Consiglio Nazionale delle Ricerche

ISTITUTO DI ELABORAZIONE
DELLA INFORMAZIONE

PISA

THE DESIGN AND IMPLEMENTATION OF DETAREL

C.Carlesi, O.Fraccalini, C.Thanos, G.Gasparotto

Nota Interna B77-14

Settembre 1977

THE DESIGN AND IMPLEMENTATION OF DETAREL

C. Carlesi, O. Fraccalini +, C. Thanos
Istituto di Elaborazione dell'Informazione
National Research Council
56100 PISA Italy

G. Gasparotto
Istituto di Matematica Finanziaria
University of Pisa
56100 PISA Italy

ABSTRACT.

This paper describes the currently operational version of a mini computer relational data base management system: the DEcision TAble RELational system (DETAREL).

DETAREL is a generalized data base management system based on Codd's relational model of data. It is designed and implemented at the Istituto di Elaborazione dell'Informazione of the National Research Council and it is running on the HP 21MX computer. It has a decision table interface to enable the user to specify requests and data manipulation commands without the aid of a programmer.

The DETAREL system, in addition of being a stand-alone data base management system, gives the basis for the HP 21MX to be a node in a distributed data base.

+ Present address Syntax S.p.A., Via G. Negri,8, Milan

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 volume, simpler utilities and a small budget. In addition, there is the possibility to insert small machines in a distributed system: a mini dbms with a 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. The advantages of a relational model for data base management systems have been extensively discussed in the literature [Codd 70, Codd 74, Date 74] and hardly require further elaboration.

In choosing the relational model, we were particularly motivated by 1) the high degree of data independence that such a model affords and 2) the possibility of providing a high level and entirely procedure-free facility for data definition, retrieval, update, access control, support of views and integrity verification.

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 dbms 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 74]. They serve two different types of users:

- casual users (non programmers) and
- application programmers.

Clerks and managers access the data base via the nonprogrammer interface. Application programmers and data base administrator 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. In the current version of DETAREL the programmer interface is not implemented.

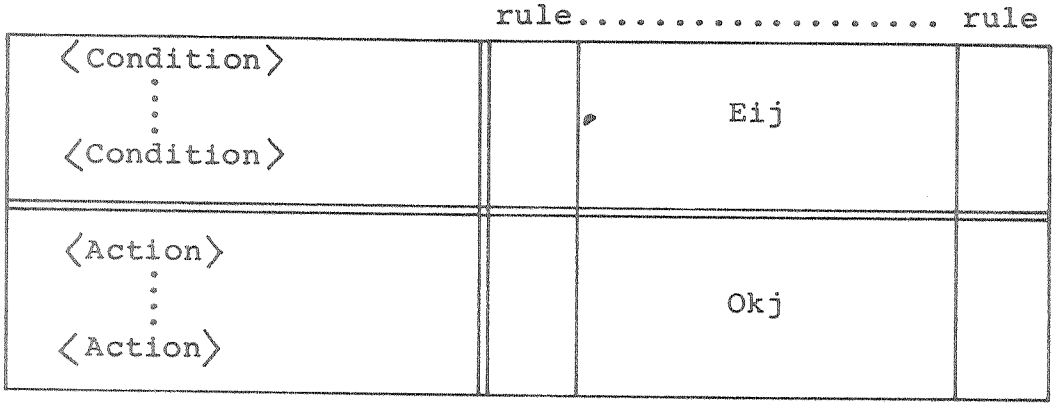
THE NONPROGRAMMER INTERFACE.

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 71]. We propose to build them in our language as its main feature. The decision table language is a non procedural language which frees the user from concern for how data structures are implemented and what algorithms are operating on stored data. As such it facilitates a considerable degree of data independence. Absent from the language are: cursors, explicit interaction, control

structures, variables, quantifiers e.t.c. We believe that this language can provide a nice environment for business applications. The version of the language, which has been implemented, includes facilities for query, insertion, deletion and update of relations [Thanos 76]. The system also provides a set of utilities to aid the user, e. g. data entry, report generating and system functions.

A DETAREL program consists of one decision table as illustrated in fig. 1.

⟨DATA BASE NAME⟩ ⟨USER CODE⟩
⟨DECISION TABLE NAME⟩
⟨FOR STATEMENT⟩



⟨On condition⟩ ⟨action⟩

Fig. 1

Data Base name, User code. The user must furnish his ⟨User code⟩ and the ⟨Data base name⟩ in order to validate the authenticity of user.

FOR Statement. 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 OF ⟨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⟩.

Conditions. 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 predicate with limited entries Y, N, on blank, whose syntax is of the form:

Simple Condition:

⟨ATTRIBUTE NAME⟩⟨OPERATOR⟩⟨VALUE⟩|⟨STRING⟩|⟨ATTRIBUTE NAME⟩

Compound Condition:

⟨RELATION NAME 1.ATTRIBUTE NAME⟩⟨OPERATOR⟩⟨RELATION NAME 2. ATTRIBUTE NAME⟩

⟨OPERATOR⟩::=>|=|<|>=|<=|≠

Actions. The actions are data manipulation commands i. e., invocations of system functions, whose syntax is presented in the Appendix A. Actions are performed within rules in the integer order given. The actions with blank entries are ignored.

Rules. 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.

On Condition statement. On condition statement 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.

EXAMPLES.

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

EMPLOYEE (ENO, NAME, DEPT, UNIT, JOB CODE, TITLE, PRIMARY SKILL, SECONDARY SKILL, SALARY)

DEPARTMENT (DEPT, MGR, FLOOR, INVOICED)

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 engi-

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 BY 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

MGR = "ANDERSON"		Y		
FLOOR = 2			Y	
INVOICED> 100				Y
EMPLOYEE.DEPT = DEPARTMENT. DEPT		Y	Y	Y
LIST NAME, JOB CODE, SALARY		1		
LIST NAME			1	
UPDATE SALARY BY MULTIPLY BY 0,05				1

Fig.3

neer but are occuping 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 company.

Example 2 (Fig. 3)

- List NAME, JOB CODE, SALARY of all those having as MGR Anderson
- List NAME of all those working for departments on the second floor.
- Increase SALARY by 5% to every one working for departments with INVOICED over 100.000.

DATA DEFINITION FACILITY (DDF).

DETAREL offers facilities for data definition.

Data definition in relational systems has three main aspects:

- Specification of the characteristics of data to be stored e. g. the attributes names, data types e.t.c. for each relation;
- changes of the data base after it has been created e. g. insertions of new relations or droppings of relations; and
- definition of alternative "riews" which are derived from the stored data. A view is a dynamic "window" of a data base [D. Chamberlin 75].

The DETAREL data definition facility covers the first two

aspects and partially covers the third. A data base can be created based on its description and destroyed. In addition, after a data base has been created, it may be changed by inserting new permanent relations or by dropping existing stored relations.

An important observation to be made is that the definition of a view is simply a process of deriving a relation from the set of stored relations. A view may be a selected subset of a stored relation, or it may span more than one stored relation, as in the case of a join.

DETAREL offers the ability to define "limited views". These views can be thought of as derived relations (only projections) which cannot be operated on as can stored relations in a data base. Any change of the underlying relation is reflected in these limited views but the views cannot be modified directly, they only can be retrieved.

The syntax of the DDF is presented in the appendix B

In our system, the Data Base Administrator (DBA) is responsible for defining, creating and maintaining one or more data bases. The data definition facility as well as other utility programs are provided by DETAREL for a DBA.

IMPLEMENTATION CONSIDERATIONS.

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

System Catalogs. The data base description is given in the system catalogs or schema. The system catalogs describe all units of logical data and the relationships that exist among them. In particular the USER DESCRIPTION catalog contains the user identifier of all the data bases users and the access rights they have on every relation owned by them.

The DATA BASE DESCRIPTION catalog contains a list of the data bases (by name); for every data base it contains a relative address of each relation within a RELATION DESCRIPTION catalog and the relation type (primary relation or view).

The RELATION DESCRIPTION catalog provides information about the structure and content of each relation in the data base (name of the relation, number of tuples in relation, number of attributes in relation, width (in bytes) of a tuple, attribute names, attribute number (position) in relation, data type of attribute (integer or character string), length (in bytes) of attributes, key information (if an attribute is part of the key) e.t.c.) All catalogs are stored in files provided by the DOS/3.

Storage Structure. Each data base contains two types of components: relations and datatypes. Relations are the most important part of the data base conceptually and contain the interdata relationship information. The datatypes contain the actual data items that are related by the relations (in the data base). A relation does not contain the data items themselves, but rather contains pointers to the data items in the datatypes. Thus, for example, the name "A. Rossi" is stored at most once in the data base (in the datatype "name"). All occurances of "A. Rossi" are actually pointers to the string "A. Rossi" in the datatype "name". Thus, for each relation, there is a tuple file containing an entry for each tuple in the relation. Each entry is a list of pointers (possibly null) one for each attribute of the relation. These entries are stored sequentially as fixed-length blocks in the current implementation. The pointers in the tuple file point to the actual data items, which are stored in the attribute value files. Actually, if the data type of the data items is integer, then it is stored in place of the pointer.

THE DECISION TABLE LANGUAGE INTERPRETER.

The Decision Table Language Interpreter implements the Decision Table Language using an Access Method Interface (AMI)

DEPARTMENT relations is made, based on that tuple, yielding a set of tuples in the EMPLOYEE relation. One then returns to the DEPARTMENT relation to find the next department and repeat the same process.
An alternative approach is the relation at a time execution mode. That is, one operation is performed on all tuples of the one or two relations before the next operation is done.

ACCESS METHOD INTERFACE (AMI).

The AMI handles all actual processing of data from relations. The AMI language is implemented as a set of functions whose calling conventions are indicated below.
The eight implemented calls are as follows:

1) START (data-base-name, user-id)

   The purpose of the START command is to prepare a data base for processing. It enables the user ⟨user-id⟩ to access the data base ⟨data-base-name⟩.

2) STOP (data-base-name, user-id)

   The STOP command cleans up processing when access to a data base in no longer desired.

3) OPEN (relation-name, descriptor, access-mode)

   Before a relation may be accessed it must be opened. This function opens the DOS/3 file for the relation and fills in a descriptor with information about the relation from the RELATION DESCRIPTOR catalog. The descriptor is used in subsequent calls on AMI routines as an input parameter to indicate what relation is involved. Consequently the AMI data accessing routines need not themselves check the system catalogs for the description of a relation. Access-mode specifies whether the relation is being opened for update or for retrieval only.

4) CLOSE (relation-name, descriptor)

This function closes the relation's DOS/3 file and rewrites the information in the descriptor back into the system catalog if there has been any change.

5) GET (descriptor, tuple-id, work-area)

This function retrieves into work-area a single tuple from the relation indicated by the descriptor. Tuple-id is the tuple identifier. The GET function is intended to be called successively to retrieve all tuples of the relation indicated by the descriptor.Each time GET is called the tuple identifier of the next tuple is placed into tuple-id in readiness for the next call. Reaching of the last tuple is indicated by a special return code.

6) INSERT (descriptor, work-area)

This function inserts a new tuple in the relation.

7) REPLACE (descriptor, tuple-id, work-area)

This function replaces by new values the tuple indicated by tuple-id. The tuple identifier of the affected tuple will have been obtained by a previous GET.

8) DELETE (descriptor, tuple-id)

The tuple indicated by tuple-id is deleted from the relation altogether. The tuple identifier of the affected tuple will have been obtained by a previous GET.

UTILITIES.

DETAREL provides several data base utilities:

1) DATA ENTRY

It is the loading facility of the DETAREL.It reads the data initially punched on cards with a free format, in the form of n-ary relations, it processes them according with the relational schema and stores them in the files managed by the DETAREL system (Appendix C) [Gasparotto 77].

2) LIST USER ⟨User-code⟩.

This utility returns the user ⟨user-code⟩ description.

3) LIST DATA BASE ⟨blank⟩|⟨data-base-name⟩.

This utility returns a list of the names of all data bases or the data base ⟨data-base-name ⟩ description is printed out.

4) LIST RELATION ⟨relation-name⟩ ⟨data-base-name⟩.

The relation ⟨relation-name⟩ description is printed out.

5) COPY RELATION ⟨relation-name⟩⟨data-base-name⟩ ⟨relation-name⟩.

This utility is very useful, not only for making copies of relations, but even for restructuring relations.

6) CREATE USER ⟨user-code⟩⟨data-base-name⟩.

This utility creates an entry in the USER DESCRIPTION catalog. The DBA specifies the user code.

7) DELETE USER⟨ user-code⟩.

This utility deletes an entry in the USER DESCRIPTION catalog.

In addition there are other utility programs which serves the following purposes:

- data base initialization;
- data base backup;
- data base recovery;
- maintain the system catalogs;
- print diagnostic messages.

LIMITATIONS.

At this point, it seems appropriate to mention what appear to be some of the most significant limitations of the DETAREL system:

- Retrieval involving more than two relations is not possible;
- No great optimizing strategies were employed for storage allocation and retrieval. In designing a dbms for such a small

computer the main difficulties are presented by the machine limitations.
- The first version of DETAREL is relatively slow.
- It provides only an elementary data protection and data security.

However, it should be Kept in mind that this is a prototype system. It was built as a feasibility study of a mini dbms. In our Institute we build prototypes mainly for research purposes. Other people can use the ideas in commercially oriented systems.

CONCLUDING REMARKS.

In this paper we presented the DEcision TAble RELational system. It is implemented on HP 21MX with DOS/3. It has been shown that a relational data base management system is practical in the mini computer environment. Although some compromises in power and generality were necessary, a usefull and cost-effective implementation has been produced. Only a single user program may execute at-a-time. We have tested the system with a number of small data bases. We are currently generating a data base for a librarian application.
Improvements and additions to the implemented version of the DETAREL will be made,in particular we hope to introduce extended entry decision tables and to implement inverted files as the basic tool for minimizing tuple retrieval operations in interpreting a query. In addition we plan to implement a version of DETAREL to operate on distributed data bases.

ACKNOWLEDGMENT.

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

REFERENCES

1. Thanos, C. and Fraccalini, O. "DETAREL: un linguaggio relazionale per utenti non programmatori", Nota Interna B76-9 Istituto di Elaborazione dell'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 dell'Informazione del C.N.R., Pisa, Italy, July 1976.

3. Carlesi, C., Fraccalini, O. and Thanos, C. "DETAREL: A generalized minicomputer relational data base management system" Proc. Mini-and Microcomputers and their applications Symposium, Zürich, Switzerland, June 7-9, 1977.

4. 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.

5. Astrahan, M. M. and Chamberlin, D. D. "Implementation of a structured english query language" Research Report R. J. 1964, IBM Research Laboratory, San Jose, Calif, October 1974.

6. Codd, E. F. "A relational model of data for large shared data banks" Com. ACM, vol. 13 N. 6, June 1970.

7. Codd, E. F. "Recent investigations in data base systems" Information Processing 74 North Holland, Amsterdam, 1974.

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

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

10. Stonebraker, M., Wong, E., Kreps, P. and Held, G. "The design and implementation of INGRES" Memo ERL-M577, Electronics Research Laboratory, University of California, Berkeley, August 1974.

11. McLeod, A. J., and Meldam, M. J. "RISS-a generalized minicomputer relational data base management system" Proc. 1975 Ncc, AFIP Press, 1975.

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

13. Chamberlin, D. D., Gray, J. N. and Traigger, I. L. "Views, Authorization and Locking in a relational data base system" Proc. 1975 NCC, AFIPS Press. 1975.

14. Schumacher, H. "The synthesis of optimal decision trees form decision tables" M. Sc. Thesis, University of Toronto, December 1974.

15. SIGPLAN notices special issue on Decision Tables, vol. 6, n. 8, September 1971.

16 Gasparotto, G. "A Data Entry System" Nota Interna to appear Istituto di Elaborazione dell'Informazione del C.N.R. di Pisa, Italy.

APPENDIX A

Data Manipulation Commands

```
<COMMAND> ::={<UPDATE PHRASE>|<INSERT PHRASE>|<DELETE PHRASE>|
             <OUTPUT PHRASE>}

<UPDATE PHRASE> ::= UPDATE<ATTRIBUTE NAME>{BY{<VALUE>|<STRING>|<ATTRIBUTE
                    NAME>}|{BY_ADDING|BY_SUBTRACTING|BY_MULTIPLYING|BY|
                    BY_DIVIDING BY}{<VALUE>|<ATTRIBUTE NAME>}}[,<ATTRIBUTE
                    NAME>{BY{<VALUE>|<STRING>|<ATTRIBUTE NAME>}|{BY_ADDING|
                    BY_SUBTRACTING|BY_MULTIPLYING|BY|BY_DIVIDING BY}{<VALUE>|
                    <ATTRIBUTE NAME>}}]

<INSERT PHRASE> ::= INSERT<ATTRIBUTE NAME>={<VALUE>|<STRING>}[,<ATTRIBUTE
                    NAME>={<VALUE>|<STRING>}]

<DELETE PHRASE> ::= DELETE

<OUTPUT PHRASE> ::= OUTPUT[<FUNCTION>]<ATTRIBUTE NAME>[,[<FUNCTION>]
                    <ATTRIBUTE NAME>].
```

APPENDIX B

Data Definition Facility (DDF)

The syntax below is intended primarily to give the reader an intuitive understanding of the structure of the DDF.

```
1.<CREATE DATA BASE> ::=CR DB<DATA_BASE_NAME>

2.<DELETE DATA BASE> ::=DE DB<DATA_BASE_NAME>[<USER_CODE>]

3.<ASSIGN RELATION>  ::=A  RE<RELATION_NAME><DATA_BASE_NAME><USER_CODE>
                           <ACCESS_MODE>

4.<CREATE RELATION>  ::=CR RE<RELATION_NAME><DATA_BASE_NAME><ATTRIBUTES_
                           NUMBER><ATTRIBUTE_NAME><ATTRIBUTE_TYPE><ATTRIBUTE_
                           LENGTH>[<KEY>|<INDEXED>][,<ATTRIBUTE_NAME>
                          <ATTRIBUTE_TYPE><ATTRIBUTE_LENGTH>[<KEY>|<INDEXED>]]
```

5.⟨DELETE RELATION⟩ ::=DE RE⟨RELATION_NAME⟩⟨DATA_BASE_NAME⟩

6.⟨CREATE PROJECTION⟩ ::=CR PR⟨RELATION_NAME⟩⟨DATA_BASE_NAME⟩⟨RELATION_NAME⟩
⟨ATTRIBUTES_NUMBER⟩⟨ATTRIBUTE_NAME⟩[⟨KEY⟩|⟨INDEXED⟩]
[,⟨ATTRIBUTE_NAME⟩[⟨KEY⟩|⟨INDEXED⟩]].

7.⟨DELETE PROJECTION⟩ ::=D PR⟨RELATION_NAME⟩⟨DATA_BASE_NAME⟩

APPENDIX C

Data Entry Data Input format (example)

```
DATA_BASE_NAME  =  DAB
USER_CODE       =  UTE
DELIMITER_MARK  =   /
RELATION_NAME   =  DEPARTMENT
ATTRIBUTE_NAME  =  YES
MGR/FLOOR/DEPT/INVOICED//
RELATION_BEGIN
SMITH/1/TOY/100K//
JONES/2/SALES/150K//
   :
ANDERSON/3/ADMIN/100K///
RELATION_END
RELATION_NAME   = EMPLOYEEE
ATTRIBUTE_NAME  = NO
RELATION_BEGIN
   :
   :
/* RELATION_END
/*/* DATA_BASE_END
/*/*/*
```