PATCOSY—A database system for the National Health Service

J. M. Kerridge

Sheffield Polytechnic, Pond Street, Sheffield S1 1WB

PATCOSY (PATient COmputer SYstem) is a data base system which has been designed specifically for the National Health Service environment. The structure of PATCOSY therefore reflects the structure and inter-relationships which exist between the information created by the different parts of the health services when a total medical information system is considered. This paper describes the major features of PATCOSY and their relationship to the CODASYL recommendations.

(Received May 1975)

1. Introduction

A PATCOSY (Kerridge, 1975) data base is defined by a definition program and the data is then manipulated by applications programs.

A definition program is created from a series of PATCOSY Definition Language (PDL) commands. The output from the program is a sequence of data maps, one for each record type in the data base.

The data maps are accessed by the PATCOSY Manipulation Language (PML) commands so that particular record occurrences can be made available to the applications program which originated the command. Hence the PML is a host language access mechanism.

A medical information system requires that the following should be considered when a data base system is being defined.

1. Maintenance of a non-hierarchical privacy structure.
2. Flexible data structures.
3. A means of specifying the type of storage device upon which a record is to be stored.
4. A flexible access mechanism.
5. A means of utilising already existing programs.

These requirements can be related to the provisions of the Data Base Task Group report for the CODASYL committee (1971), but it will be shown that the CODASYL provisions are unable to provide a solution for all these requirements. The report defines both a Data Description Language (DDL) and a Data Manipulation Language (DML).

The major difference between PDL and DDL is that PDL defines the storage media to which a particular record occurrence will be allocated. This facility has meant that an explicit access mechanism has had to be defined so that the full benefits of such a facility can be exploited. Another difference is that DDL is essentially a procedure orientated language. Examples of this are the privacy and the encoding/decoding mechanisms. It will be shown that a procedural privacy system is infeasible in a medical information system. Similarly having procedures to encode and decode data means that only one data item can be processed at one time, unless the procedure itself contains table or dictionary mechanisms which are part of PDL.

DDL uses a schema, sub-schema mechanism for defining the whole data base and that part of the data base which is required for a particular applications program. PDL has a similar mechanism provided by basis and sub-basis definitions which provide a high level privacy mechanism as well as restricting access to part of the data base. One difference is that the DDL sub-schema varies with host languages. This variance is not needed in PDL due to the manner of construction of data maps (described in Section 3) and due to the development of all encompassing programming languages such as PL/I and ALGOL 68 which allow pre-processing facilities.

PDL requires the designer to specify explicitly the access mechanism to a record whereas DDL uses the set mechanism which leaves the actual intricacies of access undefined. One advantage of explicitly defining the access mechanism is that it allows high level privacy to be incorporated as well as allowing flexibility of data transfers.

The DML and PML carry out the same function, that is, they access the data base as required by the particular manipulation. The major difference is in the manner of access in that PML uses an explicit index mechanism to locate a record occurrence whereas DML uses the set mechanism of DDL. In DML this leads to the problem of currency as sets are processed. In PML all relationships are explicitly defined and because a stack is used to implement the commands it will be shown that the problem of currency cannot occur.

The other significant difference between PML and DML is that PML incorporates the program into the body of the command. This splits an applications program into sections each corresponding to a particular manipulation command. Each command has associated with it the basis or sub-basis which is to be used to control the access thus providing a high level privacy mechanism and the reason why only one form of PDL is required for all host languages.

2. The PATCOSY definition language

The constructs of the PDL are used to form a definition program which is compiled and run to produce data maps. The PDL provides a means of

(*a*) specifying the computer environment within which the data base will operate

(*b*) defining the records which make up the data base

(*c*) specifying the privacy associated with quantities in the data base

(*d*) specifying the relationships which exist between the different entities

(*e*) defining the sequence of storage devices upon which a particular record is to be stored.

2.1. *Operating environment*

Each computer installation is necessarily different and therefore, because PATCOSY includes a mechanism for specifying storage requirements, a means must be provided of defining the installation to PATCOSY so that necessary initialising calculations can be carried out.

2.2. *Record definition*

2.2.1. *Data items*

At the lowest level (**Fig. 1**) of a PATCOSY data base are data items which contain the pieces of information from which the records are constructed. PATCOSY allows fixed and variable

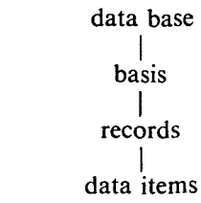

Fig. 1 Hierarchic levels of a PATCOSY data base

Table 1

```
STRUCTURE struct-1 PRIVACY 1 TO 10 CONTAINS
  FIXED item-1 LENGTH 4 PRIVACY 2, 4 TO 8
  FIXED item-2 LENGTH 8 COMPUTATIONAL
  VARIABLE item-3 PRIVACY WRITE 1 TO 5;
                          READ 2, 5 TO 10;
                          7 TO 10
  FIXED item-4 LENGTH 10
```

length scalers to be created as well as fixed and variable length repeating groups. Fixed scalers can have associated with them an attribute which indicates the mode of use, for example, as a string of characters or as a numeric quantity in a particular format. A fixed scaler always has associated with it the number of characters which it occupies. Characters have been used as a means of measuring lengths of data items because these are the only units which are machine independent.

Table 1 contains the definition of a record containing only fixed and variable length scalers. In this record four items are defined; items 1, 2 and 4 are of fixed length and item-3 is of variable length. In addition item-2 is to be held in a computational form so that calculations can be performed more efficiently. (It is realised that different machines employ different representations for numeric quantities, occupying different numbers of characters. Therefore PATCOSY has a means of translating data held on one machine into a form suitable for another machine, hence the length associated with numeric quantities will vary with machine). All other items are held as a string of characters.

In **Table 2** item-8 is a fixed length repeating group the contents of which are repeated twice. The contents are contained in parenthesis. The items of the group contents are items 9 and 10. Item-10 is itself a variable length repeating group whose contents are items 11 and 12.

The other major type of data item is called the associate item. In a medical information system vast quantities of information are going to be created and necessarily this information will have to be encoded in some way. In fact, a simple encoding procedure is not sufficient because in certain situations a piece of encoded information may be constructed from more than one data item. For example, in the encoding of pharmaceutical information, a situation may require just the name of the drug, whilst another may additionally require the maximum and minimum dose, other drugs having the same properties, side effects of the drug, and a means of obtaining a list of all drugs which should not be prescribed with that particular drug. PATCOSY contains a mechanism which provides all these facilities.

In Table 2 there is an instance of such an encoding mechanism. Item-6 is an associate item, that is of itself it occupies no storage space but when accessed obtains information from another source. In this example t-item-1 and 4 of t-table are made available. The particular occurrence of the encoding mechanism is specified by assigning the value of item-7 to t-item-5, another element of t-table. Item-7 occurs within the record as a whole.

The encoding mechanism can take one of two forms; a fixed length contiguous sequence of records called a table or a non-contiguous sequence of possibly variable length records called a dictionary.

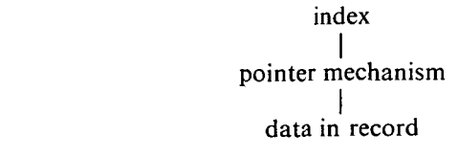

Fig. 2 Access to data items

Table 2

```
STRUCTURE struct-2 CONTAINS
  FIXED item-5 LENGTH 4
  ASSOCIATE item-6 WITH t-item-1, t-item-4 OF
           t-table FOR t-item-5 = item-7
  FIXED item-7 LENGTH 3
  FIXED item-8 LENGTH 2 (FIXED item-9 LENGTH 2
                         VARIABLE item-10
                           (FIXED item-11 LENGTH 1
                            FIXED item-12 LENGTH 2))
```

2.2.2. *Records*

The next level of the data base (Fig. 1) is that of the record. That is, a data base is constructed of one or more records. In PATCOSY classes of record are predefined, and all record definitions must use one of these classes. The hierarchic structure of the access mechanism to a particular data item is shown in **Fig. 2.**

The access mechanism shown implies that all data is obtained via an index and pointer mechanism. This is only true when data belongs to a particular key value which can be contained within an index. This means that there must be more than one class of record for holding prime data.

The classes of record can be grouped as follows

1. *Prime data records* hold data which has been created by a user application program. A *structure* is the physical record of the data base and always belongs to a particular index key value. A *sub-structure* is the logical record created from parts of one or more structures. A *list* is similar to a structure except that the data does not belong to a particular index key value.
2. *Accessing records* For data recorded in structures a mechanism has to be provided for obtaining the particular structure which is required. This process is achieved by using two or three different classes of record. A *pointer-list* provides a means of finding the start of any string of structures. An *index* points to a particular pointer-list so that the start of a particular string of structures can be found. The index contains the key value which is associated with those strings of structures. A *pointer-array* is used when a structure definition can be used for several applications differentiated from each other by a key value which is recorded in the pointer-array. **Figs. 3** and **4** show these mechanisms using pointer-lists and pointer-arrays respectively.
3. *Encoding records* Two classes of record are available, *dictionary* and *table*. In either case a sequence of records is maintained which contain data items which may or may not be used as key items. To access the dictionary or table a value is given to one of the key items, the particular record (or records) containing that key value is found and then the encoded or decoded information is extracted from the rest of the record. Two classes of record are used so that two essentially different encoding mechanisms can be provided. A table is constructed from fixed length records and therefore can be kept as a contiguous sequence and can be accessed by a binary search.

   A dictionary can be constructed from variable length records using many key items and is likely to be implemented using a combination of indexed sequential and inverted file techniques.

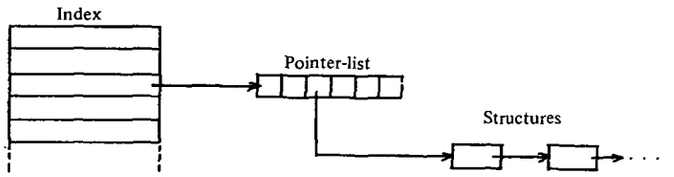

Fig. 3 Access using pointer-list only

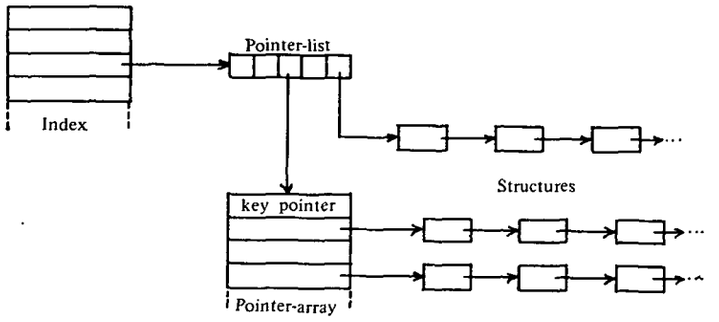

Fig. 4 Access using pointer-list and pointer-array

4. *Transfer records* One major problem associated with the introduction of a data base system is the waste of previous programming effort, because the majority of previously written applications programs cannot be utilised. PATCOSY overcomes this problem by transferring data to and from the data base to traditionally organised sequential files maintained outside the control of the data base. The *transfer structure* provides a mapping function between the data base and the sequential file. The transfer structure can also be used to transfer information from one computer to another, or different types of computer in a multicomputer system.

2.3. *Privacy*

It is obviously vitally important that any data should be maintained so that only authorised users can access the information. The National Health Service is an environment in which an hierarchic privacy structure is inadequate. For example, in an hierarchic system it is impossible for a user to be able to write a piece of data but to be unable to read the same data. This is particularly true of the socio-economic status of the person.

A home nurse may note that a child has no shoes and can record that fact. However the nurse does not need to be able to read other similar information. A social worker must be able to both read and write such information. The nurse must be able to write technical medical information certain parts of which the social worker might be allowed to read but definitely not write. This example would therefore falsify an hierarchical privacy structure.

PATCOSY employs an absolute numerical privacy system in which users of the system are given one or more privacy ratings. A check is made to make sure that the particular manipulation to be carried out is covered by at least one of the user's privacy ratings.

All syntactic forms of the privacy system are shown in Table 1.

Struct-1 as a whole is limited to users who have a privacy rating of 1 to 10 inclusive. Only users with a rating of 2, 4, 5, 6, 7 or 8 can access item-1. Users with privacy 1 to 5 can write item-3; item-3 can be read by users with privacy 2 or 5 to 10 inclusive. Hence only users with 2 or 5 can both read and write item-3. The restriction for all other users manipulation commands is for users with privacy of 7 to 10.

Such a system would be prohibitively expensive to program in a procedure based privacy system because a procedure would have to be written for each combination of privacy ratings. Therefore a procedure orientated system would more than likely invoke an hierarchic system.

2.4. *Relationships*

So far all that has been created are record occurrences with some undefined way of relating them. Relationships have to be defined to reflect the structure existing between records.

First, basic linking mechanisms have to be provided; *link* provides a 'pointer to next' mechanism; *double-link* provides a 'pointer to next and prior' mechanism. Link and double-link are used to form strings of structures and lists and to provide the basic linkage within dictionaries. A particular link or double-link relationship can only join records of the same definition.

The *tree* mechanism provides a means of joining similarly defined records into the form of a binary tree. That is, the tree relationship maintains two 'pointers to next'; one for subsequently written records with lower key values, and the other for those with higher key values.

Within tables or dictionaries it is necessary to indicate those items of the table or dictionary which can act as a key, because necessarily internal system relationships will have to be created. *Accessed by* indicates the items which can act as a key item, and the order in which the items are to be held.

Secondly, inter-record relationships are provided by the *group* and *chain* relationships.

Any record defined to belong to a *group* can be added to that group by an applications program; that is, there is no automatic addition of a record to a group, or chain relationship. In a data base only one group of a particular identifier can exist and therefore that start of that group is maintained by the system.

The chain relationship provides the data base designer with a means of creating inter-record relationships which start within a particular record. Hence a record can be the *start* of a chain or a *member* of a chain. Hence many chains of the same identifier can exist. For example, for each record in a drugs dictionary a chain could be created of other drugs which must not be prescribed with the drug which was the start of the chain. Hence it is possible for a record to be a member of a chain, with the same identifier, a variable number of times, each one started by a different record. Further it is possible for a record to be both the start and a member of the same chain identifier.

Dictionaries could become very large, a drugs dictionary could contain about 25,000 records, hence it is vital that some means of affecting the structure of the dictionary is available. The *frequency* relationship provides a means of counting the number of accesses to a record, per unit time. At the end of this period the dictionary is re-organised to reflect the rate of usage of particular records. The most frequently used records are grouped together and searched first and so on until the required record is found.

In certain instances it may prove desirable to order the elements of a repeating group in some way. Therefore the accessed by relationship can be used to specify the order in which the elements of the repeating group are to be maintained.

The relationships are implemented using one of three different mechanisms

(*a*) direct

(*b*) displacement

(*c*) algorithmic.

In the direct mechanism the physical location of the record is used as the pointer in any record which is to be related to the record.

The displacement mechanism uses the displacement of the record from the start of the record string as the pointer. Hence in inter-string relationships the pointer will be con-

structed from the displacement plus an indication of the particular string in which the record exists.

The algorithmic method allows a data base designer to implement his own relationship mechanism using a procedure specifically designed to reflect the structure of the data base.

Table 3 shows the format of most of the relationships. The privacy of each relationship is defined by the item which acts as the key item, except for groups and chains which have no key item and therefore the privacy must be defined.

2.5. *Storage*

One major aspect of a medical information system is that it would require complete control over the allocation of storage space. When a patient is receiving treatment the relevant parts of his medical record must be immediately available while all other information can be held in a less immediate but still accessible form.

PATCOSY provides a means of indicating the sequence of storage devices to which a particular record will be subjected. The controls which can be applied to the sequence are

1. Records will be removed from one device to another when a specified period of time has elapsed.
2. Records will be moved when a particular *flag* is *set*. *Set* is one of the manipulation commands which can be used by applications programs. A flag is defined by its inclusion in a storage control sequence, and all flag identifiers are unique.
3. A combination of the above two cases in which the sooner to occur will cause the data transfer.

Table 4 shows the format of the storage control commands. The sequence of storage commands results in one of two events. Either the records are destroyed or the records are archived onto a suitable storage media. Once the records have been archived it is possible for them to be on-lined again and restart the storage control sequence as necessary. In any on-lining operation the amount of data transferred is controlled by the set command.

2.6. *The basis*

In Fig. 1 the highest level of a data base, apart from the data base itself is described as a *basis*. A basis is a collection of records each of which can be accessed at the same time. Each manipulation command has associated with it the basis which is to be used for the particular operation. If the manipulation extends to records not contained in the basis, the command will be halted. If a basis includes structures then necessarily the basis must also contain an index and a pointer-list.

A *sub-basis* is a collection of one or more parts of one or more bases. It is also possible to define a sub-basis which contains only part of a record, providing a high level privacy mechanism. Hence it is possible to describe a sub-basis which only contains part of a pointer-list which will automatically restrict the structures which are available to the manipulation.

The basis structure means that more than one index can be created, each being used for a particular requirement. In such a system there must be one index which is capable of pointing to all records; this is called the *master* index. Hence in a medical information system many bases and sub-bases could be defined for specific purposes, each with their own index, and therefore their own access mechanisms.

(*a*) the basis for containing the master index.

(*b*) a basis for those currently receiving hospital care.

(*c*) a basis for those currently receiving general practitioner care.

(*d*) a sub-basis derived from (*b*) of those receiving care under particular specialities.

Table 3

```
DOUBLE-LINK d-link-1 BY item-1 ASCENDING MECHANISM
      IS DIRECT
LINK link-1 BY item-2 DESCENDING MECHANISM IS
      DISPLACEMENT
TREE a-tree BY item-4 USING 'ABCD1234' AS INITIAL NODE
      MECHANISM IS DIRECT
GROUP group-1 MECHANISM IS DIRECT PRIVACY 5;
      group-2 MECHANISM IS DISPLACEMENT PRIVACY
      4, 6 TO 10
MEMBER CHAIN chain-1 MECHANISM IS DIRECT PRIVACY
  ERASE 5
START CHAIN chain-2 MECHANISM IS DIRECT
OWNED BY index-1
```

Table 4

```
KEPT FOR 10 DAYS ON DRUM THEN
KEPT UNTIL flag-1 ON FIXED-DISC THEN
KEPT FOR 7 DAYS OR UNTIL flag-2 ON EXCHANGEABLE-
  DISC THEN
KEPT FOREVER ON TAPE
ON-LINED WHEN flag-3 TO index-1, flag-4 TO index-2
READ WHEN flag-5 RETURN IMMEDIATELY,
          flag-6 RETURN AFTER 6 DAYS
```

(*e*) a sub-basis derived from (*c*) of those receiving care from each of doctor, dentist, optician.

Further it may happen that information created in one basis may have been archived and is then required in another basis, which essentially means tranferring the data from one index to another. This can be achieved by the setting of the appropriate flag in the storage control sequence.

2.7. *Other features*

PATCOSY contains other features, not specifically designed for the National Health Service, that would be useful in a commercial environment. During the development and subsequent running of computer based information systems certain eventualities may not have been accounted for. PATCOSY provides a means of editing secondary applications programs which will carry out the required modifications. A secondary applications program is one which will be executed every time a particular applications program is executed so that the integrity of the data base can be maintained. Further these eventualities may require redefinition of the data base which can be carried out using a different form of the data definition program which allows definitions to be altered, deleted or new ones inserted.

To help in the initial introduction of the data base a utility feature has been created to allow dictionaries, tables and indexes to be rapidly initialised.

3. Data maps

The intermediary between the PDL definition of a record and its use in the course of a manipulation command is the data map. A data map is a sequence of contiguous elements which reflect the constructs of the PDL. The purpose of each element or group of elements of a data map is to indicate the location of the start of data item or relationship within the record, together with any other information which was defined for the item. The data map also maintains the details of the storage commands but these do not occupy any storage space within the record. Hence a discussion of the output from a data definition program devolves into how the data map is formed.

In Kerridge (1975) a description of a system is given in which each element of a data map contains five fields. These indicate: the identifier associated with the definition; the type of data

map element; the length of storage space occupied by the data item; the starting position of the data item in the record, or a means of finding the start of the item; and finally an indication of the number of following elements which are also used to form the definition. These elements indicate privacy restrictions and some definitions require more than one element.

Five fields were chosen because such a number is required for the majority of the constructs. The use of five fields also ensures that the minimum amount of space is wasted if not all fields are used.

4. PATCOSY manipulation language

The manipulation language (PML) is of the host language type, that is, applications programs are written in any language together with PML commands which are able to access the data base and obtain the records which are required for processing. The PML can be divided into three types

1. Prime data commands
2. Relational commands
3. Other commands.

The structure of the first two types of command can be generally discussed as: verb; type; specification; basis; mode; optional program.

The verb indicates the particular command which is to be used. The type defines the record definition which is to be processed. The specification indicates which record or records are to be processed. It is formed by assigning a value or range of values to one or more data items occurring within the record.

The basis indicates which basis or sub-basis is to be used to control access to the data base. In a high security system it would be advisable to create a different sub-basis for each-applications program.

The mode specifies the manner in which the command will proceed. The mode will only be necessary in multiprogramming environments, in which it is possible for two or more accesses to overlap. If *concurrently* is specified then the different access will be ordered according to the command priorities specified in the definition program. If *exclusively* is specified then all other commands will be halted and the current command, being the only one to use the basis or sub-basis defined, will proceed.

Depending on the command the mode could then be followed by the actual program which is to process the data obtained by the command. The program has been incorporated into the body of the command so that it becomes obvious which parts of the program relate to particular commands. It also means that an automatic looping mechanism can be incorporated into processing so that all records which are presented by the command are processed by the program.

4.1. *The prime data commands*

1. *The write command* allows new data to be added to the data base. Any type of record can be added but the majority will be structures and lists. A new record will be positioned according to the value of the key item which is used in a direct or algorithmic relationship. If more than one such relationship is defined the first so defined will be used.

2. *The read command* allows the application program to read data from the structures, sub-structures and lists within the data base. A particular read command is associated with only one list, structure or sub-structure.

One form of the read command allows lists to be used for operator and technician tasking. In this form the read starts from the point where the last such read finished. Thus if a technician is to be given three jobs at the same time the read will obtain the next three elements from the list and a record will be kept of the point reached.

One alternative of read allows relationships which may cross structure/sub-structure boundaries, to be identified for processing.

3. *The delete command* allows one record to be deleted from the data base at one time.

4. *The alter command* allows a record, which has been previously written, to be changed. Only one record can be altered at a time. However, many items of the record may be changed at the same time. The format of the command requires specification of the value of the item which is to be changed as well as the value to which it is to be changed. This mechanism provides a means of safeguarding what has already been written.

If, instead of altering a record, a particular application needs to keep adding similarly defined data to a record, the write command can be used. The data to be added is defined as a sub-structure. The related structure is the record to which data is repeatedly added. In the structure a variable repeating group is defined, the items of which are those defined as the sub-structure. When the write command operates upon the sub-structure another occurrence of the variable repeating group is added to the record. If the sub-structure is used to read from the record, only one occurrence of the repeating group will be released at one time. The whole of the repeating group can be made available either by reading the structure or defining a sub-structure which contains the identifier of the repeating group.

This facility is required to allow patient records to be built up in a form which was defined by Weed (1971). Weed describes a 'problem-orientated record' to which information is repeatedly added after the initial consultation.

The doctor states what is wrong with the patient as a series of headings or problems. He also describes what treatment he is prescribing for each problem. If the treatment cures the problem the doctor can then state precisely what was wrong with the patient, which in due course will lead to more accurate illness types statistics. If the problem was not alleviated another course of treatment can be prescribed, possibly after redefining the problem.

5. *The set command* is used to initiate the operation of a flag. This causes data to be moved from one storage device to another. It is also possible to specify the amount of data which is to be moved.

6. *The find* is similar to the read command except that it has been extended in power. It allows logical combinations of specifiers to be formed (the read command logically 'ands' all the specifiers together) and more than one key index value to be specified. The records found can be processed as found or saved for subsequent processing. These extensions mean that any program using find proceeds more slowly than one using read even if the effect of each program, in a particular case, is the same.

7. *The sort command* allows a sequence of records to be ordered upon any item contained in the record. The sorted order can either be saved for subsequent processing or the records can be processed as they are sorted. Strings of records, relationships and files containing information which has been saved can all be sorted.

8. *The destroy command* deletes any information which has been saved by the find or sort commands, or by the commands which operate upon relationships.

4.2. *The relationship commands*

1. *The add command* allows a record, which is currently being

processed or which already exists, to be included in a particular relationship. The relationship must be a group or a chain. These are the only relationships which cannot be completed when a record is included in the data base. Link, double-link and tree all join records of a similar type, and are dependent for their logical order upon the value of an item contained within the record. Group and chain can join dissimilar records and therefore have no item which can act as a key for the logical ordering of the relationship.

2. *Erase* allows the current record to be removed from a group or chain relationship.

3. *The give command* allows the user to step his way through a relationship, without retaining any trace of the records which have been processed. Give also makes available any records which have been saved by other commands.

4. *Follow* allows a user to process relationships, at the same time maintaining a trace of the path followed so that he can return to any previous point. Follow commands can be nested to as great a depth as necessary to reflect the data structure which is being processed. Other commands can be issued from within the follow sequence, although no trace will be kept of where these commands led. The trace created by a follow command is implemented by a first in, last out stack. The *return* and *retrace* commands, which can only be issued from within a follow sequence, are used for back-tracking.

5. *The return command* In a nested sequence, one option of the follow command allows the user to specify that a particular follow command in the subsequent program sequence should be more fully traced. This is achieved by the user associating a name with the particular follow command. Whenever this particular command is encountered during the dynamic program sequence, the name specified by the user and the information about the appropriate record occurrence are entered on the stack. The return command allows the user to jump to the most recent occurrence of that name in the stack.

6. *The retrace command* allows the user to process his way backwards through the stack until a particular record is found. The stack is not destroyed so that it is possible for the user to return to the place from which the retrace was issued because the retrace command employs the same naming mechanism as the follow command.

7. *Create* allows the initiation of new relationships which are equivalent to groups. The created relationship is intended to be used for a short period of time only and not as a redefinition of the data base. Pointers needed to maintain the relationship are external to any records which may have been added to the relationship. The relationship can be subsequently destroyed by the user.

8. *Membership commands* allow a user to ascertain of which relationships a particular record is the start or a member. One form of the command yields all such relationships; the other yields information about specified relationships.

4.3. *Other commands*
These commands are mainly for the use of the data base designer so that he can interrogate certain system maintained items to find out how efficiently the data base is running. Examples are: the frequency counts of dictionaries can be interrogated and altered so that elements can be artificially moved when the next re-organisation takes place; the balance of a tree relationship can be investigated and altered so as to produce a more balanced tree.

5. The operating environment of a PATCOSY data base

The interaction of data bases in general and PATCOSY in particular with the operating system is one in which many functions overlap. For example, storage device allocation has been the role of the operating system but PATCOSY assumes this role because of the need for dynamic control of storage allocation.

To this end data bases will either become extensions of operating systems or the operating system will become part of the data base system.

Whatever attitude is taken there is need for an easily specified means of assigning storage to records and then being able to easily join records into strings. PATCOSY contains three different pointer mechanisms; namely direct, displacement and algorithmic (see relationships).

In both the direct and algorithmic mechanism a strategy has to be developed for assigning storage of the required type to a record at whatever stage of a storage control sequence is reached. The strategy must also ensure that relinquished storage space is immediately made available for other records.

One possible strategy is that the total available storage space on each storage device type is divided into fixed length units, the length chosen depending on the actual length of the record. A 'free-list' will have to be maintained of all elements which are unused so that the direct mechanism can obtain an element of storage as necessary.

If algorithmic pointers are used then no free list will be required but the algorithm will have to cope with the problems of synonyms.

If both direct and algorithmic pointers are used a free-list will have to be maintained in which 'next and prior' pointers are maintained so that elements can be removed from the free-list when they are used by the algorithmic pointers.

The operating system will also have to obtain the identity of users so that privacy ratings of the user can be established. This information will then be passed to the data base system so that the control of data access can function.

6. Conclusions

PATCOSY is a system which has sufficient facilities to enable a designer to implement a total medical information system. The main conclusion which can be drawn from carrying out this piece of work is that PATCOSY highlights the interaction of a data base system and its operating environment, and that the latter can only be excluded to the detriment of the design of the former.

Acknowledgements

This work was carried out whilst the author was a research student at the Department of Computer Science, University of Manchester. The author acknowledges the guidance and assistance provided by Professor F. H. Sumner and Miss H. J. Kahn.

The author is indebted to Mr. I. W. Draffan for his help in the preparation of the manuscript.

The research was financed, in part, by the SRC.

References

KERRIDGE, J. M. (1975). *A Proposal for a Data Base for the National Health Service*, Ph.D. Thesis. University of Manchester.

CODASYL DATA BASE TASK GROUP April 1971 report, British Computer Society.

WEED, L. L. (1971). Medical Records, *Medical Education, and Patient Care: the problem-orientated record as a tool*, Cleveland Ohio: Case Western Reserve University Press.