A MINI DATA BASE FACILITY

C. Thanos, O. Fraccalini+
Istituto di Elaborazione della Informazione del C.N.R.,
Pisa, Italy.

+ Dipartimento Sperimentale di Elettrotecnica ed Elettronica,
University of Pisa, Italy.

112

ABSTRACT: Small machines have become increasingly powerful and versatile. The only limiting factor is the general software available on them. With adequate software support they can be used, not only for scientific and process control, but also for business applications. There is especially a need for improved data management facilities. DETAREL is a small relational system which is being implemented on a small HP2100. It has some elementary relational operations to facilitate user access to the data base. In addition, it has a decision table interface to enable the user to specify requests and data manipulation commands in a simple way. All the facilities are integrated in one high level user oriented language.

INTRODUCTION

The recent developments in minicomputer architecture has allowed to build small machines which offer a gamma of services comparable with those provided by large machines. Minicomputers are cost effective. Small business can now afford the cost of minicomputers. The only stumbling block is the high cost of producing or acquiring the right software. Indeed small business need more adequate software support. They do not have the expertise or resources to produce the software themselves. There are many organizations, which have applications that require data base management but which cannot afford to develop the required computer system software. Therefore it is necessary to develop data base management systems (DBMS) for small machines. These systems will serve organizations that have small scale needs such as smaller data volumes, simpler utilities and a small budget. In addition, there is the possibility to insert small machines in a distributed system: a mini DBMS together with a homogeneous network facility seems a suitable environment to solve the information needs of large organizations, which operate in a decentralized fashion.

There are three major approaches to data base management

1) The hierarchical approach
2) The network approach
3) The relational approach

We believe that the relational approach is best suited to our goals and requirements. We will not elaborate on the facilities offered by a relational system, or on its advantages. We just point out that relational systems offer some advantage to the non expert programmer [Codd 1970 , Codd & Date 1974].

The users of our system can vary including: clerks, secreteries, managers, application programmers and Data base Administrators. It is desirable for the users of a data base management system to interact with it in a way most suitable to their taste. Codd identifies different types of languages which have been developed for the query and modification of relational data bases [Codd 1974]. They serve two different types of users:

casual users (nonprogrammers) and
application programmers

Clerks and managers access the data base via the nonprogrammer interface. Application programmers and Data base Administrators access the data base via the programmer interface, but may also call on the programmer interface.

The programmer interface provides access to the data base from a host programming language (FORTRAN IV in this case) via a set of data manipulation operations. These operations operate on a tuple-at-a-time. Operations such as GET a tuple, INSERT a tuple, REPLACE a tuple, DELETE a tuple are provided. However the purpose of this paper is to concentrate on the nonprogrammer interface.

DETAREL

The nonprogrammer interface is provided to allow casual users access to the data base, without the aid of a programmer. The casual user is presented with a simple relational view of the data and a simple decision table language to operate on the data. The decision tables control the flow of his actions on the data. We believe that complicated or unrestricted control structures are not needed. Most business oriented programming logic is rather simple. We propose, therefore, decision tables as the main vehicle for representing such logic. Decision tables have been used extensively in the past in the data processing [SIGPLAN 1971]. We propose to build them in our language as its main feature. The DEcision TAble RElational Language (DETAREL) is a non procedural language which does not make use of variables, quantifiers or other mathematical concepts; rather it uses an English - keyword format. The version of DETAREL, which is now being implemented, includes facilities for query, insertion, deletion

and update of relations. [Thanos & Fraccalini 1976].
A DEATREL program consists of one decision table as illustrated in figure 1.

The FOR statement is used to define the scope of interaction with the data base. We have two types of FOR statement:

a) FOR ALL TUPLES OF <RELATION NAME>
b) FOR EACH TUPLE OF <RELATION NAME 1> WITH ALL TUPLES <RELATION NAME 2>

The first type of FOR statement involves only one relation and implies that the following decision table contains only simple conditions i.e., conditions that involve a comparison on a single value. The second type of FOR statement involves two relations and implies that the following decision table contains compound conditions as well as simple conditions, with the restriction that the simple conditions are refering to <RELATION NAME 1>. By compound conditions we mean conditions that involve a restriction of a domain of <RELATION NAME 2> to be a set of values selected from <RELATION NAME 1>. The conditions are the basic qualification facility. It enables the user to choose a subset of a relation based on Boolean combinations of conditions on domain values. As we have seen the conditions may be simple or compound. Conditions are used to determine which rule applies. The condition is stated in the left hand column and its values are given in the rule columns. The conditions are predicates with limited entries Y, N or blank. Actions are performed within rules in the integer order given. The actions with blank entries are ignored. The actions are data manipulation commands i.e., invocations of system functions, whose syntax is of the form:

```
<command> ::= {<UPDATE PHRASE>|<INSERT PHRASE>|
               <DELETE PHRASE>|<OUTPUT PHRASE>}
<UPDATE PHRASE> ::= UPDATE <DOMAIN NAME>
                {BY {<VALUE>|<STRING>|
                <DOMAIN NAME>}|{BY ADDING
                |BY SUBSTRACTING | BY MULTIPLYING
                 BY | BY DIVIDING BY}{<VALUE>|
                <DOMAIN NAME>}[,<DOMAIN NAME>{BY
                {<VALUE>|<STRING>|<DOMAIN NAME>}|
                { BY ADDING | BY SUBSTRACTING
                 |BY MULTIPLYING BY | BY DIVIDING
                 BY}{<VALUE>|<DOMAIN NAME>}}]
<INSERT PHRASE> ::= INSERT <DOMAIN NAME> =
               {<VALUE>|<STRING>}[,<DOMAIN NAME>
                = {<VALUE>|<STRING>}]
<DELETE PHRASE> ::= DELETE
<OUTPUT PHRASE> ::= OUTPUT <DOMAIN NAME>[,
                <DOMAIN NAME>]
```

On condition statements are used to detect special situations that may arise at any time during execution. When a condition is detected the associated actions are performed. On conditions have the form:

<On condition >: <action>

Conditions are restricted to those detectable by the hardware-software support of the interpreter plus an end condition to terminate execution.

Each column on the right hand side of the decision table corresponds to a rule which consists of a set of conditions and a set of actions. The rule holds if and only if each condition holds according to its entry in the column (Eij may be N for 'no', Y for 'yes' or blank for 'don't care'). Only one rule may hold at a time. When a rule holds the actions indicated and ordered by the integer entries in the action column, Okj, are performed.

DEATEREL is intended for people who have need for interaction with a data base but who are not trained programmers. The objective of the language is to provide a simple, easy-to-learn means of expressing the primitive actions used by people to obtain information from tables such as "look up a value in a column". Absent from the language are: cursors, explicit interaction, control structures, variables, quantifiers etc. We believe that this language can provide a nice environment for business applications. The system also provides a set of utilities to aid the user, e.g. data entry, report generating and system functions.

EXAMPLES

The facilities of DETAREL will be introduced by two examples against a data base which describes a small company. The data base contains the following tables:

EMPLOYEE (ENO, UNIT, JOB CODE, TITLE, PRIMARY SKILL, SECONDARY SKILL, SALARY)
DEPARTMENT (ENO, DEPT, MGR, FLOOR)

Example 1 (Fig. 2)

- List NAME, JOB CODE, TITLE, of all those working as electrical engineers or in a position where electrical engineer (skill code = 1130) is the job code of the position.
- List NAME, UNIT, JOB CODE, PRIMARY SKILL, for all those whose primary skill does not correspond to the job code required for the position they fill.
- List NAME of those who have the skill of electrical engineer (skill code = 1130).
- List NAME of those who have the skill of electrical engineer but are occupying a position requiring a different job code.
- Find and identify all those employees who are either skilled as electrical engineers and have an annual salary over 10.000 or are skilled as mechanical engineers (skill code = 1120) and have an annual salary of 11.000 or more.
- Print AVERAGE SALARY for all electrical engineers in the company.
- Find how many people are employed in the company.
- Increase SALARY by 5% to every one in the organization.

Example 2 (Fig. 3)

- List NAME, SALARY of all those working for the TOY department.
- List NAME, JOB CODE, SALARY of all those having as Manager ANDERSON.
- List NAME of all those working for departments on the second floor.
- Update SALARY of all those working for the SALES department by 1.000.

IMPLEMENTATION CONSIDERATIONS

It was felt that a simple approach to the implementation of a mini oriented system should be followed [McLeod & Meldam 1975]. Some of the intricate implementation problems in large relational systems can be avoided in the environment of a mini computer.

The DETAREL Interpreter implements the DETAREL language using a low level data management system for storage and retrieval of data in the form of n-ary relations. The Interpreter must translate the non-procedural DETAREL statements into an efficient sequence of tuple-at-a-time low level commands. The DETAREL Interpreter is a FORTRAN IV program. Its main sections are a Syntactic Analyzer, a Rule Mask Algorithm and a Translator [Thanos & Fraccalini 1976].

The syntactic Analyzer checks syntax and generates a clean form of the DETAREL commands for the Translator, as well as data structures for the Rule Mask Algorithm.
The data structures created are: a condition matrix containing an entry for every condition in the condition part in the decision table; an action matrix containing an entry for every action in the action part in the decision table, a matrix representing the action entry of the decision table. Moreover there are created two other matrices, the mask matrix and the table matrix. We prefer to create matrix data structure instead of binary trees because they are easier to manipulate.

We have selected the Rule Mask technique for processing decision tables; we will give a short description of this algorithm for a limited-entry decision table.
The entries in a limited-entry decision table are ternary. If we want to represent the decision table in binary logic, we have to use two bits for each entry. One way is to use one bit to indicate whether the entry is a dash (0) or not (1), the other bit to indicate whether the entry is a Y. Thus we get a mask matrix and a table matrix. The actual processing of the decision table is done in the following way. A binary transaction vector is built by placing a "1" in each true condition position and a "0" in all other positions. The conjunction of this vector with the first rule column of the mask matrix is built and compared with the first column of the table matrix. The conjunction filters out all non-pertinent entries. The comparison with the table matrix shows whether the pertinent Y-entries match. If they do not match, the conjunction and comparison are repeated for the second column and so on. If the decision table is complete we will certainly find a rule which corresponds to the actual conditions. If the decision table is redundant or inconsistent, we might even find more than one rule. When selecting a method for processing decision tables, the objectives are the minimization of the storage requirements for the object program, or the minimization of the average execution time. In terms of storage requirements the rule mask technique is very efficient. In our environment the storage requirement is a critical factor, so we have selected the rule mask technique.

The Translator makes the semantics checks and generates the data structures for the low level data management system: i.e., it translates the DETAREL commands represented by the matrix data structures, as has been mentioned above, into low level commands represented by command control blocks. These data structures, describing the user and the command type, are built by the Translator and are passed as parameters in a procedure call to the low level data management system.

The low level data management system is written in Assembler. The main module of this system is the monitor. It receives the low form of the data manipulation commands, emitted by the Translator , which are represented in a set of data structures (command control blocks). The monitor uses the facilities and the information in the user schema to execute the commands by calling the various commands procedures. Distinct data manipulation commands are handled by different procedures. For memory allocation and I/O operations we use the Basic Control System and the facilities offered by the Operating System DOS/M of the HP 2100F.

In our system there are two relations types: i.e., relations with independent existence (or primary relations) and relations which are formed from other relations using relational operations (derived relations). The derived relations may be of two types, i.e., either derived relations which are independent after they are formed (snapshots) or derived relations which continue to abide by their definitions and they change reflecting the data base modifications (views). The queries, in our system, are one relation at-a-time. The user can create, by using data manipulation commands, only snapshots, while the data administrator can create views using relational operations. For views the system retains only the name and reconstructs them on the fly. No later updates are allowed on views.

CONCLUDING REMARK

The paper has described the Decision Table Relational System which is being implemented at the Istituto di Elaborazione della Informazione of the National Research Council. The system is implemented on HP 2100F with DOS/M. Only a single user program may execute at-a-time. We have implemented the Interpreter and Basic Control System, and we are now working on the low level data management system. We hope to have a first version running at the end of 1976. Improvements and additions to the implemented version of the DETAREL will be made, in particular we hope to introduce extended entry decision tables, and add a macro facility. In addition we plan to implement inverted files as the basic tool for minimizing tuple retrieval operations in interpreting a query. We hope to use our system in a realistic environment to demonstrate the practical feasibility of a relational system for small business applications.

ACKNOWLEDGMENT

We are greatly indebted to Prof. Dennis Tsichritzis of Computer Science Department - University of Toronto - for his valuable guidance and help.

BIBLIOGRAPHY

1. Thanos, C. and Fraccalini, O., DEATERL: un linguaggio relazionale per utenti non programmatori, Nota Interna B76-9, Istituto di Elaborazione della Informazione del C.N.R., Pisa, Italy, March 1976.

2. Thanos, C. and Fraccalini, O., Il prototipo di un interprete per il linguaggio DETAREL, Nota Interna B76-6, Istituto di Elaborazione della Informazione del C.N.R., Pisa, Italy, July 1976.

3. Astrahan, M.M. and Chalberlin, D.D., Implementation of a structured english query language, Research Report RJ 1464, IBM Research Laboratory, San Jose, Calif., October 1974.

4. Bjorner, D., Codd, E.F., Deckert, K.L. and Traiger, I.L., The gamma zero n-ary relational data base interface: specifications of objects and operations, IBM San Jose Research Report RJ 1200, April 11, 1973.

5. Brodie, M., Chan, S., Czarnik, B., Leong, E., Schuster, S. and Tsichritzis, D., Zeta: a prototype relational data base system, CSRG Tech. Rep., Department of Computer Science, University of Toronto, 1975.

6. Brodie, M., Thanos, C. and Tsichritzis, D., A proposed business applications generating system, Nota Interna B74-26, Istituto di Elaborazione della Informazione of the C.N.R., Pisa, Italy, September 1974.

7. Chamberlin, D.D. and Boyce, R.F., SEQUEL: a structured english query language, Proc. ACM-SIGMOD Workshop on Data Description, Access and Control, Ann Arbor, Mich., May 1-3, 1974.

8. Codd, E.F., A relational model of data for large shared data banks, Comm. ACM, vol.13, No.6, June 1970.

9. Codd, E.F. and Date, C.J., Interactive support for non programmers: the relational and network approaches, Proc. 1974 ACM--SIGMOD Debate on data models: data structures set versus relational, Ann Arbor, Mich., May 1-3, 1974.

10. Codd, E.F., Recent investigations in data base systems, Information Processing '74 North-Holland, Amsterdam, 1974.

11. Date, C.J. and Codd, E.F., The relational and network approaches: comparison of the application programming interface, Proc. 1974 ACM-SIGMOD Debate on data models: data structures set versus relational, Ann Arbor, Mich., May 1-3, 1974.

12. Held, G.D., Stonebraker, M.R. and Nong,E., INGRES: a relational data base system, Proc. National Computer Conf., Anaheim, Calif., May 19-22, 1975.

13. Lorie, R.A., XRM - an extended (n-ary) relational memory, IBM Scientific Center Report G320-2096, Cambridge, Mass., January 1974.

14. McLeod, D.J. and Meldman, M.J., RISS: a generalized minicomputer relational data base management system, Proc. National Computer Conf., Anaheim, Calif., May 19-22, 1975.

15. Schumacher, H., The synthesis of optimal decision trees from decision tables, M. Sc. Thesis, University of Toronto, Toronto, December 1974.

16. SIGPLAN notices special issue on Decision Tables, vol.6, No.8, September 1971.

<USER NAME>

<DECISION TABLE NAME>

<FOR STATEMENT>

rule ..... rule

<Condition> : <Condition>		Eij	
<Action> : <Action>		Okj	

<On condition> <Action>

Fig. 1

FOR ALL TUPLES OF EMPLOYEE

ENO ≠ 0	Y											
TITLE = ELEC. ENGR.		Y										
JOB CODE = 1130			Y		N	N						
JOB CODE ≠ PRIMARY SKILL				Y								
PRIMARY SKILL = 1130					Y		Y		Y			
SECONDARY SKILL = 1130						Y		Y		Y		
SAL ≧ 10.000									Y	Y		
SAL ≧ 11.000											Y	Y
PRIMARY SKILL = 1220											Y	
SECONDARY SKILL = 1220												Y
LIST NAME, JOB CODE, TITLE		1	1									
LIST NAME, UNIT, JOB CODE, PRIMARY SKILL				1								
LIST TOTAL (ENO)	1											
LIST AVG (SAL)		2	2									
UPDATE SAL BY MULTIPLY 0,05	2											
LIST NAME					1	1	1	1				
LIST ENO									1	1	1	1

Fig. 2

FOR EACH TUPLE OF DEPARTMENT WITH ALL TUPLES OF EMPLOYEE

DEPARTMENT. DEPT = TOY	Y	N	N		
DEPARTMENT. DEPT = ADMIN	N	Y	N		
DEPARTMENT. DEPT = SALES	N	N	Y		
DEPARTMENT. MGR = ANDERSON				Y	
DEPARTMENT. FLOOR = 2					Y
EMPLOYEE. ENO = DEPARTMENT. ENO	Y	Y	Y	Y	Y
LIST NAME, SALARY	1				
LIST NAME, TITLE		1			
UPDATE SALARY BY 1000			1		
LIST NAME, JOB CODE, SALARY				1	
LIST NAME					1

Fig. 3