Anales PANEL'81/12 JAIIO
Sociedad Argentina de informática
e Investigación Operativa. Buenos Aires, 1981

A RELATIONAL DATA BASE MANAGEMENT SYSTEM

Ricardo O. Giovannone

DATA S.A.
Bdo. de Irigoyen 560
Buenos Aires - Argentina

RESUMEN

Un DBMS Relacional para la educación

Como resultado del proyecto "A Relational Data Base System", se construyó un programa llamado MINI DBMS1. Este programa, escrito en el lenguage de programación PASCAL (UCSD PASCAL), simula la actividad de un administrador de base de datos. El modelo de datos utilizado fue el relacional y el programa fue implementado en una Minicomputadora.

El objetivo principal de proyecto fue la implementación de un sistema que simule la actividad desarrollada en un DBMS, teniendose en cuenta fundamentalmente las distintas actividades que se desarrollan en la generación, mantenimiento e interacción con una base de datos a través de un DBMS. Se tuvo en cuenta que los usuarios serian estudiantes de un curso de base de datos y que tendrian que interactuar con el sistema y obtener como resultado el conocimiento conceptual del funcionamiento de un DBMS. Se asumió que los estudiantes o usuarios del sistema debian conocer el modelo relacional y las operaciones relacionales correspondientes.

Las operaciones del DBMS que se implementaron son: aquellas que realiza el administrador de la base de datos,(creación de las relaciones y destrucción de las mismas), aquellas operaciones para el mantenimiento de relaciones,(altas de tuplas,

bajas y cambias) y por último, las operaciones relacionales, proyección (projection), restricción (restriction), selección (selection), producto cartesiano (cross product) y acoplamiento (join).

El programa fue implementado en una minicomputadora, explotando las caracteristicas interactivas del lenguage y del hardware disponible (CRT, unidad central y unidad de diskette).

En el reporte correspondiente a este proyecto se cubren los siguientes puntos:

- Estructuras de datos utilizadas en el programa.
- Explicación conceptual del funcionamiento de los módulos que realizan las actividades principales del DBMS.
- Conclución.

Se quiere dejar claro que el objetivo de este sistema es educativo y no para el uso en aplicaciones comerciales. Esto se manifiesta en la implementación de la mayoria de las operaciones de un DBMS puesto que se considera una cantidad reducida de tuplas por relación en la base de datos.

1. INTRODUCTION

The result of the project "A Relational Data Base System" is a program called Mini DBMS1. This program has as its main objective the implementation of a data base management system in a minicomputer environment for educational purposes rather than applications-oriented software. The relational data model was the one used in this project.

The educational objective of the system concerns the exercise of data base administrator,relations maintenance, and relational operations activities utilizing a small number of relations and tuples. The data base administrator operations are creation and deletion of relations. The relations maintenance type of operations that can be performed are of common type to a data base. They are insertion of tuples or records, deletion, and update. The relational operations that can be exercised are projection, selection, restriction, cross product, and join. This assumes that the user of the system should have knowledge about the relational model and operations.

This project was divided into phases. Phase one concerned relational operations over one relation, and phase two involved relational operations combining two relations. The host language used in this implementation was UCSD PASCAL

version I. 4. This language allows the possibility of having machine-independent programs, which means that the programs developed in this language can run in a variety of minicomputers and microcomputers. The machines used at the moment are a Terak minicomputer and a PDP11/10. The project direction was under Prof. D. Dearholt, and the students involved were Ying-Ying Peng in phase one, and Ricardo Giovannone in phases one and two. This paper covers the work done in phases one and two of the project. The paper will cover explanations of program modules and strategies used.

2. DATA BASE MANAGEMENT SYSTEM STRUCTURE

The Mini DBMS1 is a set of program modules (procedures and functions) that make possible the handling of relations to and from the diskette. Relations are stored in the diskette and fetched to memory for further processing using basic procedures or functions. When the relational data are brought to memory, these are stored temporarily in either of two data structures (arrays) that have the same structure as the file records holding the relations in the diskette. These data structures serve as a working area for most of the procedures or functions that perform basic activities like retrieving,

printing, or recording. These arrays are also used in all the specialized procedures that perform the relational operations. They are also used by the procedures that do insertion, updating, and deletion of tuples in relations already created in the data base.

Figure 1 shows this conceptual view of the Mini DBMS1 activity. Figure 2 shows all the commands available in this Mini DBMS1 to exersice all the operations.

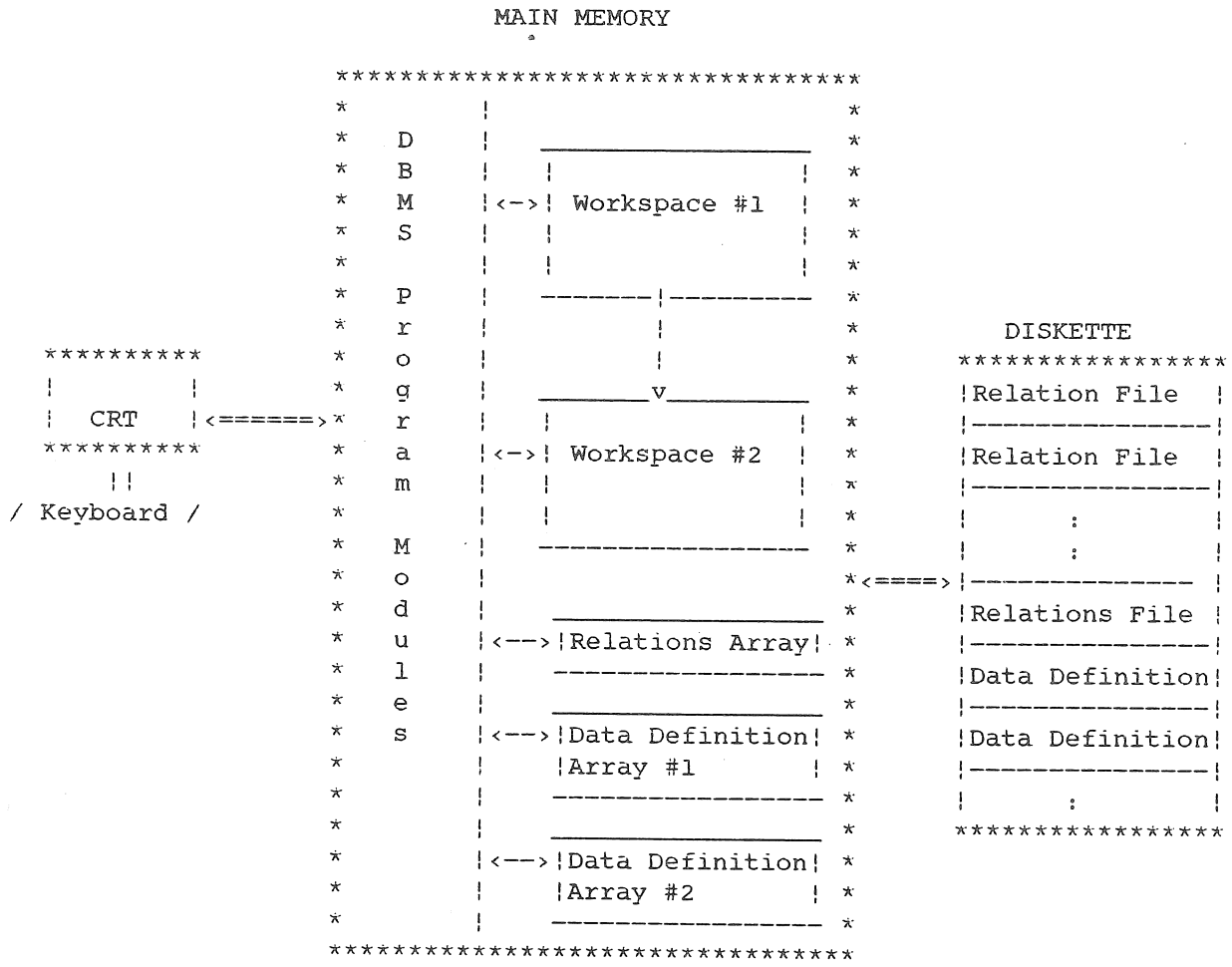

Figure 1. Conceptual view of the data base management activity.

To start the execution of the system Type: X , then a question like this will appear, What file?, and you Type: DBMS1

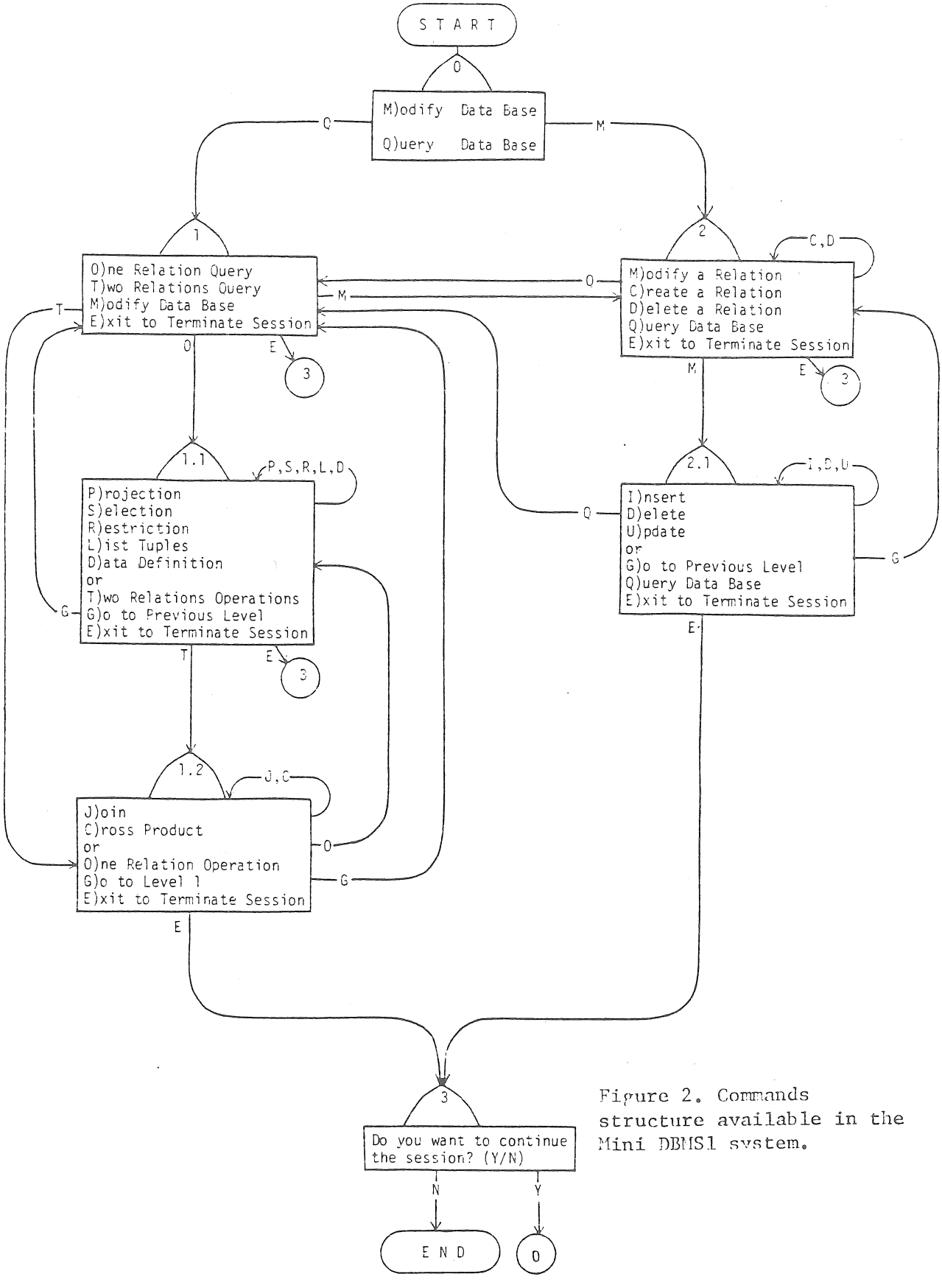

Figure 2. Commands structure available in the Mini DBMS1 system.

The following section will describe the different type of data structures that are generated and used by this Mini DBMS1.

2.1. Relation File

This file is composed of records of characters (maximum 50 characters). Each record holds one tuple of the relation. One of the constraints in using this structure is that not all the relations have tuples 50 characters long, so some space is wasted. The number of tuples that a relation can have is not a restriction. However, there is a restriction in terms of the working space created by the program in the main memory. The working area holds a maximum of 10 tuples per relation. Of course this can be changed just by changing the dimension of the working area, but one should have a memory bigger than 56K bytes.

2.2. Data Definition File

The data stored in this file refers to the different qualifications related to each attribute in a particular relation. The file name is associated to the relation name in

order to fetch it when needed. The information about each attribute is stored in a record. There is one record per attribute. The information concerns domain of the attribute, length, starting position in the tuple, designation of key or common attribute, and value ranges. There is one Data Definition file for each relation created.

2.3. Relations File

The purpose of this file is to store information about the current relations in the data base. There is a record for each relation. This record is formed by the fields relation name, number of tuples, and number of attributes.

The file structures presented are the only ones at the moment used by the program modules. The relation file and the data definition files are used for error checking purposes and to get parameters of information used in all the procedures and functions.

The relation file serves only as a means to store the relational data to be managed by the system.

3. PROGRAM MODULES

This section will refer to the different procedures or

functions used to execute data base administartor operations, relational operations and insertion, deletion, and updating of tuples.

3.1. Structure of Data Base

This command is produced automatically each time the user starts a session. The purpose of this procedure is to show the whole structure of the data base. This means that all the relation names currently in the data base and the corresponding data definition for each one will be shown. Figure 3 shows in a conceptual diagram the step followed by the procedure to accomplish this part.

3.2. Data Base Administrator Operations

3.2.1. Creation of a New Relation

This command allows the user to create a new relation for the data base. This task is always performed by the data administrator. At the moment this command has a security pass in order to prohibit access to unauthorized users. Figure 4 describes the steps involved in this command.

3.2.2. Deletion of a Relation

The delete command delete completely a relation from the data base. A special password is required of the user to perform this activity, which again should be performed by the data base administrator. The delete command appears under the M)odify Data Base command. This command erases the data file for the relation, the record from the relations file and the corresponding data definition file. Figure 5 describes this command.

3.3. Relational Operations

Relational operations are divided into one relation operations and two relations operations. The one relation operations involve projection, selection, and restriction. The two relations operations are cross product and join.

3.3.1. Projection

This relational operation is performed by following several steps. First, the relation desired is brought to main memory

and loaded to one of the working areas. The data in this working area is used as row data for a procedure that does the projection. The projection procedure selects the specified attribute in each tuple. These attribute values are transferred to the second working area. Before the transfer is done, a redundancy checking procedure over the correspondent attribute value is carried out, if the attribute value already exists in the workspace #2 then no transfer is done. Finally, all the attribute values selected are in the workspace #2. The user has the option of having this resulting projection sorted or not. This implementation of the projection operation is limited to the projection of one attribute values at a time. Figure 6 shows the steps for this operation.

3.3.2. Selection

The first part of this command is carried out like a projection. The relation is loaded into the first working area and then these data are taken to do the relational operation under the desired options and to pass the result to workspace #2 where the results of every operation are always stored. Figure 7 points out the different steps involved in this operation.

3.3.3. Restriction

This relational operation involves one attribute name and a corresponding domain value related by a relational operator. The first part of this operation is performed the same as the previous ones and the result is also placed in workspace # 2, and then printed out. Figure 8 represents the steps followed in this command.

```
LOAD Relations File
WHILE not end of Relations File DO
Begin
  PRINT Relation name
  LOAD Relational Data Definition
  PRINT Data Definition
End
```

Figure 3. Procedure that displays the Data base structure.

```
LOAD Relations File
PRINT Relation Names in the Data Base
ASK for password to create Relations
IF correct password THEN
Begin
  ASK for the name for the new Relation
  IF New Relation Name non-redundant THEN
  Begin
    Data Definition construction
    Insert first tuples for the new Relation
  End
  ELSE access denied
```

Figure 4. Steps involved in the creation of a new relation.

```
ASK for appropiate password
IF password correct THEN
Begin
  ASK for relation name to be deleted
  PURGE corresponding Relation File
  PURGE corresponding Data Definition File
  DELETE relation name from Relations File
End
ELSE delete denied
```

Figure 5. Delete relation steps.

```
LOAD Desired Relation into Workspace   1
ASK which attribute is to be projected
WHILE not end of Relation DO
Begin
  COPY attribute value selected
  IF attribute value copied exists in workspace
       2 then NO TRANSFER
  ELSE TRANSFER value to Workspace   2
End
IF sorting option true THEN
Begin
  SORT contents of Workspace   2
  PRINT Workspace   2 content
End
```

Figure 6. Steps involved in projection operation.

```
LOAD Desired relation into Workspace   1
LOAD Data Definition into Data Def. array 1
ASK for first attribute name, relational operator,
    and second attribute name
WHILE not  end of relation DO
Begin
  IF current tuple matches desired qualification THEN
  Begin
    TRANSFER current tuple to Workspace   2
  End
End
PRINT Workspace   2 content
```

Figure 7. Procedure followed to perform selection operation.

```
LOAD Desired relation into Workspace   1
LOAD Data Definition into Data Def. array 1
ASK for first attribute name, relational aperator,
    and attribute value
WHILE not end of relation DO
Begin
  IF current tuple matches desired qualification THEN
  Begin
    TRANSFER current tuple to Workspace   2
  End
End
PRINT Workspace   2 content
```

Figure 8. Procedure followed to perform restriction operation.

```
ASK for first Relation name
LOAD Relation into Workspace   1
LOAD Data Definition into Data Def. array 1
PRINT Data Definition for this relation
ASK for second Relation name
LOAD Relation into Workspace   2
LOAD Data Definition into Data Def. array 2
PRINT Data Definition for this relation
WHILE not end of First Relation DO
Begin
  WHILE not end of Second Relation DO
  Begin
    CONCATENATE to tuple from first Relation tuple
                from second Relation
    PRINT result of concatenation
  End
End
```

Figure 9. Cross product steps.

3.3.4. Cross Product

This relational operation involves two relations from the Data Base. The tuples from one relation are related to a second relation. Each tuple from the first relation is related to all the tuples in the second relation, in this way generating new tuples that are the concatenation of tuples from the first and second relation. Figure 9 shows the steps followed in this command.

3.3.5. Join

The relational operation Join also involves two relations from the Data Base. The tuples from the first relation are related to the second relation tuples by a selection type of relational operation. This operation is done taking the attributes values from one tuple (from Relation 1) and compared with the other tuple (from Relation 2). Whenever there is a success in the comparison, based on the operator chosen (=, <>, >=, etc.) a new tuple is generated. This new tuple is obtained from the concatenation of the tuples from both relations. Figure 10 represents the conceptual steps involved in obtaining the join of two relations.

3.4. Relations Maintenance

3.4.1. Insert

This commands allows the user to insert new tuples to an already created relation. The relation is loaded into the two workspaces at the same time. The insertion takes place in Workspace # 2. The tuple is inserted at the next available space on the workspace. Then the whole relation with the insertion is shown to the user. The user has the alternative of keeping this insertion or starting all over again. After user approval the relation is sorted by the key and saved into the diskette. Figure 11 shows the steps in this activity.

3.4.2. Delete

The delete command refers to tuple deletion. This is done by loading the relation into the two workspaces. The user should enter the qualification where the deletion should take place (attribute name, relational operator, and attribute value). If the user wants to delete one tuple, then the value should be the one corresponding to the desired tuple. The other way to delete more than one tuple would be to use the qualification (attribute name, relational operator, and attribute name). In

this case the attributes should be compatible.

The procedure does an exchange of the tuple attribute values by the word "TUPLE DELETED. " After searching the whole workspace # 2, the resulting modification is shown to the user, with the word "tuple deleted" where deletion took place. The user has the alternative of accepting or rejecting this result. After accepting the resulted information, tuples without the word "TUPLE DELETED" are put back into the relation file. Figure 12 describes the steps involved in this command.

3.4.3. Update

Update of a tuple is done in the following way. The user should enter the key value for the relation to be updated. This is done by means of the qualification (attribute name, relational operator, and key value). Then the system will search for this tuple in workspace # 2. If the tuple is found, then the system will show the whole relation content where the old tuple exists. Then the user has the chance to update the whole or some of the attribute values in this tuple. Following the update of the tuple, the system again shows the relation with the tuple updated. The user has the chance to approve this modification or reject it. If the user accepts the changes made, the relation will be saved. Figure 13 describes the steps for this command.

```
ASK for first Relation name
LOAD Relation into Workspace   1
LOAD Data Definition into Data Def. array 1
PRINT Data Definition for this relation
ASK for second Relation name
LOAD Relation into Workspace   2
LOAD Data Definition into Data Def. array 2
PRINT Data Definition for this relation
ASK for first attribute name from First Relation,
    relational operator, and
    second attribute name from Second Relation
WHILE not end of First Relation DO
Begin
  WHILE not end of Second Relation DO
  Begin
    GET attribute values from First Relation and Second
        Relation
    IF values match desired qualification THEN
    Begin
      CONCATENATE to Tuple from first Relation all
      attribute values of tuple from second Relation
    PRINT result of concatenation
    End
  End
End
```

Figure 10. Join operation steps.

```
LOAD Relation for insertion into Workspaces   1 &   2
LOAD Data Definition
WHILE not end insertion of tuples DO
Begin
  PROMPT lines for insertion
  ADD new tuple at the end of Workspace   2
End
SORT tuples in workspace   2 by the key
PRINT Relation with new tuples
IF affirmative answer to keep tuples THEN
Begin
  UPDATE number of tuples in Relations File
  UPDATE old Relation file in diskette
End
ELSE no insertion made
```

Figure 11. Steps involved in the insertion of tuples.

```
ASK for Relation Name
LOAD Relation into Workspaces   1 &   2
LOAD Data Definition into array 1
ASK for type of qualification to access tuple(s)
WHILE not end of relation DO
Begin
  IF tuple matches qualification THEN
  Begin
    REPLACE tuple for "TUPLE DELETED" string
  End
PRINT content Workspace   2
IF desired deletion THEN
Begin
  UPDATE number of tuples in Relations File
  UPDATE old Relation file in diskette
End
ELSE no deletion made
```

Figure 12. Steps to delete tuple(s) from a relation.

```
ASK for Relation name
LOAD Relation into Workspaces   1 &   2
LOAD Data Definition
Begin
  ASK for the tuple Key value
  SEARCH for desired tuple
  SHOW Old tuple
  ENTER new tuple
  REPLACE in workspace   2 old tuple for new one
  PRINT Workspace  2
  IF desired update THEN
  Begin
    TRANSFER Workspace   2 content to Relation File
  End
  ELSE no update made
End
```

Figure 13. Steps to update a tuple from a relation.

4. ERROR CHECKING MECHANISMS

The error checking mechanisms are present at different levels in the interaction with the system. The levels are the Data Base, the Query, the Relation, and the Interaction level. In case of some types of errors, the system responds to the user with an appropiate message. The interaction level error checking mechanisms appear in all levels and are mainly concerned with the inputs to commands and keywords to the system.

4.1. Data Base Level

The mechanisms provided at this level are put in effect when the user wants to add a new relation to the Data Base. The system will check for redundancy on the relation names. If the new relation name already exists in the data base, then the creation of the relation will be denied.

4.2. Relation Level

The error mechanisms at this level are associated with the common operations in the Data Base (tuple insertion, update,

and deletion). When the user wants to insert a new tuple, one of the first error checking mechanisms is the present number of tuples in the relation. If the tuple that is supposed to be inserted goes over the maximum number of tuples allowed, the insertion will be denied.

The other error checking mechanism appears when the actual insertion of attribute values for the new tuple is being performed. If the user inputs an attribute value that does not match the corresponding attribute domain, then the insertion is denied. The other important error checking mechanism is done to preserve the Relation integrity. This is the checking for redundant keys. If the user inputs a tuple with a key that already exists in the relation, the insertion is denied.

4.3. Query Level

The error checking mechanisms at this level correspond to the compatibility among attribute domains and values when the queries are constructed. The other error checking is using the information of the attribute value ranges. This is done when the value input for certain attribute it is not within the range that the attribute has in the data definition.

The attribute compatibility is done when the query uses two attributes, i.e. join or selection. If the attributes are not compatible in domain and length, the query is denied. The other

use of compatibility is made when a query is constructed where an attribute and a value of this attribute is placed in the query, i.e. restriction. If the value does not meet the domain and length compatibility with the attribute, the query is denied.

4.4. Interaction Level

These type of error checking mechanisms are present at the different levels of the system. There is an error checking when the user inputs the wrong attribute name and when he/she inputs a wrong relation name. When this happens, the system will give the user a chance to try the names again.

The other general error checking is done when certain answers are expected from the user, for example commands letters or Yes/No answers. The last error mechanism to mention is done at the query level; when relational operators (=,<=,<>,etc) are expected, then the system will give several chances to enter a correct relational operator.

5. SYSTEM LIMITATIONS

Limitations on the systems are caused by hardware and design objectives considerations. The limitations caused by the

hardware are those related to the size of workspaces in main memory. This implies that small relations can be brought to memory, in this case with no more than 10 tuples. The other important factor is the size of the program itself, about 1,700 lines; so there is not much space left in memory for the workspaces.

The design objectives were the development of an educational type of system that could help students studying data base management systems. The programming language used for the implementation of the Mini DBMS1 was UCSD PASCAL. The implementataion is based on the programming language facilities (data structures, built-in functions,etc) to simulate a relational data base activity. The important point was implementation of most of the relational operations. The amount of relations or tuples in them was not that important because of the type of environment for which the system was designed. The other consideration was the response time of the system to the query part or exercise of the relational operations. The system could be implemented using the diskette space for working areas, but this would lead to slower response time as a result of too much input/output activity.

The particular limitations of the system are:

* Relations cannot be more than 10 tuples each.

* Total number of relations allowed in the Data Base is 5.

* The length of each tuple cannot be more than 50 characters.

* Total number of attributes per relation is 5.

* The data types available for domain definition are strings and positive integers.

* Because of the size of the CRT (80 Characters) when a two-relations operation is performed, like Join or Cross Product, and the sum of the tuple sizes is more than 80, then the result in the last column of the CRT will be confusing. This will happen if the user is running the system in the TERAK machine; however, this will not happen if a bigger CRT or a hardcopy terminal is available.

6. CONCLUSION

This software package has been used by students in the CS482 (Spring 1980) course. They were the ones that really tested the system and made suggestions for further development, discovering some omissions that the program had. The students were asked to evaluate the system and give their ideas about it as an educational tool in the area of Relational Data Bases. The feedback obtained from the students was very interesting for the refinement of the program. Many of them found the

system helpful to understand the data base concept and exercise some of the operations in a data base, even though the relations were small.

This project has given me a very good exposure to the development of a considerable sized piece of software. This has been the result of almost a year of work, from the design of the system to the implementation and testing.

Expansions of the system can be made if a bigger minicomputer is available. This will result in bigger workspaces so that larger relations can be handled. On the other hand, if there is interest in expanding the system for real applications, then the use of bigger disks may be the way to go. For this kind of approach the basic procedures and functions developed in this program can be used. Of course, one has to consider the modification of the data structures, but the procedures for the realization of the relational operations and other utility functions can be used without modifying too much.

The other further development would be the use of this system with a CAI (Computer Aided Instruction) program in the Data Base area. The DBMS1 system can be used as a support media for the examples of relational operations and common operations in data base systems.

BIBLIOGRAPHY

* Astrahan M.M.,Blasgen M.W., et al., "SYSTEM R: Relational Approach to Database Management", ACM Transactions on Database Systems, Vol 1, #2 June 1976.

* Banerjee J. & Hsiao D.K., "Performance Study of a Database Machine in Supporting Relational Databases", Department of Computer Science, Ohio State University, 1978.

* Bernstein P.A., "Synthesizing Third Normal Form Relations from Functional Dependencies", ACM Transactions on Database Systems , Vol 1, #4, December 1976.

* Burkhard M.,"MINISEQUEL: Relational Data Management System", from "Database:Improving Usability and Responsiveness", Editor B. Shneiderman, Academic Press, 1978.

* Codd E.F., " A Relational Model of Data for Large Shared Data Banks", Communications of ACM 13, June 1970.

* Chamberlin D.D.,Astrahan M.M., et al, "SEQUEL 2: A Unified Approach to Data Definition, Manipulation and Control", IBM Journal of Resesearch and Development, November 1976.

* Gagle M.,Koehler G. & Whinston A., " Data Base Systems and Micro- Computers: an overview", Krannert Graduate School of Management, Purdue University, August 1979.

* Institute for Information Systems, "UCSD Pascal Release I.4 Manual", University of California at San Diego, January 1978.

* Kroenke D.,"Database Processing", SRA Inc.,1977.

* Martin James,"Computer Data Base Organization", Prentice Hall 1975.

* Schmidt J.W.," Some High Level Language Constructs for Data of Type Relation", ACM Trans D.B.,Vol 2, #3, September 1977.

* Tsichritzis D. & Lochovsky F.," Data Base Management Systems" Academic Press 1977.

* Wirth N. & Jensen K.,"PASCAL: User Manual & Report", Springer-Verlag, 1974.