A Multi-Level Approach for Data Description and Management of a Large Hierarchical Database Supporting a Hospital Patient Information System

K. SAUTER, W. WEINGARTEN,
J. KLONK and P.L. REICHERTZ
Department of Biometrics and Medical Informatics
Medical School Hannover

D-3000 Hannover, Fed. Rep. of Germany
P.O.B. 610180

ABSTRACT

The paper describes a systematic approach to satisfy the various - and partly conflicting - data processing requirements occurring in a large operational hospital database.

The central patient data bank of the Medical System Hannover contains at the present time the data of about 124000 patients with a large number of medical and administrative data for accounting and reporting. This data is stored in several databases under the hierarchical database management system IMS and in standard OS-files. A number of parametric update and retrieval programs is in operation on this data bank and the main design criteria for the data structures had to be the support of the efficiency of these routine programs in addition to database administrator considerations such as storage space economy, security and stability.

In the recent years, the need for more flexible tools to describe and process this large amount of data has become apparent. Therefore, a data description system handling the various views of data has been designed and implemented in a first version.

The choice of the appropriate type of data processing language for a specific task depends on a number of criteria, such as efficiency, input/output requirements and the complexity of the database problem. Therefore, the following hierarchy of user-languages had to be

developed:

(1) The conventional PL/1-programming using
the DBMS sublanguage DL/1

(2) A procedural database language

(3) A descriptive query language.

The main features of the languages (2) and (3) are described in the paper.

1. INTRODUCTION

1.1 System Environment

An integral part of the Medical System Hannover - an operational computer-supported hospital information and control system for the Medical School Hannover (MSH) (11) - is the patient information system handling medical and administrative data of 124000 patients with 178000 stays (May 1978). These data are stored in several hierarchically structured databases using IMS (6) as database management system (DBMS) in conjunction with a number of files of various organization using OS for data management (13). This combination is referred to in the following as the data bank.

Parametric program systems for updating of the data bank, for conversational enquiries, for various periodical reports and for analyses of data are in routine use, along with utilities for database administration, with particular attention being paid to the maintenance of database integrity (11, 13).

1.2 Problems and Objectives

The main layout criteria necessitate that the routine requirements of such an application-oriented data bank be concerned with system reliability and performance.

However, the ever changing information demands of a dynamic hospital environment require that information processes be modified as new needs arise. The appropriate solution of the related database processing requirements has to cope with the following problems:

- that the creation of new application programs is expensive with regard to the qualified manpower involved and should be restricted to certain program classes (see section 3)
- that there is a need to answer 'ad-hoc' questions
- that the data involved may be embedded in various subsystems of the

MSH using different data management systems
- that the maintenance of documentation consistency and actuality become more difficult.

These considerations led to the following objectives:
- Standardization and formalization of the documentation by centralized data and program description (see section 2)
- Standardized interface to presently used data management systems
- Integration of distributed data sets
- Simplification of application programming with a path-oriented procedural data manipulation language (see section 3.1)
- Provision of a descriptive query language (see section 3.2)
- Provision of interfaces to data analysis and presentation systems
- Simple establishment of permanent and temporary data aggregates.

2. DATA DESCRIPTION CONCEPT

A prerequisite for the implementation of powerful data processing tools such as data manipulation languages is a formal description of the data bank. The basic features of the 'Data and Program Description System (DAPRO)' have already been outlined (14), especially the underlying data model consisting of the data element, the record and the file/database as structural levels in addition to relations describing structural objects and their inter-relationships.

To handle the various views of data as they are seen by the data management system, by an application programmer or by a non-EDP expert at a terminal, the main ideas of the 3-schema-model proposed in the literature (1), are used for DAPRO. This approach comprises essentially:

a) The "Internal Schema" (IS) describing the data as it is stored and handled by the data management system, i.e. not on the elementary machine-dependent storage structure level. In the MSH, the IS consists essentially of several hierarchical structures.

b) The "External Schema" (ES) describing the data as it is viewed by the programmer/user and restricting the access to the application-specific subset.

c) The "Conceptual Schema" (CS) describing, in a way free of redundancy and independent of the implementation as well as of specific applications, those aspects of real world entities (i.e. patients) on which data is stored in the data bank.

The data description facility contains the description of:

- the entities within each schema level
- the relationships between these entities within each schema level
- the mappings between the entities and relationships of the different schema levels.

It is evident that the data description of the IS depends on the data(base) management systems implemented. So, the input data required to generate an IMS database description, are a subset of the IS of the respective database. With the design objective of DBMS-independent CS and ES the differences between the various DBMS are restricted to the IS, a major step towards the strategical goal of general system applicability.

Several classes of ES have been identified: the routine application programs are related to ES which are subsets of the IS whereas a query system for the database layman is based on his view of the data and therefore uses an ES which is closer to the CS.

The least consolidated level is the CS, representing the "natural" view of data, including semantic information on the substructures and the relationships between them, constraints to maintain data integrity, etc. A number of recent publications are devoted to this subject: in (9) the major candidate concepts are evaluated. But no realization of a CS is so far known to us, as has quite recently been stated during the IFIP CONGRESS 77 (16).

At the present time, the following objects and relationships are described in DAPRO:

- Internal Schema
  The IS describes "internal records", such as IMS-segments or file-records and the owner-member relationship between records. Internal records are composed of "internal items".

- Conceptual Schema
  As data model for the CS a single hierarchy of conceptual records (entities) was found to be sufficient within the MSH-environment. Conceptual records are composed of "conceptual items". Conceptual items are mapped into the IS by the following parameters:

-- identification of internal record
-- length and position within the internal record
-- format
-- simple mapping condition.

One conceptual item can be mapped onto several internal items, since there is redundancy at the internal level. Also several conceptual items can be mapped onto the same internal item, because the meaning of some internal items depends on the contents of other items from the same internal records.

- External Schema
  At present no separate ES-descriptions are stored. The IS is taken to be the ES for routine programming, whereas the ES for the query-system is a subset of the CS.

3. PROCESSING LANGUAGES

The choice of the appropriate type of data processing language for a specific application depends on a number of criteria. These are among others:
- performance
- input/output-requirements
- complexity of the database problem

Therefore the following hierarchy of languages has been developed:

LEV1) Conventional PL/1 - call DBMS:

This type of programming (host language with database sublanguage) will remain to be useful where:
- optimum performance is essential and/or
- specially formated interfaces for input/output are needed.

An example of this type of problem is the routine data-acquisition program. The support of this type of programming by the data description system and its possible replacement by the integration of general carrier systems (10, 18) into complex database processing remain under investigation.

LEV2) Procedural Data Processing Language:
Certain types of processing problems are not suitable for

formulation in a descriptive language (5). A procedural database language was implemented for solving such problems, this also serving as an intermediate language for the descriptive query processor (see 3.1).

LEV3) Descriptive Query Language:
Within a hospital environment, many of the database processing tasks consist of more or less complex retrieval requests. For such purposes a language was designed that contains features which:
- can be used without extensive training and
- can be implemented with reasonable effort and efficiency (see 3.2).

3.1 The Procedural Data Processing Language (PDPL)

An analysis of data processing requests leads to the identification of general processing functions such as: retrieve data items from data bases, compute data item values, perform logical condition testing and perform output operations. The different processing functions are placed before, between or after physical data accesses and are, in general, defined and checked out by PL/1-IMS programs.
The existing high-level database processing languages do not cope with all of these general problems, for example:
- They only support special data item types and not the whole spectrum of PL/1 - declares or more general data item types built up by sophisticated data transformation routines.
- They only support record accesses by use of special data management routines, with restrictions even on the manner of retrieving or modifying data segments.

The system DAPRO, mentioned above, defines as records all existing data item aggregates, these being records from files, tuples of relations, communication areas of subroutines, or segments of hierarchical data structures from databases. Many data processing requests involve data spread over different hierarchies or, at the very least, different records from a single hierarchy. Thus it is necessary to perform structure navigation by provision of sequences of accesses to data segments.
In the procedural language PDPL the first part of the so-called access path definition comprises the description of the navigational request to a certain record of a particular hierarchy. Data item processing is related to a segment access and therefore the second

part of an access path definition contains the related data processing functions.
DAPRO provides support for data item processing, e.g. plausibility checking, extraction from or insertion into records, or automatic conversions between the internal and external formats of data items.

Similar efforts to implement generalized data processing by defining high level languages have been published, e.g.:
A relational language of Frasson (4) supports hierarchical databases with the corresponding data type restrictions on its database description. The Lisp Data Manager LIDAM (12) uses a structure base to describe data items and hierarchies, the query being translated into COBOL-code. The University of Toronto (3) uses formal descriptions of data items and data structures for the TOTAL data management system, but has not yet implemented a data processing language. The General Information System (GIS) (7) of IBM enables search definitions to the same extent as DL/1 does for IMS databases, while Operating System files are included in addition. GIS uses a special data description facility (Data Definition Tables) and its language compiler creates modules for the execution of the therewith defined tasks. GIS is, however, not suited for the relational interactive query facilities required within the MSH.

The essential constructs of the PDPL are as follows:

The data processing task is divided into different access paths. The data items involved in the task are defined once only. Any access path is defined by a sequence of statements having the format:

    <keyword> (<statement>)

Eight different keywords can be used to describe the data processing tasks required. The format of the statements differs according to the keywords used and their intended function.

The PDPL thus provides a single solution to the problems arising when data items are
- entered from an external medium, e.g. terminal
- used for record access control (navigation)
- inserted into records
- extracted from records
- computed by use of arithmetic or string operations
- involved in logical condition testing
- transmitted to an external medium.

The execution system of the PDPL is the General Update and Retrieval System (GURS) (17).
Conditional or unconditional actions of this execution system can be specified in the data processing language by use of commands, which complement the PDPL statement types.

These commands may be classified into three groups:
- modify GURS execution parameters such as modes of output
- branch to execution points within an access path or to any other access path
- execute service functions such as: Inspect data description contents, list data definitions, list the access result code, output special messages.

The navigation within hierarchical data structures is record-oriented. The language allows hierarchical record accesses as does DL/1, but it is easier to handle than DL/1.

GURS has been implemented both for batch- and teleprocessing applications.

3.2 The Descriptive Query Language

3.2.1 Language design

The general form of the language follows the usual pattern:
    SELECT < output list > WHERE <condition>.

Implemented or proposed hierarchical languages of this type were found to be not powerful enough. The semantics of the language are therefore based on an interpretation of hierarchies as a collection of relations, where each tuple contains the key of its hierarchical 'father' - tuple. This interpretation is one of the external schemata mentioned in section 2. The <condition> is interpreted as a predicate calculus expression as in ALPHA or QUEL (15). An analysis of typical and frequent retrieval requests showed that the complexity of the <condition> should not be restricted provided that the tuple variables (in the predicate calculus sense) are existentially quantified. The constructs, however, which replace the universal quantifier in other relational languages ('GROUP BY', aggregate functions) are both too complicated for the non-specialized user and not general enough where output analysis is concerned. These considerations led to two decisions:

1.) Grouping and functions are restricted to the defined structural dependencies.
2.) The result of query execution is not a set or relation but a file, because it may contain duplicates. Grouping (counting of frequencies, cross-tabulation) or more sophisticated analysis are performed in subsequent steps by a general statistical package (SPSS (8)). Part of the input specifications (e.g. data description) for this package is also generated by the query system. In different environments other means of output analysis may be appropriate (a report generator e.g. in a business environment).

In most relational languages the structure of the database has to be specified in each query (e.g. which field is in which relation, which field of one relation is a primary key of other relations).
In the MSH query language the queries operate on a structured external schema. By use of the data description system the following structural information is added to the query:

- Names of data items are unique and predefined in the data description system (section 2), which allows the generation of the usual RANGE or FROM expressions.
- Tuple variables and quantifiers do not appear explicitly in the query but are generated through a set of defaults.
- Using the structural dependencies stored in the data description system, the corresponding 'join' - constructs are added to the query. The user is, however, free to specify additional dependencies between the relations or to suppress the inclusion of the 'join' - terms.

The following example illustrates these points. The request:

"Find the names and admission dates of all patients where the diagnoses with the codes 5423 and 4711 are stored for this admission".

is to be executed on the External Schema (ES):

$$\begin{tikzcd}[row sep=small] \text{PAT: NAME, PATKEY,...} \arrow[d] \\ \text{ADM: ADMDATE, ADMKEY,...} \arrow[d] \\ \text{DIA: DIAG} \end{tikzcd}$$

and takes the form:
SELECT NAME, ADMDATE WHERE DIAG(1) = 5423 $\wedge$ DIAG(2) = 4711.

On the corresponding set of relations:

```
PAT: NAME, PATKEY,...
ADM: PATKEY, ADMDATE, ADMKEY,...
DIA: DIAG, ADMKEY
```

the equivalent query in the relational language QUEL (14) is:

```
RANGE OF X IS PAT      }
RANGE OF Y IS ADM      }  RANGE definition
RANGE OF Z1 IS DIA     }  for the variables
RANGE OF Z2 IS DIA     }
RETRIEVE X.NAME, Y.ADMDATE
WHERE Z1.DIAG = 5423
    ∧ Z2.DIAG = 4711
    ∧ Z1.ADMKEY = Y.ADMKEY  }
    ∧ Z2.ADMKEY = Y.ADMKEY  }     'join' terms
    ∧ Y.PATKEY = X.PATKEY   }
```

The equivalent query in the relational language SEQUEL(2) is even longer.

For implementation reasons there are certain restrictions on the use of the language:

- The external structure must consist of a single hierarchy
- The query does not contain references between different occurrences of this structure. (In the example: Data from different patients cannot be compared in the <condition> or combined in one output tuple.)

3.2.2 Query Processing

In a relational file handling system there is no restriction on the sequence of tuple-retrieval calls. Query - translation can be fully oriented towards the structure of the query. This is a complicated process which is as far as we know still not fully understood (15). With a hierarchical data management system the situation is still more complex because here the sequence of DBMS calls has to follow the data structure.

For this reason the following solution was adopted:
Query-processing is divided into two parts: "retrieval" and "evaluation".

- Retrieval: All field names which appear in the query are collected and with the help of the data description system a path through the data bank is constructed which retrieves all values of these fields. This retrieval is carried out using the procedural data processing language PDPL (see 3.1).
- Evaluation: The output of the retrieval process is organized into relations and kept in core. The query is subsequently executed on these in-core relations without the need for complicated optimization.

Because of the language restriction mentioned at the end of the previous section, retrieval and evaluation can be repeated in a cycle for each occurrence of the hierarchical structure (i.e. for each patient in the example). Therefore only a moderate amount of core storage is needed for the intermediate relations.

For certain fields inverted files are maintained within the data bank, mapping these field values to the top-level key. These inversions are used whenever possible.

4. CONCLUSION

The concepts of formal data description and creation of powerful procedural as well as descriptive data manipulation languages based on a central data dictionary and representing a hierarchy of user languages has proven both applicable and useful.

A major objective of future work is the consolidation of the central data description system, especially of the conceptual schema, the incorporation of semantic information and more general mechanics for schema mapping.

REFERENCES:

(1) ANSI/X3/SPARC, Study Group on Data Base Management System, Interim Report, Washington DC: CBEMA, 1975

(2) Chamberlin, D., Boyce, R.: SEQUEL - A Structured English Query Language. IBM Technical Report RJ 1394, IBM Research Lab., San Jose, California, 1974

(3) Dubien, R.J., Corvey, H.D., Sevcik, K.C., Wigle, E.D.: A Data Base System Implementation Providing Data Independence for Medical Applications, Proceedings of the Second World Conference on Medical Informatics, Toronto, 1977

(4) Frasson, C.: A System to Increase Data Independence in a Hierarchical Structure. In: G. Goos and J. Hartmanis (Eds.): Lecture Notes in Computer Science, Vol. 34, Springer-Verlag, Berlin, 235-246, 1975

(5) Huits, M.H.H.: Requirements for Languages in Data Base Systems. In: Douque, B.C.M. and Nijssen, G.M. (Eds.): Data Base Description, North-Holland Publ. Co., Amsterdam, 1975

(6) International Business Machines Corporation (IBM): Information Management System /360, V.2 - General Information Manual, Form No. GH20-0765-1, 1975

(7) International Busines s Machines Corporation (IBM): Generalized Information System, V.2 (GIS/2) - Application Description, Form No. GN20-0360, 1970

(8) Nie, N.H., et al.: Statistical Package for the Social Sciences (SPSS), Second Edition, McGRAW-HILL Book Co., ISBN No. 0-07-046531-2, 1975

(9) Nijssen, G.M.: On the Gross Architecture for the Next Generation Database Management System, Information Proceedings 77. In: Gilchrist, B. (Ed.): IFIP Congress 77, North-Holland Publ. Co., Amsterdam, 1977, 327-335

(10) Pocklington, P.R.: The Necessity for Requirements of and Basic Design of a General Data Interpretation and Evaluation System (DIES). In: Anderson, J. and Forsythe, J.M. (Eds.): MEDINFO'74, North-Holland Publ. Co., Amsterdam, 1974, 411-418

(11) Reichertz, P.L.: The Medical System Hannover (MSH). In: Collen, M.F. (Ed.): Hospital Computer Systems, Wiley & Sons, New York, 1974, 598-661

(12) Risch,T.: LIDAM - LISP Data Manager, Datalogilaboratoriet Report DLU 77/2, Uppsala University, 1977

(13) Sauter, K.: Structure and Functions of the Patient Data Bank in the Medical System Hannover. In: Guenther, A. et al. (Eds.): International Computing Symposium 1973, Davos, North-Holland Publ. Co., Amsterdam, 1974, 585-589.

(14) Sauter, K., Reichertz, P.L., Weingarten, W., Schwarz, B.: A System to Support High Level Data Description and Manipulation of an Operational Data Base System, Medical Informatics, Vol. 1, No. 1, 1976, 15-26

(15) Stonebraker, M., Wong, E., Kreps, P., Held, G.: The Design and Implementation of INGRES, ACM Transactions on Database Systems, Vol. 1, No. 3, 1976, 189-222

(16) Tsichritzis, D. (Chairman): Data Base Organization (Panel). IFIP Congress 77, Toronto, August 7-13, 1977

(17) Weingarten, W., Klonk, J., Sauter, K., Reichertz, P.L.: Individual Data Retrieval by the Non-Programmer and System Supported Data Manipulation in a Complex Hierarchically Organized Data Base System, Proceedings of the Second World Conference on Medical Informatics, Toronto, 1977, 83-86

(18) Wolters, E., Reichertz, P.L.: Problem-Directed Interactive Transaction Management in Medical Systems, Meth. Inform. Med. 15, 1976, 135-140