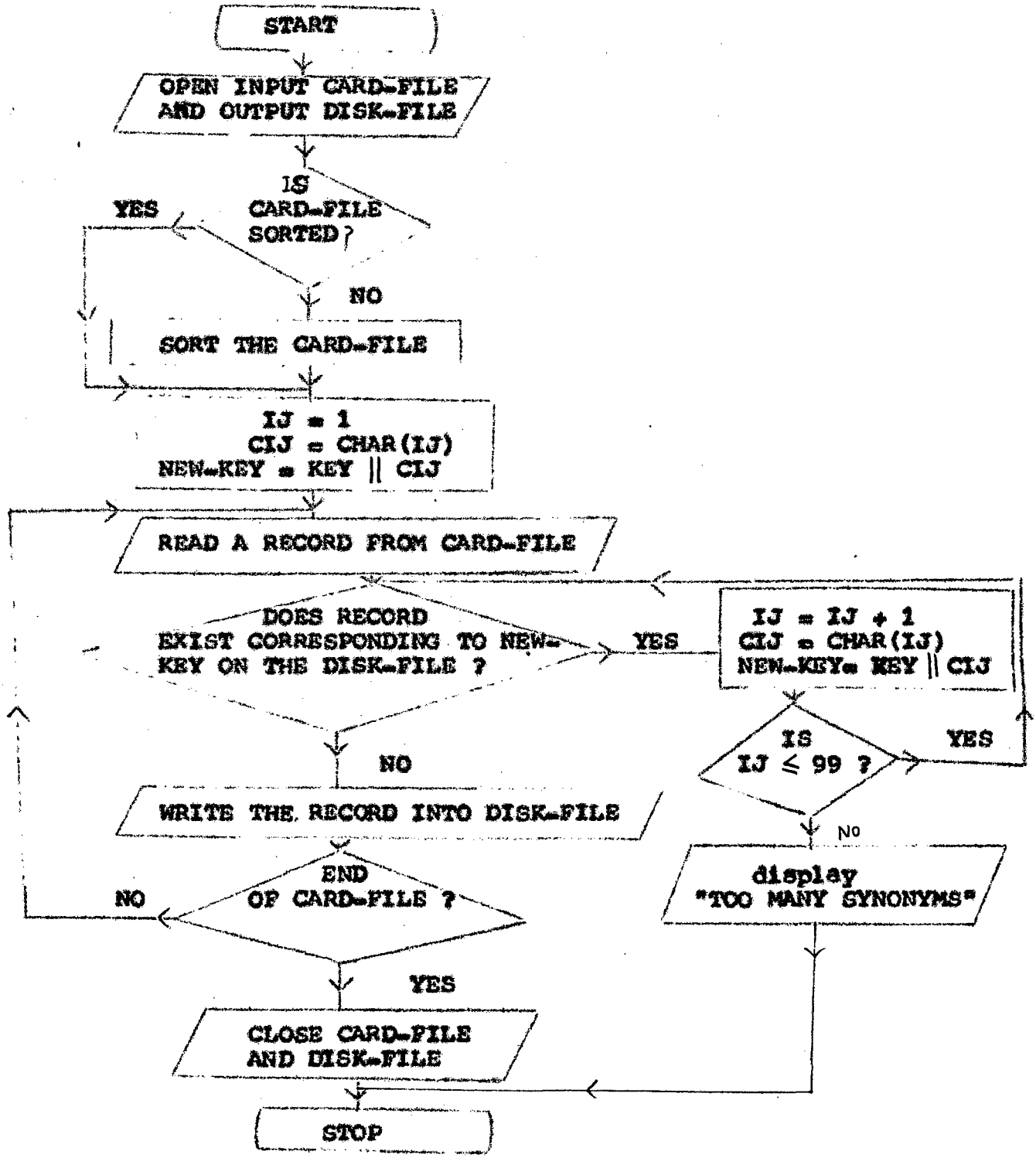
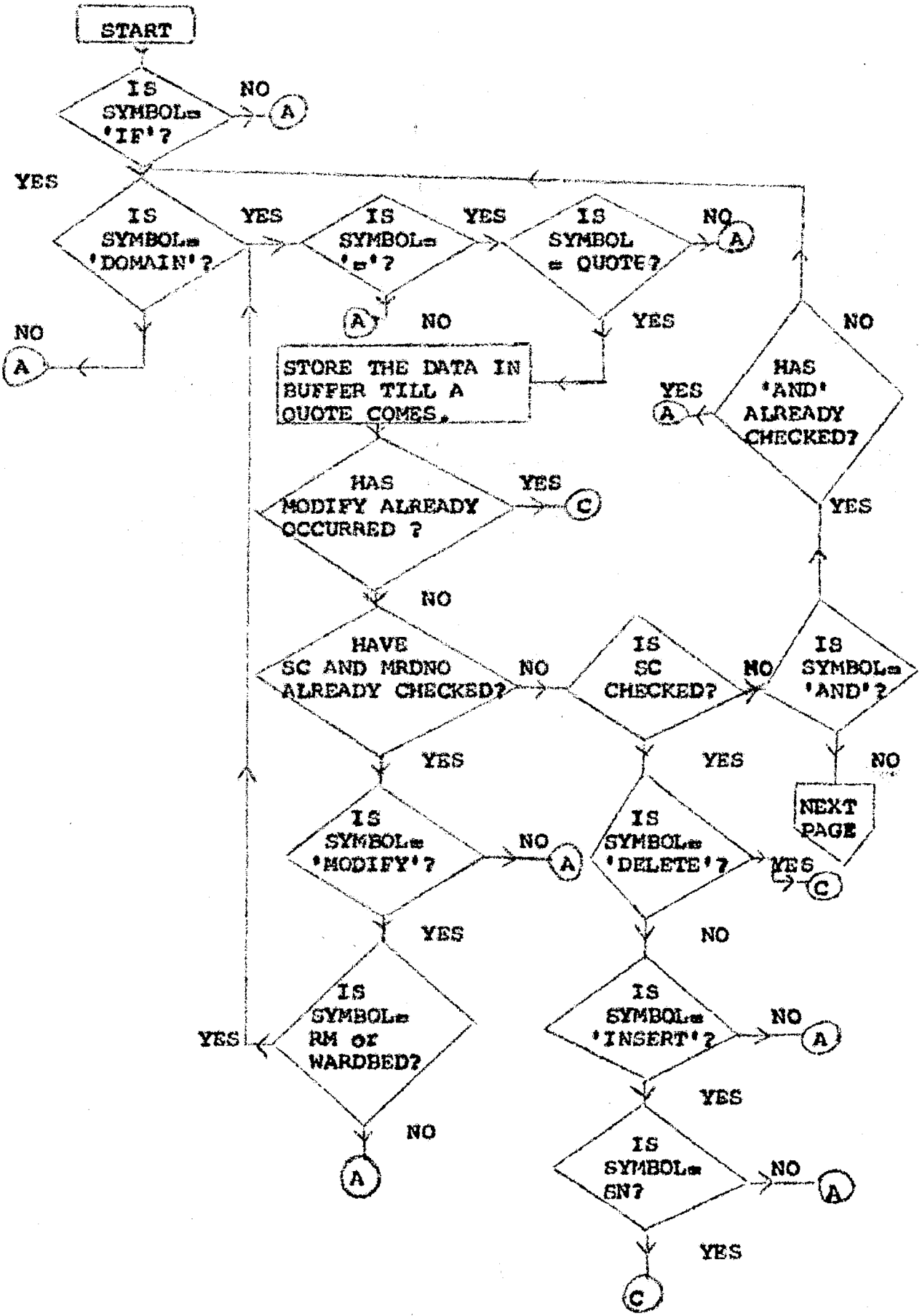
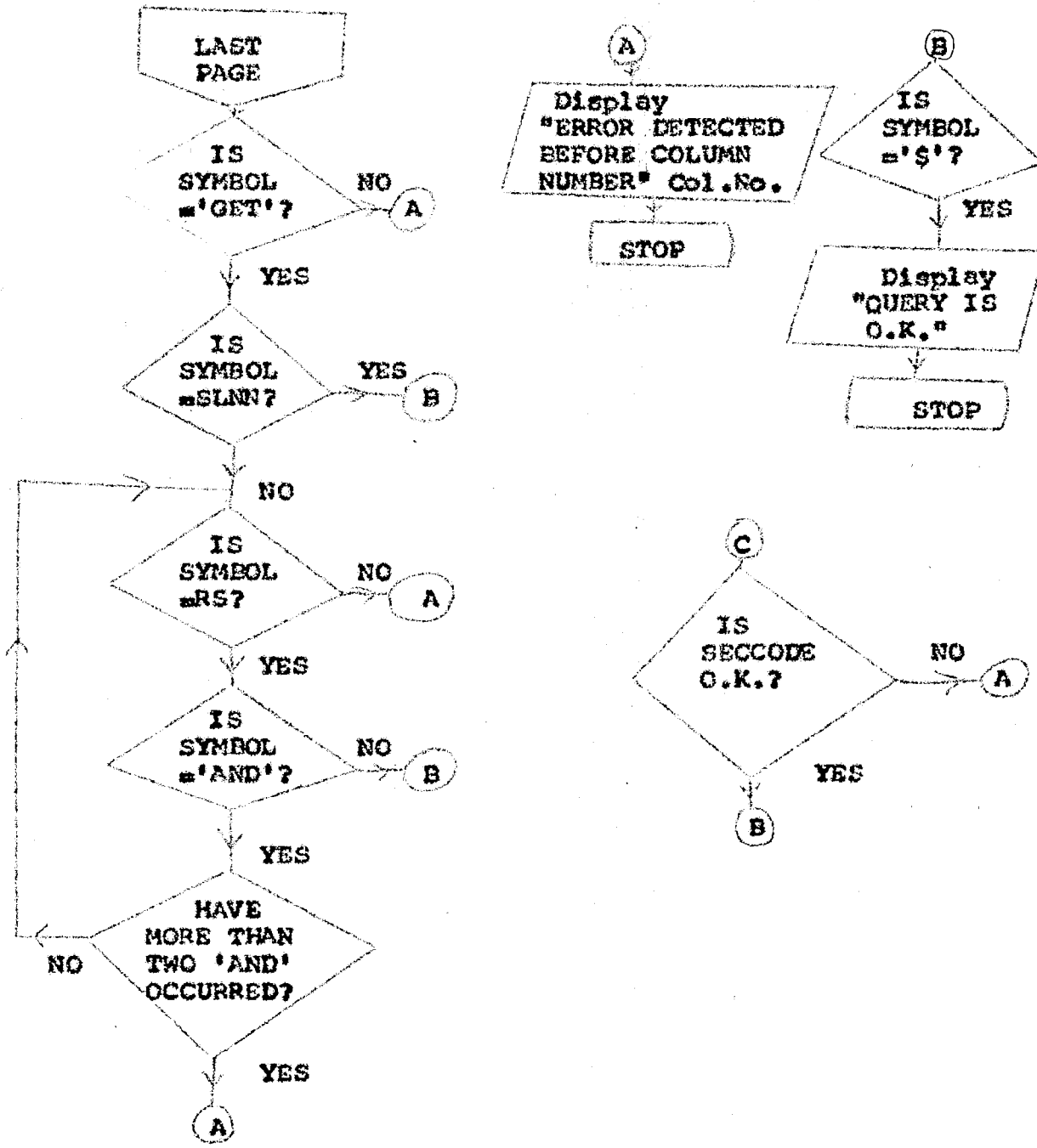


Fig. 2. FLOWCHART FOR CREATING AN INVERTED INDEXED SEQUENTIAL FILE


START
IS SYMBOL= 'IF'?
NO
A
YES
IS SYMBOL= 'DOMAIN'?
YES
IS SYMBOL= '='?
YES
IS SYMBOL = QUOTE?
NO
A
NO
A
A
NO
YES
STORE THE DATA IN BUFFER TILL A QUOTE COMES.
HAS MODIFY ALREADY OCCURRED ?
YES
C
NO
HAVE SC AND MRDNO ALREADY CHECKED?
NO
IS SC CHECKED?
NO
IS SYMBOL= 'AND'?
NO
YES
YES
YES
NEXT PAGE
IS SYMBOL= 'MODIFY'?
NO
A
IS SYMBOL= 'DELETE'?
YES
C
YES
NO
IS SYMBOL= RM or WARDBED?
YES
NO
A
IS SYMBOL= 'INSERT'?
NO
A
YES
IS SYMBOL= SN?
NO
A
YES
C
HAS 'AND' ALREADY CHECKED?
YES
A
NO
YES

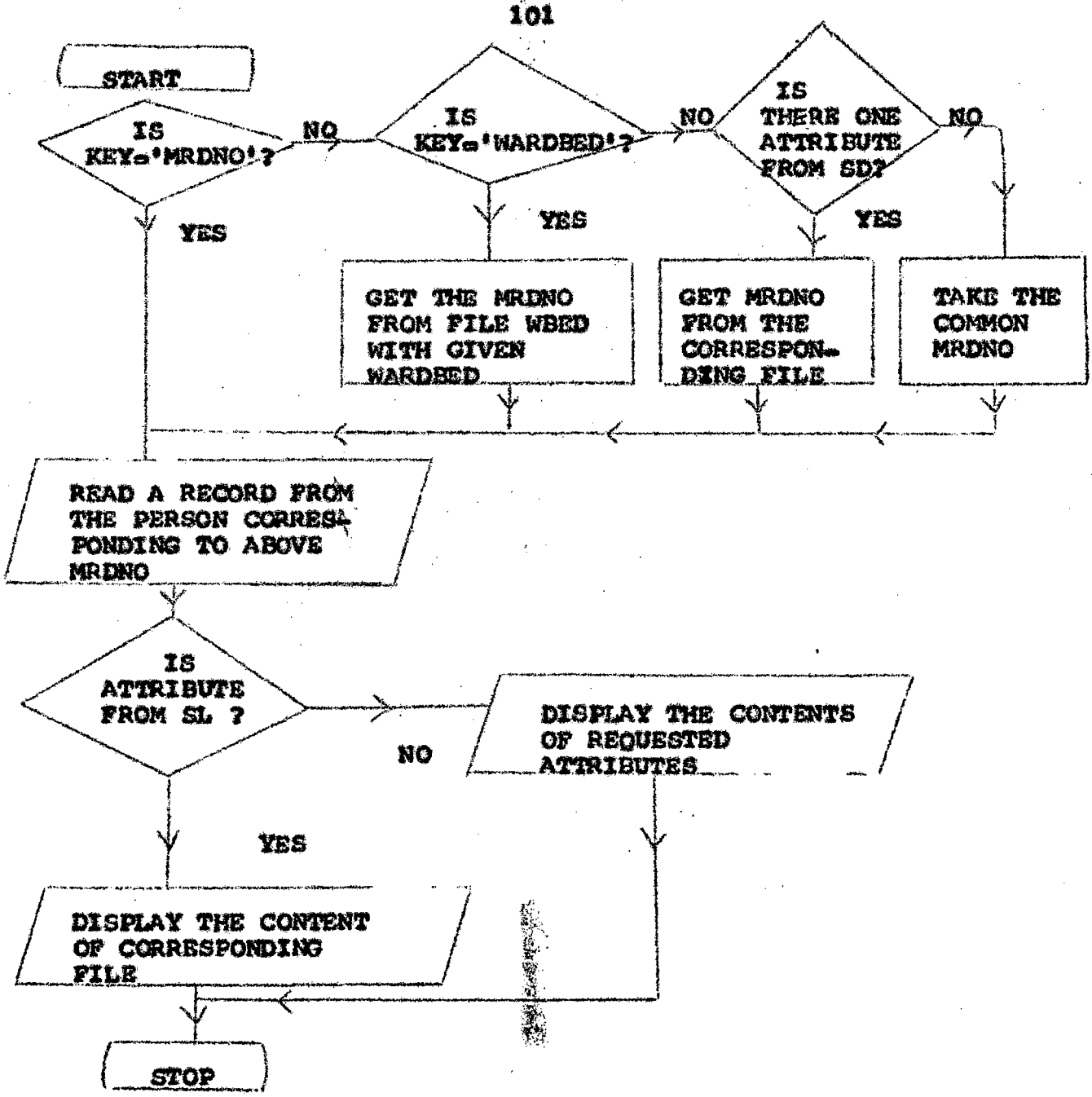
cont'd



⟨SC⟩ :: = SECCODE
⟨RM⟩ :: = LADDRESS/NKADDRESS/PHYSICEXAM/PCOMPLAINT/PSTHISTORY/
CONDITION/RESULT/ADVICE/XRAY/DELIVERY/LABEXAM/
OPERATION/OTHRTRTMNT/TESTS/FDIAGNOSIS
⟨SN⟩ :: = FULLINFO/DIAGNOSTIC/TREATMENT
⟨SLNN⟩ :: = FULLINFO/DIAGNOSTIC/TREATMENT/ALLREPORT
⟨RS⟩ :: = ⟨RM⟩ / ⟨DOMAIN⟩

⟨DOMAIN⟩ :: = SECCODE/WARDBED/MRDNO/NAME/FNAME/DATEADM/ADDRESS

Fig.3. FLOWCHART FOR SYNTAX ANALYZER



<SD> :: = NAME/FNAME/ADDRESS/DATEADM
<SL> :: = FULLINFO/DIAGNOSTIC/TREATMENT/ALLREPORT
<RS> :: = MRDNO/NAME/FNAME/WARDBED/DATEADM/ADDRESS/
LADDRESS/NKADDRESS/PHYSICEXAM/PSTHISTORY/
PCOMPLAINT/CONDITION/RESULT/ADVICE/XRAY/
DELIVERY/LABEXAM/OPERATION/ERE OTHRTRTMNT/
TESTS/FDIAGNOSIS

Fig.4. FLOWCHART FOR GET ROUTINE

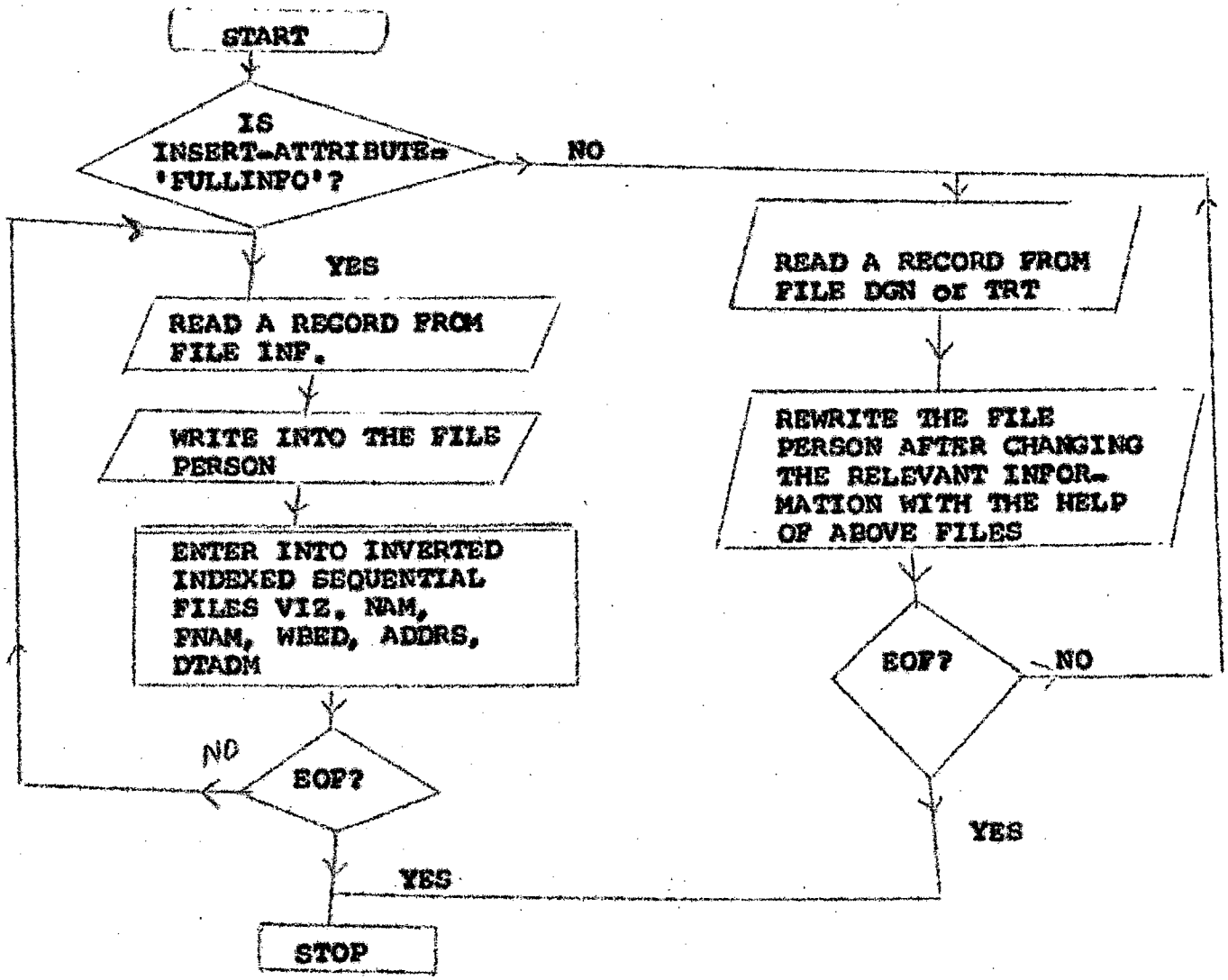


Fig.5. FLOWCHART FOR INSERT ROUTINE

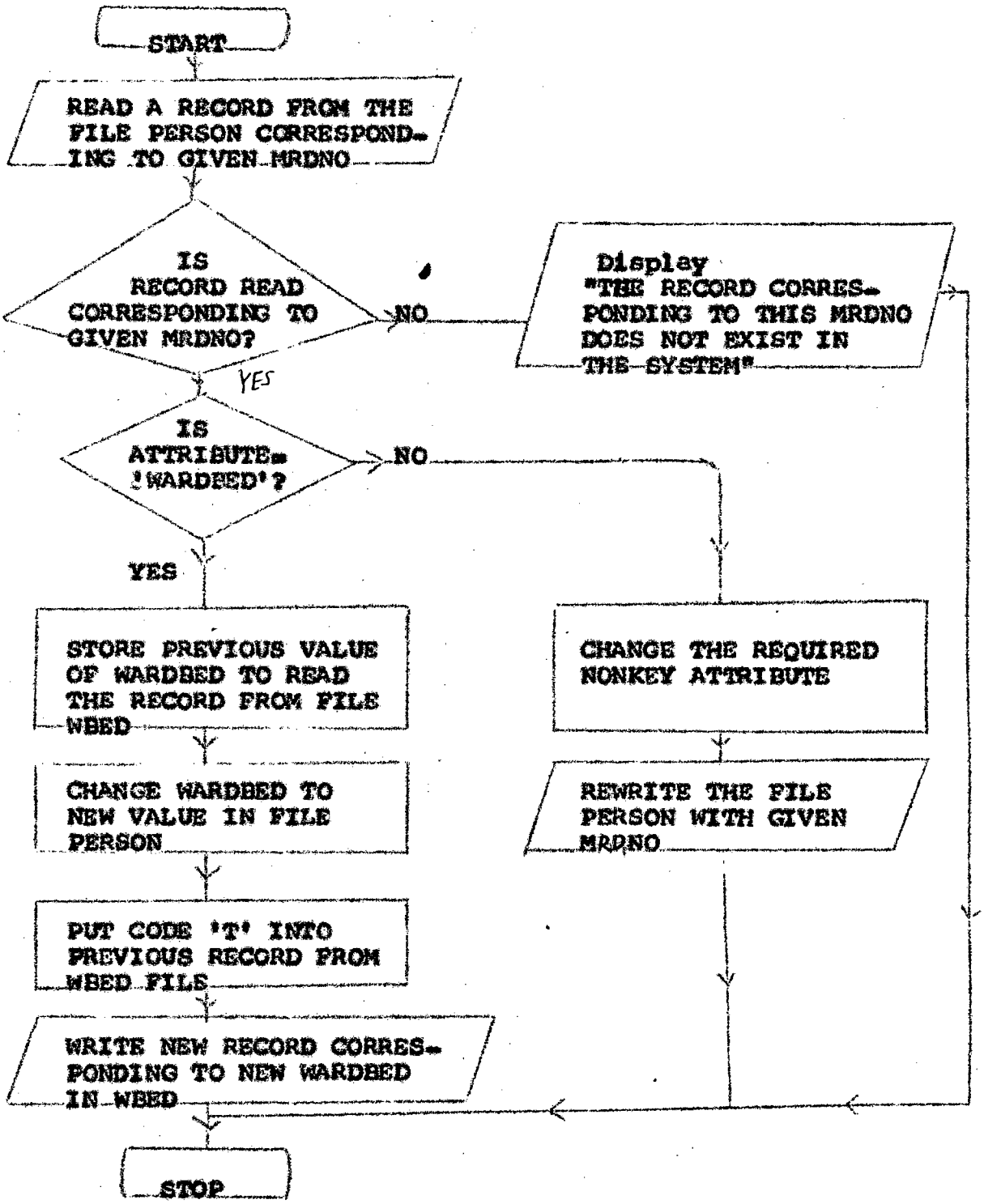


Fig.6. FLOWCHART FOR MODIFY ROUTINE

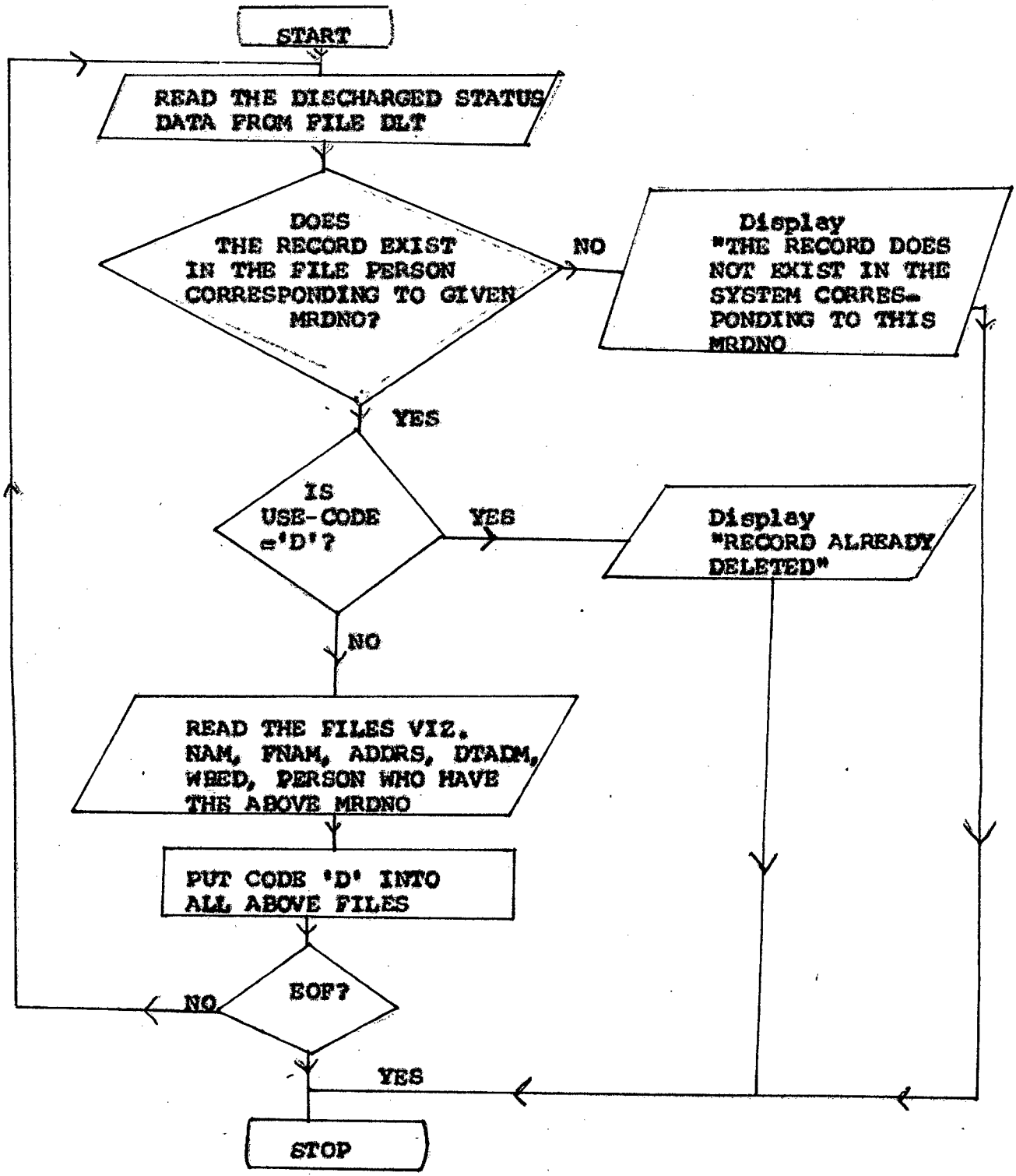
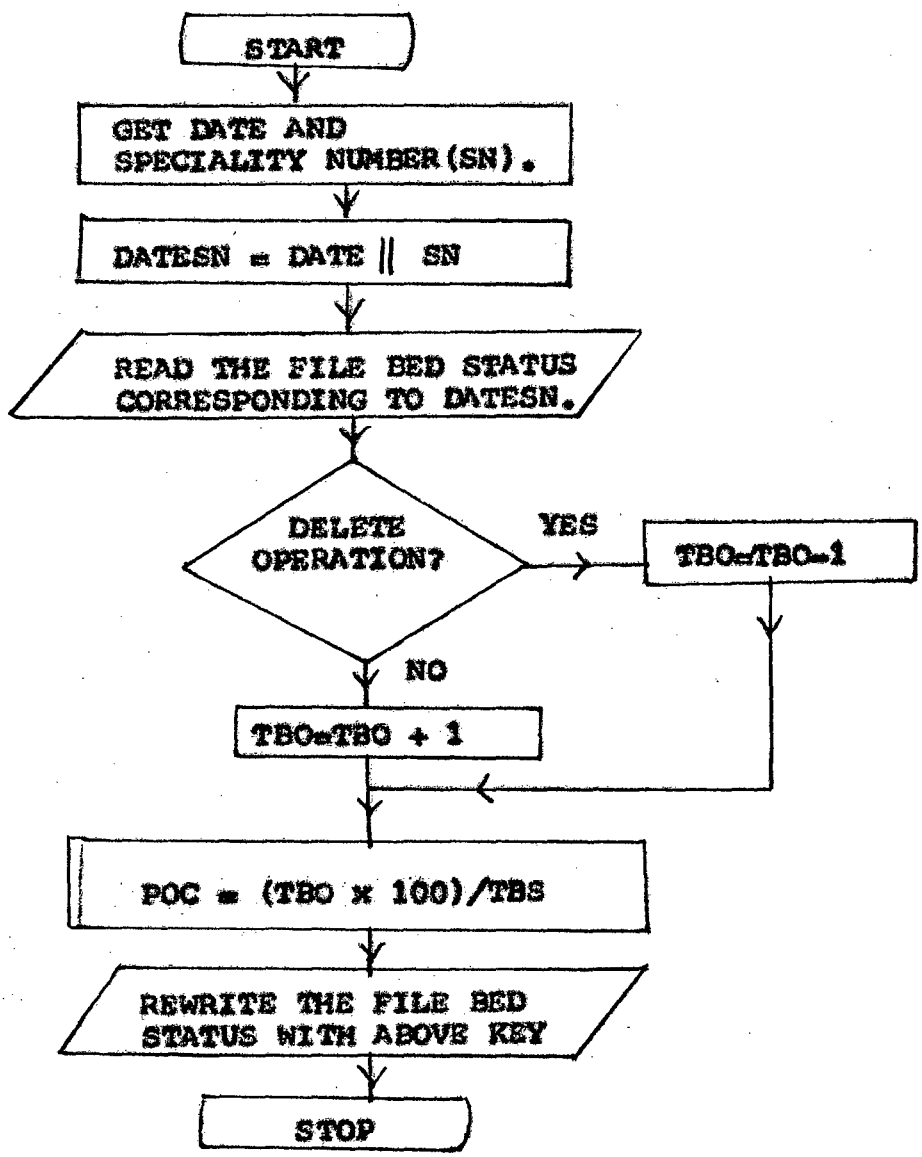


Fig.7. FLOWCHART FOR DELETE ROUTINE



Note: TOTAL-BED-OCCUPIED = TBO
TOTAL-BED-STRENGTH = TBS
PERCENTAGE-OF-OCCUPANCY = POC

Comment: Similarly for other statistics files, we can write flow-charts. Care must be taken to see which attributes are affected in INSERT/DELETE. MODIFY/GET has no effect on the statistics.

Fig. 8. FLOWCHART FOR BED STATUS

APPENDIX - D

A VIEW OF HOSPITAL STATISTICS

1 INTRODUCTION:

Hospital Statistics are one of the important components of Health Information System of a country. These can be divided into two groups, one relating to the Hospital and the other to the Patient.

The Hospital-Group comprises the following types of statistics:

i) Administrative & General Statistics

ii) Financial Statistics

iii) Hospital Service Statistics - also termed as Performance Statistics

The Patient-Group consists of:

i) Socio-economic data of the patient

ii) Morbidity data

2 VALUES OF GOOD HOSPITAL STATISTICS:

Good hospital statistics can be utilized by:

2.1. Medical Superintendent or Hospital Administrator:

for i) establishing administrative control over the functional activities;

ii) calculating average cost per unit of service rendered;

iii) planning of:

a. additional facilities
b. hospital growth
c. staff
d. equipment
e. training programmes
f. operating budgets etc.

iv) increasing the quality of care rendered to patients.

2.2. National and International Health Agencies:

for i) planning of health services;

ii) taking suitable measures for the control and prevention of diseases.

2.3. Public Safety Officials:

for i) developing measures for prevention of accidents;

ii) launching educational safety campaigns.

2.4. Census Officials and Demographers:

for i) estimating fertility of the population;

ii) predicting population growth etc.

2.5. Research Workers:

for planning and conducting researches.

3 MINIMAL HOSPITAL STATISTICS & METHODS OF COLLECTION:

The types of statistical data can vary considerably in different hospitals according to the needs of the management. But minimal statistical information, details given below, which can be compiled and published by a well organised Medical Record Department in a hospital, may be regarded as general pattern.

3.1 Hospital Service Statistics:

3.1.1. Out Patient Services:

i) Number with details of Out-patient Departments, Special Clinics and Casualty Services.

ii) Total No. of new cases

iii) Total No. of Repeat Cases

iv) Total No. of new & repeat cases

v) Department-cum-Unit wise break up of cases

vi) Geographical distribution of new cases

vii) Sex-wise distribution of cases

The above data can be compiled from the Registers maintained at the Registration Counters in the Out-patient Departments and Special Clinics and Casualty Services.

3.1.2. In-patient Service:

i) Bed Complement:

a. Total number of beds

b. Speciality-wise break up of beds & occupancy of beds.

ii) Admissions:

a. Total no. with break up according to Adults & Children and Newborns.
(In order to conform to universal practice, data on babies born in the hospital, must be shown separately)

b. Sex-wise distribution
c. Department-cum-Unit wise break up.

iii) Discharges (including Deaths):

a. Total no. with break up according to Adults & Children and Newborns.
b. Sex-wise distribution
c. Department-cum-Unit wise break up
d. Hospital days
e. Geographical distribution
f. Deaths - under 48 hours of admission & over 48 hours of admission

iv) Births:

a. Total no. of babies born in the hospital
b. Sex-wise distribution

v) Patient-movement:

a. No. of patients at the beginning of the day;
b. No. of patients admitted during the day;
c. No. of patients discharged and died during the day;
d. No. of patients transferred to and from the wards;
e. No. of patients remaining at the end of the day.

The data on Admissions can be compiled either from the Central Admission Register or from the duplicate copies of the Admission Record Forms of the patients admitted to hospital.

The data on Discharges (including Deaths) can be compiled by analysing medical reports of discharged patients. Specially designed analysis sheets are used for this purpose.

The data on Births can be compiled either from the Birth Registers or duplicate copies of Admission Record Forms prepared for babies born in the hospital.

The data on patient-movement can be compiled from the Daily Ward Census Reports prepared by the Nursing Staff on night duty. The period covered in these reports is from midnight to midnight. These reports can be collected daily by the Census Clerks of the Medical Record Department in a large hospital or can be sent to theDepartment by the Nursing Staff in a small hospital.

3.1.3. Operations:

i) Total no. of operations performed

ii) Break up showing Major and Minor Operations if these terms are well defined.

3.1.4. Laboratory Services:

i) Total no. of examinations done

ii) Types of examinations done

3.1.5. Radiology Services (including Radio-Therapy)

i) Total no. of X-ray Examinations done

ii) Total no. of Treatments given

3.1.6. E.C.G. & E.E.G

i) Total no. of Examinations done

The data on Operations, Examinations etc. done in the Pathology Labs., Radiology Departments and E.C.G. & E.E.G. Labs., can be compiled from the Registers maintained in the respective areas.

3.1.7. Rates:

The following Rates can be compiled. Formulae are given at S.No.4.

i) Mortality Rates:

a. Gross Death Rate (Non-Institutional Death Rate)
b. Net Death Rate (Institutional Death Rate)

Other Death Rates which can be compiled subject to availability of relevant data are:

c. Anaesthesia Death Rate
d. Post-operative Death Rate
e. Maternal Death Rate
f. Neo-natal Death Rate
g. Hospital Autopsy Rate & Net Autopsy Rate, if autopsies are done in the hospital and data available.

ii) Cesarean Section Rate
iii) Post-Operative Infection Rate, if data available

iv) Consultation Rate
v) Average Length of Stay - Overall
Adults &
Children
Newborns
vi) Daily average No. of patients - Total
Adults &
Children
Newborns
vii) Bed Turn Over Rate - Overall
Adults &
Children
Newborns
viii) Average Percentage of Bed Occupancy - Overall
Adults &
Children
Newborns
ix) Daily Average Out-patient Attendance.

3.2. Hospital Morbidity Statistics:

3.2.1. Out-patients:

i) Diagnostic Statistics
ii) -do- by Sex
iii) -do- by age - according to specific age groups adopted.

The Diagnostic Statistics of Out-patients are based on the diagnossis established at the first visit of the patient within a disease episode. The data which can be compiled from medical records of out-patients, are presented according to the Detailed List of I.C.D. (9th Revision) and the National List.

3.2.2. <u>In-patients</u>:

i) Diagnostic Statistics
ii) Statistics on Operative Procedures
iii) Sex-wise distribution of the above data
iv) Age-wise break up of the above data according to specific age groups adopted
v) Hospital days
vi) Result on Discharge

Diagnostic Statistics of In-patients are based on the diagnosis at discharge because these will incorporate the results of all diagnostic investigations during the patient's hospitalisation. These statistics are usually based on the principal disease or condition for which the patient was treated (or examined). Data on associated diseases may be compiled in those hospitals which have teaching and research programmes. Data are compiled from the Diagnostic Index Cards which are prepared after the diseases recorded on the medical records, are coded according to the I.C.D. The Diagnostic Data were, so far, presented according to any of the following Lists of I.C.D. (8th Revision):

a. Detailed List
b. List 'D'
c. List 'A'

According to the 9th Revision, which is the latest, the morbidity and Mortality data can be presented as under:

a. Detailed List
b. National List

Data on Operative Procedures are based on the number of Operations performed on the patient. The data can be compiled from the Operation Index Cards which are prepared after each Operative Procedure recorded on medical records of discharged patients, is coded.

As codes on Operative Procedures have not, so far, been published by W.H.O. codes given in the adapted edition of I.C.D. (Vol.II & III) by the U.S. Department of Health, Education and Welfare, can be used for the purpose.

4 RATES MOST FREQUENTLY COMPUTED:

4.1. Mortality Rate: The Death Rate:

This may be computed as gross death rate, net death rate, anaesthesia death rate, post-operative death rate, maternal death rate, infant death rate and neonatal death rate.

Deaths occurring in the emergency room of the hospital or in the ambulance on the way to the hospital, are not included when figuring in the hospital death rates:

4.1.1. Gross Death Rate:

$$\frac{\text{Total No. of deaths for a period X 100}}{\text{Total No. of discharges (including deaths) for the period}}$$

4.1.2. Net Death Rate or Institutional Death Rate:

$$\frac{\text{Total deaths 48 hours or over for a period}}{\text{Total deaths \& discharges for the period}} \times 100$$

4.1.3. Anesthesia Death Rate:

$$\frac{\text{Total No. of anesthesia deaths for a period}}{\text{Total No. of anesthetics administered for the period}} \times 100$$

4.1.4. Post-operative Death Rate:

i.e. deaths attributable to or precipitated by an operation such as deaths from haemorrhage, shock, embolism, infection, post-operative pneumonia etc. and occurring within the convalescence period (i.e. within the first 10 days post-operative)

$$\frac{\text{Total No. of post-operative deaths for a period}}{\text{Total No. of patients operated upon during the period.}} \times 100$$

4.1.5. Maternal Death Rate:

It is considered as one in which a complication of pregnancy, childbirth or of the puerperium was the cause of death. It also classifies deaths resulting from abortions as maternal deaths.

$$\frac{\text{Total No. of deaths of obstetrical patients for a period}}{\text{Total No. of discharges and deaths of obstetrical patients for the period.}} \times 100$$

4.1.6. Infant Death Rate:

$$\frac{\text{Total No. of deaths of infants born in hospital for a period} \times 100}{\text{Total No. of viable newborn infants discharged (including deaths) for the period}}$$

(A VIABLE INFANT is one that has reached a stage of development that enables it to live outside the uterus. This is usually considered as 28 weeks).

4.1.7. Neonatal Death Rate:

$$\frac{\text{Total No. of infant deaths occurring within 28 days of birth for a period} \times 100}{\text{Total No. of viable newborn infants discharged (including deaths) during the period.}}$$

(Because few hospitals have an adequate follow-up system on their patients, it is impossible for the majority of hospitals to figure their actual Neonatal Death Rate)

4.2. Autopsy Rate:

Autopsies on still births, cases dead on arrival and cases released to legal authorities, are not debited against the hospital and are not to be included in figuring the autopsy rate - This is not Autopsy Rate:

$$\frac{\text{No. of autopsies for a period} \times 100}{\text{Total No. of deaths minus unautopsied medico-legal cases}}$$

4.3. Consultation Rate:

This includes only the written consultations received by patients.

Total patients receiving consultations for a period X 100
Total patients discharged (including deaths) for the period.

4.4. Cesarean Section Rate:

Total No. of Cesarean Sections performed for a period X 100
Total No. of viable births for the period

4.5. Average Length of Stay:

In computing the length of stay, the day of admission should be counted but not the day of discharge unless the patient was admitted the same day.

Total No. of Inpatient (Exclusive of new born) days' rendered to discharged patients
Total No. of inpatients (exclusive of newborn) discharged or died.

4.6. Daily Average No. of Patients or the Average Daily Census:

Total No. of inpatients days (exclusive of newborn) care for a period
Total No. of days in the period.

4.7. Bed Turn Over Rate:

Total No. of inpatients (exclusive of Newborns) discharged including deaths
Bed complement

4.8. Percentage of Occupancy:

Daily average No. of inpatients (exclusive of newborn) for a period X 100
Daily average bed complement (exclusive of new born bassinets) for the period.

Note:- The census for the newborn nursery and the number of newborn days' care as well as rates given at S.No.4.5-8 are also worked out in the same manner.

4.9. Daily Average Out-Patient Attendance:

$$\frac{\text{Total No. of out-patient attendances during a period}}{\text{No. of working days during that period}}$$

4.10 Average Out-patient Attendance Per Patient:

This indicates the average duration of a spell of illness treated in out-patient department.

$$\frac{\text{Total No. of Out-patient Attendances}}{\text{Total No. of New Cases.}}$$

APPENDIX - E

REVIEW OF SOME MEDICAL INFORMATION SYSTEMS

A review of some of the successfully developed Medical Information Systems is presented below:

1 Massachusseta General Hospital Computer System (MGH Computer System):

The MGH computer system consists of a compact time-sharing system called MUMPS (MGH Utility Multi-Programming System). This system is dedicated to clinical data management applications. MUMPS provides a high-level interpretive language for application programming, with a flexible input/output monitor that permits conversational dialogue from a variety of terminals, and an extensive data management facility.

Within the MUMPS system, data is characterized as either local or global. Local data files are defined only within the domain of a particular program. Global data files can be referenced by all programs and, therefore, provide a common data base for the system. Each variable in a data file has a symbolic name (e.g. "AGE", "NAME", etc.

and may be given either a numerical value or a variable-length string value. The variable in global data files are organized hierarchically in a branching tree structure. Each node may possess a numerical value, string data value, or a pointer to a lower-level node in the tree. The structure is dynamic; the node content and/or structure may vary during usage. A given element of data is located within a structure by giving the symbolic names of the structure and variable. (Collen, 1974)

In MUMPS language, the structure is represented as a subscripted array where each subscript is a variable name at a different level in the tree and the hierarchical ordering of variables is represented by the order of the subscripts. The organization of data within a tree structure is both topical and chronological. The data hierarchies used in MGH are, in order from highest to lowest levels: patient ID number, major data category (chemistries, review of systems, etc.), data subcategory (name, glucose test, etc.) and date of event etc. Data subtrees are stored in reverse chronologicalorder.

Physically, global data files are stored on a fixed head disk. The records belonging to a new file are not physically contiguous, but their structure and content are

mapped into directories that contain all the data values stored at a given level and all pointers to directories at the next lower level. When a file is updated or a record deleted, the directories are modified to optimize both average access time and space utilization.

2 Hannover Medical System:

The Medical System of Hannover (MSH), is designed to provide a hospital information and data base management system for more than one million patients.The data base of this system contains both on-line and off-line (archival) files. The system uses DL/I for data manipulation (Collen, 1974).

The on-line data files are divided into two categories: files for rapid access, and slower access hierarchical files. The principle rapid access file is the Status Summary File, which contains a brief summary of the status of each patient as well as references to other files. Other rapid access files contain data linkages and administrative data and provide intermediate files for data storage and backup.

There are two on-line hierarchical files CONPAT and APAT. When records from these files are no longer relevant,

they are permanently stored off-line on corresponding archival magnetic tape files, DOKPAT and ARCHPAT.

CONPAT contains a brief information on all the treatment episodes of the patient. The file is organized into nine data categories at three hierarchical levels. The highest level consists of identification, administrative and essential medical data (blood group, medical summaries, etc.) Data categories at the next level include diagnosis, problem treatment, etc. Dated information is stored in reverse chronological order. Only formatted data is stored in CONPAT. Diagnosis are coded in one of the standard coding systems. An additional storage level is provided for data subcategories.

APAT contains detailed information on all the treatment episodes. The file is organized into 16 categories on 4 hierarchical levels. The highest level includes administrative data and a set of summary bits that describe the contents of the remainder of the record. As in CONPAT, data are stored in reverse chronological order by episode. Categories include hospital admissions, medical history, physical findings, laboratory etc. Within a data category, structures with a maximum of eight levels are possible.

3 Medical Information System for Danderyd Stockholm (MIDAS):

This system is designed to serve over 1.5 million patients in the Stockholm area. The system provides a data base for storage and retrieval in real-time from remote terminals throughout the region.

The "main file" (HR) is an on-line direct-access disk file that contains identification, administrative and essential medical data for all patients. Active patient data is stored in the "active patient file" (PR). A record is created in the PR when a patient makes an appointment or is admitted to a medical facility. The PR handles most of the information flow from daily hospital work. The PR record is formed from the HR and from an archival "medical history tape file" (MHR). The MHR is the archival counterpart of the HR and contains medical data of a noncritical nature. When a patient is discharged from the facility, the HR is automatically updated with essential data from the PR, and the remaining data is stored in the MHR. Additional work files stored on disk include a resource file (used by the appointment system), social care and insurance files. The data in each patient record in the HR file is divided into five categories:

(a) personal data (ID number, name, occupation, etc.); (b) critical medical information system (blood-group, critical diagnosis, etc.); (c) information from previous inpatient visits (location, dates, diagnosis, operations, etc.); (d) information from previous x-ray examinations (location, date, type of exam, organ diagnosis); and (e) information from previous outpatient visits (location, date, diagnosis, operations, anaesthesia).

HR is divided into three physical files: personal data file (except name), name file, and medical information file. The personal file is further divided into two parts: patients over 18 years of age and those below that age. The contents of the HR are available for direct-access via remote terminals to authorized users in the region (Collen, 1974).

Medical data are stored in the form of variable-length, variable-format records arranged in a tree-structure (limited to three levels). The highest level contains the patient's record identifiers and data common to all subrecords. The next level may contain upto six subrecords, each of which may include upto six segments. Each segment may contain multiple data elements into the lowest level. Elements are of variable length and format. It may be a binary number, a 6-bit internal code number, or a character

string. Each element is given a symbolic name by which it can be addressed. The name, type of data, lengths, and format of each element are defined in file description table All variables used must have an entry in this table.

Patients are identified to the system by a ten-digit identification number, including a six-digit birth date, a three-digit serial number, and a check digit. A cross-reference file permits patient identification by name, sex, age and other variables. A patient record in the PR contains basic patient data and pointers to addresses that contain date for various hospital functions (laboratory, scheduling, etc.).

File security is provided by requiring a terminal operator to identify himself by a unique password. It is possible in this system to store data which cannot be retrieved by anyone without the password. Psychiatric and veneral data is available only for statistical studies and cannot be retrieved for specific patients.

4 Karolinska Hospital Computer System:

This system performs a variety of clinical, educational, communication and scheduling services. The current system provides a data base for storage and

retrieval of patient records.

The primary file, the "computer medical record library" (CMRL), is stored on magnetic tape and is not available for direct access by remote terminals. All information for a patient is stored in a physically contiguous record. The file is updated periodically from disk files in the various hospital subsystems called "Computer Records" (CR). A patient may have one or more CR's. A portion of each record containing administrative and hospital admission date is kept in direct-access storage on disk. Each CR contains data associated with one registration point. Patients are identified to the system by a ten-digit identification number including a six-digit birth-date, a three-digit serial number and a check-digit.

Data for each patient, in the CMRL, is organized by tape; within type by date and within date by source location. Medical Information system is stored as variable-length, variable-format elements in a branching tree structure. Symbols and coded data are translated via dictionaries resident in the system (Collen, 1974).

In the CR, each record is divided into two parts called the "header segment" and "data segment"

respectively. The header segment is a fixed-length, fixed-format record that contains an identification number and administrative data (location, clinic, ward, date, etc.) and a code for the type of data contained in the data segment: The data segment is a variable length, variable-format record consisting of an arbitrary number of data elements arranged in sequences called "code groups".

A code group can be mapped directly into a tree structure. At each node in the tree a series of data elements are stored, beginning with a positive integer code that identifies the type of data stored on that level and a level label (except for the root node). Data elements stored at lower-level nodes in a subtree are considered to modify the data contained in the node to which they are attached. The identifying code of the root node is called the "major code" of the tree.Following each identifying code in a code group is a sequence of one or more paired data elements called "terms". The first term of a pair is a "label". A label is a negative integer which identifies the type, length and format of the second term, the data value. A data value may be a number, character string, code representing a phrase, date, time, etc.(Collen, 1974).

Data may be retrieved from the CMRL for individual patients or for a population subset. Clinical reports of all

or part of a patient's record can be printed. For research purposes, it is possible to scan the entire CMRL file for a set of conditions and retrieve only those records that satisfy the search criteria. Since the CMRL is stored off-line on magnetic tape, retrievals are not available in real-time and must be scheduled as necessary.

5 Kaiser-Permanente Medical Information System:

This data base provides a continuing integrated medical record for over one million patients in the Kaiser Health Plan. The data base is a computer file of patient identification, administrative and medical data. A patient computer medical record (PCMR) existed for each patient in the Health Plan and for any private patient who attended a Kaiser facility.

The primary file for the data base consists of 1.2 billion bytes of on-line direct-access disk storage. Each PCMR forms a physically contiguous record in the file. When a PCMR is updated with new medical data, the entire PCMR is rewritten on the disk, with the new data inserted in its proper place. This file then contains all medical data and current essential identification and administrative data. Non-essential or obsolete identification and administrative data, which are not needed for real-time

applications, are stored on magnetic tape and constitute an off-line data base in a lower hierarchy than the on-line file.

The PCMR is a logical branching tree structure in which information is represented at each branch point (node) by data elements, and in the hierarchical structure by the relationship between nodal points. Data is stored at the nodes in two different forms: fixed-length, fixed-format indices and variable-length, variable-format definitions and values. Relational data between nodes is implicitly represented by the tree structure itself: data elements within a node or subtree modify the higher level nodes from which they branch. There may be as many as 13 node levels in a PCMR.

The system provides a repository for clinical, identification and administrative data in computer-accessible form for each patient in the health care delivery system. It also provides a source for systematic retrieval of medical data both for individual patients and across the data base population (Collen, 1974).

Three types of data base retrievals are provided. They are directed at clinical, statistical or programming

requirements. First, the system provides clinicians with medical reports for individual patients. The data for these types of reports are derived from individual PCMR's. Second, health services research and epidemiological studies usually required retrieval of several variables from the data base across a defined population. In this case, data were retrieved from hundreds or thousands of PCMR's and presented in a form suitable for statistical analysis. This type of retrieval has provided both medical data for epidemiological studies of the Kaiser Health Plan population and administrative data for utilization studies. Third, the system provides listings of individual PCMR's as they appear in computer storage. This type of output is useful to programmers and systems analysts for interpretation of PCMR discrepencies, error correction and debugging of programs.

APPENDIX - P

QUERY LANGUAGE PREPROCESSOR PROGRAM

```
DURE OPTIONS(MAIN);
0     SOURCE

      DATBAS: PROCEDURE OPTIONS(MAIN);
      /*****************************************************************/
      /* THIS IS FIRST PART OF THE DATA BASE CREATION PROGRAM          */
      /* THIS PROGRAM IS FOR THE CREATION OF INDEXED SEQUENTIAL FILES */
      /* PERSON AND DTADM . IT ALSO CREATES FOUR FILES NMM,FMM,        */
      /* AMM AND WMM WHICH ARE STREAM OUTPUT FILES WHICH WILL BE       */
      /* USED FOR CREATING INVERTED FILES NAM,FNAM,ADDRS AND WBED      */
      /*****************************************************************/
       DCL MORE CHAR(1) ;
       DCL 1 PATIENT EXTERNAL,
             2 MRDNO CHAR(6) ,
             2 TITLE CHAR(5) ,
             2 NAME CHAR(30),
             2 FNAME CHAR(30),
             2 WARDBED CHAR(5),
             2 ADDRESS ,
               3 PRMNT,
                 4 STREET CHAR(37),
                 4 CITY CHAR(15),
                 4 STATE CHAR(15),
                 4 PIN CHAR(6),
               3 LOCAL CHAR(70),
               3 NEXTKIN CHAR(70),
             2 PRVDGN CHAR(20),
             2 DATEADM CHAR(6),
             2 TMADM CHAR(6),
             2 CSTATUS CHAR(20),
             2 RELIGION CHAR(6),
             2 CGHS CHAR(1),
             2 AGE CHAR(4),
             2 SEX CHAR(1),
             2 MONTHLY_INCOME CHAR(5),
             2 PHONENO CHAR(8),
             2 READM CHAR(1),
             2 SPUNIT CHAR(2),
             2 SPECIALITY CHAR(2) ,
             2 USECODE CHAR(1),
             2 MALCHLD CHAR(1),
             2 MALADLT CHAR(1),
             2 PHYSICAL_EXAM CHAR(78),
             2 PRESENT_COMPLAINT CHAR(80) ,
             2 PAST_HISTORY CHAR(100),
             2 FINALDGN CHAR(20),
             2 DISEASE_CODE CHAR(7),
             2 TESTS CHAR(100),
             2 X_RAY CHAR(60),
             2 DELIVERY CHAR(40),
             2 LAB_EXAMS CHAR(60),
             2 OPERATIONS CHAR(70),
             2 OTHER_TREATMENT CHAR(60),
             2 CONDITION CHAR(40),
             2 RESULT CHAR(40),
             2 ADVICE CHAR(40),
             2 DEATHU48H CHAR(1),
             2 DEATHO48H CHAR(1),
             2 AUTOPSY CHAR(60),
             2 CAUSE_OF_DEATH CHAR(15),
             2 FETAL_DEATH CHAR(15),
```

```
URE OPTIONS(MAIN);
    SOURCE

          2 DATE_OF_DISCHARGE CHAR(6),
          2 TOTAL_STAY  CHAR(3),
          2 BILL CHAR(10);
     DCL (ADR,NME,FNME) CHAR(30) EXTERNAL;
     DCL WBD CHAR(5) EXTERNAL;
     DCL (DTA,MRD) CHAR(6) EXTERNAL ;
    DCL PERSON FILE RECORD KEYED ENV('FO=IS,KL=6,RT=F,FL=1280,BT=C');
    DCL DTADM FILE RECORD  KEYED ENV('FO=IS,KL=6,FL=22,RT=F,BT=C');
    DCL (NMM,FMM,AMM) FILE STREAM ENV('FL=36,RT=Z,BT=C');
    DCL WMM FILE STREAM ENV('FL=11,RT=Z,BT=C') ;
    DCL 1 DTARY EXTERNAL ,
          2 DATKEY CHAR(6) INIT('0'),
          2 FSTMRD CHAR(6) INIT(' '),
          2 LSTMRD CHAR(6) INIT(' '),
          2 DACODE CHAR(1) INIT(' '),
          2 TOTALNO FIXED(3) INIT(0);
       OPEN FILE(PERSON) OUTPUT SEQL;
       OPEN FILE(DTADM) OUTPUT SEQL;
       OPEN FILE(NMM) OUTPUT ;
       OPEN FILE(FMM) OUTPUT ;
       OPEN FILE(AMM) OUTPUT ;
       OPEN FILE(WMM) OUTPUT  ;
       WRITE FILE(DTADM) FROM(DTARY) KEYFROM(DATKEY) ;
       CLOSE FILE(DTADM) ;
    AGAIN :
       GET LIST(MRDNO,TITLE,NAME,FNAME,WARDBED,STREET,CITY,STATE,PIN,
       LOCAL,NEXTKIN,PRVDGN,DATEADM,TMADM,CSTATUS,RELIGION,CGHS,AGE,
       SEX,MONTHLY_INCOME,PHONENO,READM,SPUNIT,SPECIALITY,USECODE,
       MALCHLD,MALADLT,PHYSICAL_EXAM,PRESENT_COMPLAINT,PAST_HISTORY,
       FINALDGN,DISEASE_CODE,TESTS,X_RAY,DELIVERY,LAB_EXAMS,OPERATIONS,
       OTHER_TREATMENT,CONDITION,RESULT,ADVICE,DEATHU48H,DEATHO48H,
       AUTOPSY,CAUSE_OF_DEATH,FETAL_DEATH,DATE_OF_DISCHARGE,
       TOTAL_STAY,BILL,MORE) ;
       WRITE FILE(PERSON) FROM(PATIENT) KEYFROM(MRDNO);
       ADR=STREET; WBD=WARDBED ; DTA= DATEADM  ;
       NME=NAME; FNME=FNAME ; MRD= MRDNO ;
       PUT FILE(NMM) EDIT(NME,MRD) (A(30),A(6));
       PUT FILE(FMM) EDIT(FNME,MRD) (A(30),A(6));
       PUT FILE(AMM) EDIT(ADR,MRD) (A(30),A(6));
       PUT FILE(WMM) EDIT(WBD,MRD) (A(5),A(6));
       CALL DADM;
       IF MORE = 'Y' THEN GO TO AGAIN ;
       CLOSE FILE(PERSON),FILE(NMM),FILE(FMM),FILE(WMM),FILE(AMM) ;
    /*************************************************/
    /* TO INSERT DATEADM WITH MRDNO                  */
    /*************************************************/
    DADM:PROCEDURE;
      OPEN FILE(DTADM) DIRECT UPDATE;
      ON KEY(DTADM) GO TO ABM;
      DATKEY=DTA;
      READ FILE(DTADM) INTO(DTARY) KEY(DTA);
      TOTALNO=TOTALNO+1;
      LSTMRD=MRD;
      REWRITE FILE(DTADM) FROM(DTARY) KEY(DTA);
     GO TO LMN;
    ABM: DACODE='U';
      TOTALNO=TOTALNO+1;
      FSTMRD=MRD;
```

```
DURE OPTIONS(MAIN);
1     SOURCE

        WRITE FILE(DTADM) FROM(DTARY) KEYFROM(DTA);
       LMN: CLOSE FILE(DTADM);
       END DADM;
       END DATBAS;
```

```
RE OPTIONS(MAIN);
OC /ATTRIBUTE AND REFERENCE LIST
     LABEL,     CONSTANT
     STRUCTURE,MEMBER,  (PATIENT )
     CHARACTER (     30), STATIC,    UNALIGNED
     MEMBER    (PATIENT ),CHARACTER (     40), UNALIGNED
     LABEL,     CONSTANT
     MEMBER    (PATIENT ),CHARACTER (      4), UNALIGNED
     FILE,      ENV,       STREAM
     MEMBER    (PATIENT ),CHARACTER (     60), UNALIGNED
     MEMBER    (PATIENT ),CHARACTER (     10), UNALIGNED
     MEMBER    (PATIENT ),CHARACTER (     15), UNALIGNED
     MEMBER    (PATIENT ),CHARACTER (      1), UNALIGNED
     MEMBER    (PATIENT . ADDRESS . PRMNT    ),CHARACTER (     15), UNALIGNED
     MEMBER    (PATIENT ),CHARACTER (     40), UNALIGNED
     MEMBER    (PATIENT ),CHARACTER (     20), UNALIGNED
     MEMBER    (DTARY   ),CHARACTER (      1), INITIAL,  UNALIGNED
     ENTRY,     CONSTANT
     ENTRY,     CONSTANT, OPTIONS(MAIN)
     MEMBER    (PATIENT ),CHARACTER (      6), UNALIGNED
     MEMBER    (PATIENT ),CHARACTER (      6), UNALIGNED
     MEMBER    (DTARY   ),CHARACTER (      6), INITIAL,  UNALIGNED
     MEMBER    (PATIENT ),CHARACTER (      1), UNALIGNED
     MEMBER    (PATIENT ),CHARACTER (      1), UNALIGNED
     MEMBER    (PATIENT ),CHARACTER (     40), UNALIGNED
     MEMBER    (PATIENT ),CHARACTER (      7), UNALIGNED
     CHARACTER (      6), STATIC,    UNALIGNED
     FILE,      ENV,       KEYED,     RECORD
     STRUCTURE,STATIC
     MEMBER    (PATIENT ),CHARACTER (     15), UNALIGNED
     MEMBER    (PATIENT ),CHARACTER (     20), UNALIGNED
     FILE,      ENV,       STREAM
     MEMBER    (PATIENT ),CHARACTER (     30), UNALIGNED
     CHARACTER (     30), STATIC,    UNALIGNED
     MEMBER    (DTARY   ),CHARACTER (      6), INITIAL,  UNALIGNED
     MEMBER    (PATIENT ),CHARACTER (     60), UNALIGNED
     LABEL,     CONSTANT
     MEMBER    (PATIENT . ADDRESS ),CHARACTER (     70), UNALIGNED
     MEMBER    (DTARY   ),CHARACTER (      6), INITIAL,  UNALIGNED
     MEMBER    (PATIENT ),CHARACTER (      1), UNALIGNED
     MEMBER    (PATIENT ),CHARACTER (      1), UNALIGNED
     MEMBER    (PATIENT ),CHARACTER (      5), UNALIGNED
     CHARACTER (      1), AUTOMATIC,UNALIGNED
     CHARACTER (      6), STATIC,    UNALIGNED
     MEMBER    (PATIENT ),CHARACTER (      6), UNALIGNED
     MEMBER    (PATIENT ),CHARACTER (     30), UNALIGNED
     MEMBER    (PATIENT . ADDRESS ),CHARACTER (     70), UNALIGNED
     CHARACTER (     30), STATIC,    UNALIGNED
     FILE,      ENV,       STREAM
     MEMBER    (PATIENT ),CHARACTER (     70), UNALIGNED
     MEMBER    (PATIENT ),CHARACTER (     60), UNALIGNED
```

```
URE OPTIONS(MAIN) ;
     SOURCE

      BWFAM : PROCEDURE OPTIONS(MAIN) ;
      /*****************************************************************/
      /* THIS ROUTINE IS SECOND PART OF DATA BASE CREATION .INPUT TO   - */
      /* THIS ROUTINE ARE THE SORTED FILES  SWMM,SNMM,SFMM AND SAMM     */
      /* WHICH WILL BE USED FOR CREATING INVERTED INDEXED SEQUENTIAL    */
      /* FILES FOR WARDBED,NAME,FNAME AND ADDRESS RESPECTIVELY          */
      /* WHICH WILL BE INPUT TO THE MAIN PROGRAM.                       */
      /*****************************************************************/
      DCL NAM FILE RECORD KEYED ENV('FO=IS,KL=32,FL=39,RT=F');
      DCL FNAM FILE RECORD KEYED ENV('FO=IS,KL=32,FL=39,RT=F');
      DCL ADDRS FILE RECORD KEYED ENV('FO=IS,KL=32,FL=39,RT=F');
      DCL WBED FILE RECORD KEYED ENV('FO=IS,KL=5,FL=12,RT=F');
      DCL NO CHAR(44) INIT(' COULD NOT BE INSERTED DUE TO SO MANY SYNONYMS');
      DCL SWMM FILE STREAM ENV('FL=11,RT=Z,BT=C');
      DCL SNMM FILE STREAM ENV('FL=36,RT=Z,BT=C');
      DCL SFMM FILE STREAM ENV('FL=36,RT=Z,BT=C');
      DCL SAMM FILE STREAM ENV('FL=36,RT=Z,BT=C');
      DCL BHAT FILE STREAM OUTPUT ;
      DCL (NME,FNME,ADR) CHAR(30) EXTERNAL ;
      DCL MRD CHAR(6) EXTERNAL ;
      DCL WBD CHAR(5) EXTERNAL ;
      DCL IJ FIXED(2) INIT(0) EXTERNAL ;
      DCL CIJ CHAR(2) EXTERNAL ;
      DCL 1 NMARY EXTERNAL ,
            2 NAMKEY CHAR(32),
            2 MRDNO1 CHAR(6),
            2 NCODE CHAR(1);
      DCL 1 FNMARY EXTERNAL ,
            2 FNMKEY CHAR(32),
            2 MRDNO2 CHAR(6),
            2 FCODE CHAR(1);
      DCL 1 WBARY EXTERNAL ,
            2 WBKEY CHAR(5),
            2 MRDNO3 CHAR(6),
            2 WCODE CHAR(1);
      DCL 1 ADRAR EXTERNAL ,
            2 ADDKEY CHAR(32),
            2 MRDNO4 CHAR(6),
            2 ACODE CHAR(1);
      DCL TTT FIXED(1) INIT(0) ;
      ON ENDFILE(SWMM) TTT=1 ;
         OPEN FILE(SWMM) INPUT ;
         OPEN FILE(SNMM) INPUT ;
         OPEN FILE(SFMM) INPUT ;
         OPEN FILE(SAMM) INPUT ;
         OPEN FILE(WBED) DIRECT UPDATE;
         OPEN FILE(NAM) OUTPUT SEQL;
         OPEN FILE(FNAM) OUTPUT SEQL;
         OPEN FILE(ADDRS) OUTPUT SEQL;
      AGN :
         GET FILE(SNMM) EDIT(NME,MRD) (A(30),A(6)) ;
         CALL NMCRT(NME,NAM);
         GET FILE(SFMM) EDIT(FNME,MRD) (A(30),A(6)) ;
         CALL NMCRT(FNME,FNAM);
         GET FILE(SAMM) EDIT(ADR,MRD) (A(30),A(6)) ;
         CALL NMCRT(ADR,ADDRS);
         GET FILE(SWMM) EDIT(WBD,MRD) (A(5),A(6)) ;
         CALL WBPR;
```

```
URE OPTIONS(MAIN) ;
     SOURCE

      IF TTT ^= 1 THEN GO TO AGN ;
         CLOSE FILE(ADDRS),FILE(NAM),FILE(FNAM),FILE(WBED);
      NMCRT:PROCEDURE(NKY,NM):
      /*****************************************************************/
      /* THIS SUBROUTINE IS USED FOR INSERTING NAME WITH MRDNO, FNAME  */
      /* WITH MRDNO AND ADDRESS WITH MRDNO. THIS IS USED TO CREAT 3    */
      /* INVERTED INDEXED SEQUENTIAL FILES  VIZ. NAM,FNAM AND ADDRS    */
      /*****************************************************************/
       DCL NM FILE  ;
       DCL NKY CHAR(30);
       DCL YNK CHAR(32) ;
      DCL 1 NR,
            2 NKEY CHAR(32),
            2 MRDN CHAR(6),
            2 MCD CHAR(1);
      IJ=1;
      ON KEY(NM) GO TO CHKKEY;
      ACTION: MRDN=MRD;
        CIJ = CHAR(IJ) ;
         MCD='U';
        NKEY=NKY!!CIJ;
         YNK=NKEY ;
      WRITE FILE(NM) FROM(NR) KEYFROM(YNK);
      GO TO KHTAM;
      CHKKEY: IJ=IJ+1;
      IF IJ<=99 THEN GO TO ACTION ;
      ELSE PUT SKIP EDIT ('THE GIVEN DATA',NKY,NO)
         (COLUMN(20),A,X(5),A);
      KHTAM : END NMCRT ;
      /*****************************************************************/
      /* TO INSERT WARDBED WITH MRDNO                                  */
      /*****************************************************************/
      WBPR: PROCEDURE;
      ON KEY(WBED) GO TO ACTB;
       MRDNO3=MRD;
         WCODE='U';
      WBKEY=WBD;
         READ FILE(WBED) INTO(WBARY) KEY(WBKEY) ;
         IF WCODE='D'!WCODE='T' THEN DO ;
         REWRITE FILE(WBED) FROM(WBARY) KEY(WBKEY);
         GO TO KTMA;
         END;
         IF WCODE='U' THEN GO TO CTB;
      ACTB: WRITE FILE(WBED) FROM(WBARY) KEYFROM(WBKEY);
      GO TO KTMA;
       CTB: PUT SKIP(3) EDIT('WARDBED MUST BE GIVEN UNIQUE') (COLUMN(20),A);
      KTMA:
      END WBPR;
      END BWFAM ;



URE OPTIONS(MAIN) ;
ROC /ATTRIBUTE AND REFERENCE LIST
     MEMBER   (ADRAR   ),CHARACTER (     1), UNALIGNED
     LABEL,    CONSTANT
     LABEL,    CONSTANT
     MEMBER   (ADRAR   ),CHARACTER (    32), UNALIGNED
```

```
DURE OPTIONS(MAIN);
O     SOURCE

      SYNTAX: PROCEDURE OPTIONS(MAIN);
      /*****************************************************************/
      /* THIS IS MAIN ROUTINE OF OUR DATA BASE. IT CONTAINS 24    */
      /* SUBROUTINES WHICH ARE CALLED WHENEVER THEIR JOB IS REQUIRED. */
      /* THIS ROUTINE MAY BE VIEWED AS CONSISTING OF TWO SUBPARTS :-*/
      /* 1. SYNTAX ANALYSER , 2.SEMANTIC ANALYSER . QUERY IS PROCESSED*/
      /* BY THE SEMANTIC ANAALYSER IF IT IS FOUND TO BE SYNTACTICALLY */
      /* CORRECT.ALL THE FOUR OPERATIONS VIZ. GET,INSERT,DELETE AND */
      /* MODIFY CAN BE PROCESSED WITH THIS PROGRAM.THE NOTES FOR  */
      /* SUBROGRAMS ARE GIVEN IN THE THEIR BEGINNING.          */
      /*****************************************************************/
      DCL DOMAIN(8) CHAR(10) INIT('DATE      ','MRDNO     ','WARDBED   ',
          'NAME      ','FNAME     ','DATEADM   ','ADDRESS   ','SECCODE');
      DCL TARGET(30) CHAR(10) INIT('MRDNO     ','NAME      ','ADDRESS   ',
          'FNAME     ','DATEADM   ','WARDBED   ','LADDRESS  ','NKADDRESS ',
          'PHYSICEXAM','PCOMPLAINT','PSTHISTORY','FDIAGNOSIS','TESTS     ',
          'CONDITION ','DELIVERY  ','LABEXAM   ','OPERATION ',
          'OTHRTRTMNT','XRAY      ','ADVICE    ','RESULT    ',
          'FULLINFO  ','DIAGNOSTIC','TREATMENT ','ALLREPORT ','BSTATUS   ',
          'DSTATUS   ','WSTATUS   ','MSTATUS   ','ASTATUS   ') :
      DCL KEYWRD(7) CHAR(10) INIT('IF        ','GET       ','AND       ',
          'MODIFY    ','DELETE    ','INSERT    ','$         '):
      DCL LFN FILE INPUT RECORD SEQUENTIAL ENV('FO=SQ,FL=80,RT=Z,BT=C'):
      DCL BHT FILE STREAM OUTPUT  ENV('FL=132,RT=Z,BT=C');
      DCL PERSON FILE RECORD KEYED ENV('FO=IS,KL=6,RT=F,FL=1280');
      DCL  NAM FILE RECORD  KEYED ENV('FO=IS,KL=32,FL=39,RT=F');
      DCL FNAM FILE RECORD  KEYED ENV('FO=IS,KL=32,FL=39,RT=F');
      DCL WBED FILE RECORD  KEYED ENV('FO=IS,KL=5,FL=12,RT=F');
      DCL ADDRS FILE RECORD  KEYED ENV('FO=IS,KL=32,FL=39,RT=F');
      DCL DTADM FILE RECORD  KEYED ENV('FO=IS,KL=6,FL=22,RT=F');
      DCL TRT FILE STREAM INPUT ;
      DCL DGN FILE STREAM INPUT ;
       DCL DLT FILE STREAM INPUT ;
      DCL INF FILE STREAM INPUT ;
         OPEN FILE(PERSON) DIRECT UPDATE;
         OPEN FILE(DTADM) DIRECT UPDATE;
         OPEN FILE(WBED) DIRECT UPDATE;
         OPEN FILE(NAM) DIRECT UPDATE;
         OPEN FILE(FNAM) DIRECT UPDATE;
         OPEN FILE(ADDRS) DIRECT UPDATE;
      DCL (DCODE(8),TCODE(30),KCODE(7),FLAG,FLAG1) FIXED(1) INIT(0) ;
       DCL MDMN CHAR(100) ;
       DCL (MRD,DTA,SCODE) CHAR(6) ;
       DCL (NME,FNME,ADR) CHAR(30) ;
       DCL (MARK,SIGN) FIXED(1) ;
      DCL (L,M,N) FIXED(2) INIT(0) ;
      DCL (FLG,FLGA,FLAG2,FLAG3) FIXED(1) INIT(0)  ;
      DCL (LM,FM) FIXED(6)  ;
      DCL (DA(99),DAB(99),DKM(99),D(999)) CHAR(6)  ;
      DCL (NK,NKM,NKS,NKB,NKH) FIXED(2) INIT(0)  ;
      DCL WBD CHAR(5)  ;
      DCL BATIS CHAR(32)  ;
      DCL IJ FIXED(2) INIT(0)  ;
      DCL CIJ CHAR(2)  ;
      DCL (FMRD,LMRD) CHAR(6)  ;
      DCL DASHES CHAR(120) INIT((120)'*');
      DCL 1 CARDREC CHAR(80);
      DCL CARD(80) DEFINED CARDREC CHAR(1);
```

```
DURE OPTIONS(MAIN);
)     SOURCE

        DCL KARD DEFINED CARDREC CHAR(80);
       DCL SLTH CHAR(40) INIT('LENGTH OF SYMBOL EXCEEDS 10 CHARACTERS');
       DCL SPACES CHAR(100) INIT(' ');
       DCL EQUAL CHAR(1) INIT('=');
       DCL QUOTE CHAR(1) INIT('''');
       DCL BLANK CHAR(1) INIT(' ');
       DCL SYMBOL CHAR(10) INIT(' ');
       DCL DBUFF(30) CHAR(100) INIT(' ') ;
       DCL (SL,CTR,SCTR) FIXED(3) INIT(0);
       DCL STARS CHAR(120) INIT((120)'*');
       DCL ERR CHAR(40) INIT(' ERROR DETECTED BEFORE  COLUMN NUMBER  ');
        DCL BLNK CHAR(120) INIT((120)' ');
        DCL 1 PATIENT ,
              2 MRDNO CHAR(6),
              2 TITLE CHAR(5),
              2 NAME CHAR(30),
              2 FNAME CHAR(30),
              2 WARDBED CHAR(5),
              2 ADDRESS ,
                3 PRMNT,
                  4 STREET CHAR(37),
                  4 CITY CHAR(15),
                  4 STATE CHAR(15),
                  4 PIN CHAR(6),
                3 LOCAL CHAR(70),
                3 NEXTKIN CHAR(70),
             2 PRVDGN CHAR(20),
             2 DATEADM CHAR(6),
             2 TMADM CHAR(6),
             2 CSTATUS CHAR(20),
             2 RELIGION CHAR(6),
             2 CGHS CHAR(1),
             2 AGE CHAR(4),
             2 SEX CHAR(1),
             2 MONTHLY_INCOME CHAR(5),
             2 PHONENO CHAR(8),
             2 READM CHAR(1),
             2 SPUNIT CHAR(2),
             2 SPECIALITY CHAR(2) ,
             2 USECODE CHAR(1),
             2 MALCHLD CHAR(1),
             2 MALADLT CHAR(1),
             2 PHYSICAL_EXAM CHAR(78),
             2 PRESENT_COMPLAINT CHAR(80) ,
             2 PAST_HISTORY CHAR(100),
             2 FINALDGN CHAR(20),
             2 DISEASE_CODE CHAR(7),
             2 TESTS CHAR(100),
             2 X_RAY CHAR(60),
             2 DELIVERY CHAR(40),
             2 LAB_EXAMS CHAR(60),
             2 OPERATIONS CHAR(70),
             2 OTHER_TREATMENT CHAR(60),
             2 CONDITION CHAR(40),
             2 RESULT CHAR(40),
             2 ADVICE CHAR(40),
             2 DEATHU48H CHAR(1),
             2 DEATHO48H CHAR(1),
```

```
JRE OPTIONS(MAIN);
     SOURCE

          2 AUTOPSY CHAR(60),
          2 CAUSE_OF_DEATH CHAR(15),
          2 FETAL_DEATH CHAR(15),
          2 DATE_OF_DISCHARGE CHAR(6),
          2 TOTAL_STAY  CHAR(3),
          2 BILL CHAR(10);
    DCL (PAST,TEST) CHAR(100)  ;
    DCL (DELV,COND,RSLT,ADV) CHAR(40)  ;
    DCL (XRAY,LAB,OT,OTOPSY) CHAR(40)  ;
    DCL PEXAM CHAR(78)  ;
    DCL PCOMP CHAR(80)  ;
    DCL OPRTN CHAR(70)  ;
    DCL (CDEATH,FDEATH) CHAR(15)  ;
    DCL (DU48H,DO48H) CHAR(1)  ;
    DCL BIL CHAR(10)  ;
    DCL DCD CHAR(7)  ;
    DCL TSTAY CHAR(3)  ;
    DCL FDGN CHAR(20)  ;
    DCL DTDIS CHAR(6)  ;
    DCL 1 NMARY  ,
          2 NAMKEY CHAR(32),
          2 MRDNO1 CHAR(6),
          2 NCODE CHAR(1);
    DCL 1 FNMARY  ,
          2 FNMKEY CHAR(32),
          2 MRDNO2 CHAR(6),
          2 FCODE CHAR(1);
    DCL 1 WBARY  ,
          2 WBKEY CHAR(5),
          2 MRDNO3 CHAR(6),
          2 WCODE CHAR(1);
    DCL 1 ADRAR  ,
          2 ADDKEY CHAR(32),
          2 MRDNO4 CHAR(6),
          2 ACODE CHAR(1);
    DCL 1 DTARY  ,
          2 DATKEY CHAR(6),
          2 FSTMRD CHAR(6),
          2 LSTMRD CHAR(6),
          2 DACODE CHAR(1),
          2 TOTALNO FIXED(3) INIT(0);
    START1: OPEN FILE (LFN);
       ON ENDFILE(LFN) GO TO FINISH ;
    START:  PUT FILE(BHT) SKIP EDIT (STARS) (A(120));
        OPEN FILE(BHT) ;
        CALL NXTCRD ;
        CTR=0;
        SL=0;
       NK=0; NKH=0; NKB=0; NKS=0; NKM=0;
       DO I=1 TO 99 ;
       DA(I) =BLANK ;
       DAB(I) =BLANK ;
       DKM(I) =BLANK ;
       END ;
       DO I=1 TO 999;
       D(I) =BLANK ;
       END ;
       MRD=BLANK; WBD=BLANK; SCODE=BLANK ; DTA=BLANK ; DTE=BLANK;
```

```
JRE OPTIONS(MAIN);
     SOURCE

          FLAG1=0;
          FLAG=0;
          MARK=0;
          L=0;
          DO I=1 TO 30;
          DBUFF(I)=SPACES;
          TCODE(I)=0;
          END;
          DO I=1 TO 8;
          DCODE(I)=0;
          END;
          DO I=4 TO 6;
          KCODE(I)=0;
          END;
          CALL SYMBL;
          IF SYMBOL^= KEYWRD(1) THEN CALL ERROR ;
          CALL SYMBL;
     DMN : DO I=1 TO 8 ;
          IF SYMBOL= DOMAIN(I)  THEN DO;
          IF DCODE(I)=1 THEN CALL ERROR;
          ELSE DCODE(I)=1 ;
          J=I;
          GO TO CHEQL;
          END;
          END ;
          CALL ERROR ;
     CHEQL : CALL SYMBL ;
          I=J ;
          IF SYMBOL ^= EQUAL THEN CALL ERROR;
          CALL SYMBL;
          IF SYMBOL^= QUOTE THEN CALL ERROR ;
     LDATA : CTR=CTR+1;
          IF CTR>80 THEN CALL NXTCRD ;
          ELSE ;
          IF CARD(CTR)^= QUOTE THEN DO;
          SL=SL+1;
          IF SL>= 100 THEN CALL ERROR;
          SUBSTR(DBUFF(I),SL,1)= CARD(CTR) ;
          GO TO LDATA;
          END;
          IF KCODE(4)=1 THEN DO ;
          MDMN=DBUFF(I);
          CALL SYMBL ;
          GO TO CHDLT ;
          END ;
          IF DCODE(2)=1 & DCODE(8)=1 THEN GO TO LAST ;
          IF DCODE(8)=1 THEN GO TO CHAND ;
          IF FLAG=1 & (DCODE(1)=1!DCODE(2)=1!DCODE(3)=1) THEN CALL ERROR ;
          IF DCODE(1)=1 ! DCODE(2)=1 ! DCODE(3)=1 THEN GO TO CHGET ;
     CHAND : CALL SYMBL ;
          IF SYMBOL  = KEYWRD(3) THEN DO ;
          IF FLAG =1 THEN CALL ERROR ;
          ELSE FLAG=1;
          CALL SYMBL ;
          GO TO DMN ;
          END ;
          ELSE GO TO CHGT ;
     CHGET: CALL SYMBL;
```

```
E OPTIONS(MAIN);
   SOURCE

    CHGT : IF DCODE(8)=1 THEN GO TO LASTA ;
       IF SYMBOL^= KEYWRD(2) THEN CALL ERROR ;
        CALL SYMBL ;
        IF DCODE(1)=1 THEN GO TO CH2630;
      /* TO CHECK TARGET EDIT */
    CH2225: DO I=22 TO 25 ;
        IF SYMBOL = TARGET(I) THEN DO ;
        IF TCODE(I)=1 THEN CALL ERROR ;
        ELSE TCODE(I)=1;
      MARK=1;
        CALL SYMBL ;
        GO TO CHDLT;
        END;
        END;
    CH18 : DO I=1 TO 21;
    TGT : IF SYMBOL=TARGET(I) THEN DO;
        IF TCODE(I)= 1 THEN CALL ERROR ;
        ELSE TCODE(I)= 1 ;
        CALL SYMBL ;
        IF SYMBOL = KEYWRD(3) THEN DO ;
        IF FLAG1=0 THEN L=I;
        IF FLAG1=1 THEN M=I;
        IF FLAG1=2 THEN N=I;
        FLAG1 = FLAG1 + 1 ;
        IF FLAG1>2 THEN CALL ERROR ;
        ELSE DO ;
        CALL SYMBL ;
        GO TO CH18;
        END ;
        END ;
        ELSE GO TO CHDLT ;
        END;
        END ;
        CALL ERROR ;
    CH2630: DO I=26 TO 30;
        IF SYMBOL=TARGET(I) THEN DO;
        IF TCODE(I)=1 THEN CALL ERROR ;
        ELSE TCODE(I)=1;
        CALL SYMBL ;
        GO TO CHDLT;
        END ;
        END;
        CALL ERROR ;
    CHDLT : IF SYMBOL=KEYWRD(7) THEN DO :
       /*PUT FILE(BHT) SKIP EDIT ('QUERY IS O.K.') (X(20),A); */
       /* IF DCODE(1)=1 THEN CALL STATUS ; */
       IF DCODE(2)=1 THEN CALL GETRTN ;
        GO TO START ;
        MRD=DBUFF(2) ; WBD=DBUFF(3) ; DTA=DBUFF(6) ; ADR=DBUFF(4) ;
        NME=DBUFF(4) ; FNME=DBUFF(5) ; DTE=DBUFF(1) ;SCODE=DBUFF(8);
        IF SCODE='BHATIA'!SCODE='NARAIN'!SCODE='SAXENA' THEN DO ;
       IF KCODE(4)=1 & TCODE(6) THEN CALL  WBN ;
        IF KCODE(4)=1 THEN CALL MDFRTN ;
        IF KCODE(5)=1 THEN CALL DLTRTN ;
       IF TCODE(23)=1 ! TCODE(24)=1 THEN CALL INSDT ;
        IF KCODE(6)=1 THEN CALL INSRTN ;
        ELSE CALL ERROR ;
        END ;
```

```
IRE OPTIONS(MAIN);
    SOURCE

        ELSE PUT FILE(BHT)  SKIP EDIT (' THE SECURITY CODE',SCODE,
        'IS NOT A VALID ONE ')  (COL(20),A,A,A);
        GO TO START;
        END ;
        ELSE CALL ERROR ;
    LAST : CALL SYMBL;
    LASTA: IF DCODE(2)=1 & DCODE(8)=1 THEN GO TO CHMDA ;
        IF SYMBOL=KEYWRD(5) THEN DO;
    /* TO CHECK DELETE */
        KCODE(5)=1;
        CALL SYMBL ;
        GO TO CHDLT ;
        END ;
        ELSE IF SYMBOL = KEYWRD(6) THEN DO ;
        KCODE(6)=1 ;
        GO TO CHIN ;
        END ;
        ELSE CALL ERROR ;
    CHIN : CALL SYMBL ;
        DO I=22 TO 24 ;
        IF SYMBOL = TARGET(I) THEN DO ;
        TCODE(I)=1;
        CALL SYMBL ;
        GO TO CHDLT;
        END ;
        END ;
        CALL ERROR ;
    CHMD : CALL SYMBL ;
    CHMDA : IF SYMBOL=TARGET(6) THEN DO ;
        TCODE(6)=1;
        GO TO CHEQL ;
        END ;
        DO I=7 TO 21 ;
        IF SYMBOL=TARGET(I) THEN DO ;
        NKB=I-6;
        J=I;
        GO TO CHEQL;
        END;
      END ;
      CALL ERROR ;
    SYMBL : PROCEDURE ;
         SCTR=0;
        SYMBOL=BLNK10 ;
        CTR=CTR+1;
        IF CTR >80 THEN CALL NXTCRD ;
    LOOP : IF CARD(CTR)= BLANK THEN DO ;
        IF SCTR^=0 THEN GO TO RETN ;
                                CTR=CTR+1;
         IF CTR>80 THEN CALL NXTCRD ;
         ELSE GO TO LOOP;
                                END;
        IF CARD(CTR)>='A' & CARD(CTR)<='I' ! CARD(CTR)>='J' &
        CARD(CTR)<='R' ! CARD(CTR)>='S' & CARD(CTR)<='Z' THEN DO :
        SCTR=SCTR+1;
        IF SCTR >10 THEN DO;
        PUT FILE(BHT) SKIP EDIT(SLTH) (COL(20),A);
         CALL ERROR ;
        END;
```

```
RE OPTIONS(MAIN);
    SOURCE

        SUBSTR(SYMBOL,SCTR,1)= CARD(CTR);
        CTR=CTR+1;
        IF CTR>80 THEN CALL NXTCRD ;
        GO TO LOOP ;
        END ;
    /* HERE WE FIND SPECIAL CHR */
        IF SCTR^=0 THEN GO TO RETN ;
        SUBSTR(SYMBOL,1,1)= CARD(CTR);
        GO TO OUT ;
    RETN : CTR=CTR+1;
    OUT : END ;
    ERROR :PROCEDURE ;
        PUT FILE(BHT) SKIP EDIT (ERR,CTR) (X(20),A,F(3,0));
        GO TO START ;
        END;
    NXTCRD : PROCEDURE ;
        READ FILE (LFN) INTO (CARDREC);
       PUT FILE(BHT) SKIP EDIT(KARD) (COL(20),A);
        CTR = 1 ;
        IF CARD(1) ='X' THEN GO TO FINISH ;
        END ;
    FINISH: PUT FILE(BHT) SKIP EDIT('ALL QUERIES ARE ANSWERED')
      (COL(20),A);
        CLOSE FILE (LFN),FILE(BHT),FILE(PERSON),FILE(DTADM),
        FILE(WBED),FILE(NAM),FILE(FNAM),FILE(ADDRS) ;
        STOP ;
    INSRTN: PROCEDURE ;
    /*****************************************************/
    /* THIS PART INSERTS THE REQUIRED DATA IN THE FILES  */
    /* PERSON,DTADM,NAM,FNAM,WBED AND DTADM . PERSON IS  */
    /* THE MAIN FILE ALL THE REST BEING INVERTED FILES   */
    /*****************************************************/
        ON ENDFILE(INF) GO TO START ;
    AG : GET FILE(INF)  LIST(MRDNO,TITLE,NAME,FNAME,WARDBED,STREET,
        CITY ,STATE,
    PIN,LOCAL,NEXTKIN,PRVDGN,DATEADM,TMADM,CSTATUS,RELIGION,CGHS,
    AGE,SEX,MONTHLY_INCOME,PHONENO,READM,SPUNIT,SPECIALITY,USECODE,
    MALCHLD,MALADLT,PHYSICAL_EXAM,PRESENT_COMPLAINT,PAST_HISTORY,
    FINALDGN,DISEASE_CODE,TESTS,X_RAY,DELIVERY,LAB_EXAMS,OPERATIONS,
    OTHER_TREATMENT,CONDITION,RESULT,ADVICE,DEATHU48H,DEATHO48H,
    AUTOPSY,CAUSE_OF_DEATH,FETAL_DEATH,DATE_OF_DISCHARGE,
    TOTAL_STAY,BILL) ;
       ADR=STREET;
       NME=NAME;
       FNME=FNAME;
       WBD=WARDBED;
       DTA=DATEADM;
       MRD=MRDNO;
       WRITE FILE(PERSON) FROM(PATIENT) KEYFROM(MRD);
       CALL NMCRT(NME,NAM);
       CALL NMCRT(FNME,FNAM);
       CALL NMCRT(ADR,ADDRS);
       CALL WBPR;
       CALL DADM;
       GO TO AG ;
        END INSRTN ;
    /*****************************************************/
```

```
JRE OPTIONS(MAIN);
     SOURCE

      /* FILE PERSON WHICH IS THE MAIN FILE . AS THESE RECORDS        */
      /* ALREADY EXIST THIS WILL BE SORT OF UPDATION                  */
      /***************************************************************/
      INSDT : PROCEDURE ;
       ON KEY(PERSON) GO TO MSGC ;
       ON ENDFILE(TRT) GO TO START ;
       ON ENDFILE(DGN) GO TO START ;
       IF TCODE(23)=1 THEN GO TO DINS ;
       AGA : GET FILE(DGN) LIST(MRD,PEXAM,PCOMP,PAST,FDGN,DCD) ;
      READ FILE(PERSON) INTO(PATIENT) KEY(MRD) ;
      PHYSICAL_EXAM =PEXAM ;
      PRESENT_COMPLAINT=PCOMP;
      PAST_HISTORY=PAST ;
      FINALDGN=FDGN ;
      DISEASE_CODE=DCD ;
      REWRITE FILE(PERSON) FROM(PATIENT) KEY(MRD) ;
      GO TO AGA ;
      DINS : GET FILE(TRT) LIST(MRD,TEST,XRAY,DELV,LAB,OPRTN,
      OT,COND,RSLT,ADV) ;
      READ FILE(PERSON) INTO(PATIENT) KEY(MRD) ;
      TESTS= TEST ;
      X_RAY = XRAY ;
      DELIVERY= DELV ;
      LAB_EXAMS = LAB ;
      OPERATIONS= OPRTN ;
      OTHER_TREATMENT= OT ;
      CONDITION = COND ;
      RESULT = RSLT ;
      ADVICE= ADV ;
      REWRITE FILE(PERSON) FROM(PATIENT) KEY(MRD) ;
      GO TO DINS ;
      MSGC : PUT FILE(BHT) SKIP EDIT('PATIENT WITH MRDNO=',
      MRD,'DOES NOT EXIST IN THE SYSTEM ')  (COL(20),A,A,A);
      END INSDT ;
      /***************************************************************/
      /* NOW THE SUBROUTINES START WHICH WILL PREPARE INVERTED FILES  */
      /***************************************************************/
      NMCRT:PROCEDURE(NKY,NM);
      DCL NM FILE ;
       DCL NKY CHAR(32);
      DCL 1 NR,
            2 NKEY CHAR(30),
            2 MRDN CHAR(6),
            2 MCD CHAR(1);
      DCL NI CHAR(25) INIT('COULD NOT BE INSERTED');
      DCL NIN CHAR(25) INIT('DUE TO SO MANY SYNONYMS') ;
      IJ=1;
      CIJ= CHAR(IJ) ;
      ON KEY(NM) GO TO CHKKEY;
      ACTION: MRDN=MRD;
         MCD='U';
        NKEY=NKY!!CIJ;
         BATIS = NKEY ;
      WRITE FILE(NM) FROM(NR) KEYFROM(BATIS);
      GO TO KHTAM;
      CHKKEY: IJ=IJ+1 ; CIJ=CHAR(IJ) ;
      IF IJ<=99 THEN GO TO REPEAT;
      ELSE PUT FILE(BHT)  SKIP EDIT ('THE GIVEN DATA',NKY,NI,NIN)
```

```
RE OPTIONS(MAIN);
    SOURCE

      (COL(20),A,A,A,A) ;
    GO TO KHTAM;
      REPEAT: NKEY=NKY!!CIJ;
    GO TO ACTION;
     KHTAM:
    END NMCRT;
    /*****************************************************************/
    /* TO INSERT WARDBED WITH MRDNO                                  */
    /*****************************************************************/
    WBPR: PROCEDURE;
    DCL WDD CHAR(40) INIT('WARDBED MUST BE GIVEN UNIQUE');
    ON KEY(WBED) GO TO ACTB;
     MRDNO3=MRD;
      WCODE='U';
    WBKEY=WBD;
      READ FILE(WBED) INTO(WBARY) KEY(WBD) ;
      IF WCODE='D'!WCODE='T' THEN DO ;
      REWRITE FILE(WBED) FROM(WBARY) KEY(WBD);
      GO TO KTMA;
      END;
      IF WCODE='U' THEN GO TO CTB;
    ACTB: WRITE FILE(WBED) FROM(WBARY) KEYFROM(WBD);
    GO TO KTMA;
    CTB: PUT FILE(BHT) SKIP EDIT(WDD)   (COL(20),A);
     FLAG2=1 ; /* TO SIGNAL ERROR IN INSERTION FOR FILE WBN  */
    KTMA:
    END WBPR;
    /*****************************************************************/
    /* TO INSERT DATEADM WITH MRDNO                                  */
    /*****************************************************************/
    DADM:PROCEDURE;
      ON KEY(DTADM) GO TO ABM;
      DATKEY=DTA;
      READ FILE(DTADM) INTO(DTARY) KEY(DTA);
      TOTALNO=TOTALNO+1;
      LSTMRD=MRD;
      REWRITE FILE(DTADM) FROM(DTARY) KEY(DTA);
     GO TO LMN;
    ABM: DACODE='U';
      TOTALNO=1;
      FSTMRD=MRD;
     WRITE FILE(DTADM) FROM(DTARY) KEYFROM(DTA);
    LMN :
    END DADM;
     /*****************************************************************/
    /* THE FOLLOWINGS ARE USED FOR SEARCHING RECORD FOR KEYS  */
    /*****************************************************************/
    WRTN : PROCEDURE ;
    DCL PWN CHAR(50) INIT('THE PATIENT CORRESPONDING TO THIS WARDBED');
     ON KEY(WBED) GO TO MSGE ;
     IF WCODE ='D' THEN GO TO DW ;
    IF WCODE='T' THEN GO TO DSW ;
    MRD = MRDNO3 ;
    GO TO CLSE ;
    DW:PUT FILE(BHT) SKIP EDIT(PWN,WBD,'IS ALREADY DISCHARGED')
      (COL(20),A,A,A);
    GO TO CLSE ;
    DSW : PUT FILE(BHT)  SKIP EDIT('PATIENT TRANSFERRED ')  (COL(20),A);
```

```
RE OPTIONS(MAIN);
    SOURCE

     PUT FILE(BHT)  SKIP EDIT('PLEASE TRY WITH MRDNO =',MRDNO3)
       (COL(20),A,A);
     GO TO CLSE ;
    MSGE : PUT FILE(BHT)  SKIP EDIT(PWN,
       WBD, 'DOES NOT EXIST IN THE SYSTEM ')  (COL(20),A,A,A);
    CLSE :
    END WRTN ;
    /*****************************************************************/
    /* TO SEARCH ON DATE OF ADDMISSION , THAT IS DATEADM             */
    /*****************************************************************/
    DRTN : PROCEDURE ;
    DCL RCD CHAR(45) INIT('RECORD CORRESPONDING TO DATEADM=') ;
     ON KEY(DTADM) GO TO MSGF ;
     READ FILE(DTADM) INTO(DTARY) KEY(DTA) ;
     IF DACODE ='D' THEN GO TO DDA ;
     FMRD = FSTMRD ;
     LMRD = LSTMRD ;
       LM=LMRD ; FM=FMRD ;
     IF FLAG=0 THEN DO ;
     INK= LM - FM +1 ;
     DO I= 1 TO INK ;
     N=  INK=1 ;
     LL=FM+N ;
     D(I)=CHAR(LL) ;
     END ;
     CALL DGT(INK,D) ;
     END ;
     GO TO CLSF ;
    DDA : PUT FILE(BHT)  SKIP EDIT(RCD,DTA,'IS ALREADY DELETED')
       (COL(20),A,A,A);
     GO TO CLSF ;
    MSGF : PUT FILE(BHT)  SKIP EDIT(RCD,DTA,'COULD NOT BE FOUND')
       (COL(20),A,A,A) ;
    CLSF : CLOSE FILE(DTADM) ;
    END DRTN ;
    /*****************************************************/
    /* THIS PROCEDURE IS FOR FINDING THE MRDNO RELATING  */
    /* TO NAME ,FNAME AND ADDRESS FOR QUERIES WITH GET   */
    /*****************************************************/
    GFAN : PROCEDURE(FL,KYA) ;
     DCL KYA CHAR(30) ;
    DCL FL FILE ;
     DCL 1 STR ,
           2 KY CHAR(32) ,
           2 MRDNX CHAR(6) ,
           2 MXCD CHAR(1) ;
     ON KEY(FL) GO TO MSGG ;
     IF FLG=1 THEN GO TO MR ;
    AXZ : READ FILE(FL) INTO(STR) KEY(KY) ;
      FLG = 1 ;
     IF MXCD ='U' THEN DO ;
     D(IJ) = MRDNX ;
     NK = NK + 1 ;
     IJ = IJ + 1 ;
     KY = KYA !! CIJ ;
     IF IJ <= 99 THEN GO TO AXZ ;
     END ;
```

```
RE OPTIONS(MAIN);
    SOURCE

      KY = KYA !! CIJ ;
      GO TO AXZ ;
     MR : NKH = NK ;
      DO I= 1 TO NK ;
      DA(NK) = D(NK) ;
      END ;
      IJ = 1 ;
      KY = KYA !! CIJ  ;
      GO TO AXZ ;
       MSGG : IF FLG = 1 THEN GO TO CLSH ;
       PUT FILE(BHT)  SKIP EDIT('RECORD CORRESPONDING TO ',KYA,
        'COULD NOT BE FOUND ') (COL(20),A,A,A);
      CLSH : CLOSE FILE(FL) ;
     END GFAN ;
     /*****************************************************************/
     /* THE PROCEDURE 'KOMNA'IS USED TO FIND THE COMMAN     */
     /* MRDNO FOR DATEADM WITH NAME OR FNAME OR ADDRESS       */
     /*****************************************************************/
     KOMNA : PROCEDURE ;
      NKM=0 ;
      DO I = 1 TO NK ;
      IF D(NK)>=FMRD & D(NK)<=LMRD THEN DO ;
      NKM = NKM + 1 ;
      DKM(NKM) = D(NK) ;
      END ;
      ELSE ;
      END ;
      END KOMNA ;
     /*****************************************************************/
     /* THE PROCEDURE KOMANI IS USED FOR FINDING THE COMMAN    */
     /* MRDNO FROM NAME ,FNAME AND ADDRESS TAKING TWO AT A     */
     /* TIME AND MRDNO IS THE MAIN KEY FOR FILE PERSON         */
     /*****************************************************************/
     KOMNI : PROCEDURE ;
      NKS=0 ;
      DO I = 1 TO NK ;
      DO I = 1 TO NKH ;
      IF DA(NKH) = D(NK) THEN DO ;
      NKS = NKS + 1 ;
      DAB(NKS) = D(NK) ;
      END ;
      ELSE ;
      END ;
      END ;
      END KOMNI ;
     /*****************************************************************/
     /* THE FOLLOWING PROCEDURE WHEN CALLED WILL PRINT          */
     /* ONE OF THE 21 ATTRIBUTES ,                             */
     /*****************************************************************/
     PRINT : PROCEDURE(S);
         DCL SHYAM(21) LABEL ;
         DCL S FIXED(2) ;
         GO TO SHYAM(S) ;
     SHYAM(1): PUT FILE(BHT)  SKIP EDIT('MRDNO=',MRDNO) (COL(20),A,A);
        GO TO AKHIR ;
     SHYAM(2): PUT FILE(BHT)  SKIP EDIT('NAME=',TITLE,NAME)
      (COL(20),A,A,A);          GO TO AKHIR ;
      SHYAM(3): PUT FILE(BHT) SKIP EDIT('PERMANENT ADDRESS='
```

```
RE OPTIONS(MAIN);
   SOURCE

     STREET,CITY,STATE,PIN) (COL(20),A,A,A,A,A); GO TO AKHIR;
   SHYAM(4): PUT FILE(BHT)  SKIP EDIT('FNAME=',FNAME)
      (COL(20),A,A); GO TO AKHIR ;
   SHYAM(5): PUT FILE(BHT)  SKIP EDIT('DATEADM=',DATEADM)
      (COL(20),A,A) ; GO TO AKHIR ;
   SHYAM(6): PUT FILE(BHT)  SKIP EDIT('WARDBED=',WARDBED)
      (COL(20),A,A); GO TO AKHIR ;
    SHYAM(7):PUT FILE(BHT) SKIP EDIT('LOCAL_ADDRESS=',LOCAL)
      (COL(20),A,A);  GO TO AKHIR ;
      SHYAM(8):PUT FILE(BHT) SKIP EDIT('NEXT_OF_KIN_ADDRESS=',
      NEXTKIN)  (COL(20),A,A) ; GO TO AKHIR ;
   SHYAM(9): PUT FILE(BHT) SKIP EDIT('PHYSICAL_EXAM=',
     PHYSICAL_EXAM) (COL(20),A,A) ; GO TO AKHIR ;
    SHYAM(10): PUT FILE(BHT) SKIP EDIT('PRESENT_COMPLAINT=',
     PRESENT_COMPLAINT) (COL(20),A,A); GO TO AKHIR ;
   SHYAM(11): PUT FILE(BHT) SKIP EDIT('PAST_HISTORY=',PAST_HISTORY)
      (COL(20),A,A) ; GO TO AKHIR ;
   SHYAM(12): PUT FILE(BHT) SKIP EDIT('FINAL_DIAGNOSIS=',FINALDGN)
      (COL(20),A,A) ; GO TO AKHIR ;
   SHYAM(13): PUT FILE(BHT)  SKIP EDIT('TESTS=',TESTS)
      (COL(20),A,A); GO TO AKHIR ;
   SHYAM(14): PUT FILE(BHT)  SKIP EDIT('CONDITION=',CONDITION)
      (COL(20),A,A); GO TO AKHIR ;
   SHYAM(15): PUT FILE(BHT) SKIP EDIT('DELIVERY=',DELIVERY)
       (COL(20),A,A) ; GO TO AKHIR ;
   SHYAM(16): PUT FILE(BHT) SKIP EDIT('LAB_EXAMS=',LAB_EXAMS)
      (COL(20),A,A); GO TO AKHIR ;
   SHYAM(17): PUT FILE(BHT)  SKIP EDIT('OPERATIONS=',OPERATIONS)
      (COL(20),A,A) ; GO TO AKHIR ;
   SHYAM(18): PUT FILE(BHT) SKIP EDIT('OTHER_TREATMENT=',
    OTHET_TREATMENT) (COL(20),A,A) ; GO TO AKHIR ;
   SHYAM(19): PUT FILE(BHT) SKIP EDIT('X_RAY=',X_RAY)
      (COL(20),A,A) ; GO TO AKHIR ;
   SHYAM(20): PUT FILE(BHT)  SKIP EDIT('ADVICE=',ADVICE)
      (COL(20),A,A) ; GO TO AKHIR ;
   SHYAM(21): PUT FILE(BHT)  SKIP EDIT('RESULT =',RESULT)
      (COL(20),A,A) ;
   AKHIR : END PRINT ;
   /*****************************************************/
   /* THE FOLLOWING PRINTS ONE ATTRIBUTE AT A TIME      */
   /*****************************************************/
   PRINT1: PROCEDURE(G) ;
     DCL G FIXED(2) ;
     PUT FILE(BHT)  SKIP EDIT('THE REQUIRED INFORMATION FOLLOWS')
       (COL(20),A);
     PUT FILE(BHT)  SKIP EDIT(DASHES) (A) ;
     CALL PRINT(G) ;
      PUT FILE(BHT)  SKIP EDIT (DASHES)  (A) ;
   END PRINT1 ;
   /*****************************************************/
   /* THE FOLLOWING PRINTS TWO ATTRIBUTES AT A TIME     */
   /*****************************************************/
   PRINT2 : PROCEDURE(M,N);
       DCL (M,N) FIXED(2) ;
       PUT FILE(BHT)  SKIP EDIT('THE REQUIRED INFORMATION FOLLOWS')
      (COL(20),A);
       PUT FILE(BHT)  SKIP EDIT(DASHES) (A) ;
       CALL PRINT(M);
```

```
E OPTIONS(MAIN);
   SOURCE

      CALL PRINT(N);
      PUT FILE(BHT)  SKIP EDIT(DASHES) (A) ;
   END PRINT2 ;
   /*************************************************************/
   /* THE FOLLOWING PRINTS THREE ATTRIBUTES AT A TIME           */
   /*************************************************************/
   PRINT3 : PROCEDURE(L,M,N);
      DCL (L,M,N) FIXED(2) ;
      PUT FILE(BHT)  SKIP EDIT('THE REQUIRED INFORMATION FOLLOWS')
     (COL(20),A) ;
      PUT FILE(BHT)  SKIP EDIT(DASHES) (A) ;
      CALL PRINT(L);
      CALL PRINT(M);
      CALL PRINT(N);
      PUT FILE(BHT)  SKIP EDIT(DASHES) (A) ;
   END PRINT3 ;
   /*************************************************************/
   /* THE FULRTN IS FOR WRITING FULLINFO OR DIAGNOSTIC OR       */
   /* TREATMENT OR ALL REPORT                                   */
   /*************************************************************/
   FULRTN : PROCEDURE ;
      IF TCODE(22)=1 THEN DO ;
      PUT FILE(BHT)  SKIP EDIT ('THE IDENTIFICATION INFORMATION  FOLLOWS')
      (COL(20),A);
       PUT FILE(BHT)  SKIP EDIT(DASHES) (A) ;
      GO TO BAD ;
      END ;
      IF TCODE(23)=1 THEN DO ;
      PUT FILE(BHT)  SKIP EDIT('THE DIAGNOSTIC INFORMATION FOLLOWS')
      (COL(20),A) ;
       PUT FILE(BHT)  SKIP EDIT(DASHES) (A) ;
      GO TO BAD ;
      END ;
      IF TCODE(24)=1 THEN DO ;
      PUT FILE(BHT)  SKIP EDIT('THE TREATMENT INFORMATION FOLLOWS')
     (COL(20),A) ;
       PUT FILE(BHT)  SKIP EDIT(DASHES) (A) ;
      GO TO BAD ;
      END ;
      IF TCODE(25)=1 THEN DO ;
      PUT FILE(BHT)  SKIP EDIT('THE ALL REPORT FOR PATIENT FOLLOWS')
    (COL(20),A) ;
    PUT FILE(BHT) SKIP EDIT(DASHES) (A) ;
    GO TO BAD ;
    END ;
    ELSE :
   BAD : PUT FILE(BHT) SKIP EDIT('MRDNO=',MRDNO) (COL(20),A,A);
      PUT FILE(BHT) SKIP EDIT('NAME=',TITLE,NAME)
      (COL(20),A,A,A);
      PUT FILE(BHT) SKIP EDIT('FNAME =',FNAME) (COL(20),A,A);
      PUT FILE(BHT) SKIP EDIT('WARDBED =',WARDBED) (COL(20),A,A);
   PUT FILE(BHT)SKIP EDIT('PROVISIONAL_DIAGNOSIS=',PRVDGN)
     (COL(20),A,A);
    PUT FILE(BHT) SKIP EDIT('DATEADM=',DATEADM) (COL(20),A,A);
      PUT FILE(BHT) SKIP EDIT('AGE=',AGE) (COL(20),A,A);
      PUT FILE(BHT) SKIP EDIT('SEX=',SEX) (COL(20),A,A);
      PUT FILE(BHT) SKIP EDIT('READMISSION=',READM) (COL(20),A,A);
     IF TCODE(22)=1 THEN GO TO FINF ;
```

```
OPTIONS(MAIN);
 SOURCE

    IF TCODE(23)=1 THEN GO TO DINF ;
    IF TCODE(24)=1 THEN GO TO TINF ;
  FINF: PUT FILE(BHT) SKIP EDIT('PERMANENT_ADDRESS=',
   STREET,CITY,STATE,PIN)        (COL(20),A,A,A,A,A);
   PUT FILE(BHT) SKIP EDIT('LOCAL_ADDRESS=',LOCAL) (COL(20),A,A);
  PUT FILE(BHT) SKIP EDIT('NEXT_OF_KIN_ADDRESS=',NEXTKIN)
    (COL(20),A,A);
  PUT FILE(BHT) SKIP EDIT('TIME_OF_ADMISSION=',TMADM)
    (COL(20),A,A);
  PUT FILE(BHT) SKIP EDIT('CIVIL_STATUS=',CSTATUS)
     (COL(20),A,A);
  PUT FILE(BHT) SKIP EDIT('RELIGION=',RELIGION) (COL(20),A,A);
  PUT FILE(BHT) SKIP EDIT('CGHS=',CGHS)  (COL(20),A,A);
  PUT FILE(BHT) SKIP EDIT('SPECIALITY=',SPECIALITY) (COL(20),A,A);
  PUT FILE(BHT) SKIP EDIT('SPUNIT=',SPUNIT)  (COL(20),A,A);
  PUT FILE(BHT) SKIP EDIT('MALE_CHILD=',MALCHLD) (COL(20),A,A);
  PUT FILE(BHT) SKIP EDIT('MALE_ADULT',MALADLT)(COL(20),A,A);
  PUT FILE(BHT) SKIP EDIT('MONTHLY_INCOME',MONTHLY_INCOME)
    (COL(20),A,A);
  PUT FILE(BHT) SKIP EDIT('PHONE_NUMBER=',PHONENO)(COL(20),A,A);
    IF TCODE(22)=1 THEN GO TO ANTIM  ;
  DINF : PUT FILE(BHT) SKIP EDIT('PHYSICAL_EXAM=',PHYSICAL_EXAM)
    (COL(20),A,A);
  PUT FILE(BHT)SKIP EDIT('PRESENT_COMPLAINT=',PRESENT_COMPLAINT)
    (COL(20),A,A);
  PUT FILE(BHT) SKIP EDIT('PAST_HISTORY=',PAST_HISTORY)
     (COL(20),A,A);
  PUT FILE(BHT) SKIP EDIT('FINAL_DIAGNOSIS=',FINALDGN)
    (COL(20),A,A);
  PUT FILE(BHT) SKIP EDIT('DISEASE_CODE=',DISEASE_CODE)
     (COL(20),A,A);
  PUT FILE(BHT) SKIP EDIT('TESTS=',TESTS)  (COL(20),A,A);
    IF TCODE(23)=1 THEN GO TO ANTIM ;
  TINF : PUT FILE(BHT) SKIP EDIT('X_RAY=',X_RAY) (COL(20),A,A);
   PUT FILE(BHT) SKIP EDIT('DELIVERY',DELIVERY) (COL(20),A,A);
  PUT FILE(BHT) SKIP EDIT('LAB_EXAMS=',LAB_EXAMS) (COL(20),A,A);
  PUT FILE(BHT) SKIP EDIT('OPERATIONS=',OPERATIONS) (COL(20),A,A);
  PUT FILE(BHT) SKIP EDIT('OTHER_TREATMENT=',OTHER_TREATMENT)
     (COL(20),A,A);
  PUT FILE(BHT) SKIP EDIT('CONDITION=',CONDITION) (COL(20),A,A);
  PUT FILE(BHT) SKIP EDIT('RESULT=',RESULT) (COL(20),A,A);
  PUT FILE(BHT) SKIP EDIT('ADVICE=',ADVICE)  (COL(20),A,A);
  IF DEATHU48H^=BLANK ! DEATH048H=BLANK THEN DO  ;
  PUT FILE(BHT) SKIP EDIT('DEATH_UNDER_48_HOURS=',DEATHU48H)
    (COL(20),A,A);
  PUT FILE(BHT) SKIP EDIT('DEATH_OVER_48_HOURS=',DEATH048H)
    (COL(20),A,A);
  PUT FILE(BHT) SKIP EDIT('AUTOPSY=',AUTOPSY) (COL(20),A,A);
  PUT FILE(BHT) SKIP EDIT('CAUSE_OF_DEATH=',CAUSE_OF_DEATH)
    (COL(20),A,A);
  PUT FILE(BHT) SKIP EDIT('FETAL_DEATH=',FETAL_DEATH)
    (COL(20),A,A);
   END;
   ELSE;
  PUT FILE(BHT) SKIP EDIT('DATE_OF_DISCHARGE=',DATE_OF_DISCHARGE)
    (COL(20),A,A);
  PUT FILE(BHT) SKIP EDIT('BILL=',BILL) (COL(20),A,A);
  PUT FILE(BHT) SKIP EDIT('TOTAL_STAY=',TOTAL_STAY)
```

```
RE OPTIONS(MAIN);
    SOURCE

      (COL(20),A,A);
     ANTIM : PUT FILE(BHT)  SKIP EDIT(DASHES)    (A) ;
      GO TO START ;
        END FULRTN ;
     GETRTN : PROCEDURE ;
      IF FLAG=0 THEN DO ;
     IF DCODE(2)=1 THEN DO ;  CALL PRNT(MRD) ;
      GO TO START ; END ;
     IF DCODE(3)=1 THEN DO ;
     CALL WRTN ;
     CALL PRNT(MRD) ;
     GO TO START ;
      END ;
      IF DCODE(4)=1 THEN CALL GFAN(NAM,NME) ;
      IF DCODE(5)=1 THEN CALL GFAN(FNAM,FNME);
      IF DCODE(7)=1 THEN CALL GFAN(ADDRS,ADR);
      IF DCODE(6)=1 THEN CALL DRTN ;
      ELSE ;
      CALL DGT(NK,D) ;
      END ;
      IF FLAG=1 THEN DO ;
      IF DCODE(4)=1 & DCODE(5)=1 THEN DO ;
      CALL GFAN(NAM,NME);
      CALL GFAN(FNAM,FNME);
      CALL KOMNI ;
      CALL DGT(NKS,DAB);
      END ;
      IF DCODE(5)=1 & DCODE(7)=1 THEN DO ;
      CALL GFAN(FNAM,FNME);
      CALL GFAN(ADDRS,ADR);
      CALL KOMNI ;
      CALL DGT(NKS,DAB) ;
      END ;
      IF DCODE(4)=1 & DCODE(7)=1 THEN DO ;
      CALL GFAN(NAM,NME) ;
      CALL GFAN(ADDRS,ADR) ;
      CALL KOMNI ;
      CALL DGT(NKS,DAB) ;
      END ;
       IF DCODE(4)=1 & DCODE(6)=1 THEN DO ;
       CALL GFAN(NAM,NME);
       CALL DRTN ;
       CALL KOMNA ;
       CALL DGT(NKM,DKM) ;
       END ;
       IF DCODE(5)=1 & DCODE(6)=1 THEN DO ;
       CALL GFAN(FNAM,FNME);
       CALL DRTN ;
       CALL KOMNA ;
       CALL DGT(NKM,DKM);
       END ;
       IF DCODE(7)=1 & DCODE(6)=1 THEN DO ;
       CALL GFAN(ADDRS,ADR);
       CALL DRTN ;
       CALL KOMNA;
       CALL DGT(NKM,DKM) ;
       END;
       ELSE CALL ERROR ;
```

```
URE OPTIONS(MAIN);
     SOURCE

        END GETRTN ;
      /*****************************************************/
      /* BELOW IS PROCEDURE DGT WHICH WILL COLLECT         */
      /* REQUIRED MRDNOS                                   */
      /*****************************************************/
      DGT : PROCEDURE(G,DBA) ;
      DCL G FIXED(3) ;
      DCL DBA(999) CHAR(6)  ;
      DO I=1 TO G ;
      SIGN=1 ;
      MRD=DBA(G) ;
      IF I=G THEN SIGN=0 ;
      CALL PRNT(MRD) ;
      END ;
      GO TO START ;
      END DGT ;
      /*****************************************************/
      /* BELOW IS PROCEDURE PRNT WHICH PRINTS THE REQUIRED */
      /* INFORMATION FROM THE FILE PERSON                  */
      /*****************************************************/
      PRNT : PROCEDURE(MD) ;
      DCL MD CHAR(6)  ;
      ON KEY(PERSON)  GO TO NMSG ;
      READ FILE(PERSON) INTO(PATIENT) KEY(MD) ;
      IF USECODE='D'THEN GO TO MNDL ;
      IF USECODE='U'THEN DO ;
      IF FLAG1=0 THEN CALL PRINT1(L);
      IF FLAG1=1 THEN CALL PRINT2(L,M) ;
      IF FLAG1=2 THEN CALL PRINT3(L,M,N) ;
      IF MARK=1 THEN CALL FULRTN ;
      ELSE ;
      END ;
      MNDL : IF SIGN=0 THEN PUT FILE(BHT)  SKIP EDIT(
      'THE RECORD CORRESPONDING TO MRDNO',MD,
      'IS ALREADY DELETED') (COL(20),A,A,A);
      GO TO FLBN ;
      NMSG : IF SIGN=0 THEN PUT FILE(BHT)  SKIP EDIT(
      'THE RECORD CORRESPONDING TO MRDNO=',MD,
      'DOES NOT EXIST IN THE SYSTEM') (COL(20),A,A,A);
      FLBN:
      END PRNT ;
      /*****************************************************************/
      /* THIS PART MODIFIES THE EXISTING TUPLES                        */
      /*****************************************************************/
      MDFRTN: PROCEDURE ;
        DCL DEVI(15) LABEL ;
          ON KEY(PERSON) GO TO MSG;
         READ FILE(PERSON) INTO(PATIENT) KEY(MRD):
          IF USECODE='D'THEN GO TO MSGX;
          GO TO DEVI(NKB);
       /* BRANCH WILL BE MADE TO CORRESPONDING LABEL BY ABOVE STATEMENT */
      DEVI(1): PATIENT.LOCAL=MDMN; GO TO MEND ;
      DEVI(2): PATIENT.NEXTKIN=MDMN; GO TO MEND ;
      DEVI(3): PHYSICAL_EXAM=MDMN; GO TO MEND ;
      DEVI(4): PRESENT_COMPLAINT=MDMN; GO TO MEND ;
      DEVI(5): PAST_HISTORY=MDMN; GO TO MEND ;
      DEVI(6): FINALDGN=MDMN; GO TO MEND;
      DEVI(7): TESTS=MDMN; GO TO MEND;
```

```
OPTIONS(MAIN);
 SOURCE

  DEVI(8): CONDITION=MDMN; GO TO MEND ;
  DEVI(9): DELIVERY=MDMN; GO TO MEND ;
  DEVI(10): LAB_EXAMS=MDMN; GO TO MEND;
  DEVI(11): OPERATIONS=MDMN; GO TO MEND ;
  DEVI(12): OTHER_TREATMENT=MDMN;  GO TO MEND ;
  DEVI(13): X_RAY=MDMN; GO TO MEND ;
  DEVI(14): ADVICE=MDMN; GO TO MEND ;
  DEVI(15): RESULT=MDMN;
  MEND : REWRITE FILE(PERSON) FROM(PATIENT) KEY(MRD);
  PUT FILE(BHT) SKIP EDIT('SUCCESSFUL UPDATION ') (COL(20),A) ;
      GO TO CLS ;
  MSGX: PUT FILE(BHT) SKIP EDIT('THE PATIENT ALREADY DELETED')
   (COL(20),A) ;
     GO TO CLS ;
  MSG: PUT FILE(BHT)  SKIP EDIT('THE PATIENT WITH THIS MRDNO=',MRD,
      'DOES NOT EXIST IN THE SYSTEM') (COL(20),A,A,A) ;
  CLS: GO TO START ;
   END MDFRTN ;
  /*****************************************************/
  /* THE FOLLOWING PROCEDURE IS MEANT FOR UPDATION OF  */
  /* KEY ATTRIBUTE , THAT IS WARDBED                   */
  /*****************************************************/
  WBN : PROCEDURE ;
   DCL WBAD CHAR(5)  ;
   ON KEY(PERSON) GO TO MSGH ;
   ON KEY(WBED) GO TO MSGI ;
   READ FILE(PERSON) INTO(PATIENT) KEY(MRD) ;
   IF USECODE='D'THEN GO TO MSDL ;
   WBAD=WARDBED ; /* TO PUT IN FILE WBED,CODE 'T' FOR TRANSFERRED */
   WARDBED=MDMN ;
   WBD= MDMN ; /* FOR WRITING IN NEW WARDBED */
   REWRITE FILE(PERSON) FROM(PATIENT) KEY(MRD) ;
   READ FILE(WBED) INTO(WBARY) KEY(WBAD) ;
   WCODE='D';
   REWRITE FILE(WBED) FROM(WBARY) KEY(WBAD) ;
   CALL WBPR ;
   /*THIS WILL WRITE NEW RECORD */
  MSGH : PUT FILE(BHT)  SKIP EDIT
    ('NO RECORD EXISTS CORRESPONDING TO MRDNO=',
     MRD) (COL(20),A,A);
   GO TO CLSJ ;
  MSGI : PUT FILE(BHT) SKIP EDIT
   ('RECORD CORRESPONDING TO PREVIOUS WARDBED=',
     WBAD , 'COULD NOT BE FOUND ') (COL(20),A,A,A);
   GO TO CLSJ ;
  MSDL : PUT FILE(BHT) SKIP EDIT('RECORD WITH MRDNO=',MRD,
     'IS ALREADY DELETED ') (COL(20),A,A,A);
  CLSJ :
  END WBN ;
  /*****************************************************/
  /* THIS ROUTINE DELETES THE PATIENTS 'RECORDS        */
  /*****************************************************/
  DLTRTN: PROCEDURE;
    ON KEY(PERSON) GO TO MSGA;
  ON ENDFILE(DLT) GO TO START ;
  ABB :
  GET FILE(DLT) LIST(MRD,DU48H,DO48H,OTOPSY,CDEATH,
  FDEATH,DTDIS,TSTAY,BIL) ;
```

```
JRE OPTIONS(MAIN);
     SOURCE

      READ FILE(PERSON) INTO(PATIENT) KEY(MRD);
      IF USECODE='D'THEN DO;
      PUT FILE(BHT)  SKIP EDIT('THE RECORD WITH MRDNO=',MRD,
      'IS ALREADY DELETED') (COL(20),A,A,A);
      GO TO ABCD; /* IT WILL BRANCH TO THE END OF ROUTINE */
                          END;
      ELSE USECODE='D'; ADR=LOCAL ; NME=NAME ; FNME=FNAME ;
           WBD=WARDBED ; DTA=DATEADM ;
     /*************************************************************/
     /* READ ALL THE FILES AND INSERT 'D'CODE INTO THEM           */
     /*************************************************************/
     NCODE='D'; FCODE='D'; ACODE='D';
    DEATHU48H= DU48H ;
    DEATHO48H= DO48H ;
    AUTOPSY=OTOPSY ;
    CAUSE_OF_DEATH=CDEATH ;
    FETAL_DEATH= FDEATH ;
    DATE_OF_DISCHARGE= DTDIS ;
    TOTAL_STAY=TSTAY ;
    BILL=BIL ;
    REWRITE FILE(PERSON) FROM(PATIENT) KEY(MRD) ;
    GO TO ABB ;
     ON KEY(WBED) GO TO MSGB ; /* MESSAGE FOR NOT FOUND */
     READ FILE(WBED) INTO(WBARY) KEY(WBD) ;
     IF WCODE='U'! WCODE='T'THEN DO ;
     WCODE='D';
     REWRITE FILE(WBED) FROM(WBARY) KEY(WBD) ;
     GO TO AB ;
     END ;
     ELSE DO ;
     PUT FILE(BHT)  SKIP EDIT('THE PATIENT WITH WARDBED=',WBD,
       'IS ALREADY DELETED') (COL(20),A,A,A);
     GO TO AB ;
     END ;
    MSGB: PUT FILE(BHT) SKIP EDIT('NO RECORD FOUND WITH WARDBED=',WBD)
      (COL(20),A,A,A);
    AB : ON KEY(DTADM) GO TO MSGC ;
    READ FILE(DTADM) INTO(DTARY) KEY(DTA) ;
    IF DACODE='U'THEN DO ;
    TOTALNO=TOTALNO-1 ;
    IF TOTALNO=0 THEN DO ;
    DACODE='D';
    REWRITE FILE(DTADM) FROM(DTARY) KEY(DTA) ;
    GO TO ABC ;
    END ;
    ELSE REWRITE FILE(DTADM) FROM(DTARY) KEY(DTA) ;
    END ;
    PUT FILE(BHT) SKIP EDIT('THERE IS NO PATIENT CORRESPONDING TO DATE='
     ,DTA) (COL(20),A,A);
    GO TO ABC ;
    MSGC:PUT FILE(BHT) SKIP EDIT('THE RECORD NOT FOUND WITH DATE=',DTA)
      (COL(20),A,A);
    ABC: CALL FAN(NME,NAM);
         CALL FAN(FNME,FNAM) ;
         CALL FAN(ADR,ADDRS) ;
    MSGA : PUT FILE(BHT)  SKIP EDIT('THE RECORD WITH MRDNO=',MRD,
           'DOES NOT EXIST IN THE SYSTEM') (COL(20),A,A,A);
    /*************************************************************/
```

```
JRE OPTIONS(MAIN);
     SOURCE

      /* THE PROCEDURE GIVEN BELOW IS A PART OF DLTRTN ROUTINE          */
      /* FAN IS A PROCEDURE FOR PUT FILE(BHT) TING  'D'CODE IN FILES FNAM       */
      /* ADDAR AND NAM RELATING TO FNAME,ADDRESS AND NAME               */
      /*****************************************************************/
      FAN : PROCEDURE(NKYA,NMA) ;
        DCL 1 NRA ,
              2 NKEYA CHAR(32),
              2 MRDNA CHAR(6),
              2 MCDA CHAR(1);
        DCL NKYA CHAR(30);
      DCL NMA FILE ;
       ON KEY(NMA) GO TO REPIT ;
       OPEN FILE(NMA) ;
       IJ=1 ;
       CIJ=CHAR(IJ) ;
      AGNA : NKEYA=NKYA !! CIJ ;
         BATIS=NKEYA ;
       READ FILE(NMA) INTO(NRA) KEY(BATIS) ;
       IF MCDA='D'THEN GO TO DLTX ;
       IF MRDNA= MRD THEN GO TO BADAL ;
       ELSE IJ=IJ+1 ;
        CIJ=CHAR(IJ) ;
       GO TO AGNA ;
      BADAL : MCDA= 'D' ;
       REWRITE FILE(NMA) FROM(NRA) KEY(BATIS) ;
       GO TO XYZ ;
      REPIT : PUT FILE(BHT)  SKIP EDIT('THE RECORD WITH ',NKYA,
         'DOES NOT EXIST IN THE SYSTEM')  (COL(20),A,A,A);
      DLTX : PUT FILE(BHT)  SKIP EDIT('THE RECORD WITH',NKYA,
          'IS A DELETED ONE ') (COL(20),A,A,A);
      XYZ : CLOSE FILE(NMA) ;
      END FAN ;
      ABCD : END DLTRTN :
        END SYNTAX ;
```

```
IF NAME = 'SURENDER KUMAR BOSE' GET WARDBED AND CONDITION AND ADDRESS $
THE REQUIRED INFORMATION FOLLOWS

WARDBED= 15032
CONDITION= IMPROVING
PERMANENT_ADDRESS= 124, D.D..A. FLAT MUNIRKA , NEW DELHI _110 067

**********************************************************************
IF WARDBED='12095' GET NAME AND ADDRESS $
THE REQUIRED INFORMATION FOLLOWS

NAME= MISS KUSAM LATA JAIN
PERMANENT_ADDRESS= A-2/120, JANAKPURI, NEW DELHI- 110058

**********************************************************************
IF MRDNO= '654845' GET NAME AND WARDBED AND CONDITION $
THE REQUIRED INFORMATION FOLLOWS

NAME= DR.   ALLAH BUX AHMED
WARDBED= 02038
CONDITION= NOT SO SERIOUS

**********************************************************************
IF NAME= 'RAJ PAL SINGH' AND DATEADM ='810224' GET WARDBED AND FINALDGN $
THE REQUIRED INFORMATION FOLLOWS

WARDBED= 12059
FINAL_DIAGNOSIS= MALARIA

**********************************************************************
IF MRDNO='652426' GET TESTS AND ADVICE $
THE REQUIRED INFORMATION FOLLOWS

TESTS= STOOL,SPIT,,BLOOD AND URINE
ADVICE= FULL REST,LIGHT DIET

**********************************************************************
IF PATIENT='JOHN ALFERD BINY' GET CONDITION $
ERROR DETECTED BEFORE COLUMN NUMBER  10
**********************************************************************
IF SECCODE='SAXENA' AND MRDNO='622442' MODIFY CONDITION='IMPROVING' $
SUCCESSFUL UPDATION
**********************************************************************
IF MRDNO='222266' GET CONDITION $
THE RECORD CORRESPONDING TO MRDNO= 222266 COULD NOT BE FOUND
**********************************************************************
```

```
                    THE RECORD WITH MRDNO= 555544 IS ALREADY DELETED
                    THE RECORD WITH MRDNO= 777744 DOES NOT EXIST IN THE SYSTEM
**********************************************************************************************************
                    IF SECCODE='NARAIN' INSERT FULLINFO $
**********************************************************************************************************
                    IF SECCODE='BHATIA' INSERT DIAGNOSTIC $
                    PATIENT WITH MRDNO= 555588 DOES NOT EXIST IN THE SYSTEM
                    PATIENT WITH MRDNO= 999966 DOES NOT EXIST IN THE SYSTEM
**********************************************************************************************************
                    IF SECCODE='SAXENA' INSERT TREATMENT $
                    PATIENT WITH MRDNO= 444446 DOES NOT EXIST IN THE SYSTEM
**********************************************************************************************************
                    IF SECCODE='LLLLMM' INSERT FULLINFO $
                    THE SECURITY CODE LLLLMM IS NOT A VALID ONE
**********************************************************************************************************
                    X
                    ALL QUERIES ARE ANSWERED
```

APPENDIX - G

RESULTS OF SYNTAX OF THE QUERIES

```
E OPTIONS(MAIN);
   SOURCE

    SYNTAX: PROCEDURE OPTIONS(MAIN);
    DCL DOMAIN(8) CHAR(10) INIT('DATE      ','MRDNO     ','WARDBED   ',
        'NAME      ','FNAME     ','DATEADM   ','ADDRESS   ','SECCODE');
    DCL TARGET(30) CHAR(10) INIT('MRDNO     ','NAME      ','ADDRESS   ',
        'FNAME     ','DATEADM   ','WARDBED   ','LADDRESS  ','NKADDRESS ',
        'PHYSICEXAM','PCOMPLAINT','PSTHISTORY','FDIAGNOSIS','TESTS     ',
        'CONDITION ','DELIVERY  ','LABEXAM   ','OPERATION ',
        'OTHRTRTMNT','XRAY      ','ADVICE    ','RESULT    ',
        'FULLINFO  ','DIAGNOSTIC','TREATMENT ','ALLREPORT ','BSTATUS   ',
        'DSTATUS   ','WSTATUS   ','MSTATUS   ','ASTATUS   ') EXTERNAL;
    DCL LFN FILE INPUT RECORD SEQUENTIAL ENV('FO=SQ,FL=80,RT=Z,BT=C');
    DCL BHT FILE STREAM OUTPUT  ENV('FL=132,RT=Z,BT=C');
    DCL KEYWRD(7) CHAR(10) INIT('IF        ','GET       ','AND       ',
        'MODIFY    ','DELETE    ','INSERT    ','$         ');
    DCL (DCODE(8),TCODE(30),KCODE(7),FLAG,FLAG1) FIXED(1) INIT(0) EXTERNAL;
     DCL MDMN CHAR(100) EXTERNAL;
     DCL (MRD,DTA,SCODE) CHAR(6) EXTERNAL;
     DCL (NME,FNME,ADR) CHAR(30) EXTERNAL;
     DCL NKB FIXED(2) EXTERNAL;
     DCL (MARK,SIGN) FIXED(1) EXTERNAL;
    DCL (L,M,N) FIXED(2) INIT(0) EXTERNAL;
     DCL DEVI(15) LABEL EXTERNAL;
    DCL (FLG,FLGA,FLAG2,FLAG3) FIXED(1) INIT(0) EXTERNAL ;
    DCL D(999) CHAR(6) EXTERNAL ;
    DCL (LM,FM) FIXED(6) EXTERNAL ;
    DCL (FMRD,LMRD) CHAR(6) EXTERNAL ;
    DCL DASHES CHAR(120) INIT((120)'_');
    DCL 1 CARDREC CHAR(80);
    DCL CARD(80) DEFINED CARDREC CHAR(1);
     DCL KARD DEFINED CARDREC CHAR(80);
    DCL SLTH CHAR(40) INIT('  LENGTH OF SYMBOL EXCEEDS 10 CHARACTERS');
    DCL SPACES CHAR(100) INIT(' ');
    DCL EQUAL CHAR(1) INIT('=');
    DCL QUOTE CHAR(1) INIT('''');
    DCL BLANK CHAR(1) INIT(' ');
    DCL SYMBOL CHAR(10) INIT(' ');
    DCL BLNK10 CHAR(10) INIT(' ');
    DCL DBUFF(30) CHAR(100) INIT(' ') EXTERNAL;
    DCL (SL,CTR,SCTR) FIXED(3) INIT(0);
    DCL STARS CHAR(120) INIT((120)'*');
    DCL ERR CHAR(40) INIT(' ERROR DETECTED BEFORE  COLUMN NUMBER  ');
    START1: OPEN FILE (LFN);
    START:  PUT FILE(BHT) SKIP(3) LIST (STARS);
        OPEN FILE(BHT) ;
        CALL NXTCRD ;
        CTR=0;
        SL=0;
        FLAG1=0;
        FLAG=0;
        MARK=0;
        L=0;
        DO I=1 TO 30;
        DBUFF(I)=SPACES;
        TCODE(I)=0;
        END;
        DO I=1 TO 8;
        DCODE(I)=0;
        END;
```

```
E OPTIONS(MAIN);
   SOURCE

        DO I=4 TO 6;
        KCODE(I)=0;
        END;
        CALL SYMBL;
        IF SYMBOL^= KEYWRD(1) THEN CALL ERROR ;
        CALL SYMBL;
    DMN : DO I=1 TO 8 ;
        IF SYMBOL= DOMAIN(I)  THEN DO;
        IF DCODE(I)=1 THEN CALL ERROR;
        ELSE DCODE(I)=1 ;
        J=I;
        GO TO CHEQL;
        END;
        END ;
        CALL ERROR ;
    CHEQL : CALL SYMBL ;
        I=J ;
        IF SYMBOL ^= EQUAL THEN CALL ERROR;
        CALL SYMBL;
        IF SYMBOL^= QUOTE THEN CALL ERROR ;
    LDATA : CTR=CTR+1;
        IF CTR>80 THEN CALL NXTCRD ;
        ELSE ;
        IF CARD(CTR)^= QUOTE THEN DO;
        SL=SL+1;
        IF SL>= 100 THEN CALL ERROR;
        SUBSTR(DBUFF(I),SL,1)= CARD(CTR) ;
        GO TO LDATA;
        END;
        IF KCODE(4)=1 THEN DO ;
        MDMN=DBUFF(I);
        CALL SYMBL ;
        GO TO CHDLT ;
        END ;
        IF DCODE(2)=1 & DCODE(8)=1 THEN GO TO LAST ;
        IF DCODE(8)=1 THEN GO TO CHAND ;
        IF FLAG=1 & (DCODE(1)=1!DCODE(2)=1!DCODE(3)=1) THEN CALL ERROR ;
        IF DCODE(1)=1 ! DCODE(2)=1 ! DCODE(3)=1 THEN GO TO CHGET ;
    CHAND : CALL SYMBL ;
        IF SYMBOL  = KEYWRD(3) THEN DO ;
        IF FLAG =1 THEN CALL ERROR ;
        ELSE FLAG=1;
        CALL SYMBL ;
        GO TO DMN ;
        END ;
        ELSE GO TO CHGT ;
    CHGET: CALL SYMBL;
    CHGT : IF DCODE(8)=1 THEN GO TO LASTA ;
        IF SYMBOL ^= KEYWRD(2) THEN CALL ERROR ;
        KCODE (2) = 1 ;
        CALL SYMBL ;
        IF DCODE(1)=1 THEN GO TO CH2630;
      /* TO CHECK TARGET LIST */
    CH2225: DO I=22 TO 25 ;
        IF SYMBOL = TARGET(I) THEN DO ;
        IF TCODE(I)=1 THEN CALL ERROR ;
        ELSE TCODE(I)=1;
      MARK=1;
```

```
RE OPTIONS(MAIN);
    SOURCE

        CALL SYMBL ;
        GO TO CHDLT;
        END;
        END;
     CH18 : DO I=1 TO 21;
     TGT : IF SYMBOL=TARGET(I) THEN DO;
        IF TCODE(I)= 1 THEN CALL ERROR ;
        ELSE TCODE(I)= 1 ;
        CALL SYMBL ;
        IF SYMBOL = KEYWRD(3) THEN DO ;
        IF FLAG1=0 THEN L=I;
        IF FLAG1=1 THEN M=I;
        IF FLAG1=2 THEN N=I;
       ELSE ;
        FLAG1 = FLAG1 + 1 ;
        IF FLAG1>2 THEN CALL ERROR ;
        ELSE DO ;
        CALL SYMBL ;
        GO TO CH18;
        END ;
        END ;
        ELSE GO TO CHDLT ;
        END;
        END ;
        CALL ERROR ;
    CH2630: DO I=26 TO 30;
        IF SYMBOL=TARGET(I) THEN DO;
        IF TCODE(I)=1 THEN CALL ERROR ;
        ELSE TCODE(I)=1;
        CALL SYMBL ;
        GO TO CHDLT;
        END ;
        END;
        CALL ERROR ;
    CHDLT : IF SYMBOL='$' THEN DO ;
        PUT FILE(BHT) SKIP(3) EDIT ('QUERY IS O.K.') (X(20),A);
        GO TO START;
        END ;
        ELSE CALL ERROR ;
     LAST : IF DCODE(2)=1 & DCODE(8)=1 THEN GO TO CHMD ;
        CALL SYMBL ;
        SCODE=DBUFF(8);
        MRD=DBUFF(2);
    LASTA : IF DCODE(8)=1 & DCODE(2)=1  THEN GO TO CHMDA ;
            IF SYMBOL=KEYWRD(5) THEN DO;
        KCODE(5)=1;
        CALL SYMBL ;
        GO TO CHDLT ;
        END ;
        ELSE IF SYMBOL = KEYWRD(6) THEN DO ;
        KCODE(6)=1 ;
        GO TO CHIN ;
        END ;
        ELSE CALL ERROR ;
    CHIN : CALL SYMBL ;
        DO I=22 TO 24 ;
        IF SYMBOL = TARGET(I) THEN DO ;
        TCODE(J)=1;
```

```
URE OPTIONS(MAIN);
     SOURCE

          CALL SYMBL ;
          GO TO CHDLT;
          END ;
          END ;
          CALL ERROR ;
     CHMD : CALL SYMBL ;
     CHMDA : MRD = DBUFF(2) ;
          IF SYMBOL = KEYWRD(4) THEN DO ;
          KCODE(4) = 1 ;
          CALL SYMBL ;
          END ;
          ELSE CALL ERROR ;
          DO I=7 TO 21;
          IF SYMBOL=TARGET(I) THEN DO ;
          NKB=I-6;
          J=I;
          GO TO CHEQL;
          END;
         END ;
           CALL ERROR;
     SYMBL : PROCEDURE ;
           SCTR=0;
          SYMBOL=BLNK10 ;
          CTR=CTR+1;
          IF CTR >80 THEN CALL NXTCRD ;
     LOOP : IF CARD(CTR)= BLANK THEN DO ;
          IF SCTR^=0 THEN GO TO RETN ;
                                  CTR=CTR+1;
           IF CTR>80 THEN CALL NXTCRD ;
           ELSE GO TO LOOP;
                                  END;
          IF CARD(CTR)>='A' & CARD(CTR)<='I' ! CARD(CTR)>='J' &
          CARD(CTR)<= 'R' ! CARD(CTR)>='S' & CARD(CTR)<='Z' THEN DO ;
          SCTR=SCTR+1;
          IF SCTR >10 THEN DO;
          PUT FILE(BHT) SKIP(3) LIST(SLTH) ;
           CALL ERROR ;
          END;
          SUBSTR(SYMBOL,SCTR,1)=CARD(CTR) ;
          CTR=CTR+1;
          IF CTR>80 THEN CALL NXTCRD ;
          GO TO LOOP ;
          END ;
     /* HERE WE FIND SPECIAL CHR */
          IF SCTR^=0 THEN GO TO RETN ;
          SUBSTR(SYMBOL,1,1) = CARD(CTR) ;
          GO TO OUT ;
     RETN : CTR=CTR-1;
     OUT : END ;
     ERROR :PROCEDURE ;
          PUT FILE(BHT) SKIP(3) EDIT (ERR,CTR) (X(20),A,F(3,0));
          GO TO START ;
          END;
     NXTCRD : PROCEDURE ;
          READ FILE (LFN) INTO (CARDREC);
         PUT FILE(BHT) SKIP(2) EDIT(KARD) (COL(20),A);
          CTR = 1 ;
          IF CARD(1) = 'X' THEN GO TO FINISH ;
```

```
END SYNTAX ;
```

```
IF MRDNO= '456321' GET NAME  AND FNAME AND ADDRESS $

QUERY IS O.K.

**********************************************************************************************************

IF MRDNO='142536' GET NKADDRESS AND DATEADM         AND NAME    $

QUERY IS O.K.

**********************************************************************************************************

IF NAME ='NARENDER K BHATIA' AND ADDRESS =' 124 POORVANCHAL HOSTEL J N U NEW
DELHI 110067' GET FNAME AND ADDRESS    $

QUERY IS O.K.

**********************************************************************************************************

' IF NAME ='NARENDER K BHATIA' AND ADDRESS =' 124 POORVANCHAL HOSTEL J N U NEW
DELHI 110067' GET FNAME AND LADDRESS $

QUERY IS O.K.

**********************************************************************************************************

IF MRDNO='142536' GET NKADDRESS AND DATEADM     AND FNAME    $

QUERY IS O.K.

**********************************************************************************************************

IF MRDNO='142536' GET FNAME AND ADDRESS       AND DATEADM   $

QUERY IS O.K.

**********************************************************************************************************

IF MRDNO='142536' GET ADDRESS AND NAME       AND WARDBED   $

QUERY IS O.K.

**********************************************************************************************************

IF MRDNO='142536' GET NAME AND LADDRESS AND ADDRESS $

QUERY IS O.K.
```

```
IF MRDNO='142536' GET LADDRESS AND FNAME AND        AND NAME $

  ERROR DETECTED BEFORE  COLUMN NUMBER     54
```

```
IF MRDNO='142536' GET FNAME AND NAME AND        WARDBED $

 QUERY IS O.K.
```

```
IF MRDNO='142536' GET DATEADM AND NAME AND FNAME       $

 QUERY IS O.K.
```

```
IF MRDNO='142536' GET DATEADM AND NKADDRESS AND WARDBED        $

 QUERY IS O.K.
```

```
IF MRDNO='142536' GET ADDRESS AND WARDBED AND LADDRESS             $

 QUERY IS O.K.
```

```
IF MRDNO='142536' GET ADDRESS AND LADDRESS AND NAME      $

 QUERY IS O.K.
```

```
IF MRDNO='142536' GET NAME AND LADDRESS AND NAME $

  ERROR DETECTED BEFORE  COLUMN NUMBER     48
```

```
IF MRDNO='142536' GET WARDBED AND ADDRESS $

 QUERY IS O.K.
```

```
NF MRDNO='142536' GET NAME AND WARDBED $
```

ERROR DETECTED BEFORE COLUMN NUMBER 2

****

IF MRDNO='142536' GET ADDRESS AND DATEADM $

QUERY IS O.K.

****

IF MRDNO='142536' GET NKADDRESS AND LADDRESS AND ADDRESS $

QUERY IS O.K.

****

IF DATE='801226' GET DSTATUS $

QUERY IS O.K.

****

IF DATE='801226' GET BSTATUS $

QUERY IS O.K.

****

IF DATE='801010' GET WSTATUS $

QUERY IS O.K.

****

IF DATE = '811002' GET MSTATUS $

QUERY IS O.K.

****

IF DATE = '810707' GET ASTATUS $

QUERY IS O.K.

****

IF MRDNO='810925' GET ALLREPORT $

QUERY IS O.K.

****

IF MRDNO='142536' GET NAME AND DATEADM AND LADDRESS $

QUERY IS O.K.

****

IF MRDNO='142536' GET NAME AND LADDRESS $

QUERY IS O.K.

****

IF NAME= 'JOHN FORG' AND FNAME='LIMPY FORG' GET WARDBED AND ADDRESS $

QUERY IS O.K.

****

IF NAME='SUBHASH KHOSLA' AND DATEADM='811212' GET WARDBED AND CONDITION $

QUERY IS O.K.

****

IF ADDRESS='C 2 545 JANAKPURI ' GET NAME AND FNAME AND WARDBED $

QUERY IS O.K.

****

IF WARDBED='12032' GET NAME AND ADDRESS AND CONDITION $

QUERY IS O.K.

****

IF FNAME='JIMPY MULLER' AND DATEADM='810508' GET ALLREPORT $

QUERY IS O.K.

****

IF NAME='RAM SAWARUP' GET FULLINFO $

QUERY IS O.K.

```
**********************************************************************************************************

IF ADDRESS='7 PARLIAMENT STREET ' AND NAME='AMAR NATH KOHLI' GET DIAGNOSTIC $

QUERY IS O.K.

**********************************************************************************************************

IF SECCODE='BHATIA' AND MRDNO='652426' MODIFY NAME='NARENDER BHATIA' $

 ERROR DETECTED BEFORE  COLUMN NUMBER     50

**********************************************************************************************************

IF SECCODE='BHATIA' AND MRDNO='652426' MODIFY CONDITION='IMPROVING'    $

QUERY IS O.K.

**********************************************************************************************************

IF SECCODE='BHATIA'  INSERT       FULLINFO $

QUERY IS O.K.

**********************************************************************************************************

IF SECCODE='BHATIA'  INSERT       DIAGNOSTIC    $

QUERY IS O.K.

**********************************************************************************************************

IF SECCODE='BHATIA'  INSERT    TREATMENT $

QUERY IS O.K.

**********************************************************************************************************

          W R O N G  Q U E R I E S

 ERROR DETECTED BEFORE  COLUMN NUMBER      6

**********************************************************************************************************

  IF NAME ='NARENDER K BHATIA' AND ADDRESS =' 124 POORVANCHAL HOSTEL J N U NEW
DELHI 110067' $

 ERROR DETECTED BEFORE  COLUMN NUMBER     15
```

```
**********************************************************************************************************

                IF NAME ='NARENDER K BHATIA' AND ADDRESS =' 124 POORVANCHAL HOSTEL J N U NEW

              DELHI 110067    $

                ERROR DETECTED BEFORE  COLUMN NUMBER     49

**********************************************************************************************************

                IF DATEADMISSION= '800906'  GET FNAME $

 LENGTH OF SYMBOL EXCEEDS 10 CHARACTERS'

                ERROR DETECTED BEFORE  COLUMN NUMBER     15

**********************************************************************************************************

                X

' 'ALL QUERIES ARE ANSWERED'
```