(500)

# DESIGN OF A RELATIONAL DATABASE AND A QUERY LANGUAGE

## AND IMPLEMENTATION OF ITS DML

Dissertation Submitted in partial fulfilment of the requirements
for the award of the degree of
**Master of Philosophy**

**Anjana Sahai**

SCHOOL OF COMPUTER AND SYSTEMS SCIENCES
JAWAHARLAL NEHRU UNIVERSITY
NEW DELHI-110067
1983

DEDICATED TO

MY PARENTS

SCHOOL OF COMPUTER AND SYSTEMS SCIENCES

JAWAHARLAL NEHRU UNIVERSITY

NEW DELHI - 110067

The research work embodied in this dissertation has been carried out at the School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi-110067.

This work is original and has not been submitted in part or full for any other degree or diploma of any other university.

( Anjana Sahai )
Student

N.P. Mukherjee

(Prof. N.P. Mukherjee)
Dean

(Mr. G.V. Singh )
Supervisor

## ACKNOWLEDGEMENT

I wish to thank Mr. G.V. Singh, Assistant Professor School of Computer and Systems Sciences, for his help, guidance, valuable suggestions and cooperation throughout the course of this work.

I am indebted to Professor Dilip K. Banerji, Dean of the School for his help and for providing the necessary facilities.

I also wish to thank Mrs. Indu Singh for her cooperation.

I take this opportunity to thank Mr. R.N. Singh, my project partner, for his cooperation and friendship. I also thank my friends and seniors Mr. A.K. Bansal, Mr. Baijnath, Ms. Hemlatha, Mr. Hawa Singh and Mr. S.K. Sinha for their help from time to time. Thanks are also due to Mr. S.K. Dhanawat through whose 'Word Processor' I took the listings of my programs.

Thanks are also due to office staff for their cooperation and finally, I thank Mr. S.K. Sapra for his careful and painstaking typing of the manuscript.

## REFERENCES


1. MARTIN J. — Computer Data Base Organization.
2. — Infotech state of the Art Report Data Base Technology Vol. 1.
3. ULLMAN J.D. — Principles of Data Base Systems.
4. DATE C.J. — An Introduction to Database Systems.
5. — Infotech State of the Art Report : On-line Data Bases Vol.1 and Vol.2.
6. CODD E.F. — "A Relational Model of Data for Large Shared Data Banks." CACM, Vol.13, No.6, June 1970.
7. CODD E.F. — "Further Normalization of the Relational Model." Courant Computer Science Simposium 6, Data Base Systems, New York, Prentice-Hall, New York, 1971.
8. REISNER P. — "Human Factors Studies of Data Base Query Languages." ACM Computing Surveys Vol.13, No.1, March 1981.
9. — "End User Query Language" Infotech State of the Art Report Series 9, No.8.
10. CHAMBERLAIN D.D., ASTRAHAN M.M., ESWARAN K.P., GRIFFITHS P.P. LORLE R.A., MEHL J.W., REISNER P. and WADE B.W. — "A Unified Approach to Data Definition, Manipulation and Control", IBM Journal (R and D).
11. — HP 1000 System Manual.
12. ZLOOF M.M. — "Query By Example", Proc. NCC 44 (May 1975).
13. CHAMBERLAIN D.D., and BOYCE R.F. — "SEQUEL:- Structured English Query Language." Proc. 1974 ACM SIGMOD.
14. ASTRAHAN M.M. et al — "System R: A Relational Approach to Data Base Management", ACM Transactions on Data Base Systems 1, 97 (June 1976).
15. DONOVAN and MADNICK, Operating Systems.

CONTENTS


| | | PAGE |
|---|---|---|
| CHAPTER 1 | INTRODUCTION - A REVIEW | |
| 1.1 | INTRODUCTION TO THE PROBLEM | 1 |
| 1.2 | AN OVERVIEW OF DATABASE SYSTEM | 3 |
| 1.3 | THE DATABASE MANAGEMENT SYSTEM | 7 |
| 1.4 | SECURITY AND INTEGRITY | 8 |
| 1.5 | DATA MODELS | 15 |
| 1.6 | COMPARISON OF THE THREE MODELS | 28 |
| 1.7 | DATABASE SUBLANGUAGE | 35 |
| CHAPTER 2 | THE DATABASE | |
| 2.1 | DEFINITIONS | 39 |
| 2.2 | DESIGN OF A CANONICAL SCHEMA | 42 |
| 2.3 | THE PROPOSED CONCEPTUAL SCHEMA FOR THE LIBRARY DATABASE | 46 |
| 2.4 | DESCRIPTION OF THE DIRECTORIES | 63 |
| CHAPTER 3 | THE QUERY LANGUAGE AND ITS PROCESSOR | |
| 3.1 | LANGUAGE FACILITIES | 74 |
| 3.2 | THE QUERY PROCESSOR | 83 |
| APPENDIX 1 | SYNTAX OF THE QUERY LANGUAGE | 86 |
| APPENDIX 2 | DATA FORMATS IN HP1000 AND THE HASHING PROGRAM FLOWCHART | 91 |
| APPENDIX 3 | HP1000 FILE MANAGEMENT SYSTEM | 93 |
| APPENDIX 4 | SOME SAMPLE RESULTS | 101 |

# CHAPTER - 1

## INTRODUCTION - A REVIEW

### 1.1 INTRODUCTION TO THE PROBLEM

The need to organize data of any enterprise in the form of a data base has been felt long ago. Its advantages over the simple filing system are well known in terms of the amount of redundancy that can be reduced, the problems of inconsistency in the stored data that can be avoided, the stored data that can be shared, security and integrity constraints that can be applied and last but not the least, the data independence that can be achieved.

Three of the well known models for the logical design of the data-base are, the heirarchical model, the network model and the relational model. Of these, the third one is a comparitively new model, which is easy to understand and offers advantages over its predecessors.

Handling data about the vast number of books, journals and other kinds of literature in the library is a difficult task which needs to be done efficiently, to provide a quick response to the user who wants some information about a book or a journal etc. This information could be, to name a few, about the status of that book or journal in the library or about its contents. Also, the librarian and his staff of helpers must maintain records about all the different kinds of literature and for this he will require information which is of interest to him, but which, if retreived manually may take a very long time. He will also need to constantly update

his records to reflect new additions which continually take place, and deletions in case of loss or damage. There are also other kinds of records that he must keep eg. to name a few, about the literature recommended by different faculties, whether the recommendation has been accepted, details of orders placed, orders received and information about the suppliers with whom the orders are placed. The details of the kind of information which is maintained in the library are given in Chapter 2. The conceptual schema (explained later) of the library database of this project has been implemented as a Relational database. The relations of the database are described in Chapter 2. The proposed database has been implemented on the HP 1000 system at the JNU Computer Centre. It was our endeavour to implement a library database on it which responds to a query language having query, manipulation, data definition and data control facilities in it.

The library database was implemented jointly with my colleague Mr. R.N. Singh. Some portions of the project were commonly done, while others were done separately.

The common portions were :

1. Design of the logical schema
2. Design of the syntax of the query language used.
3. Implementation of the lexical analyzer.
4. The hashing technique.

The parts of the project which I have implemented are :

Syntax checking and query processor implementation of the following statements, which have been described in detail in

Chapter 3 :

(i) CREATE R($A_1$ Type $[S_1][L_1]$ $A_2$ type $[S_2][L_2]$ $A_3$ type $[S_3][L_3]$ ... An type $[S_n][L_n]$) ∅

(ii) SELECT $A_1$, $A_2$ ... An FROM R WHERE $A_s=[V_s]$ $A_t=[V_t]$ ... $A_z=[V_z]$ ∅

(iii) SELECT $A_1$ ... An FROM $R_1$ $R_2$ WHERE $A_1$, $A_x = R_2$, $A_x$ ∅

(iv) SELECT $A_1$ ... An FROM R WHERE PART $A_z = [V]$ ∅

(v) SELECT $A_1$ ... An FROM R WHERE PART $A_x$ AND $A_y$ $= [V_1]$ AND $[V_2]$ ∅

(vi) SECOUN $A_1$ ... An FROM R WHERE $A_z = [V]$ ∅

The next section of this chapter gives general ideas about the overall database system with an emphasis on the relational model.

## 1.2 AN OVERVIEW OF A DATABASE SYSTEM

James Martin [1] describes a database as : "A database may be defined to be a collection of interrelated data stored together without harmful or unnecessary redundancy to serve multiple applications, the data are stored so that they are independent of programs which use the data; a common and controlled approach is used in adding new data and in modifying and retreiving existing data within the data base. The data is structured so as to provide a foundation for future application development."

One of the major objectives in the design of a data base system is the provision of data independence, which can be defined as the immunity of application programs to changes

in storage structure and access strategy. To achieve this objective, one of the most widely used architectures for a database system is the one which is divided into three general levels: internal, conceptual and external.

1.2.1 External schema : The information content of the database as viewed by a certain user is known as the external model i.e. to that user, the external model is the entire database. Each external model is defined by means of the external schema, which represents the particular views of data required by application programs and consists of an enumeration of types of entities, and relationships between these types of entities. It is assumed that all main types of data management can be represented, i.e. heirarchies networks and relations. An instance of external data is not stored but is materialized from data in the internal schema under control of the conceptual schema. In the DBTG network model, [4] external and conceptual sbhema are known as the subschema and schema, respectively. In the IMS heirachical model, these are known as the Program Specification Block (PSB) and logical DBD respectively. As per MERCZ [2] : Our interest is really focussed on the external schema, the particular user's views derived from the conceptual schema. Both the application programmer and the inter-active terminal user look at the information base through their own coloured window, i.e. each one's external schema.

In this sense, the external schema is at the same level of abstraction as the conceptual schema, however

there are senses in which an external scheme can be 'more abstract' than the conceptual scheme, as the data dealt with by a view may be constructable from the conceptual scheme but not actually be present in that scheme.

1.2.2 Conceptual schema : The entire information content of a database is known as the conceptual model. It is intended to be a view of the data as it really is rather than as individual users see it. The conceptual model is defined by means of the conceptual schema which consists of definitions of the different types of entities and their relationships and the ways in which the entities and relationships at this level of abstraction are expressed at the next lower level, i.e. the internal level. For data independence to be achieved, these definitions must not involve any considerations of storage structure or access strategy - they must be definitions of information content only. Thus there must be no reference to stored field representations, physical sequence, indexing, hash-addressing or any other storage/access details. The external schema (section 1.2.1) which are defined on the conceptual schema made truly independent in this way, will also be data independent. Some desirable properties of a conceptual schema as given by SENKO [2] are :

- Data independence - stability of user programs in the face of stored structure file organization changes.
- Logical level orthogonality - stability of user programs in the face of real world changes that do not concern his program.
- Canonical structure - a 'single' location for each

fact for control of data base integrity and security; simplicity for the maintenance programmer.

- Accessing language simplicity - 'non procedural', perhaps more similarities to natural language, but with easy rules.
- Faithful representation of real world relationships - no spurious relationships implied by presence of fields in a record.

Two more important properties as given by Date [4] are :

The conceptual view of data should be :

As simple as practically possible.

Should have a sound theoretical base.

1.2.3 Internal schema : Internal schema describes the internal model. The Internal model is a very low level representation of the entire database; it consists of multiple occurences of multiple types of stored records. The internal model is thus one remove away from the physical level, which deals in terms of blocks and bits for the data. The internal schema defines the various types of stored records and also specifies what indexes exist, how stored files are represented, what physical sequence the stored records are in, and so on. The overall structure of a database is as shown in the fig. 1.1

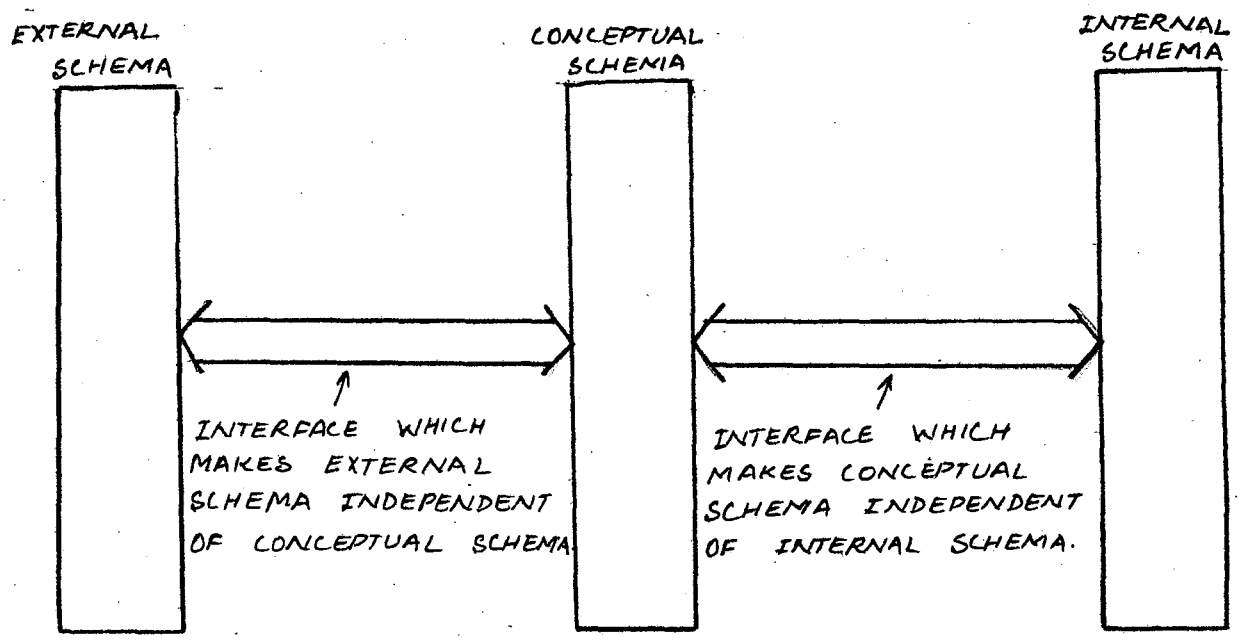



FIG. 1.1.

## 1.3 The Database Management System (DBMS) [3]

DBMS is the software that handles all access to the database. This software allows one or many persons to use and/or modify the database. A major role of the DBMS is to allow the user to deal with the data in abstract terms, rather than as the computer stores data. It allows the user to specify what must be done, with little or no attention on the user's part to the detailed algorithms or data representation used by the system. There are many other functions that can be carried by the DBMS, such as :

- Security - Not every user should have access to all the data.
- Integrity - Certain kinds of consistency constraints (i.e., required properties of the data) can be checked by the DBMS.

- Synchronization - Often many users are running programs that access the database at the same time. The DBMS should provide protection against inconsistencies that result from two simultaneous operations on a data item.
- Crash protection and recovery - There should be facilities to make regular backup copies of the database and to reconstruct the database after a hardware or software error.

1.4 This section describes in some more detail two of the functions of the DBMS enumerated above - (a) security, and (b) integrity.

1.4.1 Security : Security means the protection of data in the database against unauthorized disclosure, alteration or destruction [4]. The first security check that can be done is the identification and authentication of a user. The DBMS must not allow any operation to be performed on the database unless the user is authorized for the operation concerned. For this, the Data Base Administrator must define to the system the operations each user is allowed to perform and provide a means for the users to identify themselves to the system (for example by entering the user code at the terminal). The user must then authenticate his identification i.e. prove he really is, what he claims to be. This can be done by supplying a password which only that user and the system know, or by some kind of transformation which the user does on a random number supplied by the system.

The system then does the same transformation itself and checks the result with that which the user has supplied, to validate the authentication. This method is quite foolproof because it is difficult for an unauthorized user to perform the transformation on another random number generated by the system even if he knows the begining and end numbers of some transformation carried out by a valid user.

The identification and authentication carried out by the DBMS along with the operations a particular user is allowed to perform is known as the user profile. In practice the user profile alone may not suffice to ensure complete security, so it may become necessary for the system to check a password supplied with the request (for example), or to interrogate some other part of the database to see whether access should be granted. In general, the DBMS must check each operation as it is issued to see whether it violates any security restrictions, and must suppress it if so. In particularly sensitive situations, the system may react to an attempted breach of security by cancelling the program or locking the terminal and also recording the attempted security violation in a special log file. This will permit subsequent analysis of such attempts and may in itself act as a deterrent against illegal infiltration.

In general, an access constraint may refer to any data whatsoever in the entire database. There can thus be a highly priveleged and protected program called the 'arbiter' (part of DBMS), and used by Date [4], whose function

is to check each user request for access and to grant or deny permission, as appropriate, and which has unconstrained access to the entire database. On the basis of the combination, of user identification and operation requested, the arbiter will go through a series of tests to determine whether to grant or deny access.

In some of the relational systems eg. System R authorization mechanism is based on the concept of view. In this, a user cannot define a view involving those attributes which he is prohibited to read. In addition a more sophisticated security scheme is provided by the grant operation performed by the data control facility of SEQUEL, the language supported by system R.

To ensure against the danger of the DBMS being bypassed eg. in case of theft of the disc pack, sensitive data can be scrambled using some scrambling techniques (also known as encryption and privacy transformation). These could be, for example, simply shuffling the characters of each tuple (or record or message ...) into a different order, or replacing the characters of a tuple by characters of the same, or a different alphabet.

1.4.2 Integrity : [4], [5] maintaining the integrity of the database can be viewed as protecting the data against invalid (as opposed to illegal) alteration or destruction, and thus ensuring that the data is accurate at all times. Integrity can be maintained in single-user systems by means of a set of integrity constraints. In multiple-user systems, in

addition to these constraints, data must be shared in such a way that the integrity is preserved. Some of the Integrity Constraints are mentioned below.

- The primary key of any relation must be unique and there should be no null attributes in it.
- If the relation concerned is in 3rd normal form, integrity is maintained via the functional dependencies.
- The constraint on the value of one attribute which is determined by the value of another attribute, must be specified to the system by comparison expressions eg. in the library database the due date for a book issued must always be 15 days ahead of the issuedate.
- The values of an attribute must lie within certain 'bounds', which must be specified.
- There may be a very small set of values permitted for an attribute eg. the values of 'STATUS' attribute can be only INLIB,LOST or ISSUED.
- Values of an attribute may have to conform to a certain format.
- The set of values of an attribute may have to satisfy some 'statistical' constraint.
- The set of values appearing in a particular column within a relation may be required to be the same as, or a subset of, the set of values appearing in the same or different relation eg. 'ACCNO' value for a

book which is issued and appears in ISSUEFILE (fig. 2.4) must be present in the relation BOOKINLIB and the 'STATUS' corresponding to that 'ACCNO' in the BOOKINLIB relation must be set to 'ISSUED'. Also, it should be ensured that only one value of 'ACCNO' should correspond to a particular 'SERIALNO' of a 'CARDNO' in the relation ISSUEFILE.

- Static constraints are those that specify conditions which must hold for every given state of database. There are other constraints which must hold after a transaction has taken place i.e. the new value at the end of a transaction must be in some fixed relation to the old value. A transaction can be defined as a unit of work which is atomic. It consists of a number of input messages and their corresponding responses (output messages) together with the CPU work and data base accesses required to process the information provided in the input messages. During the execution of a transaction the database may not be in a stable state, but it must be so before and after the transaction. This implies that certain constraints must not be imposed during a transaction, but only on its completion. Such constraints are known as deferred constraints and must be specified to the system. By contrast, constraints that are enforced continuously are known as immediate constraints.

In system R, integrity constraints are provided by the assertion statements in SEQUEL 2.

In case of data being shared by a number of users, there are a number of inter-actions which may effect data base integrity these are :

- Contention: The situation where more than one transaction is attempting at the same point in time to update the same units of data, thus leading to inconsistencies. The method to overcome this is by providing a lock for the records that are to be updated by a particular user. But this solution may lead to the problem of deadlock or deadly embrace [15] This can be overcome by various methods which either preempt a deadlock from occuring or recover after a deadlock has occured. It is more economic to use the latter strategy because deadlocks are comparitively rare.
- Content consistency: This is the concept of a series of consistent updates such that a change to one data unit requires that one or more other units must also be changed.
- Time consistency: This is the ability to provide a snapshot of a data base which is being continually updated, so that reports correct at a given period in time can be produced from it.
- Resilience: This is the ability of the data base to recover without loss of stored data in the event of

any failure of the system or the data base. To provide resilience, back up and recovery facilities are required. There are three distinct configurations to provide these :

(a) Generation : In this configuration, transactions are combined with one physical version of a file (the father) to produce a new physically seperate version (son). Both the versions, and in some cases versions earlier than the father are also kept. In case of a failure, the file is recreated from the old version and the transactions. In the event of minor failures, when reprocessing the complete batch of transactions may be expensive, various checkpoints are introduced at which the status of the system is recorded on a log file and generally a back-up to the most recent checkpoint is enough for recovery.

(b) Dumping plus logging: In this, the database is copied or dumped at appropriate intervals on to some suitable physical medium, such as the magnetic tape, which is retained. The transactions that were processed after the dump are also stored in a log file or a journal. In the event of a failure the database is restored from the last dumped copy and the transactions processed since the last dump are reprocessed. In another method, no update-in-place is made to the database between dumps, instead the changes are maintained on a seperate file called the 'change' file or 'differential file'. At regular

intervals, the database and the change file are merged to produce the new database.

(c) Duplication : Two identical copies of the data base are maintained and are updated-in-parallel; every time a record is updated it must be written back to both copies. This approach provides high availability as, if hardware fault occurs to one copy, the other copy is immediately available.

## 1.5 DATA MODELS :

This section deals with the representation of the conceptual schema. Three most commonly used data models and their merits and demerits are discussed, and the Relational model is taken up in detail.

1.5.1 Heirarchical model : In this model the data is represented by a simple tree structure with one kind of record type superior to another kind. The basic definitions of trees apply to this model, thus the record type which has no parent is known as the root and it may have zero or more dependents and each of these may have any number of lower-level dependents, and so on, to any number of levels. Each dependent can have at most one parent or superior. A file of this model will contain several types of record, not just one; and it also contains links connecting occurences of these records. Any given record occurence takes on its full meaning only when seen in context - indeed, no dependent record occurence can even exist without its

superior. This introduces an assymetry in the model and infact becomes a drawback. Fig. 1.2 is an example of a hierarchical structure.

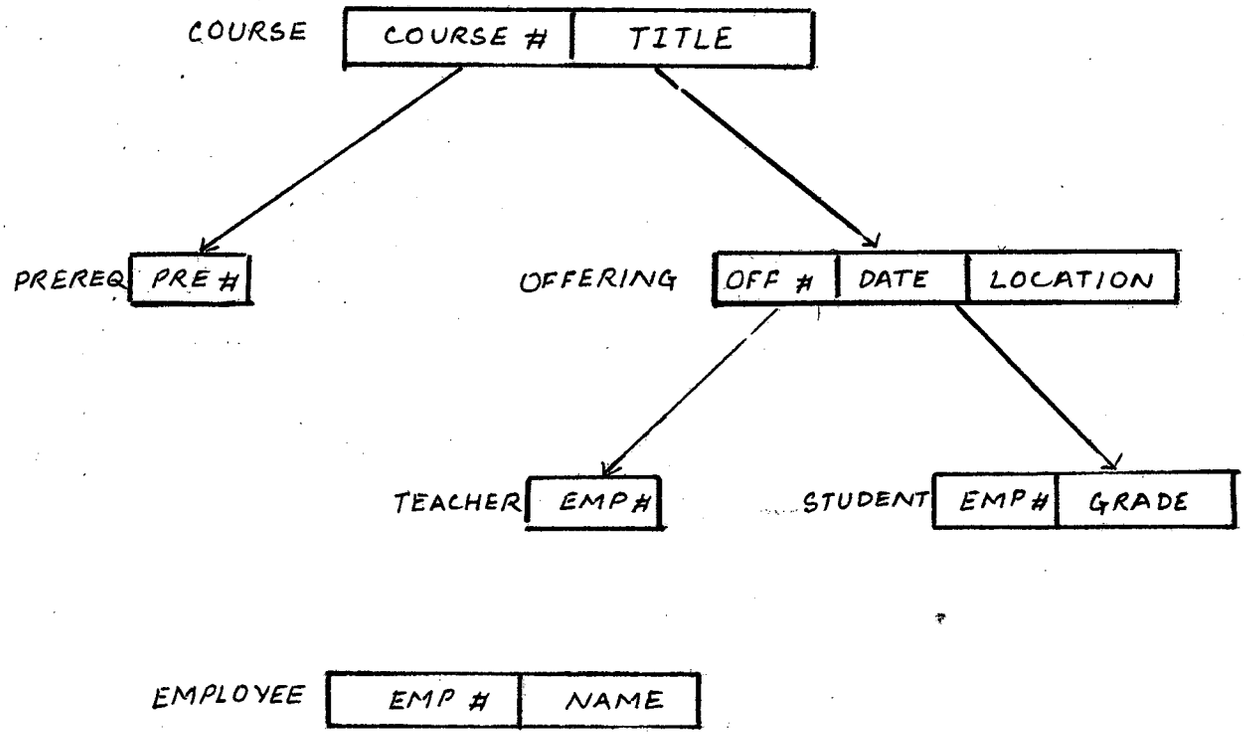



FIG. 1·2 HEIRARCHICAL VIEW OF AN EDUCATION DATABASE.

1.5.2 The Network Model : In this, as in the heirarchical approach, the data is represented by records and links. However, a network is more general in the sense that any record occurence may have any number of immediate superiors and of cource, as in the previous case, any number of immediate dependents. It thus allows to model the many-to-many correspondence more directly than does the heirarchical

model. In addition to the different record types representing the data itself, there is another type of record known as the connector which represents the associations between the different types of data, and contains data describing that association. All connector occurences for a given data type occurence are placed on a chain starting at and ending at that occurence. Each connector occurence may occur again on many chains, thus in case of certain retrieval queries a complication of deciding which connector chain to use is added. The adjustment of the connector chain must be done whenever any record occurence is deleted or a new occurence inserted. Figure 1.3 is a representation for a network model:

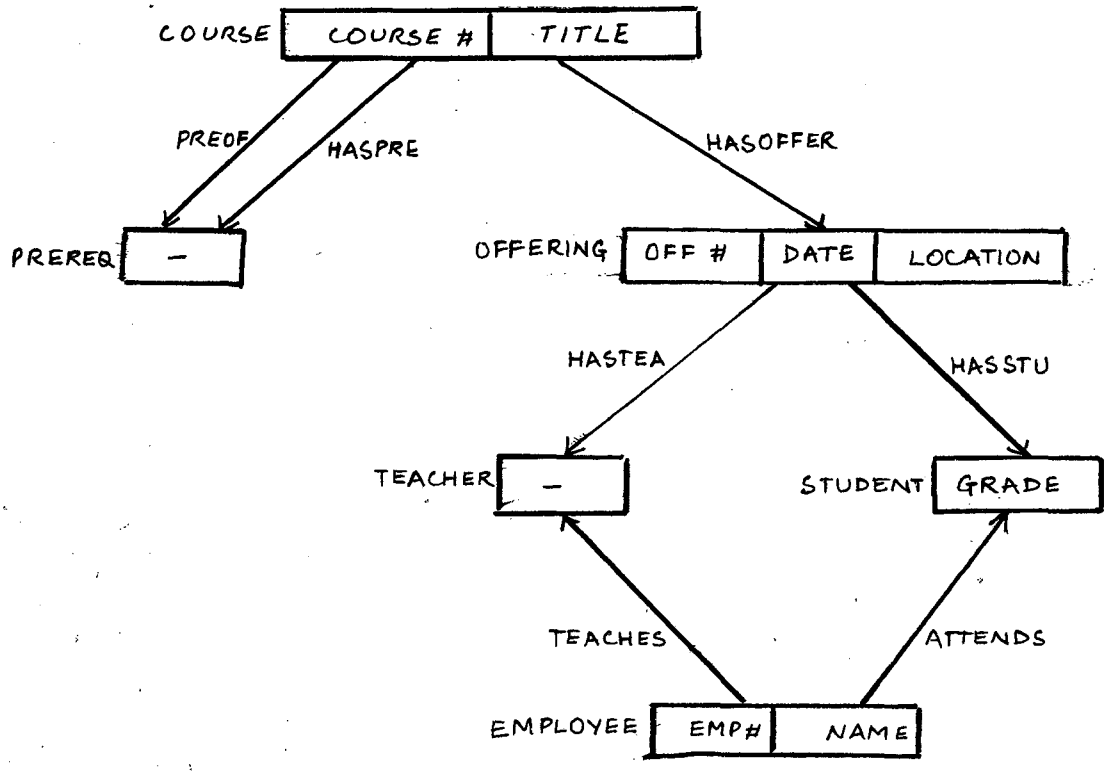



FIG. 1.3. NETWORK VIEW OF THE EDUCATION DATABASE.

1.5.3 The Relational model : In this model, data is arranged in the form of tables called relations. Much of the mathematical concepts of Relation Theory apply to these tables each of which is in fact a special case of the construct relation in mathematics. Rows of such tables are referred to as 'tuples'. Columns are referred to as 'attributes'. A 'domain' is a pool of values from which the actual values appearing in a column are drawn. Different columns in a relation can be drawn from different or same domains. A crucial feature of relational data structure is that associations between tuples are represented solely by data values in columns drawn from a common domain. In fact, all information in a database - both 'entities' and 'associations' is represented in a single uniform manner, namely in the form of tables. This characterestic is not shared by the hierarchical and network approaches, in which associations are represented by means of constructs, like links.
As per HAMMER [8], A set of high level operators can be applied to a relational database; these include projection (selection of certain column values from a set of tuples), restriction (selection of a set of tuples that satisfy a predicate), and join (a version of cross-product that essentially performs inter-tuple correlation). The relational model is simple, uniform, and flexible. The absence of intricate structuring mechanisms, and the uniformity of representation, makes the definition and understanding of the relational database a simple task. Furthermore, the

spartan nature of the database description means that it imposes little constraint on the way that a user is able to interpret and utilize the data. There are no complex tree or network structures that coerce all users into a particular limited view of the relationships between records.

Some mathematical definitions on which the relational database is based are given below :

Definitions [4]: Given a collection of sets $D_1$, $D_2$ ..., $D_n$ (not necessarily distinct), R is a 'relation' on these n sets if it is a set of ordered n tuples ($d_1$, $d_2$ ... $d_n$) such that $d_1$ belongs to $D_1$, $d_2$ belongs to $D_2$, ... $d_n$ belongs to $D_n$. Sets $D_1$, $D_2$ ... , $D_n$ are the domains of R. The value n is the degree of R. The number of tuples in a relation is called the cardinality of the relation. Relations of degree one are said to be unary, of degree two, binary and so on. A relation of degree n is said to be n-ary. Strictly speaking, there is no ordering defined among the tuples of a relation, since a relation is a set and sets are not ordered. However, we do order the tuples for retrieval purposes in, for example, the ascending order of the value of some attribute. Mathematically speaking since a relation is a set of ordered n-tuples, a rearrangement of the columns of a relation would result in a different relation, but since users normally refer to columns by name rather than their position in an ordering, this restriction is relaxed, and column order is treated as irrelevant. Each relation must have at least a single attribute or a combination of attributes which uniquely identifies the tuples of the

relation and called a 'primary key' of the relation. This is obviously true since even if there is no subset of the combination of attributes which has is property, all the attributes together will always identify the tuples uniquely. In case there is more than one such combination of attributes, which has the unique identification property, the combinations are called 'candidate keys'. Any one of the candidate keys is chosen as the primary key, the criteria often being that no attribute of the primary key should be undefined at any time in any tuple.

An attribute of relation R1 is a 'foreign key' if it is not the primary key of R1 but its values are the values of the primary key of some relation R2. A relational model of a database can thus be defined as a collection of time-varying, normalized relations of assorted degrees.

The idea of three normal forms of relations was introduced by E.F. Codd through his papers [6] and [7].

1st Normal Form

The only relations permitted in the relational model are those that satisfy the following conditions -

- Every value in the relations - i.e., each attribute value in each tuple-is atomic (nondecomposable). In other words at every row-and-column position in the relation there exists precisely one value, never a set of values. When this condition is satisfied, the relation is said to be in 1st Normal Form. So all relations which are unnormalized must be cast into the

1st NF (Normal form) before being included in the database and before further normalization is done.

The objectives of further normalization are [7]:

1. To free the collection of relations from undesirable insertion, update and deletion dependencies.
2. To reduce the need for restructuring the collection of relations as new types of data are introduced, and thus increase the life span of application programs.
3. To make the relational model more informative to the users.
4. To make the collection of relations neutral to the query statistics, where these statistics are liable to change as time goes by.

Functional dependency : [7], [4].

Attribute B of relation R is functionally dependent on attribute A of R if, at every instant of time, each value in A has no more than one value in B associated with it under R. We write R. A $\rightarrow$ R. B if B is functionally dependent on A in R, and R.A $\not\rightarrow$ R.B if B is not functionally dependent on A in R. If both R.A $\rightarrow$ R.B and R.B. $\rightarrow$ R.A hold, then at all times R.A and R.B are in one-to-one correspondence, and we write R.A $\leftrightarrow$ R.B. The notion of functional dependence can be extended to cover the case where both A and B are composite attributes. Attribute B is 'fully functionally dependent' on attribute A if it is functionally dependent on A and not functionally dependent on any subset of the attributes of A. (A must be composite).

Prime Attributes : Any attribute of relation R which participates in at least one candidate key is called a prime attribute of R. All other attributes of R are called non-prime.

With these concepts we define the 2nd Normal Form.

2nd Normal Form [7]

A relation R is in second normal form if it is in first normal form and every non-prime attribute of R is fully functionally dependent on each candidate key of R. Although each prime attribute is dependent on each candidate key of which it is a component, it is possible for a prime attribute to be non-fully dependent on a candidate key of which it is not a component.

To reduce relations which are in 1st Normal Form into an equivalent collection of 2nd N.F. relations, the relations are replaced by suitable projections such that there are no non-full dependencies in the projections. No information is lost during this process because a join of all the projections will produce the original relation again. Any information that can be derived from the original structure can also be derived from the new structure, but the converse is not true. The new structure may contain information that could not be represented in the original. In this sense, the new structure is a slightly more faithful representation of the real world.

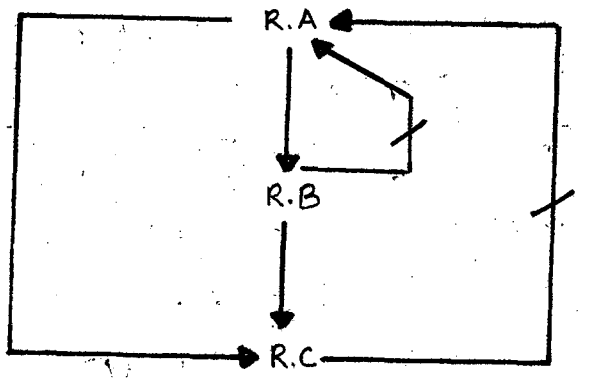
Transitive dependence [7] Suppose that A,B,C are three distinct collections of attributes of a relation R (hence R is of degree three or more), suppose that all three of the following time - independent conditions hold :

$$R.A \longrightarrow R.B \quad , \quad R.B \not\longrightarrow R.A \quad , \quad R.B \longrightarrow R.C$$

From these the following two conditions are deduced to hold :

$$R.A \longrightarrow R.C \quad , \quad R.C \not\longrightarrow R.A$$

and the entire set of relations can be represented by the following diagram. Note that $R.C \longrightarrow R.B$ is neither prohibited nor required.




In such a case, C is said to be transitively dependent on A under R. In the special case when $R.B \longrightarrow R.C$ also holds, both B and C are transitively dependent on A under R.

3rd Normal Form [7] A relation R is in third normal form if it is in second normal form and every non-prime attribute of R is non-transitively dependent on each candidate key of R.

Again, relations in 2 NF can be reduced into an equivalent collection of 3 NF relations by taking suitable

projections on the relations in 2 NF such that there are no transitive dependencies in the new relations. No information is lost because a join of the new relations will produce the original relation again i.e. the process is reversible, however the new structure may contain that information which could not be represented in the 2 NF relations structure, thus the new structure is a better representation than the 2 NF relations structure.

It is possible to give a definition of third normal form that makes no reference to the first or second normal forms as such, nor to the concepts of full and transitive dependencies. [4] This definition is useful because the first and 2nd normal forms are not important in themselves, but are only intermediate steps to obtain the more desirable 3rd or 4th Normal forms. An attribute (possibly composite) on which some other attribute is fully functionally dependent is called a 'determinant'. Then 3 NF can be defined as follows :

A normalized relation R is in third normal form (3 NF) if every determinant is a candidate key.

It happens sometimes, that though the relations are in 3 NF and cannot be reduced further according to the definitions given above for 3 NF, problems of deletion, insertion and update still remain. This happens because, through there are no transitive or non-full dependencies, another type of dependency, known as the multivalued dependency, still remains. This type of dependency models a m:m relationship

which cannot be modelled by the functional dependency. Multivalued dependency is said to exist, when, though a given attribute B of relation R is not functionally dependent on attribute A of relation R i.e. though each value of A does not have one and only one value of B associated with it, but each value of A does have a set of values of B associated with it. Functional dependence defined previously, is a special case of multivalued dependence. Problem arises with those 3 NF relations which involve multi-valued dependencies that are also not functional dependencies. The solution to this problem is to reduce such 3 NF relations by means of projections into an equivalent collection of relations which do not involve any such dependencies. There is no loss of information in this process and the new collection of relations show the M:M relationships which were hidden in the non reduced 3 NF relations.

4th Normal Form : [4] A normalized relation R is said to be in fourth normal form (4 NF) if and only if, whenever there exists a multivalued dependency in R, say of attribute B on attribute A, then all attributes of R are also functionally dependent on A. Figure 1.4 shows the process of conversion of an unnormalized relation to the 4th NF :

| S # | STATUS | CITY | P # | QTY |
|---|---|---|---|---|
| S1 | 20 | LONDON | P1<br>P2<br>P3<br>P4 | 300<br>200<br>400<br>200 |
| S2 | 10 | PARIS | P1<br>P2 | 300<br>400 |
| S3 | 10 | PARIS | P2 | 200 |

FIG. 1.4 a UNORMALIZED RELATION — NON-ATOMIC VALUES.

WHEN CONVERTED TO FIRST NORMAL FORM:

| S# | STATUS | CITY | P# | QTY |
|---|---|---|---|---|
| S1 | 20 | LONDON | P1 | 300 |
| S1 | 20 | LONDON | P2 | 200 |
| S1 | 20 | LONDON | P3 | 400 |
| S1 | 20 | LONDON | P4 | 200 |
| S2 | 10 | PARIS | P1 | 300 |
| S2 | 10 | PARIS | P2 | 400 |
| S3 | 10 | PARIS | P2 | 200 |

FIG 1.4 b FIRST NORMAL FORM — ATOMIC VALUES. BUT UPDATE, INSERTION, DELETION ANOMALIES.

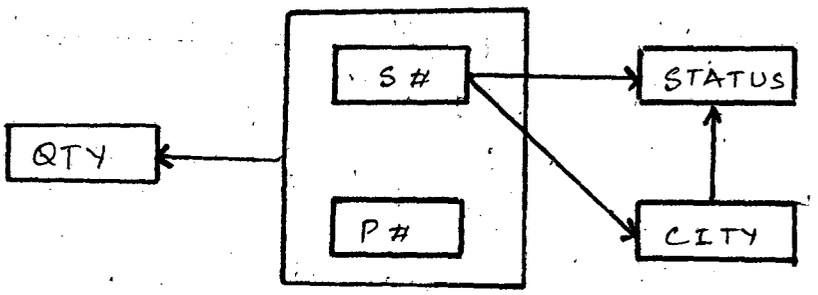



FIG 1.4 c FUNCTIONAL DEPENDENCIES IN FIRST NORMAL FORM.

WHEN CONVERTED TO 2NF, NON-FULL FUNCTIONAL DEPENDENCIES ARE REMOVED, THE 1NF RELATION IS BROKEN INTO RELATIONS SECOND AND SP:

| S# | STATUS | CITY |
|---|---|---|
| S1 | 20 | LONDON |
| S2 | 10 | PARIS |
| S3 | 10 | PARIS |
| S4 | 30 | ATHENS |

SECOND

| S# | P# | QTY |
|---|---|---|
| S1 | P1 | 300 |
| S1 | P2 | 200 |
| S1 | P3 | 400 |
| S1 | P4 | 200 |
| S2 | P1 | 300 |
| S2 | P2 | 400 |
| S3 | P2 | 200 |

SP

FIG 1.4 d SECOND NORMAL FORM — NO NON-FULL FUNCTIONAL DEPENDENCIES BUT STILL UPDATE, INSERTION, DELETION ANOMALIES IN RELATION SECOND. RELATION SP IS NOW IN THE DESIRED FORM i.e THE FOURTH NORMAL FORM.

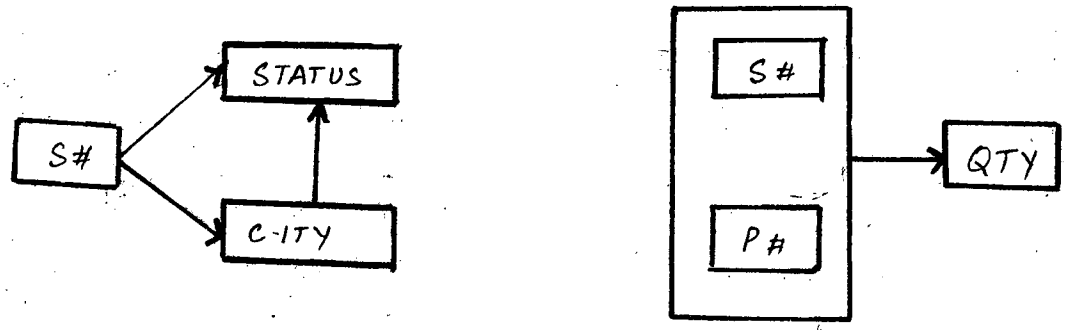



FIG 1·4 e FUNCTIONAL DEPENDENCIES IN RELATIONS SECOND AND SP.

WHEN CONVERTED TO 3NF, TRANSITIVE DEPENDENCIES ARE REMOVED, RELATION SECOND IS BROKEN INTO RELATIONS CS AND SC:

SC

| S# | CITY |
|---|---|
| S1 | LONDON |
| S2 | PARIS |
| S3 | PARIS |
| S4 | ATHENS |

CS

| CITY | STATUS |
|---|---|
| ATHENS | 30 |
| LONDON | 20 |
| PARIS | 10 |

FIG 1·4 f. THIRD NORMAL FORM — NO TRANSITIVE DEPENDENCIES AND NO UPDATE, INSERTION, DELETION ANOMALIES. RELATIONS SC AND CS ARE ALSO IN FOURTH NORMAL FORM AS THERE ARE NO MULTI-VALUED DEPENDENCIES.


FIG 1·4 g. FUNCTIONAL DEPENDENCIES IN RELATIONS SC AND CS.

1.6 Comparison of the three models : Although the hierarchical and network models have been used for a long time, it has been increasingly felt that the relational model has many important advantages over these two models. This section discusses these advantages which ultimately lead us to choose the relational model for our database design.

[6] The relational model is superior to the hierarchical or network models because it provides a means of describing data with its natural structure only-that is, without superimposing any additional structure for machine representation purposes. Accordingly, it provides a basis for a high level data language which will yield maximal independence between programs on the one hand and machine representation and organization of data on the other.

Some other advantages, listed by Martin [1] and Date [4]are

1. Ease of use : the easiest way to represent most data to users not trained in the techniques of data processing is with two dimensional tables.
2. Flexibility : Operations such as PROJECTION and JOIN permit cutting and pasting of relations so that the different logical files wanted by different application programmers can be given to them in the form they want them.
3. The directed links used in hierarchical and network structures can be misleading and may lead to wrong conclusions, specifically, to what E.F Codd calls a 'connection trap' [1].

4. Security controls can be more easily implemented. Security authorizations will relate to relations. Sensitive attributes could, for example, be moved into a separate relation with its own authorization controls. If authorization requirements are met, that attribute can be JOINed back to the original attributes.

5. Relatibility : The maximum flexibility is possible in relating attributes from different sets of tuples.

6. Ease of implementation : The physical storage of flat files which represent relations can be less complex than the physical storage of tree and plex structures.

7. Data independence : There will be need for most data bases to grow by adding new attributes and new relations. The data will be used in new ways. Tuples will be added and deleted. New data item types will be added and old ones deleted. New associations will be added. If the data base is in a normalized form in the software, the data can be restructured, and the data base can grow without, in most cases, forcing the re-writing of application programs. Good data independence can be achieved more easily with normalized logical structures than with tree or plex structures.

8. As per HAMMER [2] : The simplicity of a relational data base makes it susceptible to access by means of a very high level, non-procedural data manipulation language. Since data relationships are not expressed by intricate data structures through which the programmer must 'navigate', it is possible to design easy to use languages that allow

associative retrieval (i.e., by means of the contents of a data item, rather than by its position in a structure) and that do not limit the user to retrievals that utilize relationships expressed in the schema. Furthermore, the uniformity of data leads to the need of only one operator for each of the basic functions eg., delete, insert etc. This is not so in the network model. In DBTG data model, for example, at least two sets of operators are needed for the operation 'insert' -

(a) STORE, to create a record occurence & (b) CONNECT to create a link between the owner and its member.

9. Because of the asymmetry of record types in the hierarchical model, a query and its reverse are also asymmetric. This asymmetry occurs because different record types are treated as roots or children, i.e. one record type is superior to some other record type. This asymmetry in query, in turn leads to additional programming on the part of the application programmer and implies more complexity of programs which increases as the model becomes more complex with more kinds of records added to the database, and thus, more time is needed for debugging and more maintainence is required. Similarly the insert, delete and modify operations are more complicated. For example

(a) for insertion, a new record cannot be inserted without also inserting its dummy root if its record type does not already exist as child to some other root.

(b) for deletion : if the root record is deleted, all its dependents must also be deleted. It follows that if a

dependent, which is the only occurence of a record type, is deleted, we lose all information on that record type.

(c) for update : either all the records of the database should be searched to find every occurence of the record to be updated or an inconsistency might result.

10. In the network approach the complexity increases as a number of information-bearing constructs eg. record and links are used. And the more such constructs there are, the more operators are needed to handle them, and hence the data sublanguage becomes more complicated.

As per HAMMER 2 : Since inter-tuple associations in a Relational model are expressed by relationships between particular column values of the associated tuples, rather than with explicit structural links i.e. on the level of data manipulation rather than data definition the capacity for altenative logical views of the data base is provided by the schema, but no commitment is made to any particular one; no alterations are required for either the database definition or the application programs. Furthermore, there is no reflection at the user level of the implementation mechanisms used to represent a relation; a relation may be stored and accessed in a variety of ways, without impacting user programs except in terms of their performance. Thus, as mentioned before, relational data base provides its users with a high degree of both physical and logical independence. We will now take up in some detail the ideas of simplicity and need for a sound theoretic base for the conceptual schema as given in Sec. (1.2.2).

Simplicity :-

The conceptual view must be easy to understand and easy to manipulate. For easy understanding, the following properties are essential:

- The number of basic constructs should be small.
- Distinct concepts should be clearly seperated i.e. in a basic construct of the schema more than one concept should not be bundled.
- Symmetry should be preserved. This point has already been discussed before.
- Redundancy should be carefully controlled.

For easy manipulation the properties required are :

- The number of operator types should be small.
- Very high level operators should be available.

Theoretical Base : Because of its importance, the conceptual level must be founded on solid base of theory, because the model must behave in a totally predictable manner.

It is seen that the Relational approach measures quite well with the above requirements :
The number of basic constructs in it is only one, namely the relation or the table itself, whereas in network approach there are a number of basic constructs eg. in the DBTG model, the base set, fan set, ordering etc. Few cases of bundling ever occur in the relational approach, whereas in the network approach, the fan set usually supports a number of concepts. And as for symmetry and non redundancy, the relational model meets the requirements well, whereas in the network approach

asymmetry occurs in the form of owner and member record types. Non redundancy is achieved in relational model through normalization.

Relations are easy to manipulate, they support very high level operators, their number being quite small in any language, because only one operator is needed for each of the basic functions. In contrast, the network model requires more than one operator for a basic function eg. insert, to be performed. Symmetric exploitation is possible in the relational model because all information is represented in the same uniform way.

As for the question of underlying theory, the relational model is based on certain aspects of the mathematical set theory; it also possesses a theory for application to database itself, namely in the form of normalization theory discussed before. Relations also have a property of closure and the theory of relational completness provides a valuable tool for measuring the selective power of a language.

Looking at the advantages given, it is not to say that there are no drawbacks of the relational model. A drawback often cited is that of machine performance. As per HAMMER [2]: The potential penality to be paid for the simplicity, generality, and flexibility of a relational data base system is in its performance. Because the user expresses his transactions in terms of his logical view of the data, which is related to its physical representation in a way that may be unknown to him, it is impossible for the user to tune his programs to fit the physical data base structures; he has

lost control over the performance of the system and cannot explicity optimize his use of it.

The JOIN operation takes substantial machine time. It may be feasible with small relations, but commercial files may be hundreds of millions of bytes long. It should be noted that with interactive systems, it is unlikely that JOIN will ever take place physically infact the relations and operations on them, such as JOIN, are a logical view, and equivalent results are produced by pointer structures or indices. With batch-processing however, a physical JOIN may be permitted.

The physical layout of the relational database is generally chosen so as to give good performance to the most frequently run operations. The least frequently run operations suffer accordingly, but this is so in other database systems too. A relational database design is usually depicted as not being 'driven' by user views of data. A new unanticipated user view can be handled easily if the data it needs is stored. Although this is true in connection with the logical structure of the data, the new user view may not be handled with good machine performance because the physical structure of the data is usually designed to best serve the most common applications. The physical structure is user-'driven' even if the logical structure is not.

Another drawback is that efficient utilization of space sometimes can be easier using hierachies (or networks in the form of a hierarchy) rather than relations. While

it could be thought of representing relations as hierarchies, it is not always clear how to do so.

In conclusion, many of the apparent inefficiencies of the relational model can be eliminated. Optimization techniques can be used for relational data manipulation language that allow these languages to use time efficiently. Research aimed at producing good physical representations is also underway and progress has been made in IBM'S System R which is an experimental system for a relational database..

## 1.7 Database Sublanguage

A query language is a special purpose language for constructing queries to retrieve information from a database of information stored in a computer. It is usually intended to be used by people who are not professional programmers. Query languages are usually high level languages with a fairly limited number of functions. Query languages can differ from eachotherin their syntactic form eg., QBE [12] and SEQUEL [13] (later on known as SQL) and can differ in their overall structure, in that,QBE used a two-demensional syntax in which users write queries by filling in forms on a CRT screen. SEQUEL, on the other hand, uses a linear syntax, written in normal left-to-write, top-to-bottom fashion. There can, in turn, be variants of these, eg., in SQUARE, a precursor of SEQUEL, the query is written in a positional notation remniscent of mathematical subscripts or superscripts, together with the use of mathematical operations, which contrasts with the keyword notation used in SEQUEL [8] .

The query languages supported by relational database are many. Some languages eg. ISBL [1] are based on relational calculus, while languages eg. SEQUEL are based on an operation called a 'mapping' - the SELECT/FROM/WHERE block. SEQUEL was later revised to SEQUEL 2. [10] In this, some of the drawbacks found on the basis of experimental tests of learning in the original version were removed and it was used in system R. It can be used either as an interactive, stand alone language, or as embedded in a host language, such as PL/1. Specifically, in system R, SEQUEL 2 can be embedded in PL/1.

A language which is chosen to be supported by a database must, first of all, be relationally complete. A language is relationally complete [4] if any relation derivable from the data model by means of an expression of the relational calculus can be retrieved using a single statement of that language. In another way it can be said that for relational completness of a language, for a very large class of queries, the user will never need to use loops or branching in extracting the data required.

Relational algebra is said to be complete, and so is relational calculus. So to test for the completness of a language, it will suffice to compare it with either relational algebra or calculus. In the comparison, what must be done, is to see whether there exists, a single expression corresponding to each of the operators of the relational algebra or the relational calculus. In this sense the high level languages eg. DSL ALPHA, SQUARE, SEQUEL 2, QBE etc. are more than relationally complete since they generally have capabilities

beyond those of relational calculus. These capabilities include insertion, deletion and modification commands which are not part of relational algebra or calculus. Some additional features frequently available are :

1. Arithmetic capability.
2. Assignment or Print commands.
3. Aggregate functions, for example, average, sum, minimum or maximum can be applied to columns of a relation to obtain a single quantity.

Another desirable feature of the language is that it should be non-procedural. A non-procedural query [4] states merely what the result of the query is, not how to obtain it. Relational algebra, thus, is more procedural than Relational calculus or other languages like QBE, SEQUEL, because in it the user must specify the steps to be performed in order that a particular result may be obtained. A non-procedural language is considered better because the user does not have to spend time and effort in thinking out the process to obtain the results and this also allows the DBMS to follow the most efficient path for obtaining that result, it is not forced to follow the path (which might sometimes be inefficient) laid down by the user.

The language used for our database is like SEQUEL 2. There are many reasons for choosing this language.
A language like SEQUEL 2 is a non-procedural language. A language based on relational calculus which is also non-procedural was not chosen, because a sophisticated mathematical machinery of the predicate calculus (extra variables, quantifiers) is required to do even simple references to the

relations. As a result, in relational calculus, as the queries become more complex more variables and linking terms are required and the management of quanitifiers becomes more complex. A casual, non-programming user who will frequently require information from the library database can hardly be expected to write queries which require such mathematical sophistication.

For the comparison of ease-of-use between languages such as SQUARE, SEQUEL, QBE, many human factors studies were carried out. The results given here are taken from [8] and [9].

REISNER : The study of SQL ... was performed at the IBM Research laboratory in San Jose, California. Subjects : Five groups of students were taught. One group consisted of 18 programmers and the other of 15 non-programmers ... Two similar groups were taught SQUARE ... Experimenter's Conclusions : Reisner concludes that programmers were able to learn SQL 'more completely' than non-programmers. She also concludes that SQL was easier than SQUARE, but only for non-programmers.

Since the users of the library database would be non-programmers rather than programmers, a SEQUEL 2 based language was considered to be more appropriate.

Although in another study, [9] it is concluded about QBE,

Experimenter's conclusions : Greenblatt and Waxman concluded that QBE was superior to SQL in learning and application ease. But it was decided to use a language like SEQUEL 2 which has a linear syntax rather than QBE's 2-dimensional one and also, as stated in [9], there is no solid evidence that QBE is easier to learn.

# CHAPTER - 2

## THE DATABASE

In this chapter are described, the various steps performed to design the proposed model of the database. The resulting schema has been called the canonical schema in [1]. Before describing the actual method, a few definitions in context of the bubble chart are given

### 2.1 Definitions

(i) Canonical schema :- The canonical schema is defined [1] as a model of data which represents the inherent structure of that data and hence is independent of individual applications of the data and also of the software or hardware mechanisms which are employed in representing and using the data.

(ii) Bubble chart : a bubble chart is a graph consisting of directed links between data-item types. Note that the data item types is the same as the attributes, which is the name used in relational model. fig. 2.1 shows an example of a bubble chart :

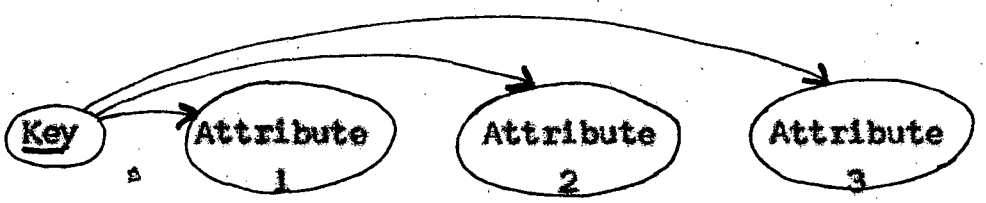



Fig. 2.1 : example of a bubble-chart.

The bubble chart is reduced to the 3rd normal form by ensuring full functional dependency of the attributes on the key attribute (or all the attributes of the concatenated key) and by removing transitive dependencies between attributes. Care is taken in deciding the transitive dependencies by understanding the role or meaning of each attribute carefully. Any M:M relationships between keys are also removed by the method given in fig. 2.2

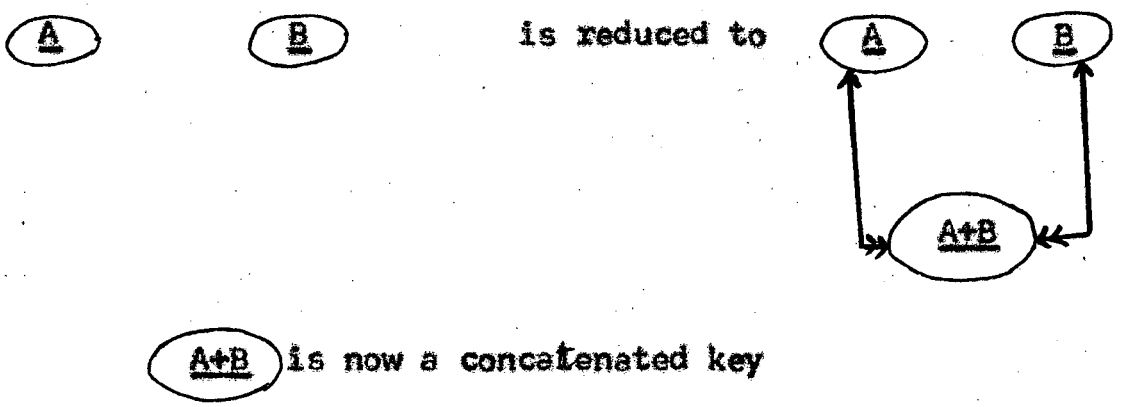



Fig. 2.2

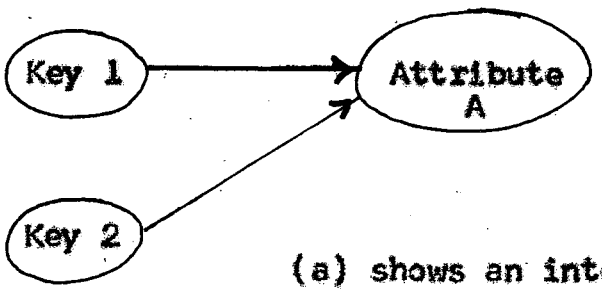
(iii) Primary key : Primary key in a bubble chart is a node with one or more single arrows leaving it. It is represented by underlining the attribute which is the primary key.

(iv) Attribute : An attribute is a node with no single arrows leaving it.

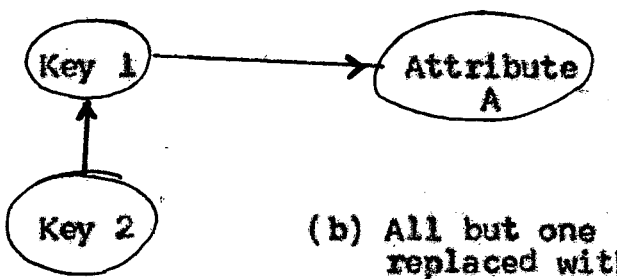
(v) Root key : is a primary key with no single arrows leaving it to another primary key.

(vi) Intersecting attribute : is an attribute which is attached to more than one key. There should be no intersecting

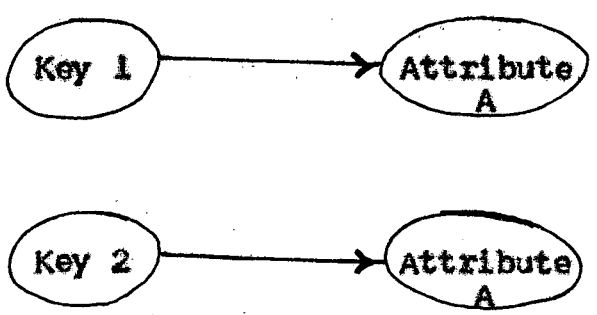
attributes on the final canconical graph, the methods of dealing with it are shown in fig. 2.3




(a) shows an intersecting Attribute A.




(b) All but one link to it may be replaced with equivalent links via an existing key.




(c) Redundant versions of it may be connected to each associated key.


(d) It may be made into a key with no attributes

Fig. 2.3

(vii) Isolated attribute :- An isolated attribute is an attribute which is not identified by a primary key. It is a bubble with no single arrows entering or leaving it, although there will be double arrow links.

There should be no isolated attributes on the canonical graph. It may be treated in one of the following ways,

(i) It may be implemented as a repeating attribute in a variable-length record.

(ii) It may be treated as a solitary key - a one data-item record.

It should however be noted that an isolated attribute often results from a wrong interpretation of the data, and so the meaning related to it should be carefully checked.

(viii) Synonyms : Two data items with different names in different users views but having the same meaning are known as synonyms.

(ix) Homonyms : Two data items having the same name in different users views but having different meanings are known as homonyms.

Homonyms are more common in occurence than synonyms and they should be distingished in the database by changing the name of one of the items.

## 2.2 Design of a canonical schema :

The canonical database structure is a minimal conceptual schema. Its records are in the third normal form and there is no M:M mapping between the records. It can be derived by a

step-by-step procedure of combining the user views of data. The procedure is an incremental one which observes the effect of incorporating a new subschema on an existing schema. Thus new user views could be added into the database without any changes in the existing application programs. The canonical form of data is finally converted into a relational schema for the proposed model. In doing so, it is assumed that all paths have same usage i.e. the high usage paths have not been discriminated.

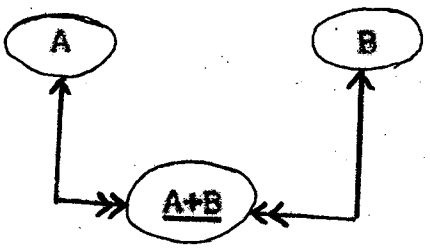
In the design of the database, as mentioned in chapter 1, three distinct levels are maintained namely, the external level (or external schema), the logical level (or logical schema) and the internal level (or internal schema). There are two interfaces, one, in between the external level and the logical level, and this maps the external level to the logical level. This mapping is done by the data definition facilities of the query language discussed in chapter 3. The second interface is inbetween the logical level and the internal level, and it maps the logical level to the internal level. This mapping is again provided by the data definition facilities of the query language. Data independence is thus provided at two levels (a) the logical independence and (b) the physical data - independence. Thus, on the first level, any addition of new relations to the logical database will not effect the existing application programs. Any deletions will also not affect those existing application programs which do not make use of the relations deleted. On the second level, data independence is maintained because

any change in the physical layout of data will not cause a change in the application programs, except, of course in their performance i.e. the physical structure could be changed from a hashed direct file storage structure to an indexed sequential one, or, the storage device could be changed in order to store data on a faster and more efficient device. Any changes that occur, either on the logical level or on the physical level, are taken care of in the interfaces, by changing the mapping as required in them. The application programs run as if no changes have occured. A description of the directories in which the data definition is maintained is be given in section 2.4.

The steps undergone in designing the canonical structure are given in order below :

1. The first user's view of data is taken and drawn in the form of a bubble chart, with the links in it representing associations of two types : 1 and M.

Where a concatenated key is used, it is drawn as a bubble, and the component data items of the concatenated key are drawn as seperate bubbles thus :




The representation is checked for any hidden transitive dependencies, which are removed, and it is ensured that

single arrow links from the concatenated key go to only those data-items which are dependent on the full concatenated key and not only on a part of it. In other words, it is ensured that the user's view is in the third normal form.

2. The next user's view is taken and represented as above. It is then merged with the graph or bubble chart of the previous step. A check is made for homonyms and synonyms which are removed if they exist.

3. In the resulting graph the primary nodes are distinguished from the attribute nodes. The primary nodes are underlined.

4. For each association of the keys, an inverse association is added if it does not already exist. If this results in an M:M link between keys, and if the inverse association is ever to be used at any time in the future, it is replaced by adding an extra concatenated key as was discussed in figure 2.2 of section 2.1.

5. The associations are examined to identify the redundant ones amongst them. These are then carefully checked for their meaning and removed if found genuinely redundant.

6. The previous four steps are repeated until all users views have been merged into the graph.

7. The root keys are identified and the diagram is rearranged with the root keys at the top i.e. all single arrow links between keys should point upwards.

8. The graph is checked for any isolated attributes, and if they exist, they are treated as mentioned in part vii of section 2.1.

9. The graph is adjusted to avoid any intersecting attributes.

The method for doing this is mentioned in part vi of section 2.1.

10. The graph is now arranged in groups or tuples each having one primary key and its associated attributes. Each group is now represented in a box.

11. All secondary keys are identified and these links drawn between boxes.

12. The canonical schema is now converted into a relational schema. For doing this, each box is represented as a relation. Since all links and associations are represented in a single uniform manner in a relational model, the upward going arrows which represent links between keys, are incorporated by adding the root keys as attributes in the relations from which the single arrows originate i.e. the root keys now become foreign keys in these relations. Through this method the information of secondary key links from attributes of one relation to primary key of another relation is also incorporated.

The next section gives the actual conceptual schema designed for the library data base.

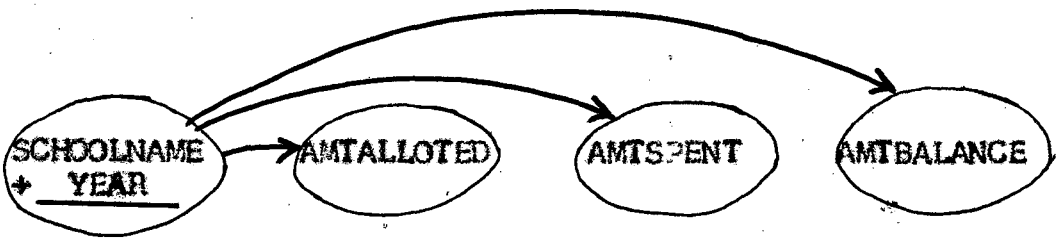
## 2.3 The proposed conceptual schema for the Library Database :

Steps for merging a few of the views for the design of the conceptual schema as explained in section 2.2 are shown below. The same method is followed for the rest of the views to construct the final conceptual schema given.

1st view :- Each school, each year, is alloted an amount of money to be spent on ordering literature for the library. An account for every year is kept of the amount spent and the balanced amount

| SCHOOLNAME | YEAR | AMTALLOTED | AMTSPENT | AMTBALANCE |
|---|---|---|---|---|

when represented as a bubble chart :

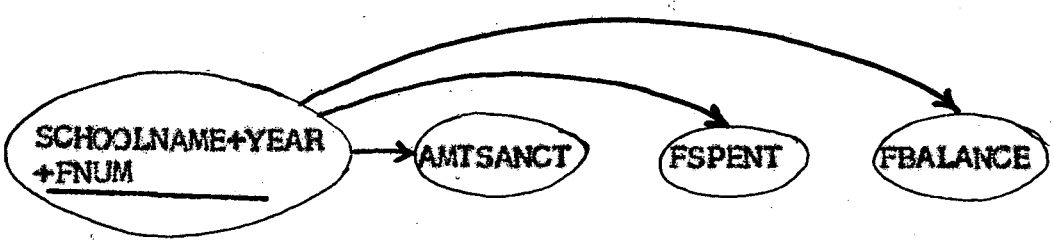



FULL FUNCTIONAL DEPENDENCY IS ENSURED AND NO TRANSITIVE DEPENDENCY OCCURS.

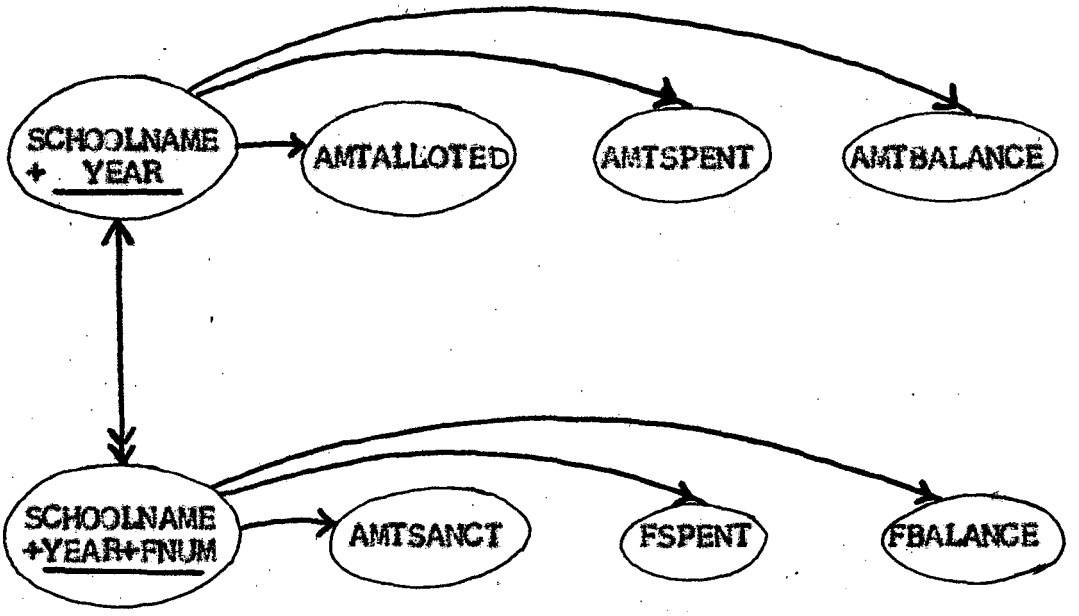
2nd View :- Each faculty member of a school is sanctioned an amount of money every year for ordering of material for the library. An account is kept of the amount spent and the balanced amount for each faculty member

| FNUM | SCHOOLNAME | YEAR | AMTSANCT | FSPENT | FBALANCE |
|---|---|---|---|---|---|

it is represented as a bubble chart, with non full functional dependencies and transitive dependencies, removed :



This view is then merged with the previous view and transitive dependencies, if any, are removed :

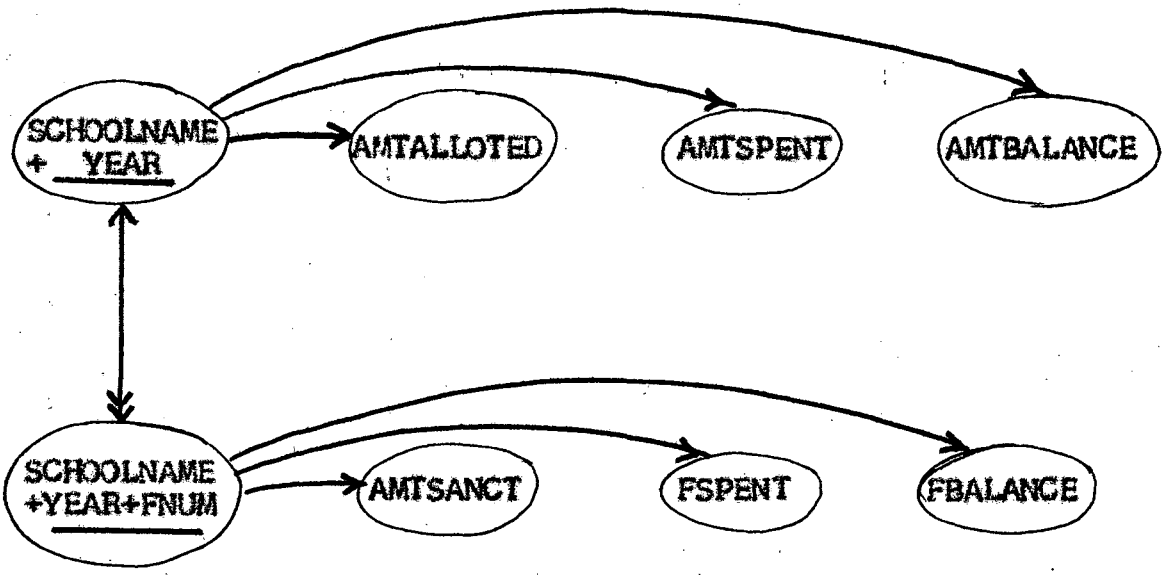



3rd View :- Details of suppliers who supply the material to the library are maintained. Each supplier has a unique supplier number.

| SUPNUMB | SUPNAME | SUPADDR |
|---|---|---|

represented as a bubble chart with no non-full or transitive dependencies :

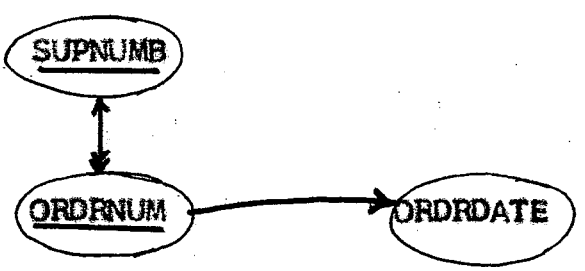

This view is then merged with the previous views



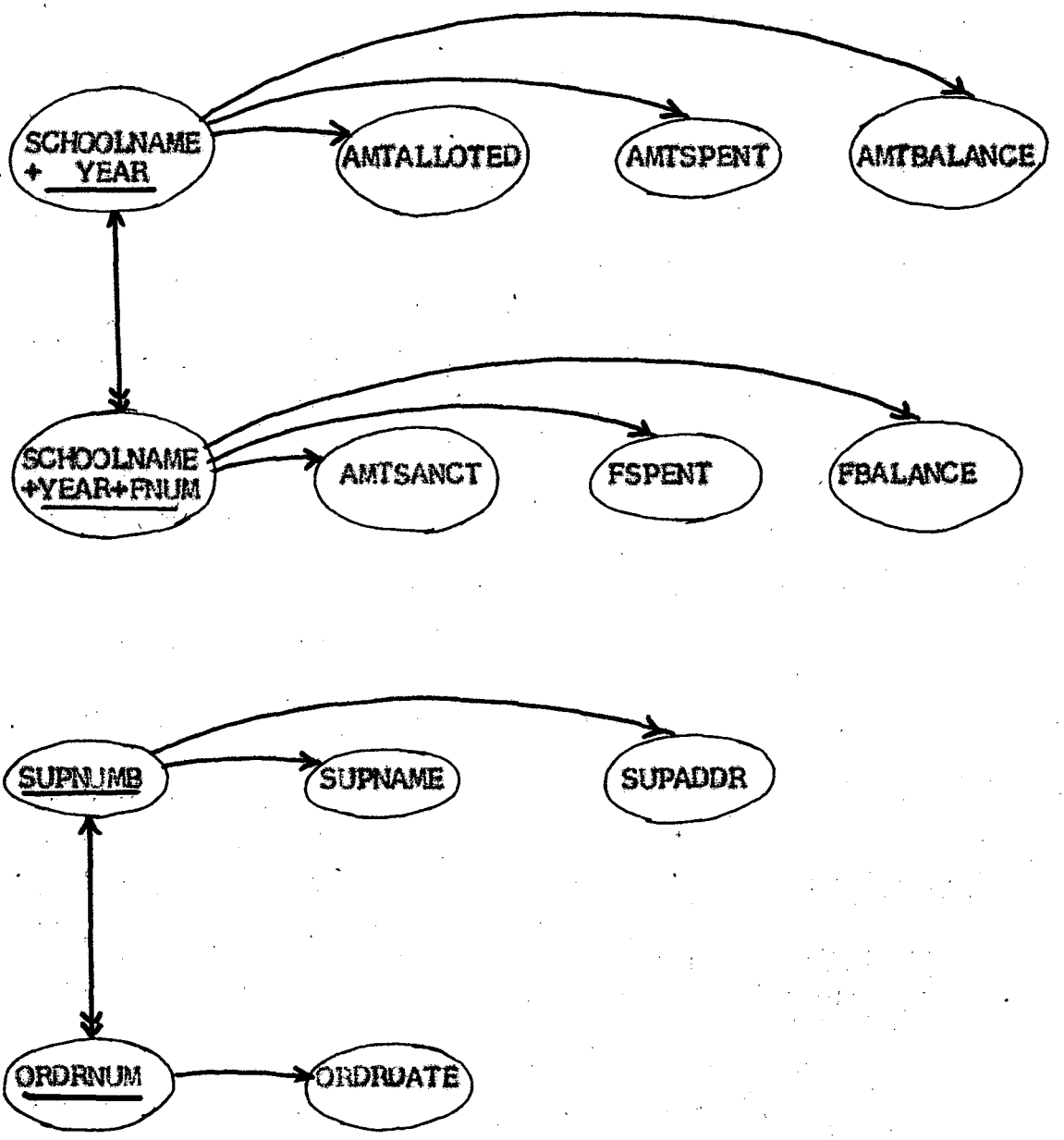


4th View :- for each order placed with a supplier, the order number and date of order are maintained


represented as bubble chart :



when merged with the previous views :

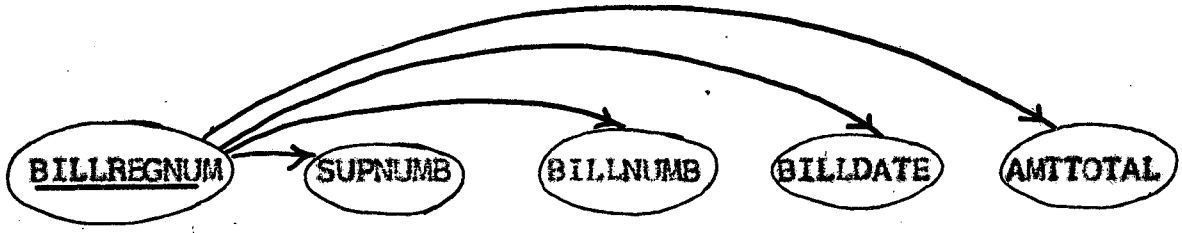



5th View :- for each bill given by a supplier, after payment has been made, each bill is registered and given a Bill registration number. The details of information maintained for each bill are : its bill number, the date of the bill,

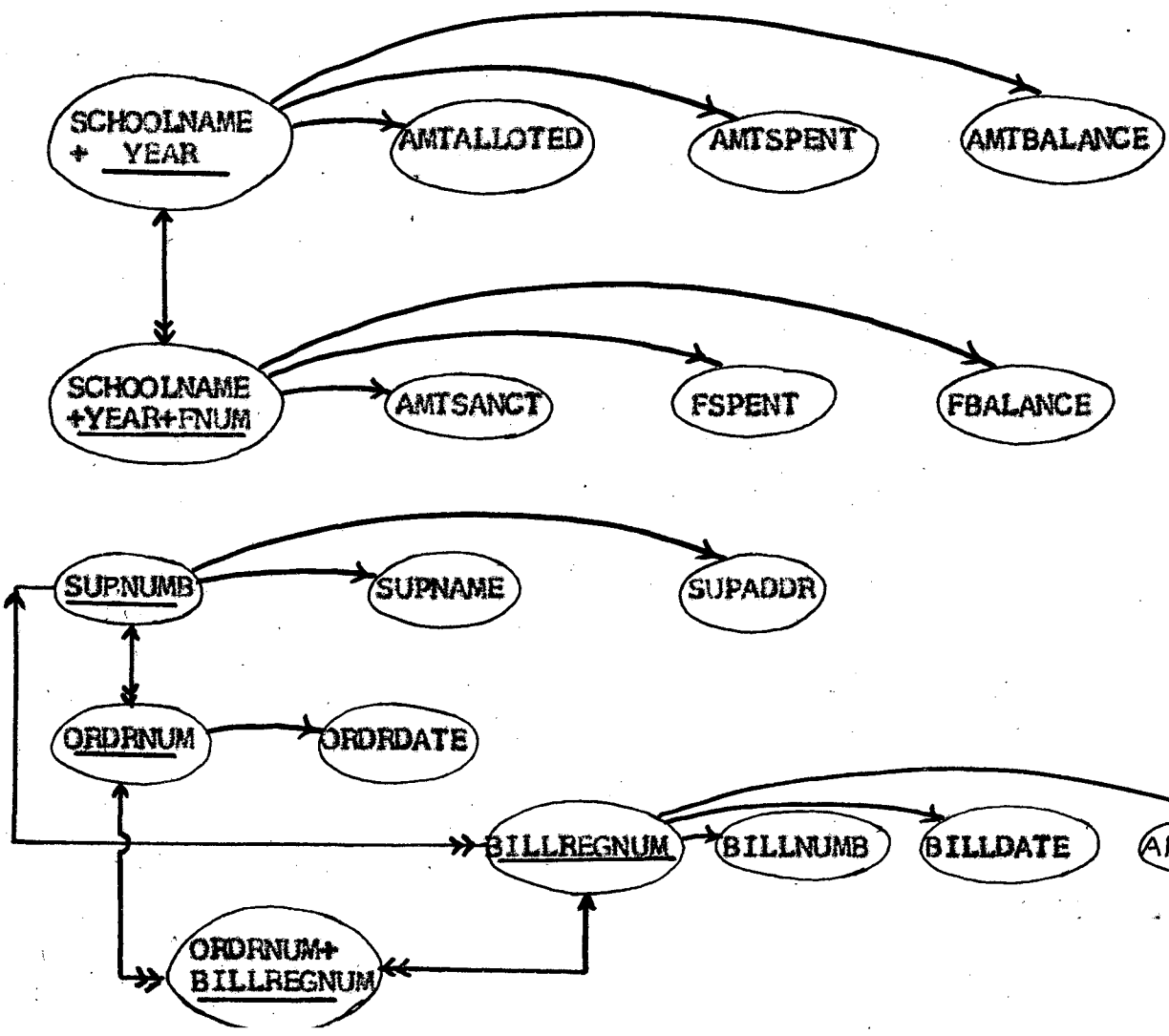
and the total amount on the bill.

| BILLREGNUM | BILLNUMB | BILLDATE | SUPNUMB | AMTTOTAL |
|---|---|---|---|---|

when represented as a bubble chart :

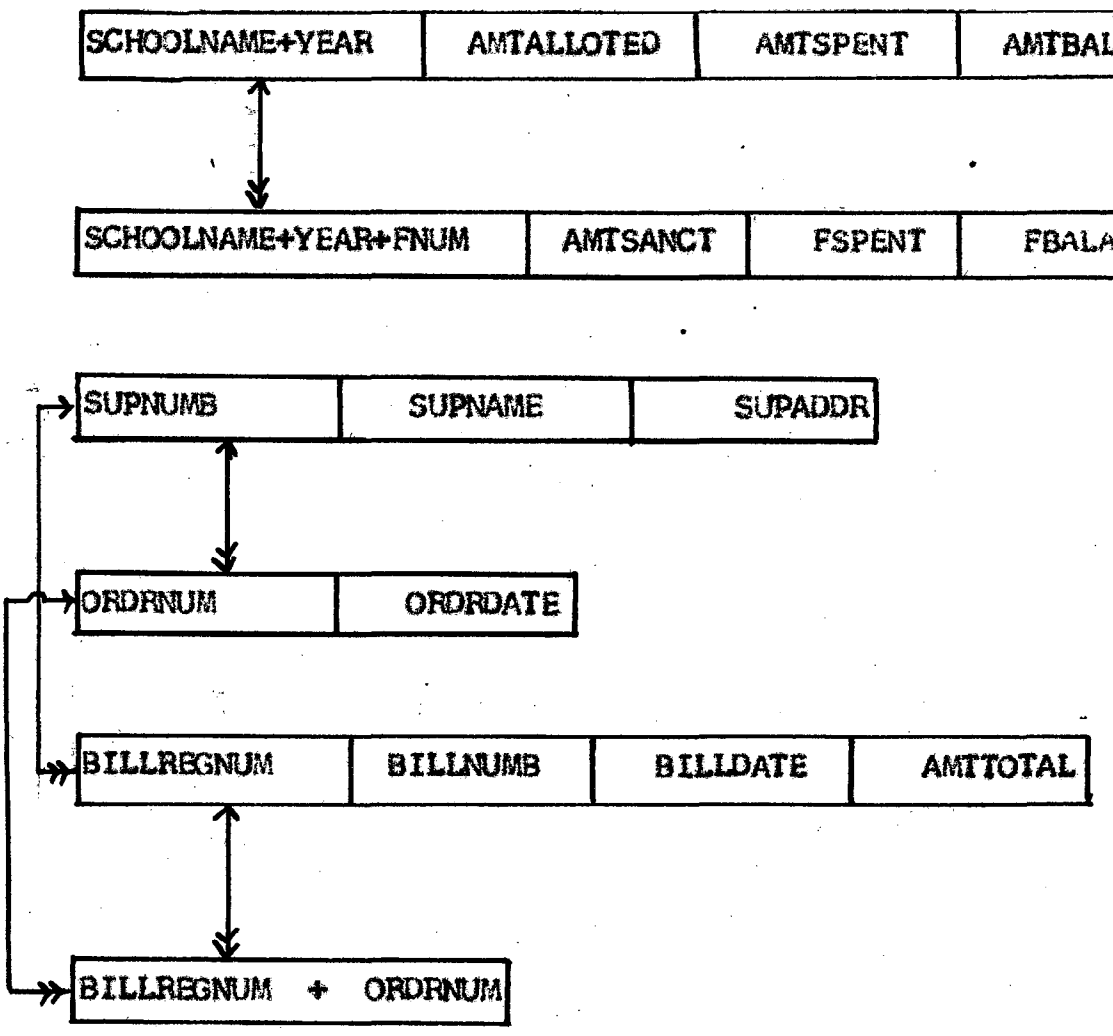



when merged with the previous views :



This process of mergring of views is carried on for all other views. The final bubble chart so obtained is a canonical structure and this is finally represented as a Relational conceptual schema. For this, the bubble chart is first represented in the form of boxes and then converted into a relational conceptual schema. These steps are shown below, taking the bubble chart which was derived above for some of the views.

(a) The canonical structure



(b) To convert it into a relational schema, each box is represented as a relation and the links between keys are represented by adding the keys as attributes in the relations according to step 12 section 2.2

| SCHOOLNAME+YEAR | AMTALLOTED | AMTSPENT | AMTBALANCE |
|---|---|---|---|

| SCHOOLNAME+YEAR+FNUM | AMTSANCT | FSPENT | FBALANCE |
|---|---|---|---|

| SUPNUMB | SUPNAME | SUPADDR |
|---|---|---|

| ORDRNUM | ORDRDATE | SUPNUMB |
|---|---|---|

supplier number is a foreign key in this relation

| BILLREGNUM | SUPNUMB | BILLNUMB | BILLDATE | AMTTOTAL |
|---|---|---|---|---|

| BILLREGNUM + ORDRNUM |
|---|

supplier number is a foreign key in this relation.

The relations in the conceptual schema so obtained are represented in the conventional way, with the relation name followed by the attributes in paranthesis. The relations given are in third normal form. The primary keys (single and

concatenated) are represented by the underlined attributes. The relations for the proposed library database are given in fig. 2.4.

SCHOOLDTA (YEAR+SCHOOLNAME, AMTALLOTED, AMTSPENT, AMTBALANCE)

FACULTYDTA (YEAR+SCHOOLNAME+FNUM, AMTSANCT,FSPENT,FBALANCE)

SUPPLIER (SUPNUMB, SUPNAME, SUPADDR)

ORDER (ORDRNUM, ORDRDATE, SUPNUMB)

BOOKRECOM (TITLE+AUTHOR+VOLUME+EDITION, SCHOOLNAME, FNUM, YEAR, ORDRNUM, PUBNAME, PRICE, RECOMCOPIS)

JURNLRECOM (TITLE, SCHOOLNAME, FNUM, YEAR, ORDRNUM, PUBNAME, PRICE, RECPERIOD)

CONFRRECOM (NAME, SCHOOLNAME, FNUM, YEAR, ORDRNUM, PRICE)

BILLSUPPL (BILLREGNUM, SUPNUMB, BILLNUMB, BILLDATE, AMTTOTAL)

BILLORDER (BILLREGNUM + ORDRNUM)

BOOKINLIB (ACCNO, LITTYPE, BILLREGNUM, ORDRNUM, TITLE, AUTHOR, VOLUME, EDITION, COPYNO, PUBNAME, PRICE, ARRIVDATE, CALLNO, LIBDIV, STATUS, ABSTRACT, NAME, PLACE, DATE, EDITOR)

FIELDETAIL (ACCNO+FIELD+SUBFLD)

CARDDETAIL (CARDNO, ISSUEENAME, ISSUEADDR, NUMOFCARDS)

ISSUEFILE (CARDNO+SERIALNO, ACCNO, ISSUEDATE, DUEDATE)

BOUNDJURNL (TITLE+VOLUME, BILLREGNUM, ORDRNUM, LIBDIV, ACCNO, CALLNO, STATUS)

JOURNAL (TITLE+VOLUME+NO, MONTH, STATUS)

JFLDSUBFLD (TITLE+VOLUME+NO+SUBFIELD)

JFIELD (TITLE+VOLUME+FIELD)

THESIS (TITLE+AUTHOR, DEGREE, YEAR, SUPERVISOR, INSTITUTE, LIBDIV)

THESISFLD (TITLE+AUTHOR+FIELD+SUBFIELD)

Fig. 2.4

### 2.3.1 Description about the Library Database

Every year an amount of money is sanctioned to each school for expenditure on the literature needed by that school. The relation SCHOOLDTA holds information of the amount alloted (AMTALLOTED), the amount spent (AMTSPENT) and the balance amount (AMTBALANCE), every year for each school.

In every school, each faculty member (represented by FNUM in that school), has a sum of money alloted to him/her every year, within which, he/she can recommend any literature to be ordered by the library. The relation FACULTYDTA has information about the amount sanctioned (AMTSANCT), amount spent (FSPENT) and the balanced amount (FBALANCE) each year, for every faculty member.

The relation BOOKRECOM has information of the details of books recommended by each faculty member. The restriction placed in the system is that the same book cannot be recommended more than once. The tuples of the relation contain complete information about the book recommended - name of the book (TITLE), Author (s) (AUTHOR), FNUM, SCHOOLNAME, YEAR, its publication name (PUBNAME) its price (PRICE), and the number of copies recommended (RECOMCOPIS). A tuple is inserted in this relation only after the recommendation of the book has been accepted. This is done by the user who is entitled to insert in this relation. If the ORDRNUM is blank, it means that the book has been recommended but not ordered as yet. The ORDRNUM is

inserted as soon as the order is placed for a particular book. A similar information is maintained about the journals and reports of conferences recommended in relations JURNLRECOM and CONFRRECOM respectively. The relation JURNLRECOM has information about the name of the journal (TITLE), its publisher name (PUBNAME) the period for which recommended (RECPERIOD) SCHOOLNAME, FNUM ORDRNUM and PRICE. Only when the journal recommendation has been accepted, is its entry done in this relation. If the ORDRNUM is blank, it means that the journal has been recommended but the order is not placed as yet.

The relation CONFRRECOM has information about the name of the conference (NAME), who recommended it i.e. FNUM, SCHOOLNAME, YEAR, and the ORDRNUM on which order was placed. Only when the recommendation has been accepted is its entry done in the relation. A blank ORDRNUM means that the conference report has been recommended but not ordered as yet. The order for any literature recommended is placed with a supplier. The supplier details are contained in the relation SUPPLIER. The details are SUPNUMB, SUPNAME and supplier address (SUPADDR). Relation ORDER contains details of the order placed. These are order number (ORDRNUM), date on which the order is placed (ORDRDATE) and the supplier with whom that order is placed. When the supplier supplies the literature ordered, an account is kept of the bill against which the payment is to be made to the supplier. The bill is registered and its registration number is given by

BILLREGNUM. This number is always unique and is never duplicated. The details of the bills registered are contained in the relation BILLSUPPL. The other details besides BILLREGNUM are : the date on which the bill is made (BILLDATE), the supplier by whom given (SUPNUMB), the number of the bill which is placed on it by the supplier (BILLNUMB) and the total amount which will be payed against that bill (AMTTOTAL). Information of the bill registration number (BILLREGNUM) to the order number (ORDRNUM), for which the payment is made, and vice versa, is maintained in the relation BILLORDER. As soon as the payment of a bill has been made, the literature received is accessed by the library.

The relation BOOKINLIB maintains information about books and conference reports accessed by the library. This information is the accession number (ACCNO) which is always unique and never duplicated, LITTYPE, the call number (CALLNO), the price (PRICE), the date of arrival in the library (ARRIVDATE), the library division (LIBDIV), the status in library (STATUS) i.e. whether the book or conference report is on shelf, lost or issued, the bill registration number (BILLREGNUM), ORDRNUM, publisher name (PUBNAME), the name of the book (TITLE), its Author (s) (AUTHOR), volume of the book (VOLUME) edition of the book (EDITION), the copy number of the book in the library (COPYNO), the abstract of the book (ABSTRACT), the name of the conference (NAME), the place at which the conference was held (PLACE), the date on which the conference started (DATE)

and the name of the editor (s) who produced and edited the report of each conference (EDITOR). LITTYPE is used to indicate the type of literature, i.e. whether it is a book, conference report or a journal. In this relation the LITTYPE can have only two values viz. B for books and C for conference reports.

LIBDIV is used to indicate the library division in which that literature is kept. For example, in JNU there are six library divisions.

The relation FIELDETAIL contains the information about the field (FIELD) such as Computer Science, Chemistry etc. and the subfield (SUBFIELD) such as database, or computer architecture etc. of the book or conference report or the bound journal, each identified by the accession number (ACCNO).

Each member of the library is given a number of library cards against each of which one book, or one conference report or one bound journal can be issued. The cards for each member have a unique card number on each one of them. The cards for each member are serialed. The information about card number (CARDNO) of cards issued to each member is maintained in the relation CARDDETAIL. Other information in this relation is the name of the member to whom the card number is issued (ISSUEENAME) the address of that member (ISSUEADDR) and the maximum number of cards that each member is entitled to (NUMOFCARDS).

The details of information maintained when some literature is issued to a member of the library is contained

in the relation ISSUEFILE. This information is, the card number (CARDNO), serial number of the card (SERIALNO). At a time only a single book can be issued against one card. The information of the issued material is the ACCNO, the date of issue (ISSUEDATE) and the date of return (DUEDATE).

The journals are received in the library under different numbers throughout the year, on, for example, a monthly basis, for a particular year. After all the numbers for a particular year have been received, they are bound together under one cover and an accession number and call number are given to this bound volume. The details of the journals which have been bound are maintained in the relation BOUNDJURNL, and of the journals not yet bound, in the relation, JOURNAL. The details in the relation JOURNAL are, the name of the journal (TITLE), its volume (VOLUME), its number (NO), the month (MONTH), and its status in the library i.e. whether on shelf or lost. The details of relation BOUNDJURNL are Title (TITLE) Volume (VOLUME), BILLREGNO, ORDERNUM, LIBDIV, ACCNO, CALLNO, and its status in library i.e. whether in library, lost or issued.

The details of the subfields with which each number of each volume of a journal deals are maintained in the relation JFLDSUBFLD. These are TITLE, VOLUME, NO and the subfield (SUBFIELD). Relation JFIELD contains information about the field of each volume of each journal. The attributes are TITLE, VOLUME and FIELD.

The thesis of research scholars are also kept in the library for reference purposes. The details of each thesis

are maintained in the relation THESIS. These are, the title or topic of the research thesis (TITLE) name (s) of author (s) who have written the thesis (AUTHOR) the degree for award of ... which the thesis was written (DEGREE) the year of writing the thesis (YEAR), name of supervisor under whom the work was done (SUPERVISOR), the name of institution from which the degree was awarded (INSTITUTE), and the library division (LIBDIV) in which it is kept.

The details of the fields to which the thesis belongs and the particular subfields considered in it are maintained in the relation THESISFIELD. These details are the title of the thesis (TITLE), the author (s) (AUTHOR), field (FIELD) and subfield (SUBFIELD).

The relations are in third normal form, this ensures data integrity and data independence.

2.3.2 Integrity Checks :

The integrity constraints in the database design are : that the value of LITTYPE must always be checked before insertion in the relation BOOKINLIB. Its value could either be 'B' for books or 'C' for conference report. Before insertion in the relation ISSUEFILE of a tuple when some material is issued from the library, the value of SERIALNO for the CARDNO in relation ISSUEFILE must be compared with the value of NUMOFCARDS for that CARDNO in relation CARDDETAIL. The SERIALNO must be less than or equal to the NUMOFCARDS. This is to ensure that no member can issue more material than he is entitled to at a time, for example, the maximum number

is limited to six, for a student, and to eight, for a teacher. The value of STATUS in relations BOOKINLIB and BOUNDJURNL should be only either, INLIB, if the literature concerned is on shelf, or, ISSUED, if it is issued to a member, or LOST, if it cannot be traced. The value of STATUS in relation JOURNAL could be only either INLIB or LOST, since unbound journals are not issued. When any material is issued, the STATUS must be set to 'ISSUED' and a corresponding entry made in relation ISSUEFILE of the attributes for that card and serial number.

2.3.3 Data Security :

The security of data is maintained at two levels. The first being that no unauthorized user can access the database, so the whole database is secure against invalid users. At the second level, security is maintained on individual relations for different operations to be performed on them. This is achieved by the GRANT facility in the query language, described in details in chapter 3, section 3.1.4.

Only those users who have been granted the data manipulation facilities through the GRANT command can perform the manipulations which are authorized to them and on those relations only which are specified in the GRANT command. The GRANT OPTIONS include all facilities of the language, namely, the query, manipulation and control facilities. These options can be granted as well as revoked.

All the facilities of the language are discussed in detail in chapter 3.

## 2.4 Description of the directories

The data definition is maintained in the directories DBMSRD, DBMS8[illegible], DBMSOO and DBMOl in the proposed database model. These directories are accessed by the interface programs very frequently and so are maintained as direct files using the hashing technique given in Appendix (3). It is basically a division technique, which has been found to be more satisfactory than many of the other techniques [1] in that, it gives fewer overflows. In the directory files used, each bucket contains three blocks and each block contains one record i.e. the blocking factor is 1, so, when the search is made for a record, no deblocking is to be done. The hashed files can be searched very quickly because the key on which the hashing function is applied is transformed to a number, which may be the address of the record (or bucket containing the record), or, it may be a pointer to the actual address. In the proposed technique, the key on being hashed is transformed into a number which is the actual address of the bucket containing the record in the file. The search control thus passes directly to the bucket which is then searched for the correct record. If the correct record is not found there, it is searched for in the overflow area which is a separate area on the file, and where, overflows from all buckets are stored. This area is searched sequentially. Directories can be treated as relations.

### 2.4.1 Directories Used for Data Definition

(a) DBMSRD : This directory maintains the definition of relations. As mentioned, it is a direct access file, i.e. a type 2 file in the HP 1000 filing system and is described in Appendix (3). The key is the relation name and it should be from one to ten characters in length. Corresponding to each name is a restore/suspend status word, which, if set to 1 means that the relation is in the restored state and can be used in any statement of the query language, and if set to 0 means that the relation is suspended and cannot be used in any statement of the query language except the 'RESTORE' statement which restores the suspended relation.

Fig. 2.4 (a) shows the arrangement of DBMSRD

DBMSRD

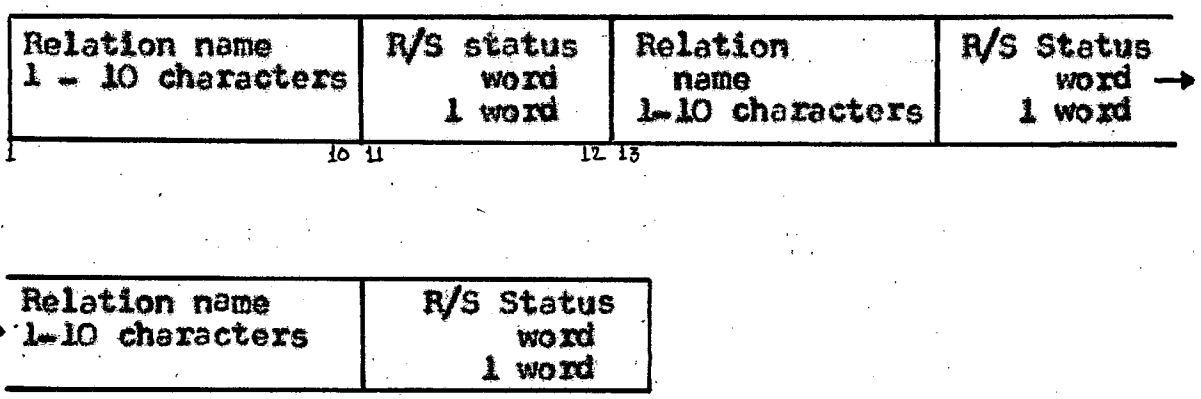



Fig. 2.4 (a)

Note that the same relation name cannot be repeated.

(b) DBMSOO : DBMSOO is a type 2 file and maintains the definition of views defined by different users. The key is 'view name' and corresponding to each view name is the suspend/restore status word which could, as previously, be 0 or 1.

The view name must be 1 to 10 characters in length and must be different from any other of the relation or view names already existing in the database.

Fig. 2.4 (b) shows the arrangement in DBMSOO

DBMSOO

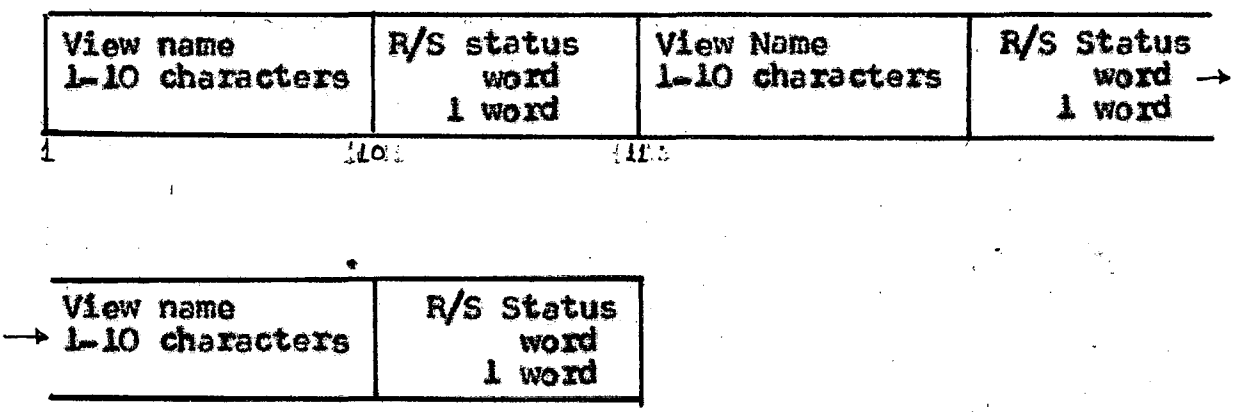



Fig. 2.4 (b)

(c) Description of DBMS8

DBMS8 is also a direct access, type 2 file. It maintains definitions of all attributes in all the relations. The key is a concatenation of relation name and attribute

name in that order. Each relation name is thus concatenated with each of its attribute names. The attribute names must be 1 to 10 characters in length, thus the key of DBMS80 has a maximum length of 20 characters. The concatenation forms a unique combination. Corresponding to each relation name + attribute name is the information about that attribute as it appears in that particular relation. The information and its order is :

(i) Type : Gives the 'type' of the attribute It is 2 characters in length. Values can be 01 for type 'alpha'. 02 for type 'alphanumeric' and 03 for type 'numeric'.

(ii) Start position : gives the starting position of the attribute in the relation. The first attribute in the relation starts from position 1. It is 4 characters in length.

(iii) Length : Gives the length of the attribute in the relation. The length is written in terms of the maximum number of characters that the value of the attribute could have. The length is written as a number of 2 characters.

(iv) A pointer to the relation in DBMSRD : This is given in terms of the bucket number followed by block number and is 4 characters in length, 2 characters for each number.

Fig. 2.4 (c) shows the arrangement in DBMS8 .

DBMS8

| (Relation name + Attribute name) 1-20 characters | Type 2 characters | Start position 4 characters | Length 2 characters → |
|---|---|---|---|

1 20 21 22

| → Bucket no. of Relation in DBMSRD 2 characters | Block no. in bucket 2 characters → |
|---|---|

| → (Relation name + Attribute name) 1 - 20 characters | type 2 characters | Start position 4 characters | Length 2 characters | bucket no. Of Relation in DBMSRD 2 characters | block no. in bucket 2 characters - |
|---|---|---|---|---|---|

| → (Relation name + Attribute name) 1 - 20 characters | type 2 characters | Start position 4 characters | Length 2 characters | bucket no. of Relation in DBMSRD 2 characters | block no. in bucket 2 characters |
|---|---|---|---|---|---|

Fig. 2.4 (c)

(d) Description of DBMSO1 : DBMSO1 is a direct access, type 2 file. It maintains definitions of all attributes in all the views defined by the users. The key is a concatenation of the view name and attribute name in that order. Each view name is concatenated with names of each of its attributes. Attribute names must be 1 to 10 characters in length, thus the maximum length of the concatenation is 20 characters. The concatenation forms a unique combination. The information corresponding to each combination is maintained in the order given below. When views are defined, no physical data is stored, i.e. the view is only a logical concept in which the view attributes are defined in terms of the relation attributes already existing in the database and having their values stored in the physical files. The definition in DBMSO1, for each view attribute, is thus the definition of the relation attribute on which the view attribute has been defined.

(i) type : gives the 'type' of the relation attribute which corresponds to the view attribute. It can have three values and is 2 characters in length. It is copied from the directory DBMSRD. Its values are :
01 for type 'alpha'
02 for type 'alphanumeric'
03 for type 'numeric'

(ii) Start position : gives the start position of the relation attribute in the actual relation. The view attribute has been defined on this relation attribute. The start position length is 4 characters. It is copied from DBMS8.

(iii) Length : gives the maximum length of the value of the attribute in the actual relation. The view attribute has been defined on this relation attribute. The length is written as a number of 2 characters.

It is copied from DBMS8?

(iv) A pointer to the relation : given by a bucket number followed by block number and is 4 characters for each number.

It is copied from DBMS8?

Fig. 2.4 (d) shows the arrangement in DBMSO1

DBMSO1

| (View name + Attribute name) 1-20 characters | type 2 characters | Start position 4 characters | Length 2 characters | Bucket no. 2 characters | Block no. 2 characters |
|---|---|---|---|---|---|
| | | | | | |

20 22

| (View name + Attribute name) 1-20 characters | type 2 characters | Start position 4 characters | Length 2 characters | Bucket no. 2 characters | Block no. 2 characters |
|---|---|---|---|---|---|
| | | | | | |

| (View name + Attribute name) 1-20 characters | type 2 characters | Start position 4 characters | Length 2 characters | Bucket no. 2 characters | Block no. 2 characters |
|---|---|---|---|---|---|
| | | | | | |

Fig. 2.4 (d)

Two other directories are maintained for the purpose of security restraints, both at the level of the entire database and at the level of individual relations.

2.4.2 Description of Directories Used for Security Maintenance :

(a) Description of USRCD : This directory maintains the options allowed to each user. The users in the proposed database model have been divided into three categories, each of which is given a distinct user code. This directory is used to determine whether a particular user is allowed to perform the action requested on any relation of the database. The directory is maintained as a direct access file. The key is the user code, corresponding to each one of which, are the options allowed. The three categories of users and their user codes are :

| | | |
|---|---|---|
| The database administrator | User code : | X99711 |
| Assistant administrator | User code : | Y88123 |
| General user | User code : | Z01234 |

Figure 2.4 (e) shows the arrangement in USRCD

USRCD

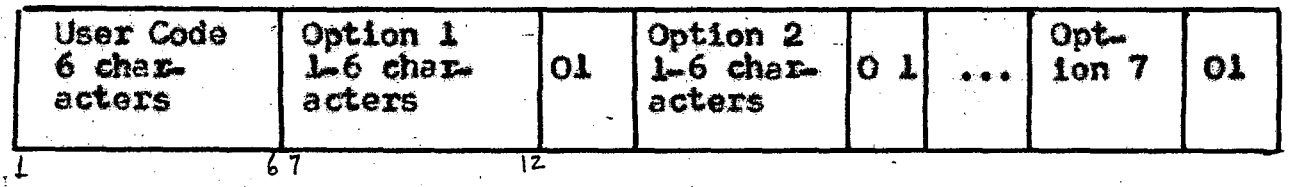



Fig. 2.4 (e)

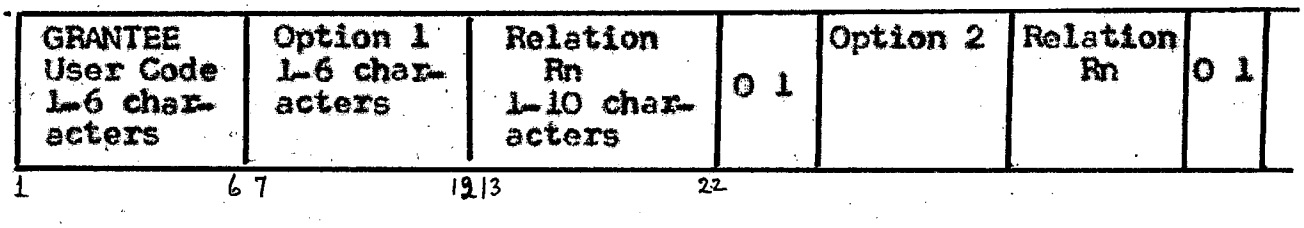
Option 1, Option 2, ..., Option 7, are any of the options which are allowed to the user these could be SELECT, MODIFY, INSRTN, etc. and shows that these options are

allowed at a particular time to the user.

Note that only the Data base administrator has the Grant option.

(b) Description of USRCOD : This directory maintains security restraints for different users according to the GRANT OPTIONS for each user. As said before, the language chosen allows security to be maintained through the GRANT statement of the data-control facility of the language. The directory is maintained as a direct access file. The key is the user code to whom the options have been granted i.e. the GRANTEE USER CODE. Corresponding to each of this code are the options granted on relations at a particular time. Overflow area for each user code is defined as a separate area in the file. The entries are similar to those described above. If an option is revoked from a user, the entry corresponding to that option is deleted from this directory. The arrangements in USRCOD are given in figure 2.4(f)

USRCOD




There can be entries for 10 options in one bucket for a grantee user code in the present directory.

Fig. 2.4 (f)

| | |
|---|---|
| Option 1, Option 2, ... | are the same as explained for the directory USRCD |
| Relation Rn ... | are any relations on which a particular option is granted |

The program to create the files including directory files in the system is the first program given in Appendix (5).

# CHAPTER - 3

## THE QUERY LANGUAGE AND ITS PROCESSOR

It has already been mentioned that the query language chosen is based on SEQUEL2 [10], which is a language intended for the inexperienced user. It is based on English Keywords. In addition to the query facility, the language has data manipulation facility which permits insertion, deletion and modification of tuples in the database, the data control facility, which enables authorization of use of data to different users and the data definition facility which enables definitions of relations, their attributes, various alternative views of relations, and their attributes.

The language is used as a stand alone language. All of its features are based on consistent keyword oriented syntax. The syntax is given in Appendix (1). This language accepts statements in free format. Words in the statements should not be broken arbitrarily and at least one blank must separate two different words ; and special characters such as [ ] ( ) , - = etc., and words.

### 3.1 Language Facilities

This section describes the various facilities mentioned above, of the language. The semantics of each statement is also given. For the detailed syntax, the reader should refer to Appendix (1).

3.1.1 Query facilities : Query facility statements are used to retrieve data from the database. The statements

are described below :

1. SELECT $A_1$ $A_2$ $A_3$ ... An FROM R WHERE $A_s$ = $[V_s]$
AND $A_t$ = $[V_t]$ ... AND $A_z$ = $[V_z]$ $

This query retrieves the values of attributes $A_1$, $A_2$ ..., An from those tuples of Relation R for which the values of attributes $A_s$, $A_t$ ... $A_z$ are $V_s$, $V_t$ ... $V_z$ respectively. Note that the equality condition could have been replaced by the 'greater than' - '>' or 'less than' - '<' conditions, however in our database these conditions have not been implemented. Also, in the actual implementation, the condition in the WHERE clause was restricted to one.

2. SECOUN $A_1$ $A_2$ ... An FROM R WHERE $A_s$ = $[V_s]$ AND
$A_t$ = $[V_t]$ ... AND $A_z$ = $[V_z]$ $

This statement retrieves as well as counts the number of tuples of relation R, for which the attributes $A_s$, $A_t$ ..., $A_z$ have values $V_s$, $V_t$ ..., $V_z$ respectively. The condition in the WHERE clause was restricted to one in the actual implementation.

3. SELECT $A_1$ $A_2$ ... An FROM R WHERE $A_s$ = $[V_s]$ AND
$A_t$ = $[V_t]$ ... $A_z$ = $[V_z]$ ORDRBY $A_k$ $

This statement selects the values of attributes $A_1$, $A_2$ ..., An from those tuples of relation R for which the attributes $A_S$, ... $A_z$ have values $V_s$ ... $V_z$ respectively. These resulting values are ordered in the ascending order of the values of attribute $A_k$. Note that $A_k$ must be one of the

attributes retrieved. Also, the WHERE clause is again restricted to one condition.

4. SELECT $A_1$ $A_2$ ... An FROM $R_1$ $R_2$ WHERE $R_1$, $A_y$ = $R_2$, $A_y$ $

This statement selects the attributes $A_1$, $A_2$ ..., An from those tuples of relations $R_1$ and $R_2$ for which the following condition is statisfied :
that, the value of attribute $A_y$ in relation $R_1$ is equal to the value of attribute $A_y$ in relation $R_2$. Each of the relations, $R_1$, and $R_2$ must contain all the attributes mentioned in the statement.

A retrieval problem important for the library database that occurs is that retrieval is always done by selecting attributes from those tuples for which the values of the attributes occur exactly as the values specified in the condition of the WHERE clause of the query statements. If this is not so, the result of a query might be blank, in which case the user will not know the real cause and will conclude that the information required does not exist in the database. This is very inconvenient, and more so for the library database, because such a situation implies that the user must know for example, the exact title of a book, journal etc. to retrieve other relevant information eg. Call No., Status etc., about it. Obviously it is very inconvenient to remember the complete and exact titles. In fact, often the user might want to select information

about books which contain, say, words like 'Data-Base' or 'Compiler Design' etc. in their titles, and not be bothered with the preceeding or succeeding parts of the title. The user might also often want to select information about books, thesis etc. by the author name. There could be more than one author. It is difficult to remember complete names of all the authors. In short, there are many instances when the user would like to retrieve information from the database knowing only 'part' of the complete value of the attribute by which retrieval is done. Such a kind of retrieval is possible as shown by query statements 5 and 6 below. The conditions in the WHERE clause of these queries are restricted to one (as in statement 5) or two (as in statement 6).

5. SELECT $A_1$ $A_2$ ... An FROM R WHERE PART $A_y$ = [V] $\not$

This query selects the values of attributes $A_1$, $A_2$ ..., An from those tuples of relation R in which part of the value of attribute $A_y$ is equal to V.

6. SELECT $A_1$ $A_2$ ... An FROM R WHERE PART $A_y$ AND $A_z$= [$V_1$] AND [$V_2$] $\not$

This query selects values of attributes $A_1$, $A_2$ ..., An from those tuples of relation R in which the part of the value of attribute $A_y$ is equal to $V_1$, and the part of the value of attribute $A_z$ is equal to $V_2$.

Note that in all the above query facilities the 'relation' could either be a relation of the conceptual schema, or a view defined by a user, since views can be used in the same ways as relations.

### 3.1.2 Data manipulation facilities

1. MODIFY R SET $A_i$ = $[V_i]$ ... An = $[V_n]$

   WHERE $A_\ell$ = $[V_\ell]$ AND $A_m$ = $[V_m]$ ... AND $A_z$ = $[V_z]$ $

This modifies the relation R by changing the existing values of attributes $A_i$ ... An by setting them equal to the new values given as $V_i$ ... $V_n$ in those tuples of relation R for which the attributes $A_\ell$, $A_m$ ..., $A_z$ have values equal to $V_\ell$, $V_m$ ... $V_z$ respectively. Note that the number of attributes to be modified as given in the SET clause was restricted to one in the actual implementation.

2. DELETE R WHERE $A_1$ = $[V_1]$ AND $A_2$ = $[V_2]$... AND $A_n$= $[V_n]$ $

This statement deletes those tuples from the relation R in which attributes $A_1$, $A_2$ ... An have values equal to $V_1$, $V_2$ ... $V_n$ respectively. Note that the 'relation' in the above statements could either be a relation of the conceptual schema, or, be a view, since views are treated in the same ways as relations.

3. INSRTN R $A_1$ $A_2$ ... $A_n$ $[V_1]$ $[V_2]$ ... $[V_n]$ $

This statement inserts in relation R, a new tuple having the values of attributes $A_1$, $A_2$ ... $A_n$ equal to $V_1$, $V_2$ ... $V_n$ respectively. It is to be noted that the attributes $A_1$, $A_2$ ... $A_n$ are the same as those in relation R.

### 3.1.3 Data Definition facilities

Data definition facilities allow users to create, suspend and restore relations dynamically, and define

alternative views of stored data.

1. CREATE R ($A_1$ type [$S_1$] [length] $A_2$ type [$S_2$] [length]
... $A_n$ type [$S_n$] [length] ) ¢

This statement Creates a new relation R with attributes $A_1$, $A_2$ ... An. The length of relation name and attribute name must not be more than 10 characters each. The word 'type' in the date definition statement gives the type declaration for each attribute. 'type' can take on the following values :
'ALPHA', if the attribute values are alpha ; 'ALFNUM', if the attribute values are alphanumeric; and 'NUMRIC', if the attribute values are numeric. $S_1$, $S_2$ ... $S_n$ give the start positions of attributes $A_1$, $A_2$ ... $A_n$ respectively in the relation R. The word 'length' in the statement denotes the maximum length allowed for values of each attribute in terms of the number of characters. Blank is counted as a character. eg. if an attribute AUTHOR is written as MARTIN J the length will be taken as 10. Alternately, if in the value specified for an attribute, the length is less than the 'length' given for that attribute in the statement of creation, the value is padded with blanks on the right. Once a relation is created, its definition must be maintained in directories. We have used two directories for this purpose. The directory named DBMSRD maintains the name of relations and their status i.e. whether 'suspended' or 'restored', A relation can be used in a query only if it is in the restored state. Directory

DBMS8 maintains the concatenated value of relations with each of their attributes, and a corresponding definition of each of these attributes is stored. A complete description of the directories contents has already been given in section 2.4.1.

2. DEFVEW R' ($A'_1$ ($R_i$ $A_1$) $A'_2$ ($R_i$ $A_2$) ... $A'_n$ ($R_i$ $A_n$)) $

Through this statement, a user can define a view R' with attributes $A'_1$, $A'_2$ ... , $A'_n$ taking attributes $A_1$, $A_2$ ... , $A_n$ from relation $R_i$. $R_i$ could be any relation in the database. A view can be defined taking some of the attributes of a relation or of another view already defined. Alternate views of stored data can thus be defined. In this implementation the view defined should have attributes taken from one relation only. The length of the view name and attribute name defined must not be more than 10 characters each. Each view must have a name different from any of the relations as well as any of the views already defined. A view can be used in the same ways as a stored relation.

A definition ofthe view must be maintained in directories as that for ordinary relations. We have used two directories for this purpose. The directories are DBMSOO and DBMSO1. A complete description of these directories has already been given in section 2.4.1.

3. SUSPND R $

This statement suspends relation R from the database i.e. no retrieval, modification, deletion, etc. can be done on that

relation. The restore status word in directory DBMSRD is put equal to zero. Views, since they act as relations can also be suspended. As is done for relations, the status word in directory DBMSOO is put equal to zero. Note that when a relation is suspended, all views defined on that relation are also suspended.

4. RESTOR R ∮

This statement 'restores' the relation or view named R. The restore status word is put equal to 1 in directory DBMSRD or directory DBMSOO according to whether R is a relation or a view respectively. Retrieval, modification, deletion, insertion etc. can be done only on relations and views which are in the restored state.

3.1.4 Data Control Facilities

The data control facilities enables the Data Base Administrator to delegate some of his functions which are deemed fit, to his assistants or other general users. In the proposed database, only the database Administrator has the right to grant other facilities amongst users. He may, if he so wishes, grant this right to his assistants too. This grant of privileges is done by means of the commend GRANT.

1. GRANT $P_1$ $P_2$ ... $P_n$ ON R TO $[U_1]$ ... $[U_n]$ ∮

By this statement, privileges $P_1$, $P_2$ ... $P_n$ on relation R are granted to users $U_1$, $U_2$ ... $U_n$. Here $P_i$ could be any privilege, for example Select, Modify, Delete, Define

view, Insert, Suspend, Restore or Revoke, we assume that only the Database Administrator and his assistant can create relations in the database, and that this privilege cannot be granted to anyother user. The values of $U_1$ ... $U_n$ are actually the user codes of the users. In the proposed model, there are three classes of users. Each class of user has a unique usercode which, alongwith his privileges are maintained in directories USRCD and USRCOD. The three classes of users are :

| Class | User code |
|---|---|
| 1. DataBase Administrator | X99711 |
| 2. Assistant Administrator | Y88123 |
| 3. General user | Z01234 |

A complete description of these directories is given in section 2.4.2.

2. Once a privilege has been granted, it may be withdrawn through use of the REVOKE command. The named privileges are withdrawn from the grantee and also from all those users to whom he has granted them, unless those users have another independent granting source of the revoked privileges. The privileges from a user may be revoked only by the user who has granted them.

REVOKE $P_1$ $P_2$ ... $P_n$ ON R FROM $[U_1]$ ... $[U_n]$ ;

This statement withdraws the privileges $P_1$, $P_2$ ... $P_n$ on relation R from users with user codes $U_1$ ... $U_n$. As before $P_1$ ... $P_n$ are privileges such as Delete, Modify, Insert etc.

These data control fecilities allow the security of the database to be maintained, since each user has first to identify himself to the system and is allowed to perform an action on the database only if he is entitled to it according to the grant options maintained in directories against his user code.

3.2 THE QUERY PROCESSOR : This section describes the steps executed during the processing of queries. The steps in order, are :

(i) the lexical analyzer
(ii) the syntax analyzer
(iii) the interpretor

The whole program has been segmented. In this method, the MAIN PROGRAM always remains in the memory, while each of the segments are over-layed over each other as they are called into the memory. Thus, only that portion of the program, which is required at a particular time, is in the memory.

A user must sign on to the database system by entering his user code. The validity of his code is checked in the MAIN PROGRAM, which is in the memory when the query processor is run. If the code is valid, the MAIN PROGRAM executes, and at the end calls into the memory, the first segment named LEXGL, which is the lexical analyzer. The lexical analyzer breaks the query statement into tokens and stores them in a two-dimensional array known as the symbol table, ISTB. The tokens consist of two integer numbers.

The first is the code of that particular token type eg. keywords of the language such as SELECT, FROM, WHERE, DELETE ... , are given a code equal to 1 identifiers of the language are given code 4, literals, code 3 and delimiters, code 2. The second number is a pointer in the table of that token type, in which the token is stored. eg. the token for keyword FROM is (1,2). Here 1 shows that it is a keyword, and 2 gives the position of 'FROM' in the table of keywords in which it is stored. Identifiers, as they occur in the query statement are stored in a table for identifiers, ITAB1. The second number in the token generated for an identifier is a pointer to that identifier in the table for identifiers ITAB1. Similarly, literals, as they occur in the query statement are stored in a table for literals, ITAB. The second number in the token generated for a literal is a pointer to that literal in the table of literals, ITAB. It is much easier to deal with tokens, which are just integer numbers than with the actual words of the query statement. The lexical analyzer calls and is overlaid by the segment in which syntax checking is done and which is called the syntax analyzer or parser. The syntax analyzer checks the query, now in the form of tokens, to see whether it is syntactically correct according to the rules of the underlying grammer of the query statement. The syntax analyzer checks each token for its syntactic correctness and enters the token along with the result of the check in a table known as the parser table IPRS. The result of a check is either 0 or 1 depending on

whether the token is syntactically correct or not eg., if the word FROM (token (1, 2) ) is in the correct place, its corresponding entry will be (1,2,0) in IPRS. If a syntactic error occurs, the third entry in IPRS for that token is 1 and the syntax analyzer gives the error diagnostic and continues to analyze the query further, giving error diagnostics wherever an error occurs, till the end of the query and stops further execution of the query. In such a case, the user must feed in his query statement again with the errors removed, or he may, of be so wishes, terminate the process. If no error occurs i.e. the query is syntactically correct, the syntax analyzer calls, and is overlaid by, the relevant segment which interprets the query now existing in the parsed form in the parser table IPRS. The interpretor program or segment interprets the semantics of each statement and performs actions according to it.

A listing of the entire program is given in Appendix 5.

The output for some example queries taken is given in Appendix 4.

## APPENDIX - 1

The syntax of the language used is given in BNF notation. Square brackets [ ] indicate optional constructs. The terminals of the language are not enclosed within angular brackets. Keywords are represented as capital letter words. Non terminals are enclosed within angular brackets < and > . Note, that in the syntax, the productions for the non-terminal < literal > are enclosed in curly brackets { and }, but in the actual query statement [ and ] are used. Since these square brackets denote something different in BNF notation, these could not be used in the productions in the representation of the syntax.

< statement > ::= < query statement > ¢
| < dml statement > ¢
| < ddl statement > ¢
| < control statement > ¢

< query statement > ::= < select-clause >
FROM < from-list >
WHERE < condition-clause >

< select-clause > ::= SELECT < attribute name list >

< attribute name list > ::= < attribute name > [< attribute name >]$_0^n$

< from-list > ::= < system-entity name > [< system entity name >]$_0^n$

< system-entity name > ::= < relation name > | < view name >

< attribute name > ::= < identifier >

< relation name > ::= < identifier >

< view name > ::= < identifier >

<identifier> ::= <alpha> [<alpha> | <digit>]$_0^n$
| <digit> [<alpha> | <digit>]$_0^n$

<condition-clause> ::= <condition> [<connector> <condition>]$_0^n$

<connector> ::= AND | OR

<condition> ::= <expression 1> <comparision> <expression 2>

<expression 1> ::= <attribute name>
| PART <attribute name>
| <system-entity name> , <attribute name>

<expression 2> ::= <literal>
| <system-entity name> , <attribute name>
| PART <literal>

<comparision> ::= = | ¬= | > | < | >= | =<

These have got their normal meanings like 'equalto', 'not equal to' etc.

<literal> ::= {<alpha>}
| {<numeric>}
| {<alphanumeric>}

<alpha> ::= <alphabet> [<alphabet>]$_0^n$

<numeric> ::= <integer>
| <real>

<alphanumeric> ::= <alphabet> [<alphabet>]$_0^n$ <digit>
[<sp.char>]$_0^n$ [<alphabet> | <digit>]$_0^n$
| <digit> [<digit>]$_0^n$ [<sp.char>]$_0^n$
<alphabet> [<alphabet> | <digit>]$_0^n$

<sp.char> ::= * | * | - | : | /

<alphabet> ::= A | B | C | D | E | F | G | H | I | J | K | L | M
| N | O | P | Q | R | S | T | U | V | W | X | Y | Z

<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<integer> ::= [<sign>] <digit> $[<digit>]_0^n$

<real> ::= <integer> . <integer 2>
| < sign > . <integer 2>
| <integer> .
| <integer> E [<sign>] <integer 1>
| <integer> . <integer> E [<sign>] <integer 1>
| [<sign>] . <integer 2> E [<sign>] <integer 1>
| <integer> . E [<sign>] <integer 1>

<integer 2> ::= <digit> $[<digit>]_0^n$

<integer 1> ::= <digit> <digit>

<sign> ::= +
| -

<dml-statement> ::= <insertion>
| <deletion>
| <modify>

<insertion> ::= INSRTN <relation name> <attribute name list> <literal list>

<literal list> ::= <literal> $[<literal>]_0^n$

<deletion> ::= DELETE <system-entity name>
<where-clause>

<where-clause> ::= WHERE <condition-clause>

<modify> ::= MODIFY <system-entity name> <set-clause>
<where-clause>

<set-clause> ::= SET <attribute name> = <literal>

<control statement> ::= <grant>
| <revoke>

<grant> ::= GRANT <option list> ON <relation name>
TO <literal list>

<option list> ::= <option> $[<option>]_0^n$

<option> ::= INSRTN | DELETE | MODIFY | RESTOR | SELECT
| SECOUN | SUSPND

<revoke> ::= REVOKE <option list> ON <relation name>
FROM <literal list>

<ddl-statement> ::= <create-relation>
|<define-view>
|<suspend>
|<restore>

<create-relation> ::= CREATE <relation name>
<attribute-definition list>

<attribute-definition list> ::= <attribute definition>
$[<$attribute definition$>]_0^n$

<attribute definition> ::= <attribute name> <type>
{<start position>} {<length>}

<start position> ::= <integer 2>

<length> ::= <integer 2>

<type> ::= ALPHA

| NUMRIC

| ALFNUM

<define-view> ::= DEFVEW <view name> (<view field list>)

<view field list> ::= <attribute name> (<relation name>
<attribute name>)

[<attribute name> (<relation name>
<attribute name>)]$_0^n$

<view name> ::= <identifier>

<suspend> ::= SUSPND <system-entity name>

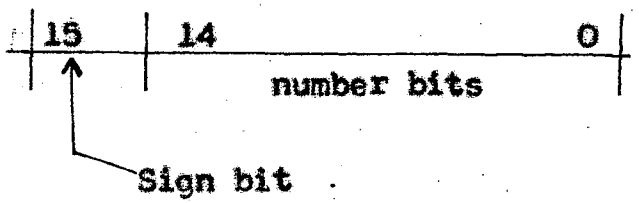
<restore> ::= RESTOR <system-entity name>

## APPENDIX - 2

The Data Format in Memory of HP 1000 system for an Integer, Real and the 3-word Double precision numbers are :

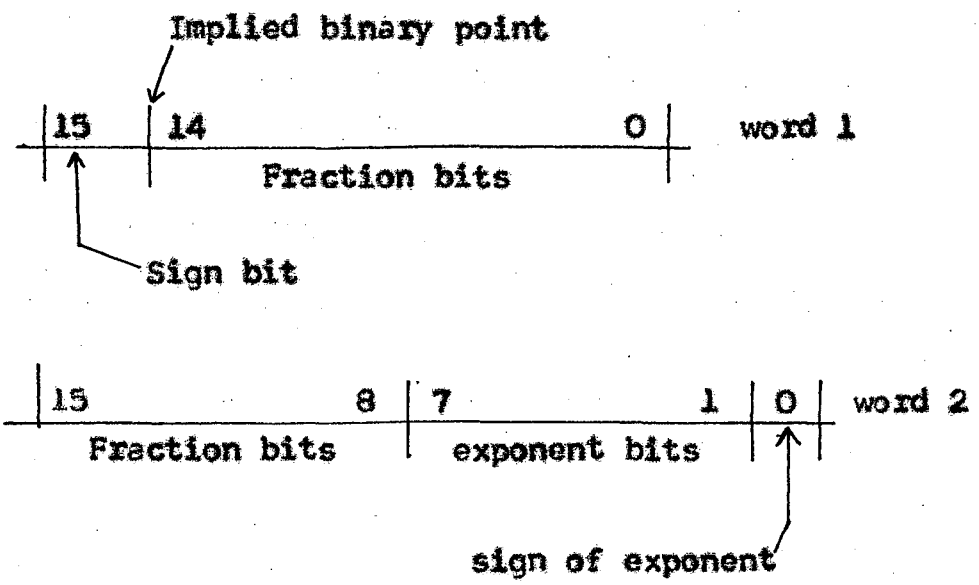
1) INTEGER FORMAT

It occupies one 16-bit word and has a range of $-2^{15}$ to $2^{15}-1$




2) REAL FORMAT

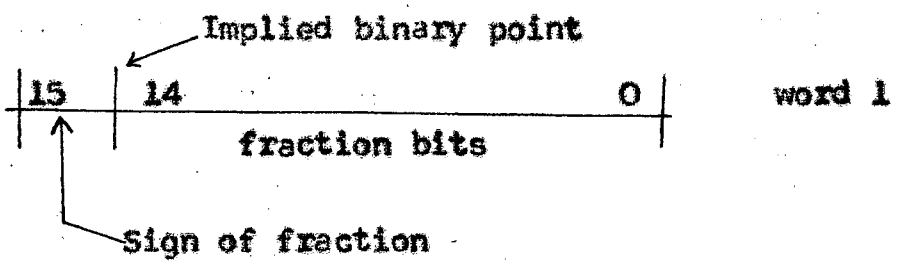
It occupies two consecutive 16-bit-words in memory and has an approximate range of $10^{-36}$ to $10^{38}$

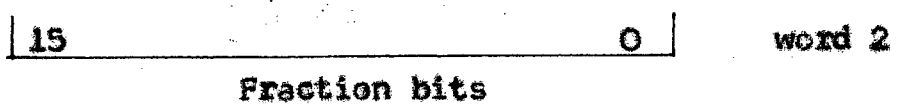


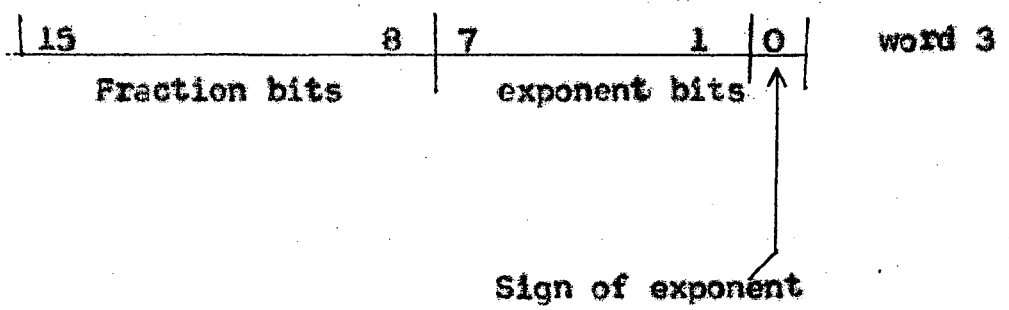

A real number has a 23-bit fraction and a 7-bit exponent.

3) 3-WORD DOUBLE PRECISION FORMAT

It occupies 3 consecutive 16-bit words in memory and has an approximate range of $10^{-38}$ to $10^{38}$

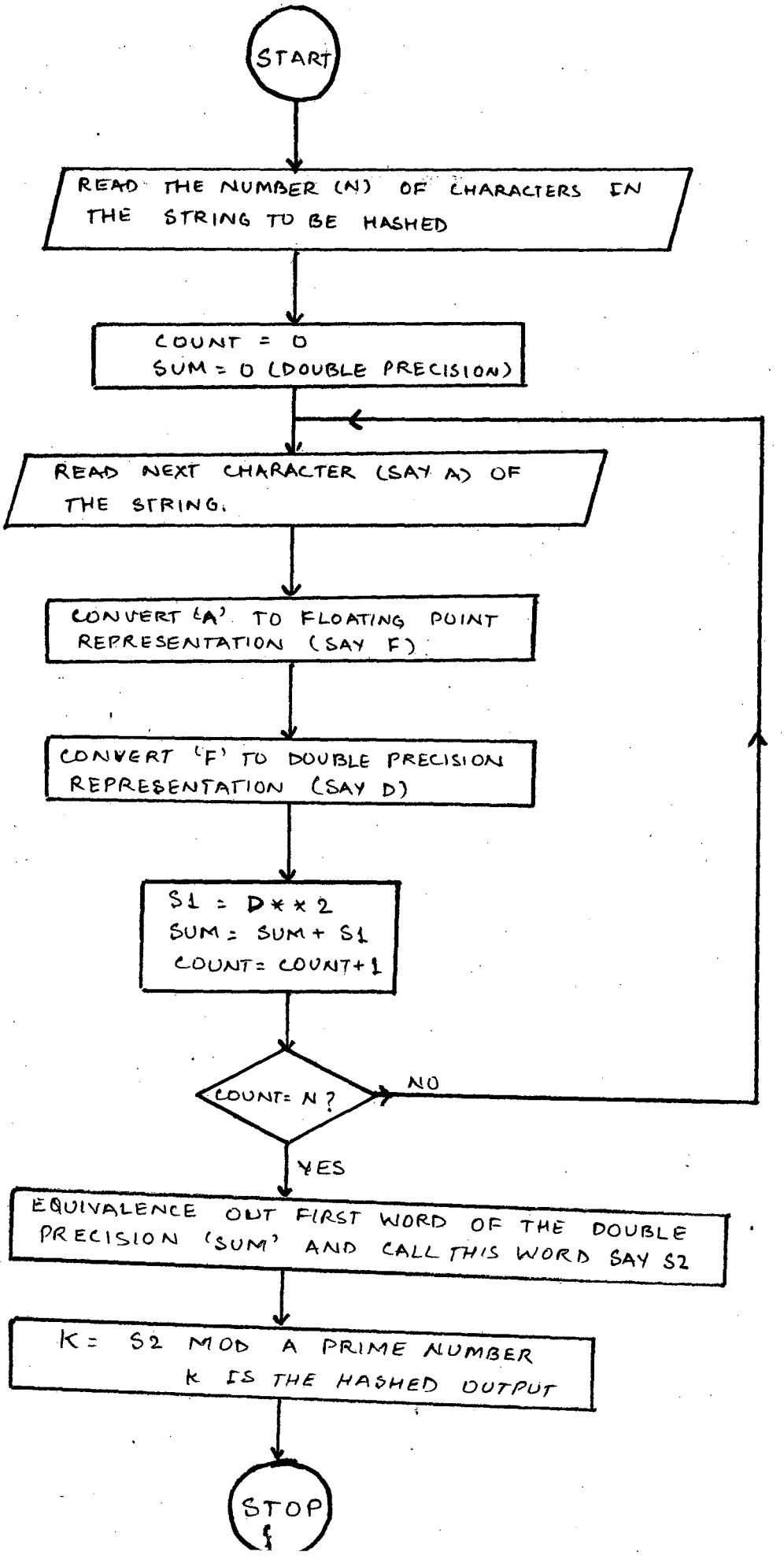



A double precision number has a 39-bit fraction and a 7-bit exponent.

HASHING TECHNIQUE FLOWCHART.



## APPENDIX - 3

## HP 1000 FILE MANAGEMENT SYSTEM

A3.1 File Management : [11] File management is performed through calls to the FMP library and by interactive operator commands to the program FMGR. The FMP calls mainly control input to and output from disc files or peripheral devices treated as files. Data may be binary or in ASCII Code in the files. Files may be stored on disc or they may refer to non-disc peripheral devices. The Batch-Spool Monitor is used to control and access files whether they are disc files or non-disc devices. The information about files is maintained in directories created and maintained by FMP. These directories are the FMP cartridge directory, which is a master index to all active FMP cartridges and the file directory, which contains information on each file on a particuler cartridge. The cartridge directory is maintained on the system disc, while file directory is maintained on each cartridge.

A3.2 FILE TYPES : Eight file types are defined by the system. Additional types may be defined by the user. Only the first four types differ in format; all subsequent types differ only in the type of data FMP expects the file to contain. The file type may be divided into three categories as shown in Fig. A1.

FIG. A1

| CATEGORY | TYPE | DESCRIPTION |
|---|---|---|
| Control | 0 | Non-disc device files |
| Fixed length, random access, non-extendable | 1 | Fixed length 128 word record files |
| | 2. | Fixed user defined record length files |
| | 3. | Variable length records; any data type |
| | 4. | Source program file; ASCII |
| | 5 | Object program file; relocatable binary |
| | 6 | Executable program file; memory image code |
| | 7 | Absolute binary |
| | 8-32767 | User defined data format |

Type 0 files : are used to reference non-disc devices by name. The record format of a type 0 file is determined by the device type.

Type 1 files : Type 1 files have fixed length records of 128 words. Because File Management Package (FMP) transfers data to and fro from disc in 128 words blocks, this file allows direct access between disc and user's buffer area in his program, thereby eliminating the need to go through a Data Control Block.

Type 2 files : The record lengths of type 2 files are fixed, but the length is defined by the user at file-creation. The end-of-file is the last word of the last block and files may

cross sector or track boundaries. Only one logical record is transferred at a time and the transfer must go through the Data Control Block. For this reason files of type 2 and above have a slower transfer rate than files of type 1. To obtain access of the maximum number of records (32767), the record length must be 128 or a multiple of 128 words.

Type 3 and above files : are variable length extendable files. These are not used in the database implementation and so are not discussed here.

A3.3 File Security : Each file has a security code. This code may be zero, negative or positive. A zero code allows the file to be opened to any caller with no restrictions; in effect this code provides zero security. A positive code restricts writing on files but not reading; that is, a user who does not know the code may open the file for read only, but may not write on the file. A negative code restricts all access to the file; this code must be specified in order to open a file protected by it. An attempt to open a file so protected without the security code results in an error message.

A3.4 FILE DEFINITION : A file may be defined in terms of its name, size, type and where it is located. The CREAT call defines a disc file in this way and causes an entry to be made for the file in the file directory. Once defined, the file may be opened for access by any program with proper security code. To open a file means to transfer the necessary information from the file directory to the control words of the Data Control Block and thus make a logical connection between the file name in the

directory and the Data Control Block for the file. The CREAT call opens the file to the calling program only, and only for update. For other types of access by other programs, the OPEN call must be used.

Following access, the file may be closed with the CLOSE call, closing a file means that connection between the Data Control Block and file directory entry for the file is severed. Some FMP calls which have been used are discussed. The parameters underlined are optional parameters. Two more commonly used parameters IDCB and IDCBS are described before individual FMP calls are described.

A3.5 IDCB : This parameter is used in FMP calls. It specifies the array used as a Data Control Block of words defined within a program that acts as an interface between the program and the FMP. It is an array which contains control information for the file including the file name, type, size and location on disc if the file is a disc file. In addition, it acts as a buffer for the physical transfer of data between a file and the program. The dimension of IDCB or the size of the Data Control Block must be at least 144 words, 16 words for file control information and 128 words for the minimum buffer. For faster processing a larger buffer may be specified.

A3.6 IDCBS : When a data control block larger than 144 words is specified in parameter IDCB, then parameter IDCBS must also be specified. It informs FMP of the number of words available in the DCB buffer for data transfer. Normally the IDCBS is specified as 16 words less than the array size specified for IDCB.

## A3.7 FMP CALLS

(i) CALL CREAT ( IDCB, IERR, NAME, ISIZE, ITYPE, ISECU, ICR, IDCBS)

Parameters

IDCB — Data control block; an array of $144^{+n}$ words where n is positive or zero.

IERR — Error return; one-word variable in which a negative error code is returned. If no error, it is set to the number of 64 words sectors in the created file.

NAME — File name; 3-word array containing ASCII file name.

ISIZE — File size; 2-word array with number of blocks in first word; if negative, rest of cartridge is allocated to file; second word, used only for type 2 files, contains record length in words.

ITYPE — File type.

ISECU — Security code; optional 1-word variable in range 0 through $\pm$32767; if ommitted, code is set to zero and file is not protected.

ICR — Cartridge reference; optional 1-word variable; if omitted ICR is set to zero and space for the file may be allocated to any cartridge; if positive, cartridge is identified by cartridge reference number, if negative, by logical unit number.

IDCBS — DCB buffer size; optional 1-word variable; It has already been described in detail in section A3.6.

(ii) OPEN : A call to OPEN opens a file for access; the file must have been created before. Files may be opened for exclusive use of the calling program or for non-exclusive use of up to seven programs. A file may be opened for update or for standard sequential write.

CALL OPEN ( IDCB, IERR, NAME, IOPTN, ISECU, ICR, IDCBS)

Parameters

IDCB — is an explained in section A3.5

IERR — Error return; 1-word variable in which negative error code is returned if unsuccessful, file type if successful.

NAME — As explained in ~~section~~ part (i) of section A3.7.

IOPTN — Open options; optional word variable set to octal value to specify non-standard opens. If omitted or set to zero, the file is opened by default.

ISECU — Security code; described in section A3.3

ICR — Cartridge reference; described in part (i) of section A3.7.

IDCBS — DCB buffer size; described in section A3.5.

(iii) CLOSE : To close a file after use, the routine, Close, is called. The file remains in the system available to other programs following the close; The Data Control Block is freed for association with other files. A disc file opened for exclusive use of calling program may be truncated to its actual length.

CALL CLOSE (IDCB, IERR, ITRUN)

IDCB — As explained in section A3.5

IERR — Error return; 1-word variable in which negative error code is returned if truncation unsuccessful, only required when ITRUN is specified.

ITRUN — Truncation; optional 1-word variable containing integer number of blocks to be deleted from the file at closing; if omitted or zero, the file is closed without truncation; if negative, only extents are truncated.

3.8 FILE ACCESS : Information in files is accessed by the READF and WRITF routines. Calls to these routine are the same whether the file is a device (type 0) or a disc file (type 1 and above). The normal mode of access of file type 3 and above is sequential. Such files are created with an

end-of-file in the first record. The first record written overrides the end of file and a new end-of-file is written immediately following the record. As each subsequent record is written, the process is repeated so that the end-of-file always follows the last record written. For file types 1 and 2, random access is the normal mode. The end-of-file is written at the end of the file according to the file size at creation. Since each record is a fixed length determined at creation, the file is easily positioned to a particular record. Generally one record is written or read at a time, although more may be transferred when accessing a type 1 file.

3.8.1 READF : This routine reads a record from an open file to the user buffer. Either one full record or a specified number of words is read. The record to be read may be the record on which the file is currently positioned or, for type 1 and 2 files, it may be any specified record.

CALL READF ( IDCB, IERR, IBUF, IL, LEN, NUM)

Parameters

| | |
|---|---|
| IDCB | as described in section A3.5. |
| IERR | Error return; 1-word variable in which negative error code is returned. |
| IBUF | User buffer, array into which the record is read; it should be large enough to contain the record; if IL is specified, it should be length IL. |
| IL | Length in words; optional 1-word variable specifying number of words to be read; should not be omitted for type 0 files, for other files, 1 record is read if IL if omitted. |
| LEN | Words read; optional 1-word variable in which actual number of words read is returned; set to - 1 if end-of-file read; if omitted, information not specified. |

NUM — Record number, optional 1-word variable set to record number to be read if positive, to number of records to backspace if negative; used only for type 1 and 2 files; if omitted, record at current position is read.

3.8.2 WRITF : A call to this routine transfers a record from the user's buffer to an open file. For files of type 0 or type 3 and above, a specified number of words is written. Type 1 files are written in blocks of 128 words. For type 2 files the exact record length specified at creation is written.

CALL WRITF ( IDCB, IERR, IBUF, IL, NUM)

Parameters

IDCB — as described in section A3.5.

IERR — Error return, 1-word variable in which negative error code is returned.

IBUF — User buffer; array containing the record to be written it should be large enough to contain the largest record to be written.

IL — Length in words, optional 1-word variable specifying number of words to be written; if omitted, one record is written to type 1 and 2 files, zero length record to other file types.

NUM — Record number, optional 1-word variable containing record number to be written if positive, number of records to backspace if negative; used only for type 1 and 2 files; if omitted, record is written to current file position.

```
If you want to continue,
please give your query . If you want to terminate,
please give $ in in First colum .
SELECT TITLE STATUS CALLNO FROM BOOKINLIB WHERE PART AUTHOR = [ HABERMAN ] $

TITLE:-       INTRODUCTION TO OPERATING SYSTEM DESIGN

STATUS:-       INLIB

CALLNO:-       681.3.06H113IN4

QUERY PROCESSED
```

```
If you want to continue,
please give your query . If you want to terminate,
please give $ in in first column .
SELECT TITLE AUTHOR ABSTRACT FROM BOOKINLIB WHERE ACCNO = [  115075  ]  $

TITLE:-      INTRODUCTION TO OPERATING SYSTEM DESIGN

AUTHOR:-      HABERMAN A.N.

ABSTRACT:-      PROCESSOR MANAGEMENT, DEVICE MANAGEMENT,MEMORY MANAGEMENT

QUERY PROCESSED
```

```
If you want to continue,
please give your query . If you want to terminate,
please give $ in in First column .
SELECT LIBDIV STATUS FROM BOOKINLIB WHERE TITLE = I ART OI COMPUTER PROGKAMMING ) $

LIBDIV/-        COMSC.

STATUS:-        LOST

QUERY PROCESSED
```

```
If you want to continue,
please give your query . If you want to terminate,
please give $ in in First column .
SELECT TITLE YEAR MONTH FROM JFLDSUBFLD WHERE FIELD = [ COMPUTER SCIENCE ] $

TITLE:-      COMPUTER DESIGN

YEAR:-       1977

MONTH:-      NOVEMBER

TITLE:-      COMMUNICATIONS OF THE ACM

YEAR:-       1975

MONTH:-      JANUARY

TITLE:-      COMPUTER JOURNAL

YEAR:-       1978

MONTH:-      FEBRUARY

TITLE:-      COMPUTER LANGUAGES

YEAR:-       1978

MONTH:-

TITLE:-      COMPUTER PHYSICS COMMUNICATIONS
```

YEAR:- 1976

MONTH:- MARCH

TITLE:- COMPUTER GRAPHICS & IMAGE.PROCESSING

YEAR:- 1981

MONTH:- JANUARY

TITLE:- COMMUNICATIONS IN MATHEMETICAL PHYSICS

YEAR:- 1981

MONTH:- JANUARY

TITLE:- COMPUTING REVIEWS

YEAR:- 1981

MONTH:- JANUARY

TITLE:- COMMUNICATIONS OF THE ACM

YEAR:- 1978

MONTH:- JANUARY

TITLE:- COMPUTER AND BIOMEDICAL RESEARCH

YEAR:- 1978

MONTH:- FEBRUARY

QUERY PROCESSED

```
If you want to continue,
please give your query . If you want to terminate,
please give $ in in First column .
SECOUN ISSUEADDR CARDNO FROM CARDDETAIL WHERE ISSUENAME = ' DAS PAMPA '  $

ISSUEADDR:-       303 JHELUM HOSTEL J N U

CARDNO:-       B.4-5-82 3

ISSUEADDR:-       303 JHELUM HOSTEL J N U

CARDNO:-       C.2-36.2D 2
COUNT=    2

QUERY PROCESSED
```

```
If you want to continue;
please give your query . If you want to terminate,
please give $ in in First colunn .
SELECT TITLE STATUS FROM JOURNAL WHERE YEAR = [ 1981 ]  $

TITLE:-      COMPUTER GRAPHICS & IMAGE PROCESSING

STATUS:-      INLIB

TITLE:-      COMPUTING REVIEWS

STATUS:-      INLIB

TITLE:-      COMMUNICATIONS OF THE ACM

STATUS:-      INLIB

QUERY PROCESSED
```

```
If you want to continue,
please give your query . If you want to terminate,
please give $ in in First column .
SELECT TITLE FROM JFLDSUBFLD WHERE FIELD = [ COMPUTER SCIENCE ]  $

TITLE:-      COMPUTER DESIGN

TITLE:-      COMMUNICATIONS OF THE ACM

TITLE:-      COMPUTER JOURNAL

TITLE:-      COMPUTER LANGUAGES

TITLE:-      COMPUTER PHYSICS COMMUNICATIONS

TITLE:-      COMPUTER GRAPHICS & IMAGE PROCESSING

TITLE:-      COMMUNICATIONS IN MATHEMETICAL PHYSICS

TITLE:-      COMPUTING REVIEWS

TITLE:-      COMMUNICATIONS OF THE ACM

TITLE:-      COMPUTER AND BIOMEDICAL RESEARCH

QUERY PROCESSED
```

```
If you want to continue,
please give your query . If you want to terminate,
please give $ in in First colvan .
SECOUN TITLE PUBNAME STATUS FROM JOURNAL WHERE YEAR  = [ 1981  ]  $

TITLE:-      COMPUTER GRAPHICS & IMAGE PROCESSING

AUTHOR:-      ACADEMIC PRESS

STATUS:-      INLIB

TITLE:-      COMPUTING REVIEWS

AUTHOR:-      ASSOCIATION OF COMPUTING M

STATUS:-      INLIB

TITLE:-      COMMUNICATIONS OF THE ACM

AUTHOR:-      ASSOCIATION OF COMPUTING M

STATUS:-      INLIB
COUNT=   3

QUERY PROCESSED
```

```
If you want to continue,
please give your query.. If you want to terminate,
please give $ in in First colUMn .
SECOUN TITLE FROM JFLDSUBFLD WHERE FILLD = [ COMPUTER SCIENCE ]  $

TITLE:-      COMPUTER DESIGN

TITLE:-      COMMUNICATIONS OF THE ACM

TITLE:-      COMPUTER JOURNAL

TITLE:-      COMPUTER LANGUAGES

TITLE:-      COMPUTER PHYSICS COMMUNICATIONS

TITLE:-      COMPUTER GRAPHICS & IMAGE PROCESSING

TITLE:-      COMMUNICATIONS IN MATHEMLTICAL PHYSICS

TITLE:-      COMPUTING REVIEWS

TITLE:-      COMMUNICATIONS OF THE ACM

TITLE:-      COMPUTER AND BIOMEDICAL RESEARCH
COUNT=  10

QUERY PROCESSED
```

```
If you want to continue,
please give your query . If you want to terminate,
please give $ in in First colum .
SELECT TITLE YEAR FROM FIELDSUBFLD WHERE PART SUBFIELD = [ ARTIFICIAL INTELLIGENCE ] $

TITLE:-      COMPUTER JOURNAL

YEAR:-       1978

TITLE:-      COMPUTER GRAPHICS & IMAGE PROCESSING

YEAR:-       1981

QUERY PROCESSED
```

```
If you want to continue,
please oive your query . If you want to terminate,
please give $ in in First coluan .
SELECT AUTHOR INSTITUTE YEAR FROM THESIS WHERE PART TITLE = [ INTEL 8085 ]  $

AUTHOR:-       RAMESH PRASAD

INSTITUTE:- SC&SS J.N.U. NEW DELHI

YEAR:-       1980

QUERY PROCESSED
```

```
If you want to continue,
please oive your query . If you want to terminate,
please give $ in in first colvan .
SELECT TITLE YEAR STATUS FROM JOURNAL WHERE PART PUBNAME = [ COMPUTING MACHINERY ]    $

TITLE:-      COMMUNICATIONS OF THE ACM

YEAR:-       1975

STATUS:-      INLIB

TITLE:-      COMPUTING REVIEWS

YEAR:-       1981

STATUS:-      INLIB

TITLE:-      COMMUNICATIONS OF THE ACM

YEAR:-       1981

STATUS:-      INLIB

QUERY PROCESSED
```

```
If you want to continue,
please give your query . If you want to terminate,
please give $ in in First column .
SELECT TITLE INSTITUTE FROM THESIS WHERE PART AUTHOR = [  BANSAL  ]  $

TITLE:-      ECMIX APORTABLE SOFTWARE SYSTEM FOR MIXA

INSTITUTE:- SC&SS J.N.U. NEW DELHI

QUERY PROCESSED
```

```
If you want to continue,
please give your query . If you want to terminate,
please give $ in in First column .
SELECT TITLE AUTHOR STATUS WHERE CALLNO = [ 681.4.06K78AR1  ]  $
Wrong Keyword after 1st Keyword  OR  Identifier        missing OR Wrong Keyword after 2nd Keyword
Either Identifier missing after 3rd Keyword OR          = is missing
Either Identifier missing after 3rd Keyword OR          = is missing
QUERY NOT PROPERLY ENDED
Sorry. Your Query is Wrong
```

```
If you want to continue,
please give your query . If you want to terminate,
please give $ in in First column .
SELECT TITLE AUTHOR STATUS FROM BOOKINFIB WHERE CALLNO = [ 681.4.06K78AR1 ] $
LITERAL DOES NOT EXIST IN DATA FILE
```

```
If you want to continue,
please give your query . If you want to terminate,
please give $ in in First column .
 CREATE SUPDETAIL ( SUPNUM NUMRIC [ 0001 ] [ 40 ] SUPNUM NUMRIC [ 0041 ] [ 10 ] SUPADDR ALFNUM [ 0051 ] [ 60 ]  )  $
```

```
If you want to continue,
please give your query . If you want to terminate,
please give $ in in First column .
CREATE JURNLDTAIL ( NAME ALPHA [ 0001 ] [ 40 ] YEAR NUMRIC [ 0041 ] [ 04 ] FIELD ALPHA [ 0045 ] [ 30 ] ) $
```

```
If you want to continue,
please give your query . If you want to terminate,
please give $ in in First column .
CREATE BOOK ( ACCNO NUMRIC [ 0001 ] [ 06 ] CALLNO ALFNUM [ 0007 ] [ 20 ] PRICE NUMRIC [ 0027 ] [ 08 ] ) $
```

```
If you want to continue,please give your      query.
 If you want to terminate,please give $ in first          column
DEFVEW BUKLIST ( TYTLE ( BOOKINLIBTITLE ) AUTHOR ( BOOKINLIBAUTHOR ) PRICE ( BOOKINLIBPRICE )  STATUS ( BOOKINLIBSTATUS ) ) ;
 Query is syntactically correct
QUERY IS PROCESSED
```

```
f you want to continue,please give your      query.
If you want to terminate,please give $ in first          column
ODIFY BOOKINLIB SET STATUS = [ INLIB ] WHERE TITLE = [ PRINCIPLES OF COMPILER DESIGN ] AND AUTHOR = [ AHO A.V. & ULMAN J.D. ] $
Tuple before update is :-
PRINCIPLES OF COMPILER DESIGN          AHO A.V. & ULMAN J.D.          2 500       LOST  COMSC.  681.3.02AH68PR2     115078CO
pdated record
PRINCIPLES OF COMPILER DESIGN          AHO A.V. & ULMAN J.D.          2 500       INLIB COMSC.  681.3.02AH68PR2     115078CO
UERY PROCESSED
```

```
If you want to continue,please give your      query.
 If you want to terminate,please give $ in first          column
DELETE BOOKINLIB WHERE TITLE = [ MINI/MICRO COMP. ] AND AUTHOR = [ CRAFT ] $
QUERY PROCESSED
```

```
If you want to continue,please give your      query.
 If you want to terminate,please give $ in first          column
INSRTN BOOKINLIB WHERE TITLE AUTHOR VOLUME EDITION [ OPERATING SYSTEM ] [ GHOSH ] [ 2 ] [ 3 ] $

  Tuple to be inserted is :-
   OPERATING SYSTEM                          GHOSH                          2 3

  Tuple to be inserted is :-
   NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
   UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU
QUERY IS PROCESSED
```

```
If you want to continue,please give your      query.
 If you want to terminate,please give $ in first          column
SUSPND BOOKINLIB $
QUERY IS PROCESSED
```

```
If you want to continue,please give your      query.
 If you want to terminate,please give $ in first       .   column
SELECT TITLE AUTHOR FROM BOOKINLIB WHERE ACCNO = [ 115078 ] $
RELATION IS SUSPENDED,QUERY CANNOT BE PROCESSED
```

```
If you want to continue,please give your      query.
 If you want to terminate,please give $ in first         column
RESTOR BOOKINLIB $
Query is syntactically correct
QUERY IS PROCESSED
```

```
If you want to continue,please give your      query.
 If you want to terminate,please give $ in first      .   column
SELECT TITLE AUTHOR FROM BOOKINLIB WHERE ACCNO = [ 115078 ] $

TITLE:-      PRINCIPLES OF COMPILER DESIGN

AUTHOR:-      AHO A.V. & ULMAN J.D.

QUERY PROCESSED
```

```
If you want to continue,please give your      query.
 If you want to terminate,please give $ in first          column
GRANT DELETE MODIFY ON BOOKINLIB TO [ Y88123 ] $
QUERY PROCESSED
```

```
If you want to continue,please give your      query.
 If you want to terminate,please give $ in first          column
GRANT DELETE MODIFY ON BOOKINLIB TO [ Z01234 ] $
 You can't give your grant right to user :- Z01234
```

```
If you want to continue,please give your      query.
 If you want to terminate,please give $ in first          column
REVOKE DELETE MODIFY ON BOOKINLIB FROM [ Y98123 ] $
QUERY PROCESSED
```

```
&CREAT T=00004 IS ON CR00016 USING 00003 BLKS R=0021

0001   FTN4,L
0002         PROGRAM KREAT ( ), THIS PROGRAM IS TO CREAT NEW FILE
0003         INTEGER IDCB(144),NAME(3),ISIZ(2),ISECU(3),IBUF(40)
0004         ISIZ(1)=1500
0005         ISIZ(2)=40
0006         IBLNK=000040B
0007         WRITE(1,1)
0008   1     FORMAT('GIVE FILE NAME:_')
0009         READ(1,20)NAME
0010         WRITE(1,5)
0011   5     FORMAT("GIVE SECURITY CODE:_")
0012         READ(1,10) (ISECU(L),L=1,3)
0013   10    FORMAT(3A2)
0014   20    FORMAT(3A2)
0015         CALL CREAT(IDCB,IERR,NAME,ISIZ,2,0,18)
0016         IF(IERR.LT.0) GO TO 99
0017         DO 145 I=1,3318
0018         CALL SFILL(IBUF,1,80,IBLNK)
0019   C     READ(1,87) (IBUF(KO),KO=1,128)
0020   87    FORMAT(128A2)
0021         CALL WRITF(IDCB,IERR,IBUF,40)
0022         IF(IERR.LT.0) GO TO 99
0023    145  CONTINUE
0024         CALL CLOSE(IDCB)
0025         STOP
0026   99    WRITE(1,100) IERR
0027   100   FORMAT('****************** FMGR ERROR ******************',I4)
0028         END
```

```
&DBMS1 T=00004 IS ON CR00016 USING 00011 BLKS R=0073

0001   FTN4,L
0002         PROGRAM TOKN,3,,,,,,,, THIS IS OUR MAIN PROGRAM
0003   C    ***************************************************************
0004   C     THIS IS OUR MAIN PROGRAM WHICH CALLS SEGMENT LEXCL
0005   C    ***************************************************************
0006         INTEGER ISTB(50,2),ITAB1(30,22),ITAB(30,42),IPRS(50,3),KNAME(3)
0007        *,ICODE(20),ISTB1(20,8),IDCB(144),IBUF(144),NAM5(3),IBUF2(96),
0008        *ISECU(3)
0009         COMMON ITAB,ITAB1,ISTB,MM,M,JM,IPRS,IQRY,IF,ISTB1,KM,ICODE,JLVAR
0010        *,JFLAG,MFLG,IBUF2
0011         DATA ICODF,KNAME,NAM5/8,2HLE,2HXC,2HL ,2HUS,2HRC,2HD /
0012        , DATA ISECU/2H-7,2H56,2H19/
0013         BLANK=020040B
0014   C---------    VALIDITY OF THE USER IS CHECKED  ----------------------
0015         WRITE(1,1)
0016   1     FORMAT(1H ,'GIVE YOUR USER CODE:_')
0017         READ(1,5) (ICODE(KI),KI=1,6)
0018   5     FORMAT(6A1)
0019         CALL HASS(ICODE,6,7,NUMBR)
0020         CALL OPEN(IDCB,IERR,NAM5,1,ISECU)
0021         IF(IERR.NE.2) GO TO 50
0022         CALL SFILL(IBUF2,1,48,BLANK)
0023         CALL READF(IDCB,IERR,IBUF2,48,LEN,NUMBR)
0024         IF(IERR.LT.0) GO TO 50
0025         CALL CODE
0026         WRITE(IBUF,10)(ICODE(IK),IK=1,6)
0027   10    FORMAT(6A1)
0028         IER=0
0029         IF(JSCOM(IBUF2,1,6,IBUF,1,IER)) 40,45,40
0030   40    WRITE(6,41)
0031   41    FORMAT(1H ,'YOU ARE NOT A VALID USER')
0032         STOP
0033   50    WRITE(1,51)IERR
0034   51    FORMAT(1H ,'FMGR ERROR **********',I4)
0035         STOP
0036   C --------- SEGMENT LEXCL IS CALLED ------------------------------------
0037   45    CALL CLOSE(IDCB)
0038         CALL EXEC(ICODF,KNAME)
0039         END
0040   C    ***************************************************************
0041         SUBROUTINE HASS(IBUFF,MAR,IDIV,NUMBR),SUBROUTINE USED FOR HASHING
0042         DIMENSION IBUFF(20), ALPH1(30),ISUM(3)
0043         DOUBLE PRECISION SUM,N ,ALPH2(30)
0044         EQUIVALENCE(SUM,ISUM(2))
0045         SUM=0.D00
0046         DO 10 J=1,MAR
```

```
0047          ALPH1(J) =FLOAT(IBUFF(J))
0048          ALPH2(J)=DBLE(ALPH1(J))
0049          N=(ALPH2(J))**2
0050          SUM=SUM+N
0051    10    CONTINUE
0052          K=ISUM(2)
0053          NUMBR=MOD(K,IDIV)
0054          WRITE(1,20)NUMBR,IBUFF
0055    20    FORMAT(1H ,I7,5X,30A1)
0056          RETURN
0057          END
0058    C     ***************************************************************************
0059          SUBROUTINE CONVR(NUMHAS,IB),CONVERSION FROM I FORMAT TO ASCII
0060          IF(NUMHAS.GT.9) GO TO 50
0061          IE=0
0062          CALL SDEA2(NUMHAS,1,2,IE)
0063          IB=NUMHAS
0064          GO TO 85
0065    50    N=MOD(NUMHAS,10)
0066          M=NUMHAS/10
0067          IE=0
0068          CALL SDEA2(N,1,2,IE)
0069          CALL SDEA2(M,1,2,IE)
0070          CALL SMOVE(M,2,2,IB,1)
0071          CALL SMOVE(N,2,2,IB,2)
0072    85    RETURN
0073          END
0074    C***************************************************************************
0075    C          THIS SEGMENT BREAKS THE QUERY INTO TOKENS
0076    C***************************************************************************
0077          PROGRAM LEXCL,5
0078          INTEGER IBUF(256),IBUF1(40),IARA(8,34),ISTB(50,2),IDCB(144)
0079         *,NAM5(3),ITAB1(30,22),LL(30),ITAB(30,42),ICODE(20),ISECU(3),
0080         *BLANK,IBUF2(96),JNAME(3),IPRS(50,3),ISTB1(20,8),ICODH(20)
0081          COMMON ITAB,ITAB1,ISTB,MM,M,JM,IPRS,IQRY,IF,ISTB1,KM,ICODE,NUMBR
0082         *,JFLAG,MFLG,IBUF2
0083    C-------------    THE KEYWORDS AND DELIMITERS USED IN THE QUERIES
0084    C                 ARE GIVEN BELOW ----------------------------------------
0085          DATA IARA,NAM5/1HS,1HE,1HL,1HE,1HC,1HT,1,1,
0086         *1HF,1HR,1HO,1HM,1H ,1H ,1,2,
0087         *1HW,1HH,1HE,1HR,1HE,1H ,1,3,
0088         *1HO,1HN,1H ,1H ,1H ,1H ,1,4,
0089         *1HO,1HR,1HD,1HR,1HB,1HY,1,5,
0090         *1HD,1HE,1HL,1HE,1HT,1HE,1,6,
0091         *1HI,1HN,1HS,1HR,1HT,1HN,1,7,
0092         *1HT,1HO,1H ,1H ,1H ,1H ,1,8,
0093         *1HR,1HE,1HA,1HD,1H ,1H ,1,9,
0094         *1HN,1HU,1HM,1HR,1HI,1HC,1,10,
```

```
0095         *1HC,1HR,1HE,1HA,1HT,1HE,1,11,
0096         *1HA,1HL,1HF,1HN,1HU,1HM,1,12,
0097         *1HD,1HE,1HF,1HV,1HE,1HW,1,13,
0098         *1HS,1HU,1HS,1HP,1HN,1HD,1,14,
0099         *1HR,1HE,1HV,1HO,1HK,1HE,1,15,
0100         *1HM,1HO,1HD,1HI,1HF,1HY,1,16,
0101         *1HS,1HE,1HT,1H ,1H ,1H ,1,17,
0102         *1HG,1HR,1HA,1HN,1HT,1H ,1,18,
0103         *1HS,1HE,1HC,1HO,1HU,1HN,1,19,
0104         *1HT,1HA,1HK,1HI,1HN,1HG,1,20,
0105         *1HW,1HI,1HT,1HH,1H ,1H ,1,21,
0106         *1HA,1HL,1HP,1HH,1HA,1H ,1,22,
0107         *1HP,1HA,1HR,1HT,1H ,1H ,1,23,
0108         *1HA,1HN,1HD,1H ,1H ,1H ,1,24,
0109         *1HR,1HE,1HS,1HT,1HO,1HR,1,25,
0110         *1H=,1H ,1H ,1H ,1H ,1H ,2,26,
0111         *1H>,1H ,1H ,1H ,1H ,1H ,2,27,
0112         *1H=,1H(,1H ,1H ,1H ,1H ,2,28,
0113         *1H*,1H ,1H ,1H ,1H ,1H ,2,29,
0114         *1H*,1H ,1H ,1H ,1H ,1H ,2,30,
0115         *1H(,1H ,1H ,1H ,1H ,1H ,2,31,
0116         *1H(,1H ,1H ,1H ,1H ,1H ,2,32,
0117         *1H),1H ,1H ,1H ,1H ,1H ,2,33,
0118         *1H,,1H ,1H ,1H ,1H ,1H ,2,34,
0119         *2HUS,2HRC,2HD /
0120          DATA ICODF,JNAME,ISECU/8,2HSY,2HTA,2HX ,2H-7,2H56,2H19/
0121          K=0
0122          KM=0
0123          L=0
0124          MM=0
0125          ICC=0
0126          KA=0
0127          JM=0
0128          IBB=0
0129          IC=0
0130          M=0
0131          N=0
0132          IKEY=0
0133          IL=0
0134          BLANK=020040B
0135          IF(MFLG.EQ.1)GO TO 45
0136    C---------        THE QUERY IS READ
0137    45    WRITE(1,1)
0138    1     FORMAT(1H ,'PLEASE DO NOT BREAK A FULL WORD BY A BLANK OR ANY',
0139         */,'SPECIAL CHARACTER    **********************')
0140          WRITE (6,2)
0141    2     FORMAT(1H1,//////,' If you want to continue,please give your
0142         *query.',/,'  If you want to terminate,please give $ in first
```

```
0143          *   column ')
0144           CALL SFILL(IBUF,1,256,BLANK)
0145   3       READ(1,4)IBUF
0146   4       FORMAT(256A1)
0147           WRITE(6,8)IBUF
0148   8       FORMAT(1H ,256A1)
0149   5       K=K+1
0150           IF(IBUF(1).EQ.1H$) GO TO 905
0151           IF(K.EQ.256) GO TO 75
0152           DO 10 I=1,40
0153   10      IBUF1(I)=1H
0154           DO 20 I=K,256
0155           IF(IBUF(I).NE.1H ) GO TO 30
0156   20      CONTINUE
0157   30      J=0
0158           J=J+1
0159           IF(IBUF(I) .EQ.1H$) GO TO 900
0160           IF(IBUF(I).EQ.1HI) GO TO 300
0161           IBUF1(J)=IBUF(I)
0162           DO 50 K=I+1,256
0163           IF(IBUF(K).EQ.1H$) GO TO 900
0164           IF(IBUF(K).EQ.1H )GO TO 150
0165           J=J+1
0166           IBUF1(J) = IBUF(K)
0167   50      CONTINUE
0168   75      IF (IBUF(K-1).EQ.1H$)GO TO 900
0169           WRITE(6,760)
0170   760     FORMAT(1H ,'INPUT ERROR')
0171           STOP
0172   99      WRITE(6,100) IERR
0173   100     FORMAT(1H ,'******************* FMGR ERROR **************',I4)
0174           STOP
0175   C*****************************************************************************
0176   C-----------------------------------------------------------------------------
0177   C       THIS PART CHECKS FOR IDENTIFIERS,KEYWORDS & DELIMITERS
0178   C       CODE FOR KEYWORDS =1
0179   C       CODE FOR DELIMITERS=2
0180   C       CODE FOR IDENTIFIERS=4
0181   C-----------------------------------------------------------------------------
0182    150    LEN=J
0183           DO 160 IJ=1,34
0184           DO 155 JJ=1,J
0185           IF(IBUF1(JJ).NE.IARA(JJ,IJ)) GO TO 160
0186   155     CONTINUE
0187           IKEY=IKEY+1
0188           JJ=JJ-1
0189           IF(JJ.NE.LEN) GO TO 160
0190   C-----------------------------------------------------------------------------
```

```
0191  C      THIS PART WRITES CODE FOR KEYWORD & DELIMS IN ISTB,THE SYMBOL
0192  C      TABLE
0193  C-------------------------------------------------------------------------
0194           IF(IKEY.NE.1) GO TO 156
0195           CALL CODE
0196           WRITE(ICODH,210) (IBUF1(LK),LK=1,6)
0197  210      FORMAT(6A1)
0198           DO 211 IVAR=1,11
0199           IBEG=7+(IVAR-1)*8
0200           IEND=13+(IVAR-1)*8
0201           IF(JSCOM(ICODH,1,6,IBUF2,IBEG,IER)) 211,212,211
0202  211      CONTINUE
0203           STOP
0204  212      CALL SMOVE(IBUF2,IEND,IEND+1,ILVAR,1)
0205           CALL CODE
0206           READ(ILVAR,213)LIVAR
0207  213      FORMAT(I2)
0208           IF(LIVAR.EQ.1) GO TO 154
0209           JFLAG=0
0210           GO TO 156
0211  154      JFLAG=1
0212  156      N=7
0213           MM=MM+1
0214           NN=1
0215           ISTB(MM,NN) = IARA(N,IJ)
0216           ISTB(MM,NN+1)=IARA(N+1,IJ)
0217          IF((ISTB(1,1).EQ.1).AND.((ISTB(1,2).EQ.15).OR.(ISTB(1,2).EQ.18).
0218         *OR.(ISTB(1,2).EQ.11)))GO TO 189
0219           GO TO 5
0220  160      CONTINUE
0221           M=M+1
0222           IAA=M
0223           IF(M.NE.1) GO TO 190
0224  165      DO 170 KK=1,20
0225  170      ITAB1(M,KK)=IBUF1(KK)
0226           KK=KK-1
0227           IL=IL+1
0228           LL(IL)=KK
0229           ITAB1(M,21)=4
0230           ITAB1(M,22)=M
0231           NANA=M
0232  185      MM=MM+1
0233           NN=1
0234           ISTB(MM,NN)=ITAB1(M,21)
0235           ISTB(MM,NN+1)=ITAB1(M,22)
0236           M=IAA
0237           GO TO 5
0238  189      KM=KM+1
```

```
0239          DO 191 IU=1,6
0240   191    ISTB1(KM,IU)=IBUF1(IU)
0241          ISTB1(KM,IU)=ISTB(MM,NN)
0242          ISTB1(KM,IU+1)=ISTB(MM,NN+1)
0243          GO TO 5
0244   190    IK=0
0245          DO 200 IC=1,M-1
0246          IK=IK+1
0247          KK=LL(IK)
0248          DO 195 IB=1,KK
0249          IF(IBUF1(IB).NE.ITAB1(IC,IB)) GO TO 200
0250   195    CONTINUE
0251          IB=IB-1
0252          IF(IB.NE.LEN) GO TO 200
0253          M=IC
0254          KK=IB
0255          IAA=IAA-1
0256          GO TO 185
0257   200    CONTINUE
0258          GO TO 165
0259   300    DO 430 IR=I,256
0260          KA=KA+1
0261          IBUF1(KA)=IBUF(IR)
0262          GO TO 312
0263   305    DO 310 JK=KA-1,40
0264   310    IBUF1(JK)=1H
0265          LI=1
0266          GO TO 322
0267   312    IF(KA.EQ.1) GO TO 430
0268          IF(IBUF1(KA).EQ.1H]) GO TO 315
0269          IF(IBUF1(KA).EQ.1H$) GO TO 380
0270          IF(IBUF1(KA).EQ.1H ) GO TO 400
0271          IBB=0
0272          GO TO 430
0273   315    IF(IBUF1(KA-1).EQ.1H[) GO TO 390
0274          IF(IBUF1(KA-1).EQ.1H ) GO TO 305
0275          LI=0
0276          DO 320 LK=KA,40
0277   320    IBUF1(LK)=1H
0278   322    DO 325 KL=1,KA-2
0279   325    IBUF1(KL)=IBUF1(KL+1)
0280          IBUF1(KA-1)=1H
0281          KA=KA-(2+LI)
0282          IF(JM.EQ.0) GO TO 350
0283          DO 340 IA=1,JM
0284          DO 330 IB=1,40
0285          IF(ITAB(IA,IB).NE.IBUF1(IB)) GO TO 340
0286   330    CONTINUE
```

```
0287          TEMP=ICC
0288          N=40
0289          ICC=ITAB(IA,N+2)
0290          MM=MM+1
0291          NN=1
0292          GO TO 360
0293  340      CONTINUE
0294  C*****************************************************************************
0295  C                LITERAL TABLE ENTRY
0296  C                CODE FOR LITERAL=3
0297  C*****************************************************************************
0298  350     MM=MM+1
0299          JM=JM+1
0300          ICC=ICC+1
0301          TEMP=ICC
0302          NN=1
0303          N=40
0304          DO 352 IVAX=1,40
0305  352     ITAB(JM,IVAX)=1H
0306          DO 355 JJ=1,N
0307  355      ITAB(JM,JJ)=IBUF1(JJ)
0308          ITAB(JM,N+1)=3
0309          ITAB(JM,N+2)=JM
0310  C*****************************************************************************
0311  C                SYMBOL TABLE ENTRY
0312  C*****************************************************************************
0313  360     ISTB(MM,NN)=3
0314          ISTB(MM,NN+1)=ICC
0315  C*****************************************************************************
0316         KA=0
0317          ICC=TEMP
0318          K=IR
0319          GO TO 5
0320  380     WRITE(6,385)
0321  385      FORMAT(1H ,'] IS MISSING FOR LITERAL &YOUR QUERY TERMINATED')
0322  396     STOP
0323  390     WRITE(6,395)
0324  395     FORMAT(1H ,'[ & ] IS PROPER BUT NO LIT. IN BETWEEN')
0325          STOP
0326  400     IF(IBUF1(KA-1).EQ.1H[) GO TO 410
0327          IBB=IBB+1
0328          IF(IBB.EQ.1) GO TO 430
0329  410     KA=KA-1
0330  430     CONTINUE
0331          K=IR-1
0332          GO TO 5
0333  900     CALL EXEC(ICODF,JNAME)
0334  905     END
```

```
0335          PROGRAM SYTAX,5
0336    C*************************************************************************
0337    C         FROM THIS SEGMENT A BRANCH IS MADE TO THE APPROPRIATE
0338    C         SEGMENT FOR SYNTAX CHECKING
0339    C*************************************************************************
0340          DIMENSION ISTB(50,2),IPRS(50,3),ITAB1(30,22),ITAB(30,42)
0341         *,LNAM1(3),LNAM2(3),LNAM3(3),LNAM4(3),LNAM5(3),LNAM6(3),LNAM7(3),
0342         *LNAM8(3),LNAM9(3),MNAM1(3)
0343          COMMON ITAB,ITAB1,ISTB,MM,MT,ML,IPRS,IQRY,IF
0344    C-----------     NAMES OF SEGMENTS CALLED BY SYTAX ARE GIVEN     ---------
0345          DATA LNAM1,ICODF/2HCH,2HEC,2HK ,8/
0346          DATA LNAM2/2HDE,2HLT,2H  /
0347          DATA LNAM3/2HIN,2HSR,2H  /
0348          DATA LNAM4/2HKR,2HEA,2HT /
0349          DATA LNAM5/2HMO,2HDF,2HY /
0350          DATA LNAM6/2HGR,2HAN,2HT /
0351          DATA LNAM7/2HSS,2HPN,2HD /
0352          DATA LNAM8/2HRE,2HST,2HR /
0353          DATA LNAM9/2HRE,2HVO,2HK /
0354          DATA MNAM1/2HDE,2HFV,2HW /
0355          IF(ISTB(1,1).NE.1) GO TO 10
0356    C-----------     CHECK IS MADE TO BRANCH TO APPROPRIATE SEGMENTS     -----
0357          IF((ISTB(1,2).EQ.1).OR.(ISTB(1,2).EQ.19)) GO TO 30
0358          IF(ISTB(1,2).EQ.11) GO TO 50
0359          IF(ISTB(1,2).EQ.6) GO TO 40
0360          IF(ISTB(1,2).EQ.7) GO TO 45
0361          IF(ISTB(1,2).EQ.16) GO TO 55
0362          IF(ISTB(1,2).EQ.18) GO TO 60
0363          IF(ISTB(1,2).EQ.14) GO TO 65
0364          IF(ISTB(1,2).EQ.15) GO TO 70
0365          IF(ISTB(1,2).EQ.13) GO TO 75
0366          IF(ISTB(1,2).EQ.25) GO TO 68
0367    10    WRITE(1,2)
0368    2     FORMAT(1H ,'SYNTAX ERROR,CHECK FIRST WORD OF YOUR QUERY ')
0369          STOP
0370    C-----------     THE APPROPRIATE SEGMENT IS CALLED     ----------------
0371    30    CALL EXEC(ICODF,LNAM1)
0372    40    CALL EXEC(ICODF,LNAM2)
0373     45   CALL EXEC(ICODF,LNAM3)
0374    50    CALL EXEC(ICODF,LNAM4)
0375    55    CALL EXEC(ICODF,LNAM5)
0376     60   CALL EXEC(ICODF,LNAM6)
0377    65    CALL EXEC(ICODF,LNAM7)
0378    68    CALL EXEC(ICODF,LNAM8)
0379     70   CALL EXEC(ICODF,LNAM9)
0380    75    CALL EXEC(ICODF,MNAM1)
0381          END
0382    C*************************************************************************
```

```
0383  C             THIS SEGMENT IS CALLED BY SEGMENT SYTAX
0384  C*************************************************************************
0385        PROGRAM CHECK,5,,,,,,,, THIS IS THE PARSER FOR QUERY FACILITIES
0386        DIMENSION IA(50,2),IB(50,3),IE(7,3),JNAM1(3),ITAB1(30,22),ITAB(30
0387       *,42),KINM(3)
0388        COMMON ITAB,ITAB1,IA,MM,MT,ML,IB,IQRY,IF
0389        DATA JNAM1,KINM,ICODF/2HQR,2HYA,2H  ,2HCL,2HEA,2HR  ,8/
0390        M1=1
0391        N1=1
0392        IX1=0
0393  C----------   IB IS THE PARSED QUERY TABLE GENERATED    ----------------
0394  C----------   ZERO OR ONE IS ENTERED IN IB DEPENDING ON  ---------------
0395  C             WHETHER EACH WORD IN THE QUERY IS SYNTACTICALLY ----------
0396  C             CORRECT OR NOT                            ----------------
0397  C    IX1 KEEPS  COUNT OF (1,2) IN ISTB
0398        IY1=0
0399  C    IY1 KEEPS COUNT OF (1,3) IN ISTB
0400        IP1=0
0401  C    IP1 KEEPS COUNT OF (2,34) IN ISTB
0402        IQ1=0
0403  C    IQ1 KEEPS COUNT OF (2,26) IN ISTB
0404        IR1=0
0405  C IR1 KEEPS COUNT OF (4,)AFTER (1,3) IN ISTB
0406        IZ1=0
0407  C    IZ1=1 TELLS WHEN QUERY ENDS OR WHEN LITERAL COMES
0408        IS1=0
0409  C    IS1 KEEPS COUNT OF (1,5) IN ISTB
0410        IR=0
0411  C    IR KEEPS  COUNT OF (4,) AFTER (1,2) IN ISTB
0412        IQRY=0
0413  C    IQRY TELLS THE TYPE OF SELECT QUERY
0414        IF=0
0415  C    IF TELLS THE NO OF ATTRIBUTES AFTER SELECT
0416        DO 15 I=1,MM
0417        DO 15 J=1,2
0418        IB(I,J)=IA(I,J)
0419  15    CONTINUE
0420        IF(IB(1,2).EQ.19) IQRY=4
0421  10    IB(M1,3)=0
0422  20    M1=M1+1
0423        IF(M1.GT.MM) GO TO 75
0424        IF(M1.NE.2) GO TO 40
0425  C----------   FIRST ID AFTER FIRST KEYWORD IS CHECKED   -------------
0426        IF(IA(M1,N1).NE.4) GO TO 11
0427        IF=IF+1
0428        GO TO 10
0429  11    WRITE(6,21)
0430  30    IB(M1,3)=1
```

```
0431          GO TO 20
0432    C-----------   CHECK FOR KEYWORD 'PART' IN THE QUERY     ----------
0433    40    IF((IY1.EQ.1).AND.(IR1.EQ.0).AND.(IA(M1,N1).EQ.1)) GO TO 130
0434    C-----------  SYNTAX CHECK AFTER KEYWORD 'WHERE'         ----------
0435          IF(IY1.EQ.1) GO TO 95
0436    C-----------   SYNTAX CHECK AFTER KEYWORD ' FROM'        ----------
0437          IF((IX1.EQ.1).AND.(IA(M1,N1).EQ.4)) GO TO 25
0438    C-----------   CHECK FOR IDENTIFIERS IN THE QUERY        ----------
0439          IF(IA(M1,N1).EQ.4) GO TO 46
0440    45    IF(IA(M1,N1).NE.1) GO TO 70
0441          IF(IZ1.EQ.1) GO TO 115
0442          IF(IA(M1,N1+1).NE.2) GO TO 50
0443          IX1=IX1+1
0444          IF(IX1.GT.1) GO TO 30
0445          GO TO 10
0446    46    IF=IF+1
0447          GO TO 10
0448    25    IR=IR+1
0449          IF(IR.EQ.1)GO TO 41
0450          IF(IR.EQ.2)GO TO 42
0451          WRITE(6,32)
0452          GO TO 30
0453    41    IF(IQRY.EQ.4) GO TO 10
0454          IQRY=1
0455          GO TO 10
0456    42    IQRY=2
0457          GO TO 10
0458    50    IF((IX1.EQ.1).AND.(IA(M1,N1+1).EQ.3).AND.(IR.GT.0)) GO TO 60
0459          WRITE(6,22)
0460          GO TO 30
0461    60    IY1=IY1+1
0462          IF(IY1.GT.1) GO TO 30
0463          GO TO 10
0464    70    IF((IY1.EQ.1).AND.(IA(M1,N1).EQ.2)) GO TO 80
0465          IF((IQ1.EQ.1).AND.(IA(M1,N1).EQ.3)) GO TO 110
0466    75    IF((IZ1.EQ.1).AND.(M1.GT.MM)) GO TO 200
0467          IF((IZ1.EQ.2).AND.(M1.GT.MM)) GO TO 200
0468          IF((IZ1.EQ.1).AND.(IA(M1,N1).EQ.1)) GO TO 115
0469          IF(M1.GT.MM) GO TO 120
0470          WRITE(6,23)
0471          GO TO 30
0472    C-----------   CHECK FOR '=' AFTER 'WHERE' IN QUERY      ----------
0473    80    IF(IA(M1,N1+1).EQ.26) GO TO 90
0474    C-----------   CHECK FOR ',' AFTER 'WHERE' IN QUERY      ----------
0475          IF(IA(M1,N1+1).EQ.34) GO TO 100
0476          WRITE(6,24)
0477          GO TO 30
0478    90    IF((IR1.EQ.1).AND.(IQ1.EQ.0)) GO TO 85
```

```
0479          IF((IR1.EQ.2).AND.(IP1.EQ.1)) GO TO 85
0480          WRITE(6,23)
0481          GO TO 30
0482    85    IQ1=IQ1+1
0483          IF(IQ1.EQ.1)GO TO 10
0484          WRITE(6,26)
0485          GO TO 30
0486    95    IF(IA(M1,N1).NE.4) GO TO 45
0487          IR1=IR1+1
0488    C-----------------    CHECK THE 'WHERE CLAUSE' IN QUERY    ----------------
0489          IF(IR1.EQ.1) GO TO 10
0490          IF((IR1.EQ.2).AND.(IS1.EQ.1)) GO TO 110
0491          IF((IR1.EQ.3).AND.(IQ1.EQ.1)) GO TO 10
0492          IF((IR1.EQ.4).AND.(IP1.EQ.2)) GO TO 110
0493          IF((IR1.EQ.2).AND.(IP1.EQ.1)) GO TO 10
0494    100   IF(IR1.EQ.1) GO TO 105
0495          IF((IR1.EQ.3).AND.(IQ1.EQ.1)) GO TO 105
0496          WRITE(6,27)
0497          GO TO 30
0498    105   IP1=IP1+1
0499          GO TO 10
0500    115   IF(IA(M1,N1+1).NE.5) GO TO 31
0501          IS1=IS1+1
0502          IQRY=3
0503          GO TO 10
0504    31    WRITE(6,29)
0505          GO TO 30
0506    110   IZ1=IZ1+1
0507          IF((IZ1.EQ.1).AND.(IP1.EQ.0)) GO TO 10
0508          IF((IZ1.EQ.1).AND.(IP1.EQ.2)) GO TO 10
0509          IF((IZ1.EQ.2).AND.(IR1.EQ.2)) GO TO 10
0510          WRITE(6,28)
0511          GO TO 30
0512    130   IF(IA(M1,N1+1).EQ.23) GO TO 140
0513          WRITE(6,23)
0514          GO TO 30
0515    140   IF((MM-M1).EQ.7) GO TO 141
0516          IF((MM-M1).EQ.3) GO TO 142
0517          GO TO 125
0518    141   IQRY=5
0519          GO TO 126
0520    142   IQRY=6
0521    126   IB(M1,3)=0
0522          K1=M1
0523          DO 145 I=1,MM-K1
0524          M1=M1+1
0525          DO 145 J=1,2
0526    145   IE(I,J)=IA(M1,J)
```

```
0527  C--------     SUBROUTINE 'PRT' IS CALLED WHEN THE QUERY HAS 'PART'
0528  C             IN ITS WHERE CLAUSE
0529          CALL PRT(IE,MM-K1,3)
0530          M1=K1
0531          DO 150 I=1,MM-K1
0532          M1=M1+1
0533          IB(M1,3)=IE(I,3)
0534  150     CONTINUE
0535          GO TO 200
0536  21      FORMAT(1H ,'Identifier is missing after 1st KEYWORD')
0537  22      FORMAT(1H ,'Wrong Keyword after 1st Keyword  OR  Identifier
0538         *missing OR Wrong Keyword after 2nd Keyword')
0539  23      FORMAT(1H ,'Either Identifier missing after 3rd Keyword OR
0540         *'' ='' is missing')
0541  24      FORMAT(1H ,'Either '' ='' is  missing OR '','' is missing')
0542  29      FORMAT(1H ,'Correct Keyword missing')
0543  26      FORMAT(1H ,' ''='' occurs more than once')
0544  27      FORMAT(1H ,'Identifier missing after'' = '' OR '','' is
0545         *missing after ''='' OR '','' after ''='' is in wrong place')
0546  28      FORMAT(1H ,'Literal in wrong place')
0547  32      FORMAT(1H ,'Wrong Number Of Identifiers After Keyword ''FROM'' '
0548  120     WRITE(6,5)
0549  5       FORMAT(1H ,'QUERY NOT PROPERLY ENDED ')
0550          GO TO 200
0551  125     WRITE(6,6)
0552  6       FORMAT(1H ,'Query Not Proper.Check Last 7 Words Of Your Query')
0553  C-------------   THE PARSER TABLE IS CHECKED FOR WRONG SYNTAX  --------
0554  200     DO 210 IP=1,MM
0555          IF(IB(IP,3).EQ.1) GO TO 991
0556  210     CONTINUE
0557  C----------      IF NO SYNTAX ERROR OCCURS, SEGMENT QRYA IS CALLED   ---
0558          CALL EXEC(ICODF,JNAM1)
0559  991     WRITE(6,995)
0560  995     FORMAT(1H ,'Sorry, Your Query is Wrong ')
0561          CALL EXEC (ICODF,KINM)
0562          END
0563  C  THIS IS SUBROUTINE 'PRT' CALLED BY SEGMENT 'CHECK'
0564          SUBROUTINE PRT (IX,M,N)
0565          DIMENSION IX(7,3)
0566          M1=0
0567          N1=1
0568  20      M1=M1+1
0569          IF(M1.GT.M) GO TO 100
0570          IF(M.EQ.7) GO TO 1
0571          GO TO 2
0572  1       GO TO(5,15,5,25,35,15,35),M1
0573  2       GO TO (5,25,35),M1
0574  5       IF(IX(M1,N1).EQ.4) GO TO 10
```

```
0575          WRITE(6,41)
0576          GO TO 30
0577    15    IF(IX(M1,N1).NE.1) GO TO 11
0578          IF(IX(M1,N1+1).EQ.24) GO TO 10
0579          WRITE(6,42)
0580          GO TO 30
0581    25    IF(IX(M1,N1).NE.2) GO TO 12
0582          IF(IX(M1,N1+1).EQ.26) GO TO 10
0583          WRITE(6,43)
0584          GO TO 30
0585    35    IF(IX(M1,N1).EQ.3) GO TO 10
0586          WRITE(6,44)
0587          GO TO 30
0588    10    IX(M1,3)=0
0589          GO TO 20
0590    11    WRITE(6,42)
0591          GO TO 30
0592    12    WRITE(6,43)
0593    30    IX(M1,3)=1
0594          GO TO 20
0595    41    FORMAT(1H ,'Identifier missing in query after Keyword  PART ')
0596    42    FORMAT(1H ,'Keyword ''AND'' missing')
0597    43    FORMAT(1H ,' ''='' is missing')
0598    44    FORMAT(1H ,' Literal missing')
0599    100   RETURN
0600          END
0601          PROGRAM KREAT,5
0602          DIMENSION IA(50,2),IB(50,3),ITAB1(30,22),ITAB(30,42),INAM(3),
0603         *ISTB1(20,8),KJNM(3)
0604          COMMON ITAB,ITAB1,IA,MM,M,JM,IB,IQRY,IF,ISTB1
0605          DATA ICODF,INAM,KJNM/8,2HKR,2HET,2H  ,2HCL,2HEA,2HR /
0606          ID=0
0607    C   ID KEEPS COUNT OF IDENTIFIER AFTER ( IN ISTB
0608          IK=0
0609    C   IK KEEPS COUNT OF KEYWORD AFTER ( IN ISTB
0610          IL=0
0611    C    IL KEEPS COUNT OF LITERAL AFTER ( IN ISTB
0612    C-----------   IB IS THE PARSED QUERY TABLE  ----------------------
0613          DO 5 I=1,MM
0614          DO 5 J=1,2
0615    5     IB(I,J)=IA(I,J)
0616          L=1
0617          K=1
0618    10    IB(L,3)=0
0619    20    L=L+1
0620          IF(L.EQ.MM) GO TO 80
0621          IF(L.GT.MM) GO TO 100
0622          IF(L.NE.2) GO TO 40
```

```
0623  C------------   CHECK 2nd WORD OF QUERY TO BE AN IDENTIFIER   ----------
0624  C------------   ENTRY I N IB FOR SYNTACTICALLY CORRECT WORD IS 0,
0625  C               AND FOR A WRONG WORD IS 1                    ----------
0626        IF(IA(L,K).EQ.4) GO TO 10
0627        WRITE(6,51)
0628        GO TO 30
0629  C------------   CHECK FOR '(' AT THE 3rd QUERY WORD          ----------
0630  40    IF(L.NE.3) GO TO 50
0631        IF(IA(L,K).EQ.2) GO TO 45
0632        WRITE(6,52)
0633        GO TO 30
0634  45    IF(IA(L,K+1).EQ.32) GO TO 10
0635        WRITE(6,52)
0636        GO TO 30
0637  C------------   CHECK FOR IDENTIFIER AFTER '(' IN QUERY      ----------
0638  50    IF((IA(L,K).EQ.4).AND.(ID.EQ.0).AND.(IL.EQ.0)) GO TO 60
0639        IF((IA(L,K).EQ.1).AND.(ID.EQ.1)) GO TO 55
0640  C------------   CHECK FOR LITERAL AFTER '(' IN QUERY         ----------
0641        IF((IA(L,K).EQ.3).AND.(IK.EQ.1)) GO TO 70
0642        WRITE(6,54)
0643        GO TO 30
0644  C------------   CHECK FOR THE 'TYPE' OF THE ATTRIBUTE        ----------
0645  55    IF((IA(L,K+1).EQ.10).OR.(IA(L,K+1).EQ.12).OR.(IA(L,K+1).EQ.22))
0646       *GO TO 65
0647        WRITE(6,53)
0648        GO TO 30
0649  60    ID=ID+1
0650        GO TO 10
0651  65    IK=IK+1
0652        GO TO 10
0653  70    IL=IL+1
0654        IF(IL.EQ.1) GO TO 10
0655        IL=0
0656        ID=0
0657        IK=0
0658        GO TO 10
0659  C------------   CHECK FOR ')' AT THE END OF QUERY             ----------
0660  80    IF(IA(L,K).EQ.2) GO TO 85
0661        WRITE(6,59)
0662  30    IB(L,3)=1
0663        GO TO 20
0664  85    IF(IA(L,K+1).EQ.33) GO TO 10
0665        WRITE(6,59)
0666        GO TO 30
0667  51    FORMAT(1H ,'identifier missing')
0668  52    FORMAT(1H ,'Delimiter ( is missing')
0669  53    FORMAT(1H ,'Wrong Keyword')
0670  54    FORMAT(1H ,'Error after ( in your query')
```

```
0671  59     FORMAT(1H ,' '')'' is missing at the end of your query')
0672  100    DO 110 LK = 1,MM
0673         IF(IB(LK,3).EQ.1) GO TO 150
0674  110    CONTINUE
0675  C----------      IF QUERY IS FOUND TO BE SYNTACTICALLY CORRECT,
0676  C                THE SEGMENT 'KRET'IS CALLED
0677         CALL EXEC(ICODF,INAM)
0678  150    WRITE(6,155)
0679  155    FORMAT(1H ,'Sorry, Your Query is Wrong')
0680         CALL EXEC(ICODF,KJNM)
0681         END
0682         PROGRAM INSR,5
0683  C -----* THIS SEGMENT IS TO CHECK SYNTAX FOR INSRTN STATEMENT *
0684         DIMENSION IA(50,2),IB(50,3),ITAB1(30,22),ITAB(30,42),INAM(3),
0685        *LRNAM(3)
0686         COMMON  ITAB,ITAB1,IA,L,M,JM
0687         DATA ICODF,INAM,LRNAM/8,2HIN,2HST,2HN ,2HCL,2HEA,2HR /
0688         DO 8 I=1,L
0689         DO 8 J=1,2
0690   8     IB(I,J)=IA(I,J)
0691  C------**********************************************************
0692  C -----*  VARIABLE ID IS USED TO DENOTE IDENTIFIER IN STATEMENT *
0693  C -----*  VARIABLE LIT IS USED TO DENOTE LITERAL IN STATEMENT   *
0694  C      *  VARIABLE IF  IS USED TO KEEP ACOUNT OF PROPER PLACE
0695  C      *  FOR IDENTIFIER & LITERAL  IN QUERY STATEMENT          *
0696  C      **********************************************************
0697         ID=0
0698         LIT=0
0699         IF=0
0700         DO 140 I=1,L
0701         IF(I.GE.4) GO TO 36
0702         GO TO(10,20,30),I
0703  C -----**********************************************************
0704  C      * NEXT STATEMENT CHECKS FOR FIRST QUERY WORD INSRTN IN    *
0705  C      **************   QUERY STATEMENT   **************************
0706   10    IF((IB(I,1).EQ.1).AND.(IB(I,2).EQ.7)) GO TO 15
0707   12    IB(I,3)=1
0708         GO TO 140
0709   15    IB(I,3)=0
0710         GO TO 140
0711  C -----**** NEXT STATEMENT CHECKS IDENTIFIER IN QUERY STATEMENT  *****
0712   20    IF(IB(I,1).EQ.4) GO TO 15
0713         GO TO 12
0714   30    IF(IB(I,1).EQ.4) GO TO 35
0715         GO TO 12
0716   35    ID=ID+1
0717         GO TO 15
0718   36    IF(IB(I,1).EQ.4) GO TO 40
```

```
0719  C -----*  LITERAL is checked in next statement  *
0720        IF(IB(I,1).EQ.3) GO TO 60
0721        GO TO 12
0722   40   ID=ID+1
0723  C -----* Check is made in next five statements  *
0724  C -----* for proper place of IDENTIFIER & LITERAL in query statement  **
0725        IF((ID.GT.LIT).AND.(I.EQ.L)) GO TO 48
0726        IF((ID.EQ.LIT).AND.(I.EQ.L).AND.(IF.EQ.0)) GO TO 50
0727        IF((ID.EQ.LIT).AND.(I.LT.L)) GO TO 51
0728        IF((ID.GT.LIT).AND.(I.LT.L).AND.(IF.EQ.0)) GO TO 15
0729        IF((ID.LT.LIT).AND.(I.EQ.L)) GO TO 48
0730        GO TO 12
0731   48   IB(I,3)=1
0732        WRITE(1,151)
0733        GO TO 170
0734   51   IF=IF+1
0735        GO TO 12
0736   50   IB(I,3)=0
0737        GO TO 169
0738   60   LIT=LIT+1
0739         IF((LIT.EQ.ID).AND.(I.EQ.L).AND.(IF.EQ.0)) GO TO 50
0740        IF((LIT.LT.ID).AND.(I.LT.L)) GO TO 15
0741        IF((LIT.LT.ID).AND.(I.EQ.L)) GO TO 48
0742        IF((LIT.GT.ID).AND.(I.LE.L)) GO TO 48
0743        IF((LIT.EQ.ID).AND.(I.LT.L)) GO TO 51
0744        IB(I,3)=1
0745   140  CONTINUE
0746  C -----* Flags affected during syntax checking are checked  *
0747  C      * for error possibility ********************************
0748   170  DO 149 I=1,L
0749        IF(IB(I,3).NE.1) GO TO 149
0750        GO TO 148
0751   149  CONTINUE
0752   169  WRITE(1,157)
0753        GO TO 162
0754   148  IF(IB(1,3).EQ.1) WRITE(1,153)
0755  C -----* Error is checked & its diagnosis is given *
0756        IF(IB(2,3).EQ.1) WRITE(1,154)
0757        IF(IB(3,3).EQ.1) WRITE(1,155)
0758        IF(IF.GT.0) WRITE(1,152)
0759        CALL EXEC( ICODF,LRNAM)
0760   151  FORMAT(1H ,'No correspondence between Literal & Idetifier')
0761   152  FORMAT(1H ,'Literal has come before end of Idetifier in Query')
0762   153  FORMAT(1H ,'At your 1st Query Word no keyWord. INSRTN ')
0763   154  FORMAT(1H ,'At your 2nd Query Word no Idetifier')
0764   155  FORMAT(1H ,'At your 3rd Query Word no Idetifier')
0765   157  FORMAT(1H ,'Your query is syntactically correct')
0766  C -----*  Corresponding semantics routine is called provided query is co
```

```
0767    162   CALL EXEC(ICODF,INAM)
0768          END
0769          PROGRAM REVOK,5
0770    C -----**********************************************************************
0771    C -----*** this segmrent checks the query statement to REVOKE the ***-----
0772    C       *** granted OPTIONS viz. TO MODIFY , TO DELETE , TO INSRTN ***
0773    C       *** from authorised users                                  ***
0774    C       **********************************************************************
0775          DIMENSION IA(50,2),IB(50,3),ITAB1(30,22),ITAB(30,42),INAM(3)
0776         *,LRNAM(3)
0777          COMMON ITAB,ITAB1,IA,M,M1,JM,IB
0778          DATA ICODF,INAM,LRNAM/8,2HGR,2HNT,2HT  ,2HCL,2HEA,2HR /
0779    C     **********************************************************************
0780    C -----* INS keeps acount of key word INSRTN                              *
0781    C       * IDEL keeps acount of key word DELETE                            *
0782    C       * MODFY keeps acount of key word MODIFY                           *
0783    C       **********************************************************************
0784          INS=0
0785          IDEL=0
0786          MODFY=0
0787    C     **********************************************************************
0788    C -----* Uniform symbol table from two dimensional array  IA is copied
0789    C       * to array IB in next three statements
0790    C       **********************************************************************
0791          DO 8 I=1,M
0792          DO 8 J=1,2
0793    8     IB(I,J)=IA(I,J)
0794          DO 140 I=1,M
0795          IF(I.GE.3) GO TO 30
0796          GO TO (10,20),I
0797    10    IF((IB(I,1).EQ.1).AND.(IB(I,2).EQ.15)) GO TO 15
0798    12    IB(I,3)=1
0799          GO TO 140
0800    15    IB(I,3)=0
0801          GO TO 140
0802    C -----*** IN next two statements option rights which will be    ***
0803    C       *** revoked from users are checked                       ***
0804    20    IF(((IB(I,1).EQ.1).AND.(IB(I,2).EQ.7))
0805         *.OR.((IB(I,1).EQ.1).AND.(IB(I,2).EQ.6)).OR.((IB(
0806         *I,1).EQ.1).AND.(IB(I,2).EQ.16))) GO TO 25
0807          GO TO 12
0808    30    IF(((IB(I,1).EQ.1).AND.(IB(I,2).EQ.7)).OR.
0809         *((IB(I,1).EQ.1).AND.(IB(I,2).EQ.6)).OR.((IB(I,1).
0810         *EQ.1).AND.(IB(I,2).EQ.16))) GO TO 31
0811    C ---*** Key word ON is checked          ****
0812          IF((IB(I,1).EQ.1).AND.(IB(I,2).EQ.4)) GO TO 33
0813          IF(I.GT.3) GO TO 35
0814          GO TO 12
```

```
0815  C ----*** In next statements repeatition of options are checked ***
0816  25     IF(IB(I,2).EQ.6) GO TO 40
0817         IF(IB(I,2).EQ.7) GO TO 42
0818         IF(IB(I,2).EQ.16) GO TO 50
0819         GO TO 12
0820  31     IF(IB(I-1,1).EQ.1) GO TO 25
0821         GO TO 12
0822  33     IF((IB(I-1,1).EQ.1).AND.(IB(I+1,1).EQ.4)) GO TO 15
0823         GO TO 12
0824  35     IF((IB(I,1).EQ.4).AND.((IB(I-1,1).EQ.1).AND.(IB(I-1,2).EQ.4)).AND
0825        *((IB(I+1,1).EQ.1).AND.(IB(I+1,2).EQ.2))) GO TO 15
0826         IF((IB(I,1).EQ.1).AND.(IB(I,2).EQ.2)) GO TO 36
0827         IF((IB(I,1).EQ.3).AND.((IB(I-1,1).EQ.1).AND.(IB(I-1,2).EQ.2)).O
0828        *R.(IB(I,1).EQ.3)) GO TO 15
0829         GO TO 12
0830  36     IF((IB(I-1,1).EQ.4).AND.(IB(I+1,1).EQ.3)) GO TO 15
0831         GO TO 12
0832  40     IDEL=IDEL+1
0833         IF(IDEL.GT.1) GO TO 12
0834         GO TO 15
0835  42     INS=INS+1
0836         IF(INS.GT.1) GO TO 12
0837         GO TO 15
0838  50     MOD=MOD+1
0839         IF(MOD.GT.1) GO TO 12
0840         GO TO 15
0841  140    CONTINUE
0842         DO 141 I=1,M
0843         IF(IB(I,3).NE.1) GO TO 141
0844         GO TO '42
0845  141    CONTINUE
0846         IF((INS.OR.IDEL.OR.MODFY).GT.1) GO TO 144
0847         WRITE(1,160)
0848         GO TO 143
0849  C ----*** In next five statements error diagnosis is given ***
0850  144    WRITE(1,166)
0851  142    IF(IB(1,3).EQ.1) WRITE(1,162)
0852         IF(IB(2,3).EQ.1) WRITE(1,163)
0853         IF(IB(3,3).EQ.1) WRITE(1,164)
0854         IF(IB(4,3).EQ.1) WRITE(1,165)
0855         CALL EXEC( ICODF,LRNAM)
0856  160    FORMAT(1H ,'Your Query is Syntactically correct')
0857  162    FORMAT(1H ,'At your 1st Query Word no proper keyWord REVOKE ')
0858  163    FORMAT(1H ,'At your 2nd Query Word no proper keyWord  DELETE
0859        * INSRTN MODIFY ')
0860  164    FORMAT(1H ,'Either  MODIFY INSRTN DELETE or ON MISSINGAT 3RD')
0861  165    FORMAT(1H ,'Either kyword MODIFY INSRTN DELETE or ON or
0862       *Idetier is missing at your 4th Query word ')
```

```
0863  166    FORMAT(1H ,'Repeated keyword INSRTN MODFY DELETE  in Query
0864        *  So your query is wrong')
0865  C -----*** In case correct query corresponding semantic routine is called
0866   143    CALL EXEC(ICODF,INAM)
0867          END
0868          PROGRAM SSPND ,5
0869  C       ************************************************************
0870  C -----*    This segment checks syntax of query statement *
0871  C       *    SUSPND . This query is to suspend the relation*
0872  C       *  name ,so that any user can't operate on it.     *
0873  C       ************************************************************
0874          DIMENSION IA(50,2),IB(50,3),ITAB1(30,22),ITAB(30,42),INAM(3)
0875         *,LRNAM(3)
0876          COMMON  ITAB,ITAB1,IA,M,M1,JM,IB
0877          DATA ICODF,INAM,LRNAM/8,2HRE,2HST,2HT ,2HCL,2HEA,2HR /
0878          DO 10 I=1,M
0879          DO 10 J=1,2
0880   10     IB(I,J)=IA(I,J)
0881          DO 60 I=1,M
0882          IF(I.GT.2) GO TO 45
0883          GO TO (30,40),I
0884  C -----* FIRST WORD OF QUERY STATEMENT IS CHECKED  *
0885   30     IF((IB(I,1).EQ.1).AND.(IB(I,2).EQ.14)) GO TO 35
0886   32     IB(I,3)=1
0887          GO TO 60
0888   35     IB(I,3)=0
0889          GO TO 60
0890  C -----* IDENTIFIER IS CHECKED IN NEXT STATEMENT *
0891   40     IF(IB(I,1).EQ.4) GO TO 35
0892          GO TO 32
0893    45    WRITE(1,85)
0894          GO TO 68
0895   60     CONTINUE
0896          DO 65 I=1,M
0897          IF(IB(I,3).NE.1) GO TO 65
0898          GO TO 66
0899   65     CONTINUE
0900          WRITE(1,70)
0901          GO TO 90
0902  C -----*    IN NEXT TWO STATEMENTS ERROR DIAGNOSIS IS MENTIONED *
0903   66     IF(IB(1,3).EQ.1) WRITE(1,75)
0904          IF(IB(2,3).EQ.1) WRITE(1,80)
0905   68     CALL EXEC(ICODF,LRNAM)
0906   70     FORMAT(1H ,'Query is Syntactically Correct')
0907   75     FORMAT(1H ,'At your 1st Query Word no correct keyWord SSPEND ')
0908   80     FORMAT(1H ,'At your 2nd Query Word no Idetifier ')
0909   85     FORMAT(1H ,'You are giving more than required Query ')
0910  C       ****************************************
```

```
0911  C -----* IN CASE OF CORRECT QUERY CORRESPONDING *
0912  C      * SEMANTICS ROUTINE IS CALLED            *
0913  C      ****************************************
0914  90     CALL EXEC(ICODF,INAM)
0915         END
0916         PROGRAM GRANT ,5
0917  C -----* This segment checks syntax of query statement whose first query
0918  C       ***********    is GRANT     ***********************************
0919         DIMENSION IA(50,2),IB(50,3),ITAB1(30,22),ITAB(30,42),INAM(3)
0920        *,LRNAM(3)
0921         COMMON ITAB,ITAB1,IA,L,M1,JM,IB
0922         DATA ICODF,INAM,LRNAM/8,2HGR,2HNT,2HT ,2HCL,2HEA,2HR /
0923  C      ***************************************************************
0924  C -----*  INS KEEPS ACOUNT OF GRANT OPTION INSRTN                      *
0925  C      *  MOD keeps acount of grant option MODIFY                      *
0926  C      *  IDEL keeps acount of grant option DELETE                     *
0927  C      *  ITO  keeps acount of keyword TO                              *
0928  C      *  ION  keeps acount of keyword ON                              *
0929  C      ***************************************************************
0930         INS=0
0931         IDEL=0
0932         MOD=0
0933         ITO=0
0934         ION=0
0935         DO 8 I=1,L
0936         DO 8 J=1,2
0937  8      IB(I,J)=IA(I,J)
0938         DO 140 I=1,L
0939  C    *****      first keyword GRANT in query statement is checked    *****
0940         IF((IB(I,1).EQ.1).AND.(IB(I,2).EQ.18)) GO TO 90
0941  C   ***grant OPTIONS viz. MODIFY DELETE INSRTN are checked in next statem
0942         IF((IB(I,1).EQ.1).AND.((IB(I,2).EQ.7).OR.(IB(I,2).EQ.6)
0943        *.OR.(IB(I,2).EQ.16))) GO TO 30
0944  C ----*****   keyword " ON " is checked         *******
0945         IF((IB(I,1).EQ.1).AND.(IB(I,2).EQ.4)) GO TO 45
0946  C -----****  IDENTIFIER  check in next statement  ****
0947         IF((IB(I,1).EQ.4).AND.(I.GT.3).AND.((IB(I-1,1).EQ.1).AND.(IB(I-1,2
0948        *).EQ.4)).AND.((IB(I+1,1).EQ.1).AND.(IB(I+1,2).EQ.8))) GO TO 90
0949  C ----***** keyword " TO " check in next statement   *******
0950         IF((IB(I,1).EQ.1).AND.(IB(I,2).EQ.8)) GO TO 50
0951         IF(IB(I,1).EQ.3) GO TO 60
0952         IF(((IB(I,1).EQ.1).AND.(IB(I,2).EQ.8)).AND.(IB(I-1,1).EQ.4).AND.(
0953        *IB(I+1,1).EQ.3)) GO TO 90
0954         GO TO 85
0955  30     IF(IB(I,2).EQ.6) GO TO 35
0956         IF(IB(I,2).EQ.7) GO TO 36
0957         IF(IB(I,2).EQ.16) GO TO 40
0958         GO TO 85
```

```
0959   35     IDEL=IDEL+1
0960          IF(IDEL.GT.1) GO TO 85
0961          GO TO 90
0962   36     INS=INS+1
0963          IF(INS.GT.1) GO TO 85
0964          GO TO 90
0965   40     MOD=MOD+1
0966          IF(MOD.GT.1) GO TO 85
0967          GO TO 90
0968   45     ION=ION+1
0969          IF((IB(I+1,1).EQ.4).AND.(ION.EQ.1)) GO TO 90
0970          IC=1
0971          GO TO 85
0972   50     ITO=ITO+1
0973          IF((IB(I-1,1).EQ.4).AND.((IB(I-2,1).EQ.1).AND.(IB(I-2,2).EQ.4))
0974         *.AND.(IB(I+1,1).EQ.3).AND.(ITO.EQ.1)) GO TO 90
0975          ID=1
0976          GO TO 85
0977   60     IF((IB(I-1,1).EQ.3).OR.((IB(I-1,1).EQ.1).AND.(IB(I-1,2).EQ.8))
0978         *.AND.(IB(I-2,1).EQ.4)) GO TO 90
0979   85     IB(I,3)=1
0980          GO TO 140
0981   90     IB(I,3)=0
0982   140    CONTINUE
0983          DO 145 I=1,L
0984   C---* Flags affected in syntax checking are checked for presence of erro
0985          IF(IB(I,3).EQ.0) GO TO 145
0986          GO TO 148
0987   145    CONTINUE
0988          WRITE(1,161)
0989          GO TO 150
0990   C -----*** error diagnosis is given below **********
0991   148    IF((IDEL.GT.1).OR.(INS.GT.1).OR.(MOD.GT.1)) WRITE(1,162)
0992          IF((ION.GT.1).OR.(ITO.GT.1).OR.(IC.EQ.1).OR.(ID.EQ.1))WRITE(1,163)
0993          IF(IB(1,3).EQ.1) WRITE(1,165)
0994          IF(IB(2,3).EQ.1) WRITE(1,166)
0995          CALL EXEC(ICODF,LRNAM)
0996   161    FORMAT('Query is correct')
0997   162    FORMAT('Either of INSRTN  MODIFY DELETE is Repeated')
0998   163    FORMAT('Either TO or ON is repeated which is wrong')
0999   165    FORMAT('At your 1st Query Word GRANT is missing')
1000   166    FORMAT('either INSRTN or DELETE or MODIFY  missing at 2nd')
1001   C -----* Corresponding semantic routine is called provided query is corre
1002    150   CALL EXEC(ICODF,INAM)
1003          END
1004          PROGRAM DELT,5
1005   C      ***************************************************************
1006   C -----*     This segment checks syntaxof query statement whose first
```

```
1007  C      *    query word is ' DELETE '
1008  C      ************************************************************
1009         DIMENSION IA(50,2),IB(50,3),ITAB1(30,22),ITAB(30,42),JNAM2(3)
1010        *,LRNAM(3)
1011         COMMON ITAB,ITAB1,IA,M,L,JM,IB
1012         DATA ICODF,JNAM2,LRNAM /8,2HUP,2HDA,2HT ,2HCL,2HEA,2HR /
1013         DO 5 I=1,M
1014         DO 5 J=1,2
1015  5      IB(I,J)=IA(I,J)
1016         DO 100 I=1,M
1017         IF(I.GE.4) GO TO 40
1018         GO TO (10,20,30),I
1019  C -----**** first query word   DELETE is checked          ******
1020  10     IF((IB(I,1).EQ.1).AND.(IB(I,2).EQ.6)) GO TO 15
1021  12     IB(I,3)=1
1022         GO TO 100
1023  15     IB(I,3)=0
1024         GO TO 100
1025  C -----*  2nd query is checked for IDENTIFIER   *****
1026  20     IF(IB(I,1).EQ.4) GO TO 15
1027         GO TO 12
1028  C -----*  3rd Query word is checked for Keyword WHERE   ****
1029  30     IF((IB(I,1).EQ.1).AND.(IB(I,2).EQ.3)) GO TO 15
1030         GO TO 12
1031  C -----* WHERE clause is checked in next four statement ****
1032  40     IF((IB(I,1).EQ.4).AND.(IB(I-1,1).EQ.1).AND.((IB(I+1,1).EQ.2).AND.
1033        *(IB(I+1,2).EQ.26))) GO TO 15
1034         IF(((IB(I,1).EQ.2).AND.(IB(I,2).EQ.26)).AND.(IB(I-1,1).EQ.4).AND.
1035        *(IB(I+1,1).EQ.3)) GO TO 15
1036         IF((IB(I,1).EQ.3).AND.((IB(I-1,1).EQ.2).AND.(IB(I-1,2).EQ.26)))
1037        *GO TO 15
1038         IF(((IB(I,1).EQ.1).AND.(IB(I,2).EQ.24)).AND.(IB(I-1,1).EQ.3).AND.
1039        *(IB(I+1,1).EQ.4)) GO TO 15
1040         GO TO 12
1041  100    CONTINUE
1042  C -----***  flag is checked for error      ****
1043         DO 99 J=1,M
1044         IF(IB(J,3).NE.1) GO TO 99
1045         GO TO 110
1046  99     CONTINUE
1047         WRITE(1,109)
1048         GO TO 112
1049  C -----** error diagnosis is given below   *****
1050  110    IF(IB(1,3).EQ.1)WRITE(1,102)
1051         IF(IB(2,3).EQ.1)WRITE(1,103)
1052         IF(IB(3,3).EQ.1)WRITE(1,104)
1053         IF(IB(4,3).EQ.1)WRITE(1,105)
1054         IF(IB(5,3).EQ.1)WRITE(1,106)
```

```
1055          IF(IB(6,3).EQ.1)WRITE(1,107)
1056          CALL EXEC(ICODF,LRNAM)
1057   102    FORMAT(1H ,'At Your 1st Query Word no proper Kyword DELETE')
1058   103    FORMAT(1H ,'At Your 2nd Query Word no Identifier')
1059   104    FORMAT(1H ,'At Your 3rd Query Word no proper Kyword WHERE')
1060   105    FORMAT(1H ,'At Your 4th Query Word no Identifier')
1061   106    FORMAT(1H ,'At Your 5th Query Word no Proper Delimiter = ')
1062   107    FORMAT(1H ,'At Your 6th Query Word no Proper Literal ')
1063   109    FORMAT(1H ,'Your Query is Syntactically correct')
1064   C -----*** corresponding semantics routine is called in next statement **
1065   112    CALL EXEC(ICODF,JNAM2)
1066          END
1067          PROGRAM MODFY,5
1068   C      ************************************************************************
1069   C -----*    This segment checks syntax of query statement whose first quer
1070   C      ********************  is MODIFY            ***************************
1071          DIMENSION IA(50,2),IB(50,3),ITAB1(30,22),ITAB(30,42),JNAM3(3)
1072         *,LRNAM(3)
1073          COMMON ITAB,ITAB1,IA,M,L,JM,IB
1074          DATA ICODF,JNAM3,LRNAM/8,2HUP,2HDA,2HT ,2HCL,2HEA,2HR /
1075          DO 5 I=1,M
1076          DO 5 J=1,2
1077   5      IB(I,J)=IA(I,J)
1078          DO 130 I=1,M
1079          IF(I.GE.8) GO TO 80
1080          GO TO (10,20,30,40,50,60,70),I
1081   C -----******** First query word MODIFY is checked  in Query statement
1082   10     IF((IB(I,1).EQ.1).AND.(IB(I,2).EQ.16)) GO TO 15
1083   12     IB(I,3)=1
1084          GO TO 130
1085   15     IB(I,3)=0
1086          GO TO 130
1087   C -----*****  2nd query word is checked for identifier          **********
1088   20     IF(IB(I,1).EQ.4) GO TO 15
1089          GO TO 12
1090   C -----*** 3rd query word is checked for keyword SET    ************
1091   30     IF ((IB(I,1).EQ.1).AND.(IB(I,2).EQ.17)) GO TO 15
1092          GO TO 12
1093   C -----*** 4th query word is checked for IDENTIFIER  again  ********
1094   40     IF(IB(I,1).EQ.4) GO TO 15
1095          GO TO 12
1096   50     IF((IB(I,1).EQ.2).AND.(IB(I,2).EQ.26)) GO TO 15
1097          GO TO 12
1098   C -----*** 6th query word is checked for LITERAL      *******
1099   60     IF(IB(I,1).EQ.3) GO TO 15
1100          GO TO 12
1101   70     IF(((IB(I,1).EQ.1).AND.(IB(I,2).EQ.3)).AND.(IB(I+1,1).EQ.4))
1102         *GO TO 15
```

```
1103          GO TO 12
1104    80    IF((IB(I,1).EQ.4).AND.((IB(I+1,1).EQ.2).AND.(IB(I+1,2).EQ.26)).AND
1105         *.(IB(I-1,1).EQ.1)) GO TO 15
1106          IF(((IB(I,1).EQ.2).AND.(IB(I,2).EQ.26)).AND.(IB(I-1,1).EQ.4).AN
1107         *D.(IB(I+1,1).EQ.3)) GO TO 15
1108          IF((IB(I,1).EQ.3).AND.((IB(I-1,1).EQ.2).AND.(IB(I-1,2).EQ.26)))
1109         *GO TO 15
1110          IF(((IB(I,1).EQ.1).AND.(IB(I,2).EQ.24)).AND.(IB(I+1,1).EQ.4).AN
1111         *D.(IB(I-1,1).EQ.3)) GO TO 15
1112          GO TO 12
1113    130   CONTINUE
1114    C -----****      Flags affected during syntax checking are processed     **
1115          DO 135 I=1,M
1116          IF(IB(I,3).NE.1) GO TO 135
1117          GO TO 138
1118    135   CONTINUE
1119          WRITE(1,147)
1120          GO TO 139
1121    C -----***  Error diagnosis is given in next few statements     ******
1122    138   IF(IB(1,3).EQ.1) WRITE(1,141)
1123          IF(IB(2,3).EQ.1) WRITE(1,142)
1124          IF(IB(3,3).EQ.1) WRITE(1,143)
1125          IF(IB(4,3).EQ.1) WRITE(1,144)
1126          IF(IB(5,3).EQ.1) WRITE(1,145)
1127          IF(IB(6,3).EQ.1) WRITE(1,146)
1128          IF(IB(7,3).EQ.1) WRITE(1,150)
1129          IF(IB(8,3).EQ.1) WRITE(1,151)
1130          IF(IB(9,3).EQ.1) WRITE(1,152)
1131          IF(IB(10,3).EQ.1) WRITE(1,153)
1132          CALL EXEC(ICODF,LRNAM)
1133    141   FORMAT(1H ,'At your 1st Query Word no proper kyword MODIFY ')
1134    142   FORMAT(1H ,'At your 2nd Query Word no idetifier')
1135    143   FORMAT(1H ,'At Your 3rd Query word no proper Kyword SET ')
1136    144   FORMAT(1H ,'At your 4th Query Word no Identifier')
1137    145   FORMAT(1H ,'At Your 5th Query Word no proper Delimeter = ')
1138    146   FORMAT(1H ,'At Your 6th Query Word no Literal ')
1139    147   FORMAT(1H ,'Your Query is Syntactically Correct ')
1140    148   FORMAT(1H ,'Your Query is Wrong Please Give Again ')
1141    150   FORMAT(1H ,'At your 7th Query Word no proper KyWord WHERE ')
1142    151   FORMAT(1H ,'At your 8th Query Word no Idetifier ')
1143    152   FORMAT(1H ,'At your 9th Query Word no proper Delimeter = ')
1144    153   FORMAT(1H ,'At your 10th Query Word no proper Literal ')
1145    C -----*** Corresponding semantic routine is called in case      ******
1146    C     *** of correct query statement by system utility routine ******
1147    C     ********************  CALL EXEC (------,------)  ********************
1148    139   CALL EXEC(ICODF,JNAM3)
1149          END
1150          PROGRAM RESTR,5
```

```
1151  C      ************************************************************
1152  C -----*    This segment checks syntax of query statement whose first
1153  C      *    query word is RESSTOR . This will make the relation in state
1154  C      *    of use which was previously suspended by some user
1155  C      ************************************************************
1156         DIMENSION IA(50,2),IB(50,3),ITAB1(30,22),ITAB(30,42),INAM(3)
1157        *,LRNAM(3)
1158         COMMON ITAB,ITAB1,IA,M,L,JM,IB
1159         DATA ICODF,INAM,LRNAM/8,2HRE,2HST,2HT  ,2HCL,2HEA,2HR  /
1160         DO 10 I=1,M
1161         DO 10 J=1,2
1162  10     IB(I,J)=IA(I,J)
1163         DO 60 I=1,M
1164         IF(I.GT.2) GO TO 45
1165         GO TO (30,40),I
1166  C -----**** First query word RESTOR is checked in query statement     ****
1167  30     IF((IB(I,1).EQ.1).AND.(IB(I,2).EQ.25)) GO TO 35
1168  32     IB(I,3)=1
1169         GO TO 60
1170  35     IB(I,3)=0
1171         GO TO 60
1172  C -----****  2nd query word is checked for IDENTIFIER        *********
1173  40     IF(IB(I,1).EQ.4) GO TO 35
1174         GO TO 32
1175  45     WRITE(1,85)
1176         GO TO 68
1177  60     CONTINUE
1178  C -----*** Flags are checked for presence of error     ********
1179         DO 65 I=1,M
1180         IF(IB(I,3).NE.1) GO TO 65
1181         GO TO 66
1182  65     CONTINUE
1183         WRITE(6,70)
1184         GO TO 90
1185  C -----***  Error diagnosis is given here       *****
1186  66     IF(IB(1,3).EQ.1) WRITE(1,75)
1187         IF(IB(2,3).EQ.1) WRITE(1,80)
1188  68     CALL EXEC(ICODF,LRNAM)
1189  70     FORMAT(1H ,'Query is syntactically correct ')
1190  75     FORMAT(1H ,'At your 1st query word keyword RESTOR is missing')
1191  80     FORMAT(1H ,'No relation name after keyword')
1192  85     FORMAT(1H ,'You are giving more infomation to restor ')
1193  C -----***  Corresponding semantic routine is called in case  ****
1194  C      ***  query statement is correct                        ****
1195  90     CALL EXEC(ICODF,INAM)
1196         END
1197         PROGRAM DEFVW,5
1198  C      ************************************************************
```

```
1199  C ----*  This segment checks syntax of query statement whose first      *
1200  C      *  query word is DEFVEW                                           *
1201  C      ***************************************************************
1202         DIMENSION IA(50,2),IB(50,3),ITAB1(30,22),ITAB(30,42),JNAM4(3)
1203        *,LRNAM(3)
1204         COMMON ITAB,ITAB1,IA,M,L,JM,IB
1205         DATA ICODF,JNAM4,LRNAM/8,2HDE,2HFV,2HE ,2HCL,2HEA,2HR /
1206         DO 5 I=1,M
1207         DO 5 J=1,2
1208  5      IB(I,J)=IA(I,J)
1209         DO 100 I=1,M
1210         IF(I.EQ.M) GO TO 50
1211         IF(I.GT.6) GO TO 40
1212         GO TO (10,20,30,20,30,20),I
1213  C ----***  First query word  DEFVEW is checked in query statement     ***
1214  10     IF((IB(I,1).EQ.1).AND.(IB(I,2).EQ.13)) GO TO 25
1215  15     IB(I,3)=1
1216         GO TO 100
1217  20     IF(IB(I,1).EQ.4) GO TO 25
1218         GO TO 15
1219  25     IB(I,3)=0
1220         GO TO 100
1221  30     IF((IB(I,1).EQ.2).AND.(IB(I,2).EQ.32)) GO TO 25
1222         GO TO 15
1223  40     IF(((IB(I,1).EQ.2).AND.(IB(I,2).EQ.33)).AND.(IB(I-1,1).EQ.4).AND.
1224        *((IB(I+1,1).EQ.4).OR.((IB(I+1,1).EQ.2).AND.(IB(I+1,2).EQ.33))))
1225        * GO TO 25
1226         IF((IB(I,1).EQ.4).AND.((IB(I-1,1).EQ.2).AND.((IB(I-1,2).EQ.33).OR
1227        *.(IB(I-1,2).EQ.32))) .AND. ((IB(I+1,1).EQ.2).AND.((IB(I+1,2).EQ.
1228        *33).OR.(IB(I+1,2).EQ.32)))) GO TO 25
1229          IF(((IB(I,1).EQ.2).AND.(IB(I,2).EQ.32)).AND.(IB(I+1,1).EQ.4).AND
1230        *.(IB(I-1,1).EQ.4).AND.((IB(I+2,1).EQ.2).AND.(IB(I+2,2).EQ.33)))
1231        * GO TO 25
1232         GO TO 15
1233  50     IF(((IB(I,1).EQ.2).AND.(IB(I,2).EQ.33)).AND.((IB(I-1,1).EQ.2).AND.
1234        *(IB(I-1,2).EQ.33))) GO TO 25
1235          GO TO 15
1236  100    CONTINUE
1237  C -----***  Flags are checked for presence of error  ***-----
1238         DO 110 I=1,M
1239         IF(IB(I,3).NE.1) GO TO 110
1240         GO TO 120
1241  110    CONTINUE
1242         WRITE(6,150)
1243         GO TO 130
1244  C ----***  Error diagnosis is given in next few statements   ***------
1245  120    IF((IB(2,3).EQ.1).OR.(IB(4,3).EQ.1).OR.(IB(6,3).EQ.1))
1246        * WRITE(1,140)
```

```
1247          IF((IB(3,3).EQ.1).OR.(IB(7,3).EQ.1)) WRITE(1,145)
1248          CALL EXEC(ICODF,LRNAM)
1249   140    FORMAT(1H ," Check ID for proper place")
1250   145    FORMAT(1H ," Check for left & right parenthesis")
1251   150    FORMAT(1H ," Query is syntactically correct")
1252   C ------***  Corresponding semantic routine is called     ***------
1253   C ------***  provided query statement is correct          ***------
1254   130    CALL EXEC(ICODF,JNAM4)
1255          END
```

```
&DBMS   T=00004 IS ON CR00015 USING 00481 BLKS R=3334

0001          PROGRAM QRYA,5
0002          COMMON ITAB,ITAB1,ISTB,MM,MT,ML,IPRS,IQRY,IF
0003          INTEGER IPRS(50,3),IDCB(144),NAME(3),NAM1(3),NAM2(3),NAM3(3),
0004         *NAM4(3),NAM5(3),IBUF(160),IBUF1(10),IBUF2(256),ITAB(30,42),
0005         *ITAB1(30,22),IB(128),IC(80),ID(10),BLANK,IBUF3(20),IE(128),
0006         *ISTRT(10),ILEN(10),IRHS(10),ISECU(3),KF(2),IDCB1(144),NF(2),
0007         *IBUFL(41),IBUFF(10),NAM6(3),IBUFC(40),IBUFP(40),IBUFX(40),
0008         *ISTB(50,2),NAM7(3),NAM8(3),NAM9(3),NAM10(3),NAM11(3),
0009         *NAM12(3),IDCB2(272),NE(256),ICNAM(3)
0010          DATA NAME,NAM1,NAM2,NAM3,NAM5,NAM4,NAM6/2HDB,2HMS,2HRD,
0011         *2HDB,2HMS,2H8 ,2HDB,2HMS,2H00,2HDB,2HMS,2H11,2HIS,2HUF,2HYL,
0012         *2HCA,2HRD,2HAT,2HDB,2HMS,2H01/
0013          DATA NAM7,NAM8,NAM9,NAM10,NAM11,NAM12/2HJU,2HRN,2HAL,2HJU,2HNE,
0014         *2HLD,2HCN,2HFR,2HNC,2HCN,2HFF,2HLD,2HTH,2HSI,2HS ,2HAB,2HST,
0015         *2HRC/
0016          DATA BLANK/000040B/
0017          DATA ISECU/2H-7,2H56,2H19/
0018          DATA ICOE,ICNAM /0,2HCL,2HEA,2HR /
0019          M=0
0020          N=0
0021          KOUNT=0
0022          KONT=0
0023          IFL=0
0024          KA=0
0025          KB=0
0026          KC=0
0027          IVEW=0
0028          ICONT=0
0029  C      ICONT IS THE COUNT FOR SECOUN
0030          IN=0
0031          IFLGT=0
0032          DO 5 I=1,MM
0033          DO 5 J=1,2
0034          IF(IPRS(I,1).NE.1) GO TO 5
0035          IF(IPRS(I,2).NE.2) GO TO 5
0036          GO TO 10
0037  5       CONTINUE
0038          WRITE(6,1)
0039  1       FORMAT(1H ,'FROM MISSING')
0040          STOP
0041  C-----------     IDENTIFIERS IN 'FROM CLAUSE' ARE RECOVERED FROM
0042  C                ID TABLE ITAB1
0043  10      DO 15 II=1,MT
0044          IF(IPRS(I+1,1).NE.ITAB1(II,21)) GO TO 15
0045          IF(IPRS(I+1,2).NE.ITAB1(II,22)) GO TO 15
0046          GO TO 20
```

```
0047   15     CONTINUE
0048   20     CALL SFILL(IBUF1,1,20,BLANK)
0049          DO 25 IJ=1,10
0050          IF(ITAB1(II,IJ).EQ.1H )GO TO 52
0051          IBUF1(IJ)=ITAB1(II,IJ)
0052   25     KOUNT=KOUNT+1
0053   C------------   CHECK WHETHER THE RELATION GIVEN IN QUERY EXISTS
0054   C              IN THE RELATION DIRECTORY
0055   52     CALL OPEN(IDCB,IERR,NAME,1,ISECU)
0056          IF(IERR.NE.2) GO TO 101
0057          IF(KOUNT.LT.10)GO TO 53
0058          GO TO 54
0059   53     CALL SFILL(IBUF1,19,20,BLANK)
0060   54     CALL HASS(IBUF1,10,6,NUMBR)
0061          IF(NUMBR.EQ.0)NUMBR=6
0062          N1=NUMBR
0063          CALL CODE
0064          WRITE(IBUF,26)(IBUF1(IX),IX=1,10)
0065   26     FORMAT(10A1)
0066          CALL SFILL(IC,1,160,BLANK)
0067          CALL READF(IDCB,IERR,IC,80,LEN,NUMBR)
0068          IBUC=1
0069          DO 50 IBUC=1,4
0070          INO=1+(IBUC-1)*12
0071          IBEG1=6+(IBUC-1)*6
0072          IER=0
0073          IF(JSCOM(IBUF,1,10,IC,INO,IER))50,55,50
0074   50     CONTINUE
0075          DO 62 N2=7,10
0076          CALL SFILL(IC,1,160,BLANK)
0077          CALL READF(IDCB,IERR,IC,80,LEN,N2)
0078          IF(IERR.LT.0)GO TO 101
0079          IBUC=1
0080          DO 62 IBUC=1,4
0081          INO=1+(IBUC-1)*12
0082          IBEG1=6+(IBUC-1)*6
0083          IER=0
0084          IF(JSCOM(IBUF,1,10,IC,INO,IER))62,55,62
0085   62     CONTINUE
0086          CALL CLOSE(IDCB)
0087   C------------   IF RELATION NOT FOUND IN RELATION DIRECTORY,
0088   C              CHECK IN THE VIEW DIRECTORY
0089         CALL OPEN(IDCB,IERR,NAM2,1,ISECU)
0090         IF(IERR.NE.2)GO TO 101
0091         CALL HASS(IBUF1,KOUNT,19,NUMBR)
0092         IF(NUMBR.EQ.0)NUMBR=19
0093         CALL SFILL(IC,1,160,BLANK)
0094         CALL READF(IDCB,IERR,IC,80,LEN,NUMBR)
```

```
0095          IF(IERR.LT.0)GO TO 101
0096          DO 85 IBUC=1,5
0097          INO=1+(IBUC-1)*16
0098          IBEG=6+(IBUC-1)*8
0099          ITYP=13+(IBUC-1)*16
0100          IER=0
0101          IF(JSCOM(IBUF,1,2*KOUNT,IC,INO,IER))85,88,85
0102    85    CONTINUE
0103          DO 86 N2=20,22
0104          CALL SFILL(IC,1,160,BLANK)
0105          CALL READF(IDCB,IERR,IC,80,LEN,N2)
0106          IF(IERR.LT.0)GO TO 101
0107          IBUC=1
0108          DO 86 IBUC=1,5
0109          INO=1+(IBUC-1)*16
0110          IBEG=6+(IBUC-1)*8
0111          ITYP=13+(IBUC-1)*16
0112          IER=0
0113          IF(JSCOM(IBUF,1,2*KOUNT,IC,INO,IER))86,88,86
0114    86    CONTINUE
0115          WRITE(6,82)
0116    82    FORMAT(1H ,'RELATION NOT FOUND')
0117          CALL CLOSE(IDCB)
0118          STOP
0119    88    CALL SMOVE(IC,ITYP,ITYP+3,NF,1)
0120          CALL CODE
0121          READ(NF,90)N1,IBUC1
0122    90    FORMAT(2I2)
0123          CALL SMOVE(IC,2*IBEG-1,2*IBEG,IZ,1)
0124          CALL CODE
0125          READ(IZ,89)IU
0126    89    FORMAT(I2)
0127    C-----------    CHECK WHETHER RELATION IS SUSPENDED    ----------
0128          IF(IU.EQ.0)GO TO 58
0129          WRITE(6,83)
0130    83    FORMAT(1H ,'RELATION FOUND IN VIEW DIRECTORY')
0131          IVEW=1
0132          CALL CLOSE(IDCB)
0133          GO TO 500
0134    55    IBUC1=IBUC
0135          CALL SMOVE(IC,2*IBEG1-1,2*IBEG1,IZ,1)
0136          CALL CODE
0137          READ(IZ,63)IU
0138    63    FORMAT(I2)
0139          IF(IU.EQ.0)GO TO 58
0140          WRITE(1,56)
0141    56    FORMAT(1H ,"RELATION FOUND")
0142          GO TO 59
```

```
0143  58     WRITE(6,51)
0144  51     FORMAT(1H ,'RELATION IS SUSPENDED,QUERY CANNOT BE PROCESSED')
0145         STOP
0146  59     CALL CLOSE(IDCB)
0147         CALL SFILL(IBUF3,1,40,BLANK)
0148         CALL SMOVE(IBUF1,1,2*KOUNT,IBUF3,1)
0149         CALL OPEN(IDCB,IERR,NAM1,1,ISECU)
0150         IF(IERR.NE.2)GO TO 101
0151  C------------     ATTRIBUTES IN SELECT AND WHERE CLAUSES ARE
0152  C                 RETRIEVED FROM ITAB1 ----------------------------
0153  57     DO 60 KI=1,MT
0154         IF(IPRS(I-1,1).NE.ITAB1(KI,21))GO TO 60
0155         IF(IPRS(I-1,2).NE.ITAB1(KI,22))GO TO 60
0156         M=M+1
0157         GO TO 65
0158  60     CONTINUE
0159         WRITE(6,61)
0160  61     FORMAT(1H ,"ATTRIBUTE NOT IN ITAB1")
0161         STOP
0162  65     CALL SFILL(IBUF1,1,20,BLANK)
0163         KONT=0
0164         DO 66 IG=1,10
0165         IF(ITAB1(KI,IG).EQ.1H )GO TO 67
0166         IBUF1(IG)=ITAB1(KI,IG)
0167  66     KONT=KONT+1
0168  67     CALL SFILL(IBUF3,2*KOUNT+1,40,BLANK)
0169         CALL SMOVE(IBUF1,1,2*KONT,IBUF3,2*KOUNT+1)
0170         IF((KOUNT+KONT).LT.20)GO TO 64
0171         GO TO 68
0172  C------------     CHECK WHETHER THE ATTRIBUTES GIVEN IN QUERY
0173  C                 EXIST IN THE DIRECTORY ---------------------------
0174  64     CALL SFILL(IBUF3,2*(KOUNT+KONT)+1,40,BLANK)
0175  68     CALL HASS(IBUF3,20,31,NUMBR)
0176         IF(NUMBR.EQ.0)NUMBR=31
0177         CALL CODE
0178         WRITE(IBUFF,69)(IBUF3(IX),IX=1,20)
0179  69     FORMAT(20A1)
0180         CALL SFILL(IE,1,256,BLANK)
0181         CALL READF(IDCB,IERR,IE,128,LEN,NUMBR)
0182         IF(IERR.LT.0)GO TO 101
0183         KKO=KOUNT+KONT
0184         IER=0
0185         IBUC=1
0186         DO 72 IBUC=1,3
0187         INO=1+(IBUC-1)*32
0188         IF(JSCOM(IBUFF,1,KKO,IE,INO,IER))72,73,72
0189  72     CONTINUE
0190         DO 79 N2=32,35
```

```
0191          CALL SFILL(IE,1,256,BLANK)
0192          CALL READF(IDCB,IERR,IE,128,LEN,N2)
0193          IF(IERR.LT.0)GO TO 101
0194          IBUC=1
0195          DO 79 IBUC=1,3
0196          INO=1+(IBUC-1)*32
0197          IER=0
0198          IF(JSCOM(IBUFF,1,KKO,IE,INO,IER))79,73,79
0199    79    CONTINUE
0200          WRITE(6,77)
0201    77    FORMAT(1H ,'ATTRIBUTE NOT IN DIRECTORY')
0202          STOP
0203    73    WRITE(1,76)
0204    76    FORMAT(1H ,'ATTRIBUTE FOUND')
0205          KKB=23+(IBUC-1)*32
0206          CALL SMOVE(IE,KKB,KKB+3,KF,1)
0207          CALL CODE
0208          READ(KF,75)K2
0209    75    FORMAT(I4)
0210          KA=KA+1
0211          ISTRT(KA)=K2
0212          CALL SMOVE(IE,KKB+4,KKB+5,KG,1)
0213          CALL CODE
0214          READ(KG,78)K3
0215    78    FORMAT(I2)
0216          KB=KB+1
0217          ILEN(KB)=K3
0218          IF(N.EQ.1)GO TO 200
0219          IF(M.EQ.IF)GO TO 150
0220          I=I-1
0221          KI=1
0222          GO TO 57
0223   150    N=1
0224          I=1
0225          M=0
0226          DO 155 I=1,MM
0227          DO 155 J=1,2
0228          IF(IPRS(I,1).NE.2)GO TO 155
0229          IF(IPRS(I,2).NE.26)GO TO 155
0230          KI=1
0231          GO TO 57
0232   155    CONTINUE
0233          WRITE(6,156)
0234   156    FORMAT(1H ,"= IS MISSING")
0235          STOP
0236   C------------    THE LITERALS GIVEN IN QUERY ARE RETRIEVED
0237   C               FROM THE LITERAL TABLE ITAB ----------------------------
0238   200    DO 220 IK=1,ML
```

```
0239          IF(IPRS(I+1,1).NE.ITAB(IK,41))GO TO 220
0240          IF(IPRS(I+1,2).NE.ITAB(IK,42))GO TO 220
0241          GO TO 225
0242    220   CONTINUE
0243          WRITE(6,221)
0244    221   FORMAT(1H ,"LITERAL NOT IN ITAB")
0245          STOP
0246    225   CALL SFILL(IBUFL,1,80,BLANK)
0247          DO 230 L=1,40
0248    230   IBUFL(L)=ITAB(IK,L)
0249          CALL CLOSE(IDCB)
0250    C-----------      APPROPRIATE DATA FILRS ARE OPENED AND RETRIEVAL
0251    C                 DONE ACCORDING TO QUERY
0252          IF((N1.EQ.1).AND.(IBUC1.EQ.1))GO TO 235
0253          IF((N1.EQ.2).AND.(IBUC1.EQ.1))GO TO 236
0254          IF((N1.EQ.2).AND.(IBUC1.EQ.2))GO TO 237
0255          IF((N1.EQ.4).AND.(IBUC1.EQ.1))GO TO 238
0256          IF((N1.EQ.4).AND.(IBUC1.EQ.2))GO TO 239
0257          IF((N1.EQ.5).AND.(IBUC1.EQ.1))GO TO 240
0258          IF((N1.EQ.6).AND.(IBUC1.EQ.1))GO TO 241
0259          IF((N1.EQ.5).AND.(IBUC1.EQ.2))GO TO 242
0260          WRITE(6,243)
0261    243   FORMAT(1H ,'WRONG RELATION NAME')
0262    235   CALL OPEN(IDCB,IERR,NAM3)
0263          IF(IERR.NE.2)GO TO 101
0264          GO TO 248
0265    236   CALL OPEN(IDCB,IERR,NAM4)
0266          IF(IERR.NE.2)GO TO 101
0267          GO TO 248
0268    237   CALL OPEN(IDCB,IERR,NAM7,1,ISECU)
0269          IF(IERR.NE.2)GO TO 101
0270          GO TO 248
0271    238   CALL OPEN(IDCB,IERR,NAM5)
0272          IF(IERR.NE.2)GO TO 101
0273    239   CALL OPEN(IDCB,IERR,NAM11,1,ISECU)
0274          IF(IERR.NE.2)GO TO 101
0275          GO TO 248
0276    240   CALL OPEN(IDCB,IERR,NAM8,1,ISECU)
0277          IF(IERR.NE.2)GO TO 101
0278          GO TO 248
0279    241   CALL OPEN(IDCB,IERR,NAM9,1,ISECU)
0280          IF(IERR.NE.2)GO TO 101
0281          GO TO 248
0282    242   CALL OPEN(IDCB,IERR,NAM10,1,ISECU)
0283          IF(IERR.NE.2) GO TO 101
0284    248   IF(IQRY.NE.6)GO TO 250
0285    342   KUNT=0
0286          LO=0
```

```
0287          LP=1
0288          DO 345 LP=1,40
0289          IF(IBUFL(LP).EQ.1H )GO TO 344
0290          LO=LO+1
0291          IBUFC(LO)=IBUFL(LP)
0292          KUNT=KUNT+1
0293   345    CONTINUE
0294   344    IF(IFLGT.EQ.1)GO TO 346
0295          IF(IFLAG.EQ.1)GO TO 300
0296   250    DO 300 IN=1,10
0297          ITB=0
0298          CALL SFILL(IB,1,256,BLANK)
0299          CALL READF(IDCB,IERR,IB,128,LEN,IN)
0300          IF(IERR.LT.0) GO TO 101
0301          CALL SFILL(IBUF2,1,512,BLANK)
0302          CALL CODE
0303          READ(IB,232)(IBUF2(IX),IX=1,256)
0304   232    FORMAT(256A1)
0305          IER=0
0306          IST=ISTRT(KA)
0307          ISTT=2*IST-1
0308          ILT=ILEN(KA)
0309          LITL=2*(IST+ILT-1)
0310   346    IF(IQRY.NE.6)GO TO 292
0311          CALL SMOVE(IBUF2,ISTT,LITL,IBUFP,1)
0312          LH=0
0313          IF(ITB.EQ.1)GO TO 350
0314          LF=1
0315          DO 350 LF=1,40
0316          LH=LH+1
0317          IF(IBUFP(LF).EQ.1H )GO TO 352
0318          IBUFX(LH)=IBUFP(LF)
0319          ITB=1
0320   350    CONTINUE
0321   352    IF(JSCOM(IBUFC,1,2*KUNT,IBUFX,1,IER))353,354,353
0322   353    CALL SFILL(IBUFX,1,80,BLANK)
0323          LH=0
0324          IF(LF.GT.40)GO TO 300
0325          IF(IBUFP(LF+1).EQ.1H )GO TO 300
0326          GO TO 350
0327   354    CALL SFILL(IBUFX,1,80,BLANK)
0328          CALL SFILL (IBUFC,1,80,BLANK)
0329          IFLGT=1
0330          IF(IBUFL(LP+1).EQ.1H )GO TO 245
0331          IF(LP+1.GT.40)GO TO 245
0332          KUNT=0
0333          LO=0
0334          GO TO 345
```

```
0335  292    IF(JSCOM(IBUF2,ISTT,LITL,IBUFL,1,IER))300,245,300
0336  300    CONTINUE
0337         IF(IFLAG.EQ.1)GO TO 910
0338         GO TO 900
0339  245    IFLAG=1
0340         IFLGT=0
0341         ICONT=ICONT+1
0342  C------------------  RESULTS OF RETRIEVAL ARE PRINTED  ------------------
0343         DO 251 LS=KA-1,1,-1
0344         GO TO 301
0345  251    CONTINUE
0346         IF(IQRY.EQ.6)GO TO 342
0347         GO TO 300
0348  301    IF((ISTRT(LS).EQ.1).AND.((N1.EQ.6).OR.((N1.EQ.6).AND.(IBUC1.EQ.2))
0349        *)GO TO 382
0350         IF(ISTRT(LS).EQ.1)GO TO 371
0351         IF((ISTRT(LS).EQ.41).AND.((N1.EQ.1).OR.(N1.EQ.2).OR.(N1.EQ.4)))
0352        *GO TO 377
0353         IF((ISTRT(LS).EQ.41).AND.(((N1.EQ.5).AND.(IBUC1.EQ.1)).OR.((N1.EQ
0354        *.2).AND.(IBUC1.EQ.2))))GO TO 385
0355         IF(ISTRT(LS).EQ.61)GO TO 386
0356         IF((ISTRT(LS).EQ.67).AND.(N1.EQ.4).AND.(IBUC1.EQ.2))GO TO 387
0357         IF(ISTRT(LS).EQ.67)GO TO 373
0358         IF((ISTRT(LS).EQ.101).AND.(N1.EQ.2))GO TO 390
0359         IF(ISTRT(LS).EQ.101)GO TO 391
0360         IF((ISTRT(LS).EQ.81).AND.(N1.EQ.4))GO TO 392
0361         IF(ISTRT(LS).EQ.81)GO TO 393
0362         IF(ISTRT(LS).EQ.69)GO TO 380
0363         IF(ISTRT(LS).EQ.111)GO TO 394
0364         IF((ISTRT(LS).EQ.85).AND.((N1.EQ.5).OR.((N1.EQ.2).AND.(IBUC1.EQ.2
0365        *))))GO TO 396
0366         IF(ISTRT(LS).EQ.77)GO TO 397
0367         IF((ISTRT(LS).EQ.73).AND.(N1.EQ.1))GO TO 376
0368         IF(ISTRT(LS).EQ.73)GO TO 375
0369         IF(ISTRT(LS).EQ.113)GO TO 398
0370         IF(ISTRT(LS).EQ.95)GO TO 400
0371         IF(ISTRT(LS).EQ.83)GO TO 370
0372         IF((ISTRT(LS).EQ.85).AND.(N1.EQ.2).AND.(IBUC1.EQ.1))GO TO 381
0373         IF((ISTRT(LS).EQ.85).AND.(N1.EQ.4))GO TO 383
0374         IF((ISTRT(LS).EQ.115).AND.(N1.EQ.6))GO TO 401
0375         IF((ISTRT(LS).EQ.115).AND.(N1.EQ.5))GO TO 402
0376         IF((ISTRT(LS).EQ.97).AND.(N1.EQ.1))GO TO 384
0377         IF(ISTRT(LS).EQ.97)GO TO 403
0378         IF(ISTRT(LS).EQ.107)GO TO 404
0379         IF(ISTRT(LS).EQ.89)GO TO 378
0380         IF((ISTRT(LS).EQ.135).AND.(N1.EQ.6))GO TO 405
0381         IF((ISTRT(LS).EQ.99).AND.(N1.EQ.2))GO TO 407
0382         IF((ISTRT(LS).EQ.135).AND.(N1.EQ.5))GO TO 406
```

```
0383          IF((ISTRT(LS).EQ.99).AND.(N1.EQ.5))GO TO 408
0384          IF(ISTRT(LS).EQ.147)GO TO 409
0385          IF(ISTRT(LS).EQ.123)GO TO 410
0386          IF(ISTRT(LS).EQ.155)GO TO 411
0387          IF(ISTRT(LS).EQ.105)GO TO 412
0388          IF(ISTRT(LS).EQ.119)GO TO 413
0389          IF(ISTRT(LS).EQ.167)GO TO 414
0390          IF(ISTRT(LS).EQ.143)GO TO 415
0391          IF(ISTRT(LS).EQ.161)GO TO 416
0392          IF(ISTRT(LS).EQ.125)GO TO 417
0393          IF(ISTRT(LS).EQ.183)GO TO 418
0394          WRITE(6,265)
0395   265    FORMAT(1H ,'CORRECT ATTRIBUTE IN DIRECTORY NOT FOUND')
0396          STOP
0397   370    WRITE(6,420)(IBUF2(IX),IX=83,88)
0398   420    FORMAT(1H ,/,' STATUS:- ',5X,6A1)
0399          GO TO 395
0400   371    WRITE(6,421)(IBUF2(IX),IX=1,40)
0401   421    FORMAT(1H ,/,' TITLE:- ' ,5X,40A1)
0402          GO TO 395
0403   372    WRITE(6,422)(IBUF2(IX),IX=71,72)
0404   422    FORMAT(1H ,/,' COPYNO:- ' ,5X,2A1)
0405          GO TO 395
0406   373    WRITE(6,423)(IBUF2(IX),IX=67,68)
0407   423    FORMAT(1H ,/,' VOLUME:- ' ,5X,2A1)
0408          GO TO 395
0409   375    WRITE(6,425)(IBUF2(IX),IX=73,84)
0410   425    FORMAT(1H ,/,' CARDNO:- ' ,5X,12A1)
0411          GO TO 395
0412   376    WRITE(6,426)(IBUF2(IX),IX=73,80)
0413   426    FORMAT(1H ,/,' PRICE:- ' ,5X,8A1)
0414          GO TO 395
0415   377    WRITE(6,427)(IBUF2(IX),IX=41,66)
0416   427    FORMAT(1H ,/,' AUTHOR:- ' ,5X,26A1)
0417          GO TO 395
0418   378    WRITE(6,428)(IBUF2(IX),IX=89,96)
0419   428    FORMAT(1H ,/,' LIBDIV/- ' ,5X,8A1)
0420          GO TO 395
0421   380    WRITE(6,430)(IBUF2(IX),IX=69,70)
0422   430    FORMAT(1H ,/,' EDITION:- ' ,5X,2A1)
0423          GO TO 395
0424   381    WRITE(6,431)(IBUF2(IX),IX=85,110)
0425   431    FORMAT(1H ,/,' ISUEENAME:- ' ,5X,16A1)
0426          GO TO 395
0427   382    WRITE(6,432)(IBUF2(IX),IX=1,60)
0428   432    FORMAT(1H ,/,' NAME:- ' ,5X,60A1)
0429          GO TO 395
0430   383    WRITE(6,433)(IBUF2(IX),IX=85,94)
```

```
0431   433    FORMAT(1H ,/,' DUEDATE:- ' ,5X,10A1)
0432          GO TO 395
0433   384    WRITE(6,434)(IBUF2(IX),IX=97,116)
0434   434    FORMAT(1H ,/,' CALLNO:- ' ,5X,20A1)
0435          GO TO 395
0436   385    WRITE(6,437)(IBUF2(IX),IX=41,80)
0437   437    FORMAT(1H ,/,' PUBNAME:- ' ,5X,40A1)
0438          GO TO 395
0439   386    WRITE(6,438)(IBUF2(IX),IX=61,100)
0440   438    FORMAT(1H ,/,' PLACE:- ' ,5X,40A1)
0441          GO TO 395
0442   387    WRITE(6,439)(IBUF2(IX),IX=67,76)
0443   439    FORMAT(1H ,/,' DEGREE:- ' ,5X,10A1)
0444          GO TO 395
0445   390    WRITE(6,436)(IBUF2(IX),IX=101,130)
0446   436    FORMAT(1H ,/,' ISSUEADDR:- ' ,5X,30A1)
0447          GO TO 395
0448   391    WRITE(6,440)(IBUF2(IX),IX=101,110)
0449   440    FORMAT(1H ,/,' DATE:- ' ,5X,10A1)
0450          GO TO 395
0451   392    WRITE(6,441)(IBUF2(IX),IX=81,106)
0452   441    FORMAT(1H ,/,' SUPERVISOR:- ' ,5X,26A1)
0453          GO TO 395
0454   393    WRITE(6,442)(IBUF2(IX),IX=81,84)
0455   442    FORMAT(1H ,/,' YEAR:- ' ,5X,4A1)
0456          GO TO 395
0457   394    WRITE(6,423)(IBUF2(IX),IX=111,112)
0458          GO TO 395
0459   396    WRITE(6,444)(IBUF2(IX),IX=85,94)
0460   444    FORMAT(1H ,/,' MONTH:- ' ,5X,10A1)
0461          GO TO 395
0462   397    WRITE(6,442)(IBUF2(IX),IX=77,80)
0463          GO TO 395
0464   398    WRITE(6,446)(IBUF2(IX),IX=113,114)
0465   446    FORMAT(1H ,/,' NUMBER:- ' ,5X,2A1)
0466          GO TO 395
0467   400    WRITE(6,423)(IBUF2(IX),IX=95,96)
0468          GO TO 395
0469   401    WRITE(6,448)(IBUF2(IX),IX=115,134)
0470   448    FORMAT(1H ,/,' EDITOR:- ' ,5X,20A1)
0471          GO TO 395
0472   402    WRITE(6,449)(IBUF2(IX),IX=115,134)
0473   449    FORMAT(1H ,/,' FIELD:- ' ,5X,20A1)
0474          GO TO 395
0475   403    WRITE(6,450)(IBUF2(IX),IX=97,98)
0476   450    FORMAT(1H ,/,' COPYNO:- ' ,5X,2A1)
0477          GO TO 395
0478   404    WRITE(6,451)(IBUF2(IX),IX=107,146)
```

```
0479   451    FORMAT(1H ,/,' INSTITUTE:- ' ,40A1)
0480          GO TO 395
0481   405    WRITE(6,434)(IBUF2(IX),IX=135,154)
0482          GO TO 395
0483   406    WRITE(6,453)(IBUF2(IX),IX=135,174)
0484   453    FORMAT(1H ,/,' SUBFIELD:- ' ,5X,40A1)
0485          GO TO 395
0486   407    WRITE(6,435)(IBUF2(IX),IX=99,104)
0487   435    FORMAT(1H ,/,' ACCNO:- ' ,5X,6A1)
0488          GO TO 395
0489   408    WRITE(6,449)(IBUF2(IX),IX=99,118)
0490          GO TO 395
0491   409    WRITE(6,449)(IBUF2(IX),IX=147,166)
0492          GO TO 395
0493   410    WRITE(6,449)(IBUF2(IX),IX=123,142)
0494          GO TO 395
0495   411    WRITE(6,435)(IBUF2(IX),IX=155,160)
0496          GO TO 395
0497   412    WRITE(6,434)(IBUF2(IX),IX=105,124)
0498          GO TO 395
0499   413    WRITE(6,453)(IBUF2(IX),IX=119,158)
0500          GO TO 395
0501   414    WRITE(6,453)(IBUF2(IX),IX=167,206)
0502          GO TO 395
0503   415    WRITE(6,453)(IBUF2(IX),IX=143,182)
0504          GO TO 395
0505   416    WRITE(6,420)(IBUF2(IX),IX=161,166)
0506          GO TO 395
0507   417    WRITE(6,420)(IBUF2(IX),IX=125,130)
0508          GO TO 395
0509   C-------------     IF THE QUDERY IS ON A VIEW THE CORRESPONDING
0510   C                  CHECKS ARE MADE ON VIEW DIRECTORIES AND RETRIEVAL
0511   C                  IS DONE ACCORDINGLY
0512   418    CALL SMOVE(IB,183,186,NF,1)
0513          CALL CODE
0514          READ(NF,419)NG
0515   419    FORMAT(I4)
0516          CALL SFILL(NE,1,512,BLANK)
0517          CALL OPEN(IDCB2,IERR,NAM12,1,ISECU)
0518          IF(IERR.NE.2)GO TO 101
0519          CALL READF(IDCB2,IERR,NE,256,LEN,NG)
0520          IF(IERR.LT.0)GO TO 101
0521          WRITE(6,465)(NE(IX),IX=1,256)
0522   465    FORMAT(1H ,/,' ABSTRACT:- ' ,5X,256A2)
0523          CALL CLOSE(IDCB2)
0524   395    IF(IVEW.EQ.1)GO TO 720
0525          GO TO 251
0526   500    CALL SFILL(IBUF3,1,40,BLANK)
```

```
0527          CALL SMOVE(IBUF1,1,2*KOUNT,IBUF3,1)
0528          CALL OPEN(IDCB,IERR,NAM6,1,ISECU)
0529          IF(IERR.NE.2)GO TO 101
0530          IF(IQRY.EQ.6)GO TO 502
0531          DO 505 KI=1,MT
0532          IF(IPRS(I+3,2).NE.ITAB1(KI,22))GO TO 505
0533          GO TO 506
0534    505   CONTINUE
0535    503   WRITE(6,501)
0536    501   FORMAT(1H ,'ATTRIBUTE NOT IN ITAB1')
0537          STOP
0538    502   DO 509 KI=1,MT
0539          IF(IPRS(I+4,2).NE.ITAB1(KI,22))GO TO 509
0540          GO TO 506
0541    509   CONTINUE
0542          GO TO 503
0543    506   KONT=0
0544          CALL SFILL(IBUF1,1,20,BLANK)
0545          DO 510 IG=1,10
0546          IF(ITAB1(KI,IG).EQ.1H )GO TO 507
0547          IBUF1(IG)=ITAB1(KI,IG)
0548    510   KONT=KONT+1
0549    507   CALL SFILL(IBUF3,2*KOUNT+1,40,BLANK)
0550          CALL SMOVE(IBUF1,1,2*KONT,IBUF3,2*KOUNT+1)
0551          KKO=KOUNT+KONT
0552          CALL HASS(IBUF3,KKO,31,NUMBR)
0553          IF(NUMBR.EQ.0)NUMBR=31
0554          CALL SFILL(IE,1,256,BLANK)
0555          CALL READF(IDCB,IERR,IE,128,LEN,NUMBR)
0556          IF(IERR.LT.0)GO TO 101
0557          CALL CODE
0558          WRITE(IBUFF,508)(IBUF3(IX),IX=1,20)
0559    508   FORMAT(20A1)
0560          IBUC=1
0561          DO 515 IBUC=1,4
0562          INO=1+(IBUC-1)*36
0563          IER=0
0564          IF(JSCOM(IBUFF,1,KKO,IE,INO,IER))515,520,515
0565    515   CONTINUE
0566          DO 525 N3=32,34
0567          CALL SFILL(IBUFF,1,20,BLANK)
0568          CALL SFILL(IE,1,256,BLANK)
0569          CALL READF(IDCB,IERR,IE,128,LEN,N3)
0570          IF(IERR.LT.0)GO TO 101
0571          IBUC=1
0572          DO 525 IBUC=1,4
0573          INO=1+(IBUC-1)*36
0574          IF(JSCOM(IBUFF,1,KKO,IE,INO,IER))525,520,525
```

```
0575  525    CONTINUE
0576           WRITE(6,526)
0577  526    FORMAT(1H ,'ATTRIBUTE NOT IN VIEW DIRECTORY')
0578           STOP
0579  520    WRITE(1,527)
0580  527    FORMAT(1H ,'ATTRIBUTE FOUND IN VIEW')
0581           KKB=27+(IBUC-1)*36
0582           CALL SMOVE(IE,KKB,KKB+3,KF,1)
0583           CALL CODE
0584           READ(KF,528)K2
0585  528    FORMAT(I4)
0586           KA=KA+1
0587           ISTRT(KA)=K2
0588           CALL SMOVE(IE,KKB+4,KKB+5,KG,1)
0589           CALL CODE
0590           READ(KG,529)K3
0591  529    FORMAT(I2)
0592           KB=KB+1
0593           ILEN(KB)=K3
0594           CALL SMOVE(IE,KKB+6,KKB+7,KG,1)
0595           CALL CODE
0596           READ(KG,530)K4
0597  530    FORMAT(I2)
0598           KC=KC+1
0599           IRHS(KC)=K4
0600           IF(IFL.EQ.1)GO TO 615
0601           K5=IRHS(1)
0602           GO TO (600,605,237,606),K5
0603  600    CALL OPEN(IDCB1,IERR,NAM3)
0604           IF(IERR.NE.2)GO TO 101
0605           GO TO 610
0606  605    CALL OPEN(IDCB1,IERR,NAM4)
0607           IF(IERR.NE.2)GO TO 101
0608           GO TO 610
0609  606    CALL OPEN(IDCB1,IERR,NAM5)
0610           IF(IERR.NE.2)GO TO 101
0611  610    IFL=1
0612           IF(IQRY.EQ.6)GO TO 622
0613           IF(IPRS(I+5,2).NE.ITAB(1,42))GO TO 620
0614  611    DO 625 L=1,40
0615  625    IBUFL(L)=ITAB(1,L)
0616  615    I=I-1
0617           IF(I.EQ.1)GO TO 700
0618           KI=1
0619           DO 630 KI=1,MT
0620           IF(IPRS(I,2).NE.ITAB1(KI,22))GO TO 630
0621           IG=1
0622           GO TO 506
```

```
0623   630    CONTINUE
0624          WRITE(6,631)
0625   631    FORMAT(1H ,'VIEW ATTRIBUTE NOT IN ITAB1')
0626          STOP
0627   620    WRITE(6,621)
0628   621    FORMAT(1H ,'VIEW LITERAL NOT IN ITAB')
0629          STOP
0630   622    IF(IPRS(I+6,2).NE.ITAB(1,42))GO TO 620
0631          GO TO 611
0632   700    IST=ISTRT(1)
0633          ISTT=2*IST-1
0634          IFLGL=0
0635          ILT=ILEN(1)
0636          LITL=2*(IST+ILT-1)
0637          IF(IQRY.NE.6)GO TO 702
0638   699    KUNT=0
0639          LO=0
0640          LP=1
0641          DO 703 LP=1,40
0642          IF(IBUFL(LP).EQ.1H )GO TO 704
0643          LO=LO+1
0644          IBUFC(LO)=IBUFL(LP)
0645          KUNT=KUNT+1
0646   703    CONTINUE
0647   704    IF(IFLGT.EQ.1)GO TO 706
0648          IF(IFLGL.EQ.1)GO TO 710
0649   702    DO 710 IN=1,10
0650          ITB=0
0651          CALL SFILL(IB,1,256,BLANK)
0652          CALL SFILL(IBUF2,1,256,BLANK)
0653          CALL READF(IDCB1,IERR,IB,128,LEN,IN)
0654          IF(IERR.LT.0)GO TO 101
0655          CALL CODE
0656          READ(IB,701)(IBUF2(IX),IX=1,256)
0657   701    FORMAT(256A1)
0658          IER=0
0659   706    IF(IQRY.NE.6)GO TO 705
0660          CALL SMOVE(IBUF2,ISTT,LITL,IBUFP,1)
0661          LH=0
0662          IF(ITB.EQ.1)GO TO 707
0663          LF=1
0664          DO 707 LF=1,40
0665          LH=LH+1
0666          IF(IBUFP(LF).EQ.1H )GO TO 708
0667          IBUFX(LH)=IBUFP(LF)
0668          ITB=1
0669   707    CONTINUE
0670   708    IF(JSCOM(IBUFC,1,2*KUNT,IBUFX,1,IER))709,711,709
```

```
0671   709    CALL SFILL(IBUFX,1,80,BLANK)
0672          LH=0
0673          IF(LF.GT.40)GO TO 710
0674          IF(IBUFP(LF+1).EQ.1H )GO TO 710
0675          GO TO 707
0676   711    CALL SFILL(IBUFX,1,80,BLANK)
0677          CALL SFILL(IBUFC,1,80,BLANK)
0678          IFLGT=1
0679          IF(IBUFL(LP+1).EQ.1H )GO TO 715
0680          IF(LP+1.GT.40)GO TO 715
0681          KUNT=0
0682          LO=0
0683          GO TO 703
0684   705    IF(JSCOM(IBUF2,ISTT,LITL,IBUFL,I,IER))710,715,710
0685   710    CONTINUE
0686          IF(IFLGL.EQ.1)GO TO 910
0687          GO TO 900
0688   715    IFLGL=1
0689          IFLGT=0
0690          ICONT=ICONT+1
0691          DO 720 LS=KA,2,-1
0692          GO TO 301
0693   720    CONTINUE
0694          IF(IQRY.EQ.6)GO TO 699
0695          GO TO 710
0696   101    WRITE(6,912)IERR
0697   912    FORMAT(1H ,'IERR=',I2)
0698          STOP
0699   900    WRITE(6,911)
0700   911    FORMAT(1H ,'LITERAL DOES NOT EXIST IN DATA FILE')
0701          STOP
0702   910    CALL CLOSE(IDCB)
0703          IF(IVEW.NE.1)GO TO 950
0704          CALL CLOSE(IDCB1)
0705   950    IF(IQRY.NE.4)GO TO 1000
0706          WRITE(6,960)ICONT
0707   960    FORMAT(1H ,'COUNT=',I4)
0708   1000   WRITE(6,1001)
0709   1001   FORMAT(1H ,/,' QUERY PROCESSED')
0710          CALL EXEC(ICOE,ICNAM)
0711          END
0712          PROGRAM KRET,5
0713          INTEGER BLANK,ITAB1(30,22),ITAB(30,42),IPRS(50,3),ISTB1(20,8)
0714         *,IBUF(20),IDCB(144),NAM1(3),NAM2(3),IA(20),ID(20),IG(20)
0715         *,ISECU(3),IB(128),ISTB(50,2),KOMN(3)
0716          COMMON ITAB,ITAB1,ISTB,MM,MT,ML,IPRS,IQRY,IF,ISTB1
0717          DATA NAM1,NAM2/2HTE,2HST,2H1  ,2HTE,2HST,2H2 /
0718          DATA ISECU/2H-7,2H56,2H19/
```

```
0719          DATA BLANK/000040B/
0720          DATA ICDA,KOMN/8,2HCL,2HEA,2HR /
0721          N=1
0722          KON=0
0723          KOUNT=0
0724          KONT=0
0725          NT=0
0726          DO 5 K=1,20
0727          IF(ITAB1(1,K).EQ.1H )GO TO 6
0728          IBUF(K)=ITAB1(1,K)
0729    5     KOUNT=KOUNT+1
0730    6     CALL CODE
0731          WRITE(IA,7)(IBUF(IX),IX=1,20)
0732    7     FORMAT(20A1)
0733          CALL HASS(IBUF,10,6,NUMBR)
0734          IF(NUMBR.EQ.0)NUMBR=6
0735          N1=NUMBR
0736          CALL SFILL(IG,1,40,BLANK)
0737          CALL SMOVE(IBUF,1,2*KOUNT,IG,1)
0738          CALL OPEN(IDCB,IERR,NAM1,2,ISECU)
0739          IF(IERR.NE.2)GO TO 101
0740          CALL SFILL(IB,1,256,BLANK)
0741          CALL READF(IDCB,IERR,IB,128,LEN,N1)
0742          IF(IERR.LT.0)GO TO 101
0743          IBUC=1
0744          CALL SFILL(ID,1,40,BLANK)
0745          DO 10 IBUC=1,4
0746          INO=1+(IBUC-1)*12
0747          IEND=10+(IBUC-1)*12
0748          IER=0
0749          IF(JSCOM(IB,INO,IEND,IA,1,IER))15,100,15
0750   15     IF(JSCOM(IB,INO,IEND,ID,1,IER))10,30,10
0751   10     CONTINUE
0752          DO 20 IR=7,10
0753          CALL SFILL(IB,1,256,BLANK)
0754          CALL READF(IDCB,IERR,IB,128,LEN,IR)
0755          IF(IERR.LT.0)GO TO 101
0756          IBUC=1
0757          DO 20 IBUC=1,4
0758          INO=1+(IBUC-1)*12
0759          IEND=10+(IBUC-1)*12
0760          IER=0
0761          IF(JSCOM(IB,INO,IEND,IA,1,IER))16,100,16
0762   16     IF(JSCOM(IB,INO,IEND,ID,1,IER))20,29,20
0763   20     CONTINUE
0764   21     WRITE(6,22)
0765   22     FORMAT(1H ,'NO PLACE IN DIRECTORY')
0766          STOP
```

```
0767  29     N1=IR
0768  30     IBUCF=IBUC
0769         CALL SMOVE(IA,1,KOUNT,IB,INO)
0770         IM=1
0771         IE1=0
0772         CALL SDEA2(IM,1,2,IE1)
0773         CALL SMOVE(IM,1,2,IB,IEND+1)
0774         CALL WRITF(IDCB,IERR,IB,0,N1)
0775         IF(IERR.LT.0)GO TO 101
0776         CALL SFILL(IB,1,256,BLANK)
0777         CALL READF(IDCB,IERR,IB,128,LEN,N1)
0778         IF(IERR.LT.0)GO TO 101
0779         CALL CLOSE(IDCB)
0780         DO 35 IP=1,20
0781         IF(ISTB1(IP,7).EQ.2)GO TO 31
0782         IF(ISTB1(IP,7).NE.1)GO TO 35
0783         IF(ISTB1(IP,8).EQ.10)GO TO 37
0784         IF(ISTB1(IP,8).EQ.12)GO TO 38
0785         IF(ISTB1(IP,8).EQ.22)GO TO 39
0786  31     IF(ISTB1(IP,8).EQ.33)GO TO 500
0787  35     CONTINUE
0788         WRITE(6,36)
0789  36     FORMAT(1H ,'TYPE MISSING')
0790         STOP
0791  37     ITYP=3
0792         GO TO 40
0793  38     ITYP=2
0794         GO TO 40
0795  39     ITYP=1
0796  40     CALL SFILL(IBUF,1,40,BLANK)
0797         N=N+1
0798         KONT=0
0799         DO 45 IS=1,10
0800         IF(ITAB1(N,IS).EQ.1H )GO TO 46
0801         IBUF(IS)=ITAB1(N,IS)
0802  45     KONT=KONT+1
0803  46     CALL SMOVE(IBUF,1,2*KONT,IG,2*KOUNT+1)
0804         CALL HASS(IG,20,31,NUMBR)
0805         IF(NUMBR.EQ.0)NUMBR=31
0806         N2=NUMBR
0807         CALL OPEN(IDCB,IERR,NAM2,2,ISECU)
0808         IF(IERR.NE.2)GO TO 101
0809         CALL SFILL(IB,1,256,BLANK)
0810         CALL READF(IDCB,IERR,IB,128,LEN,N2)
0811         IF(IERR.LT.0)GO TO 101
0812         IBUC=1
0813         DO 50 IBUC=1,3
0814         INO=1+(IBUC-1)*32
```

```
0815          IEND=20+(IBUC-1)*32
0816          IER=0
0817          IF(JSCOM(IB,INO,IEND,ID,1,IER))50,55,50
0818    50    CONTINUE
0819          DO 51 IR1=32,35
0820          CALL SFILL(IB,1,256,BLANK)
0821          CALL READF(IDCB,IERR,IB,128,LEN,IR1)
0822          IF(IERR.LT.0)GO TO 101
0823          IBUC=1
0824          DO 51 IBUC=1,3
0825          INO=1+(IBUC-1)*32
0826          IEND=20+(IBUC-1)*32
0827          IER=0
0828          IF(JSCOM(IB,INO,IEND,ID,1,IER))51,54,51
0829    51    CONTINUE
0830          GO TO 21
0831    54    N2=IR1
0832    55    CALL SFILL(IA,1,40,BLANK)
0833          CALL CODE
0834          WRITE(IA,56)(IG(IX),IX=1,20)
0835    56    FORMAT(20A1)
0836          CALL SMOVE(IA,1,KOUNT+KONT,IB,INO)
0837          IE1=0
0838          CALL SDEA2(ITYP,1,2,IE1)
0839          CALL SMOVE(ITYP,1,2,IB,IEND+1)
0840          NT=NT+1
0841          IF(NT.GT.ML)GO TO 500
0842          CALL SFILL(IBUF,1,40,BLANK)
0843          DO 60 LT=1,20
0844    60    IBUF(LT)=ITAB(NT,LT)
0845          CALL SFILL(IA,1,40,BLANK)
0846          CALL CODE
0847          WRITE(IA,61)(IBUF(IX),IX=1,20)
0848    61    FORMAT(20A1)
0849          CALL SMOVE(IA,1,4,IB,IEND+3)
0850          NT=NT+1
0851          CALL SFILL(IBUF,1,40,BLANK)
0852          DO 62 LT1=1,20
0853    62    IBUF(LT1)=ITAB(NT,LT1)
0854          CALL SFILL(IA,1,40,BLANK)
0855          CALL CODE
0856          WRITE(IA,63)(IBUF(IX),IX=1,20)
0857    63    FORMAT(20A1)
0858          CALL SMOVE(IA,1,2,IB,IEND+7)
0859          IE1=0
0860          IF(KON.EQ.1)GO TO 70
0861          CALL SDEA2(IBUCF,1,2,IE1)
0862          KON=1
```

```
0863          IF(N1.LT.10)GO TO 65
0864          K1=MOD(N1,10)
0865          K2=N1/10
0866          IE1=0
0867          CALL SDEA2(K2,1,2,IE1)
0868          CALL SDEA2(K1,1,2,IE1)
0869          CALL SMOVE(K2,2,2,NA,1)
0870          CALL SMOVE(K1,2,2,NA,2)
0871          GO TO 70
0872    65    IE1=0
0873          CALL SDEA2(N1,1,2,IE1)
0874          CALL SMOVE(N1,1,2,NA,1)
0875    70    CALL SMOVE(NA,1,2,IB,IEND+9)
0876          CALL SMOVE(IBUCF,1,2,IB,IEND+11)
0877          CALL WRITF(IDCB,IERR,IB,0,N2)
0878          IF(IERR.LT.0)GO TO 101
0879          GO TO 35
0880    100   WRITE(6,103)
0881    103   FORMAT(1H ,'RELATION ALREADY CREATED BEFORE,GIVE ANOTHER NAME')
0882          GO TO 500
0883    101   WRITE(6,102)IERR
0884    102   FORMAT(1H ,'IERR=',I4)
0885          STOP
0886    500   CALL CLOSE(IDCB)
0887          CALL EXEC(ICDA,KOMN)
0888          END
0889          PROGRAM UPDAT,5
0890    C -----**********************************************************
0891    C     * This segment helps to MODIFY tuples or to DELETE tuples  *
0892    C     * from data file according to specifications .             *
0893    C     ***********************************************************
0894         . INTEGER IDCB(400),NAM7(3),NAM8(3),BLANK,ITAB1(30,22),IBUF(40),
0895        *IBUF1(144),IBUF2(85),NAM9(3),ITAB(30,42),IB(40),IDCB2
0896        *(400),NAM1(3),NAM2(3),IBUFL(20),ISECU(3),IBFR(5),NAM3(3),NAM4(3)
0897        *,ISTB(50,2),NAM5(3),NAM6(3),NAM10(3),NAM11(3),NAM12(3),NAM13(3)
0898        *,ISTB1(20,8),ICODE(20),IPARS(50,3),LLNAM(3)
0899         COMMON ITAB,ITAB1,ISTB,MM,NANA,IVAR5,IPARS,IQRY,IF,ISTB1,KMM,ICODE
0900        *,JJNUM,JFLAG
0901         DATA NAM7,NAM8,NAM9,NAM1,NAM2,BLANK/2HDB,2HMS,2HRD,2HDB,2HMS,2H8 ,
0902        *2HDB,2HMS,2H11,2HCA,2HRD,2HAT,2HIS,2HUF,2HYL,000040B/
0903         DATA ISECU,NAM3,NAM4/2H-7,2H56,2H19,2HDB,2HMS,2H00,2HDB,2HMS,2H01/
0904         DATA NAM5,NAM6,NAM10,NAM11,NAM12/2HJU,2HRN,2HAL,2HTH,2HSI,2HS ,
0905        *2HJF,2HLS,2HFD,2HCO,2HNF,2HFD,2HCO,2HNF,2HNC/
0906         DATA NAM13,LLNAM,ICODX/2HUS,2HRC,2HOD,2HCL,2HEA,2HR ,8/
0907         ICHAR=0
0908         NUM=0
0909         IFLAG=0
0910         ITEST=0
```

```
0911          LVAL=0
0912    C -----***  Contents of first row of array ITAB1 is transffered to IBUF *
0913    C -----***  & characters are also counted                                *
0914    C -----***  ITAB1 contains all IDENTIFIERS in query statement            *
0915          CALL SFILL(IBUF,1,40,BLANK)
0916          DO 2 KP=1,10
0917          IF(ITAB1(1,KP).EQ.1H ) GO TO 30
0918          ICHAR=ICHAR+1
0919    2     IBUF(KP)=ITAB1(1,KP)
0920    30    CALL MASS(IBUF,10,6,NUMB)
0921          IF(NUMB.EQ.0) NUMB=6
0922          CALL SFILL( IBUFL,1,40,BLANK)
0923    C -----*** Conversion from A1 format to A2 format  ***------
0924          CALL CODE
0925          WRITE(IBUFL,33) (IBUF(LK),LK=1,10)
0926    33    FORMAT(10A1)
0927          IF ( JFLAG.EQ.1 ) GO TO 22
0928          CALL SFILL (IB,1,40,BLANK)
0929          CALL SFILL (ICODE,1,40,BLANK)
0930          DO 43 IJH=1,6
0931    43    ICODE(IJH)=ISTB1(1,IJH)
0932    C -----*** Conversion from A1 format to A2 Format  ***------
0933          CALL CODE
0934          WRITE(IB,4) (ICODE(IG),IG=1,6)
0935    4     FORMAT(6A1)
0936          CALL SMOVE(IBUFL,1,ICHAR,IB,7)
0937    C -----*** File  USRCOD is opened to check the right of update for user *
0938          CALL OPEN(IDCB,IERR,NAM13,1,ISECU)
0939          IF(IERR.NE.2) GO TO 705
0940          CALL SFILL ( IBUF1,1,200,BLANK)
0941          CALL READF(IDCB,IERR,IBUF1,100,LEN,JJNUM)
0942          IF(IERR.LT.0) GO TO 705
0943          DO 5 JJJ= 1,10
0944          KSTRT=7+(JJJ-1)*18
0945        ! IF(JSCOM(IB,1,16,IBUF1,KSTRT,IER)) 5,6,5
0946    5     CONTINUE
0947          GO TO 710
0948    6     CALL CLOSE(IDCB)
0949    C -----*** File DBMSRD is opened to check the relation name specified  **
0950    C -----***-------------------  by user  ----------------------------------**
0951    22    CALL OPEN(IDCB,IERR,NAM7,1,ISECU)
0952          IF(IERR.NE.2) GO TO 705
0953          IRSIZ=30
0954    34    CALL SFILL(IBUF2,1,64,BLANK)
0955          CALL READF(IDCB,IERR,IBUF2,IRSIZ,LEN,NUMB)
0956          IF(IERR.LT.0) GO TO 705
0957          IF(IFLAG.EQ.1) GO TO 35
0958          IER=0
```

```
0959          DO 40 NVAR=1,4
0960          NSTRT=1+(NVAR-1)*12
0961          NFLAG=6+(NVAR-1)*6
0962          IF(JSCOM(IBUFL,1,ICHAR,IBUF2,NSTRT,IER)) 40,45,40
0963    40    CONTINUE
0964    C -----*** Flag is set to 1 to indicate that relation name is    ***------
0965    C -----***-------          not in original directory          ---------***------
0966          IFLAG=1
0967          CALL CLOSE(IDCB)
0968    C -----*** File DBMS00 is opened to check the view relation name  ***------
0969    C -----***-----------          in VIEW directory          ---------***------
0970          CALL OPEN(IDCB,IERR,NAM3,1,ISECU)
0971          IF(IERR.NE.2) GO TO 705
0972          IRSIZ=32
0973          CALL HASS(IBUF,ICHAR,19,NUMB)
0974          IF(NUMB.EQ.0) NUMB=19
0975          GO TO 34
0976    35    DO 36 I=1,5
0977          LSTRT=1+(I-1)*16
0978          NFLAG=7+(I-1)*7
0979          IF(JSCOM(IBUFL,1,ICHAR,IBUF2,LSTRT,1,IER)) 36,38,36
0980    36    CONTINUE
0981          WRITE(6,37) IBUFL
0982    37    FORMAT(1H ,"RELATION NAME ","***",1X,5A2,1X,"***","NEITHER IN
0983         *VIEW",/,"NOR IN ACTUAL DIRECTORY ")
0984          GO TO 710
0985    38    CALL SMOVE(IBUF2,(2*NFLAG)-1,2*NFLAG,JVAR,1)
0986    C -----***          Conversion from A2 format to I2 format          ***------
0987    C -----***          Status of view relation name is checked         ***------
0988    C -----***                in following few statements               ***------
0989          CALL CODE
0990          READ(JVAR,16) JVAR1
0991    16    FORMAT(I2)
0992          IF(JVAR1.EQ.1) GO TO 14
0993          WRITE(1,15) IBUFL
0994    15    FORMAT(1H ,"VIEW RELATION NAME SUSPENDED",2X,5A2)
0995          GO TO 710
0996    14    CALL CLOSE(IDCB)
0997    C -----*** File DBMS01 is opened to provided view attributes    ***------
0998    C -----***-------     have ocurred in query statement       ---------***------
0999          CALL OPEN(IDCB,IERR,NAM4,1,ISECU)
1000          IF(IERR.NE.2) GO TO 705
1001          IRSIZE=85
1002          GO TO 39
1003    29    CALL HASS(IBUF,ICONT,31,NUMBR)
1004          IF(NUMBR.EQ.0) NUMBR=31
1005          GO TO 31
1006    45    CALL SMOVE(IBUF2,(2*NFLAG)-1,2*NFLAG,JVAR,1)
```

```
1007  C '----*** Conversion from A2 format to I2 format & status of   ***------
1008  C -----***              relation name is checked                ***-----
1009        CALL CODE
1010        READ(JVAR,27) JVAR1
1011  27    FORMAT(I2)
1012        IF(JVAR1.EQ.1) GO TO 47
1013        WRITE(1,28) IBUFL
1014  28    FORMAT(1H ,5A2,2X,"RELATION NAME SUSPENDED")
1015        GO TO 710
1016  47    CALL CLOSE(IDCB)
1017        ITEST=1
1018  C -----*** A flag is set to 1 to indicate the presence of relation ***----
1019  C -----*** in DBMSRD directory & file DBMS8 is opened . Attributes ***----
1020  C -----*** are checked which have occured in query statement.  -----***----
1021        CALL OPEN(IDCB,IERR,NAM8,1,ISECU)
1022        IF(IERR.NE.2) GO TO 705
1023        IRSIZE=45
1024  39    IBEG=2*ICHAR+1
1025        DO 55 IV=NANA,2,-1
1026        KAUNT=0
1027        CALL SFILL(IBUF2,1,20,BLANK)
1028        DO 46 IU=1,10
1029        IF(ITAB1(IV,IU).EQ.1H ) GO TO 44
1030        KAUNT=KAUNT+1
1031  46    IBUF2(IU)=ITAB1(IV,IU)
1032  44    CALL SMOVE(IBUF2,1,2*KAUNT,IBUF,IBEG)
1033        ICONT=KAUNT+ICHAR
1034        IF(ICONT.LT.20) CALL SFILL(IBUF,(2*ICONT)+1,40,BLANK)
1035        IF(ITEST.EQ.0) GO TO 29
1036        CALL HASS(IBUF,20,31,NUMBR)
1037  31    CALL CODE
1038        WRITE(IBUFL,32) (IBUF(JK),JK=1,20)
1039  32    FORMAT(20A1)
1040        CALL SFILL(IBUF2,1,170,BLANK)
1041        CALL READF(IDCB,IERR,IBUF2,IRSIZE,LEN,NUMBR)
1042        IF(IERR.LT.0) GO TO 705
1043        IF(ITEST.EQ.0) GO TO 90
1044        DO 48 IBUC=1,3
1045        JBEG=1+(IBUC-1)*32
1046        JEND=20+(IBUC-1)*32
1047        IF(JSCOM(IBUF2,JBEG,JEND,IBUFL,1,IER)) 48,50,48
1048  48    CONTINUE
1049        WRITE(6,53)
1050  53    FORMAT(1H ,'ATTRIBUTE NOT IN BUCKET')
1051        GO TO 710
1052  50    LVAL=LVAL+1
1053        ISTRT=23+(IBUC-1)*32
1054        IEND= 28+(IBUC-1)*32
```

```
1055          CALL SMOVE(IBUF2,ISTRT,IEND,IBFR,1)
1056          CALL CODE
1057          READ(IBFR,83) ISTART,LENTH
1058    83    FORMAT(I4,I2)
1059    C -----*** According to relation names different data files are  ***-----
1060    C -----***---------           opened for specific use          ---------***-----
1061    84    IF((NUMB.EQ.1).AND.(NVAR.EQ.1)) GO TO 600
1062          IF((NUMB.EQ.2).AND.(NVAR.EQ.1)) GO TO 700
1063          IF((NUMB.EQ.2).AND.(NVAR.EQ.2)) GO TO 708
1064          IF((NUMB.EQ.4).AND.(NVAR.EQ.1)) GO TO 800
1065          IF((NUMB.EQ.4).AND.(NVAR.EQ.2)) GO TO 810
1066          IF((NUMB.EQ.5).AND.(NVAR.EQ.1)) GO TO 820
1067          IF((NUMB.EQ.5).AND.(NVAR.EQ.2)) GO TO 830
1068          IF((NUMB.EQ.6).AND.(NVAR.EQ.1)) GO TO 835
1069          WRITE(6,85)
1070    85    FORMAT(1H ,"RELATION NAME IS WRONG")
1071          GO TO 710
1072    90    DO 92 IL=1,4
1073          JSTRT=1+(IL-1)*36
1074          JEND=36+(IL-1)*36
1075          IF(JSCOM(IBUFL,1,ICONT,IBUF2,JSTRT,IER)) 92,94,92
1076    92    CONTINUE
1077          WRITE(6,93) IBUFL
1078    93    FORMAT(1H ,"ATTRIBUTE CONCATENATED WITH VIEW RELATION ",2X,
1079         *"***",1X,20A2,"***",1X,"DOESN'T EXISTS IN DIRECTORY")
1080          GO TO 710
1081    94    LVAL=LVAL+1
1082          KSTRT=27+(IL-1)*36
1083          KEND=36+(IL-1)*36
1084          CALL SFILL(IBFR,1,10,BLANK)
1085          CALL SMOVE(IBUF2,KSTRT,KEND,IBFR,1)
1086    C -----***          Conversion from A format to I format          ***-----
1087    C -----***                 in next few statements                 ***-----
1088          CALL CODE
1089          READ(IBFR,96)ISTART,LENTH,NUMB,NVAR
1090    96    FORMAT(I4,I2,I2,I2)
1091          IF(LVAL.EQ.1) GO TO 84
1092          GO TO 500
1093    600   CALL OPEN(IDCB2,IERR,NAM9,2)
1094          IF(IERR.NE.2) GO TO 705
1095          GO TO 500
1096    700   CALL OPEN(IDCB2,IERR,NAM1,2)
1097          IF(IERR.NE.2) GO TO 705
1098          GO TO 500
1099    708   CALL OPEN(IDCB2,IERR,NAM5,2,ISECU)
1100          IF(IERR.NE.2) GO TO 705
1101          GO TO 500
1102    800   CALL OPEN(IDCB2,IERR,NAM2,2)
```

```
1103          IF(IERR.NE.2) GO TO 705
1104          GO TO 500
1105   810    CALL OPEN(IDCB2,IERR,NAM6,2,ISECU)
1106          IF(IERR.NE.2) GO TO 705
1107          GO TO 500
1108   820    CALL OPEN(IDCB2,IERR,NAM10,2,ISECU)
1109          IF(IERR.NE.2) GO TO 705
1110          GO TO 500
1111   830    CALL OPEN(IDCB2,IERR,NAM11,2,ISECU)
1112          IF(IERR.NE.2) GO TO 705
1113          GO TO 500
1114   835    CALL OPEN(IDCB2,IERR,NAM12,2,ISECU)
1115          IF(IERR.NE.2) GO TO 705
1116   500    CALL SFILL(IB,1,80,BLANK)
1117          DO 601 ID=1,40
1118   601    IB(ID)=ITAB(IV-1,ID)
1119   19     CALL SFILL(IBUFL,1,40,BLANK)
1120   C -----*** Conversion from A1 format to A2 format  ***------
1121          CALL CODE
1122          WRITE(IBUFL,18) (IB(L),L=1,40)
1123   18     FORMAT(40A1)
1124          DO 690 IH=1,10
1125          CALL SFILL(IBUF2,1,130,BLANK)
1126          IF(NUM.GE.1) GO TO 696
1127          IREC=IH
1128   C -----*** Record size of data files are assigned here according  ***-----
1129   C -----***      to relation name specified in query statement     ***-----
1130          IF(NUMB.EQ.1) IRSIZ=62
1131          IF(NUMB.EQ.2) IRSIZ=65
1132          IF(NUMB.EQ.4) IRSIZ=48
1133   5705    CALL READF(IDCB2,IERR,IBUF2,IRSIZ,LEN,IREC)
1134          IF(IERR.LT.0) GO TO 705
1135          IF(JSCOM(IBUF2,ISTART,ISTART+(LENTH-1),IBUFL,1,IER)) 690,692,690
1136   690    CONTINUE
1137          IF((NUM.EQ.IVAR5-1).AND.((ISTB(1,1).EQ.1).AND.(ISTB(1,2).EQ.16
1138         *))) GO TO 698
1139   617    WRITE(6,691) (IBUFL,I=1,20)
1140   691    FORMAT(1H ,'NO RECORD CORRESPONDS TO LITERAL',2X,20A2)
1141          GO TO 710
1142   692    NUM=NUM+1
1143   55     CONTINUE
1144          IF((NUM.EQ.IVAR5).AND.((ISTB(1,1).EQ.1).AND.(ISTB(1,2).EQ.16)
1145         *)) GO TO 694
1146          IF((NUM.EQ.IVAR5).AND.((ISTB(1,1).EQ.1).AND.(ISTB(1,2).EQ.6)
1147         *)) GO TO 695
1148   698    WRITE(6,976) ( IBUF2(IF),IF=1,65)
1149   976    FORMAT(1H ," Tuple before update is :- ",/,2X,65A2)
1150          CALL SMOVE(IBUFL,1,LENTH,IBUF2,ISTART)
```

```
1151          WRITE(6,977) (IBUF2(IF),IF=1,65)
1152   977    FORMAT(1H ,"Updated record",/,2X,65A2)
1153   678    CALL WRITF(IDCB2,IERR,IBUF2,0,IREC)
1154          IF(IERR.LT.0) GO TO 705
1155          IF((ISTB(1,1).EQ.1).AND.(ISTB(1,2).EQ.16)) GO TO 694
1156          M=10-IREC
1157          IF(M.EQ.0) GO TO 694
1158          DO 680 J=1,M
1159          IP=J+IREC
1160          IQ=IP-1
1161          CALL SFILL(IBUF2,1,130,BLANK)
1162          CALL READF(IDCB2,IERR,IBUF2,65,LEN,IP)
1163          IF(IERR.LT.0) GO TO 705
1164          CALL WRITF(IDCB2,IERR,IBUF2,0,IQ)
1165          IF(IERR.LT.0) GO TO 705
1166   680    CONTINUE
1167          CALL SFILL(IBUF2,1,130,BLANK)
1168          CALL WRITF(IDCB2,IERR,IBUF2,0,IP)
1169          IF(IERR.LT.0) GO TO 705
1170    694   CALL CLOSE(IDCB2)
1171          WRITE(6,677)
1172   677    FORMAT(1H ,"QUERY PROCESSED ")
1173          GO TO 710
1174   695    CALL SFILL(IBUF2,1,130,BLANK)
1175          GO TO 678
1176   696    IREC=IREC
1177          GO TO 5705
1178   C -----*** File manager errors are issued to user if he tries to ****-----
1179   C        * access wrong relation name or data file                    *
1180   705      WRITE(1,709) IERR
1181   709      FORMAT(1H ," FMGR ERROR ",2X,I4)
1182   710      CALL EXEC( ICODX,LLNAM)
1183          END
1184          PROGRAM DEFVE,5
1185   C -----*** This segment is to define new views on existing relation  ***-
1186   C      ***---------          name & its attributes          ----------***--
1187          INTEGER IDCB(400),NAM8(3),IBUFR(40),IBUF(128),IBUF1(20),BLANK,
1188         *IHOST(128),IPARS(50,3),ITAB1(30,22),ITEM(5),IBUK(5),ISECU(3),
1189         *IBLNK(20),ISTOR(6,30),NAMR(3),ICC(5),NAM5(3),ITAB(30,42)
1190         *,IBUFL(10),LVAR(2),IDCB1(144),NAM1(3),ISTB(50,2),LRNAM(3)
1191          COMMON ITAB,ITAB1,ISTB,N,NANA,IVAR5,IPARS
1192          DATA NAM8,NAMR,ISECU,NAM5/2HDB,2HMS,2H00,2HDB,2HMS,2H01,2H-7,2H56,
1193         *2H19,2HDB,2HMS,2H8 /
1194          DATA NAM1,LRNAM,ICODC /2HDB,2HMS,2HRD,2HCL,2HEA,2HR  ,8/
1195          ICAR=0
1196          INUM=0
1197          KAUNT=0
1198          IDAT=0
```

```
1199          IBEG=0
1200          IVAR3=0
1201          IVAR9=0
1202          IXZ=0
1203          IVEW=0
1204          IER=0
1205          IBAR=0
1206          INAM=0
1207          BLANK=000040B
1208          DO 10 I=1,N
1209    C -----*** IDENTIFIERS ( relation name concatenated with its       ***-----
1210    C          ***                  attributes ) are checked           ***
1211          IF((IPARS(I,1).EQ.4).AND.((IPARS(I-1,1).EQ.2).AND.(IPARS(I-1,2).
1212         *EQ.32)).AND.((IPARS(I+1,1).EQ.2).AND.(IPARS(I+1,2).EQ.33)))
1213         * GO TO 110
1214    10    CONTINUE
1215          IF(INUM.EQ.IDAT) GO TO 135
1216          WRITE(1,20)  I
1217    20    FORMAT(1H ,"NO RECORD CORRESPONDS TO ID :",2X,I2)
1218          GO TO 288
1219    110   IDAT=IDAT+1
1220          CALL SFILL(IBUFR,1,40,BLANK)
1221    C -----*** NO. of characters are counted in IDENTIFIERS .          ***-----
1222    C          ***  Table ITAB1 contains all identifiers coming in a      ***
1223    C          ***  query statement. Contents of ITAB1 are transferred to  ***
1224    C          ***           a buffer IBUFR                               ***
1225          DO 115 II=1,20
1226          IF(ITAB1(IPARS(I,2),II).EQ.1H ) GO TO 116
1227          IBUFR(II)=ITAB1(IPARS(I,2),II)
1228    115   ICAR=ICAR+1
1229    116   IF(ICAR.NE.20) CALL SFILL(IBUFR,(2*ICAR)+1,40,BLANK)
1230          CALL HASS(IBUFR,20,31,NUMBR)
1231    C -----*** File DBMS8 is opened to check the concatenated relation with *
1232    C          ***----------------          its attributes .          ----------**
1233          CALL OPEN(IDCB,IERR,NAM5,1,ISECU)
1234          IF(IERR.NE.2) GO TO 299
1235          CALL SFILL(IBUF,1,256,BLANK)
1236          CALL READF(IDCB,IERR,IBUF,45,LEN,NUMBR)
1237          IF(IERR.LT.0) GO TO 299
1238    C -----*** Conversion from A1 format to A2 format  ***-----
1239          CALL CODE
1240          WRITE(IBLNK,113) (IBUFR(IX),IX=1,20)
1241    113   FORMAT(20A1)
1242    C -----*** Concatenated relation & attribute is searched in bucket ***----
1243          DO 120 IBUC=1,3
1244          JBEG=1+(IBUC-1)*32
1245          JEND=20+(IBUC-1)*32
1246          IF(JSCOM(IBUF,JBEG,JEND,IBLNK,1,IER)) 120,130,120
```

```
1247  120    CONTINUE
1248         WRITE(1,122) IBLNK
1249  122    FORMAT(1H ," You are giving Wrong RELATION & ATTRIBUTE  in ",
1250        */,1H ,"Hashed FORM:_",2X,10A2)
1251         GO TO 288
1252  130    INUM=INUM+1
1253         CALL SFILL(IBUF1,1,60,BLANK)
1254         MBEG=21+(IBUC-1)*32
1255         MEND=32+(IBUC-1)*32
1256         CALL CONVR(NUMBR,IHASH)
1257         CALL CONVR(IBUC,IBUKET)
1258         IBUF1(1)=IHASH
1259         IBUF1(2)=IBUKET
1260         CALL SMOVE(IBUF,MBEG,MEND,IBUF1,5)
1261         CALL SMOVE(IBUF1,13,16,LVAR,1)
1262  C -----*** Conversion from A format to I format ***-----
1263         CALL CODE
1264         READ(LVAR,128) JFILE,JBUKET
1265  128    FORMAT(I2,I2)
1266         IBUFL(INUM)=JFILE
1267         IF(INUM.EQ.1) GO TO 235
1268         IF(INUM.GT.1) GO TO 133
1269  129    DO 131 IVAR=1,8
1270  131    ISTOR(IDAT,IVAR)=IBUF1(IVAR)
1271         GO TO 10
1272  133    IF(IBUFL(INUM).EQ.IBUFL(INUM-1)) GO TO 129
1273         WRITE(6,134) (IBUFL(KO),KO=1,INUM)
1274  134    FORMAT(1H ,"You are defining view on different relations",5I2)
1275         GO TO 288
1276  C -----*** File DBMSRD is opened to check the relation name on which  ***
1277  C      ***------      user is going to define his own view        ------***
1278  235    CALL OPEN(IDCB1,IERR,NAM1,1,ISECU)
1279         IF(IERR.NE.2) GO TO 299
1280         KSTRT=11+(JBUKET-1)*12
1281         CALL SFILL ( IBUF,1,60,BLANK)
1282         CALL READF(IDCB1,IERR,IBUF,30,LEN,JFILE)
1283         IF(IERR.LT.0) GO TO 299
1284  C -----*** Relation's status is checked in few next statements whether  *
1285  C      ***-----       relation is suspended or restored           -----*
1286         CALL SMOVE(IBUF,KSTRT,KSTRT+1,MVAR,1)
1287         CALL CODE
1288         READ(MVAR,236) MVAL
1289  236    FORMAT(I2)
1290         IF(MVAL.EQ.1) GO TO 129
1291         WRITE(1,337)
1292  337    FORMAT(1H ,"RELATION NAME IS SUSPENDED ON WHICH YOU ARE DEF. VEW")
1293         GO TO 288
1294  135    CALL SMOVE(IBUF1,13,16,LVAR,1)
```

```
1295          CALL CLOSE(IDCB)
1296    C -----*** File DBMS00 is opened in which user will check the existence *
1297    C     *** of view relation name & entry will be given with its status  *
1298    C     ***----   provided it is not existing in this directory      -----**
1299          CALL OPEN(IDCB,IERR,NAM8,2,ISECU)
1300          IF(IERR.NE.2) GO TO 299
1301          CALL SFILL(IBUFR,1,40,BLANK)
1302          CALL SFILL(IBLNK,1,40,BLANK)
1303          DO 136 IM=1,10
1304          IF(ITAB1(1,IM).EQ.1H ) GO TO 147
1305          IVAR3=IVAR3+1
1306    136   IBUFR(IM)=ITAB1(1,IM)
1307    147   CALL HASS(IBUFR,IVAR3,19,NUMB)
1308          IF(NUMB.EQ.0) NUMB=19
1309          CALL SFILL(IBUF,1,128,BLANK)
1310          CALL READF(IDCB,IERR,IBUF,32,LEN,NUMB)
1311    C -----***    conversion from A1 format to A2 format       ***--------
1312          CALL CODE
1313          WRITE(IBUF1,137) (IBUFR(IT),IT=1,10)
1314    137   FORMAT(10A1)
1315          DO 138 IBUC=1,5
1316          ISTRT=1+(IBUC-1)*16
1317          IEND=10+(IBUC-1)*16
1318          IF(IVAR3.LT.10) CALL SFILL(IBUF1,IVAR3+1,10,BLANK)
1319          IF(JSCOM(IBUF,ISTRT,IEND,IBUF1,1,IER)) 148,139,148
1320    148   IF(JSCOM(IBUF,ISTRT,IEND,IBLNK,1,IER)) 138,159,138
1321    159   IF(IXZ.EQ.1) GO TO 138
1322          IXZ=IXZ+1
1323          ITERM=NUMB
1324          IBAK=IBUC
1325    138   CONTINUE
1326          DO 151 IVAX1=20,22
1327          CALL SFILL(IBUF,1,64,BLANK)
1328          CALL READF(IDCB,IERR,IBUF,32,LEN,IVAX1)
1329          IF(IERR.LT.0) GO TO 299
1330          DO 151 IVAX=1,5
1331          ISTRT=1+(IVAX-1)*16
1332          IEND=10+(IVAX-1)*16
1333          IF(JSCOM(IBUF,ISTRT,IEND,IBUF1,1,IER)) 154,139,154
1334    154   IF(JSCOM(IBUF,ISTRT,IEND,IBLNK,1,IER)) 151,155,151
1335    155   IF(IXZ.EQ.1) GO TO 151
1336          IXZ=IXZ+1
1337          ITERM=IVAX1
1338          IBAK=IVAX
1339    151   CONTINUE
1340          IF(IXZ.EQ.1) GO TO 160
1341          WRITE(1,150)
1342    150   FORMAT(1H ,"PLEASE GIVE SOME ANOTHER VIEW RELATION NAME",
```

```
1343         */,"ALL PLACE OF BUCKET IN HASHED RECORD NO & OVERFOLW AREA ",
1344         */,"*** IS FULL ")
1345          GO TO 288
1346   160    CALL SFILL(IBUF,1,256,BLANK)
1347          CALL READF(IDCB,IERR,IBUF,128,LEN,ITERM)
1348          LBEG=1+(IBAK-1)*16
1349          LEND=16+(IBAK-1)*16
1350   C -----*** View relation name with its restored status is given   ***-----
1351   C      ***-----------          in DBMS00 directory          ----------***
1352          NVAR=1
1353          CALL CONVR(NVAR,NVAR1)
1354          CALL SMOVE(NVAR1,1,2,IBUF1,11)
1355          CALL SMOVE(LVAR,1,4,IBUF1,13)
1356          CALL SMOVE(IBUF1,1,16,IBUF,LBEG)
1357          CALL WRITF(IDCB,IERR,IBUF,0,ITERM)
1358          IF(IERR.LT.0) GO TO 299
1359          GO TO 237
1360   139    KAUNT=1
1361   237    CALL CLOSE(IDCB)
1362   C -----*** File DBMS01 is opened where existence of attributes will be **
1363   C      ***-----------------------------------          checked .          ---------***-
1364          CALL OPEN(IDCB,IERR,NAMR,2,ISECU)
1365          IF(IERR.NE.2) GO TO 299
1366          DO 280 IS=2,NANA,2
1367          IYZ=0
1368          KOUNT=0
1369          IVAR9=IVAR9+1
1370          CALL SFILL(IBUF1,1,40,BLANK)
1371          DO 240 ISS=1,10
1372          IF(ITAB1(IS,ISS).EQ.1H ) GO TO 245
1373          IBUF1(ISS)=ITAB1(IS,ISS)
1374   240    KOUNT=KOUNT+1
1375   245    CALL SMOVE(IBUF1,1,2*KOUNT,IBUFR,(2*IVAR3)+1)
1376          IVAR4=IVAR3+KOUNT
1377          CALL SFILL(IBUFR,(2*IVAR4)+1,40,BLANK)
1378          CALL HASS(IBUFR,IVAR4,31,NUMBR)
1379          IF(NUMBR.EQ.0) NUMBR=31
1380          ICC(IVAR9)=NUMBR
1381          IF(IVAR9.GT.1) GO TO 230
1382   246    CALL SFILL(IBLNK,1,40,BLANK)
1383          CALL SFILL(IBUF1,1,120,BLANK)
1384          CALL CODE
1385          WRITE(IBUF1,249) (IBUFR(IU),IU=1,20)
1386   249    FORMAT(20A1)
1387          CALL SMOVE(IBUF1,1,20,IBUF,1)
1388          CALL SFILL(IBLNK,1,40,BLANK)
1389   C -----***    Attributes are searched in bucket of DBMS8 directory   ***-
1390          DO 247 IMM=1,8
```

```
1391  247    IBLNK(IMM)=ISTOR(IVAR9,IMM)
1392         CALL SMOVE(IBLNK,1,16,IBUF,21)
1393         CALL SFILL(IHOST,1,256,BLANK)
1394         CALL READF(IDCB,IERR,IHOST,85,LEN,NUMBR)
1395         IF(IERR.LT.0) GO TO 299
1396         CALL SFILL(IBLNK,1,40,BLANK)
1397         DO 258 IVAX2=1,4
1398         NBEG=1+(IVAX2-1)*36
1399         NEND=36+(IVAX2-1)*36
1400         IF(JSCOM(IHOST,NBEG,NEND,IBUF,1,IER)) 252,270,252
1401  252    IF(JSCOM(IHOST,NBEG,NEND,IBLNK,1,IER)) 258,255,258
1402  255    IF(IYZ.EQ.1) GO TO 258
1403         IYZ=IYZ+1
1404         ITEM(IVAR9)=NUMBR
1405         IBUK(IVAR9)=IVAX2
1406  258    CONTINUE
1407  C ------*** Attributes are searched in overflow area of directory ***------
1408         DO 260 IVAX1=32,34
1409         CALL SFILL(IHOST,1,256,BLANK)
1410         CALL READF(IDCB,IERR,IHOST,85,LEN,IVAX1)
1411         IF(IERR.LT.0) GO TO 299
1412         DO 260 IBHAR=1,4
1413         NBEG=1+(IBHAR-1)*36
1414         NEND=36+(IBHAR-1)*36
1415         IF(JSCOM(IHOST,NBEG,NEND,IBUF,1,IER)) 262,270,262
1416  262    IF(JSCOM(IHOST,NBEG,NEND,IBLNK,1,IER)) 260,265,260
1417  265    IF(IYZ.EQ.1) GO TO 260
1418         IYZ=IYZ+1
1419         ITEM(IVAR9)=IVAX1
1420         IBUK(IVAR9)=IBHAR
1421  260    CONTINUE
1422         IF(IYZ.EQ.1) GO TO 273
1423  261    WRITE(1,271)
1424  271    FORMAT(1H ,"PLEASE GIVE SOME ANOTHER VIEW RELATIONE NAME",
1425        *//,1H ,"THIS VEW RELATION HAS BEEN DEFINED BY SOME USER")
1426         GO TO 288
1427  230    IF(ICC(IVAR9).EQ.ICC(IVAR9-1)) GO TO 261
1428         GO TO 246
1429  273    IBAR=IBAR+1
1430         DO 272 IVAR8=1,18
1431  272    ISTOR(IBAR,IVAR8)=IBUF(IVAR8)
1432         GO TO 280
1433  270    IVEW=IVEW+1
1434  280    CONTINUE
1435         IF((IVEW.EQ.IVAR9).AND.(KAUNT.EQ.1)) GO TO 290
1436         IF((KAUNT.EQ.0).AND.(IVEW.EQ.0)) GO TO 275
1437         GO TO 261
1438  275    DO 285 IW=1,IVAR9
```

```
1439          CALL SFILL(IHOST,1,256,BLANK)
1440          ITERM=ITEM(IW)
1441          IBHAR=IBUK(IW)
1442          CALL READF(IDCB,IERR,IHOST,128,LEN,ITERM)
1443          IF(IERR.LT.0) GO TO 299
1444          LBEG=1+(IBHAR-1)*36
1445          LEND=36+(IBHAR-1)*36
1446          CALL SFILL(IBUF1,1,48,BLANK)
1447          DO 276 IE=1,18
1448   276    IBUF1(IE)=ISTOR(IW,IE)
1449          CALL SMOVE(IBUF1,1,36,IHOST,LBEG)
1450          CALL WRITF(IDCB,IERR,IHOST,0,ITERM)
1451          IF(IERR.LT.0) GO TO 299
1452   285    CONTINUE
1453          CALL CLOSE(IDCB)
1454   286    WRITE(6,287)
1455   287    FORMAT(1H ,"QUERY IS PROCESSED ")
1456   288    CALL EXEC(ICODC,LRNAM)
1457   290    WRITE(1,291)
1458   291    FORMAT(1H ,"YOUR DEFINED VIEW EXISTS IN DIRECTORY")
1459          GO TO 288
1460   C ------***  File manager errors are given to user in case of accessing **
1461   C      ***  wrong directory or beyond the directory                      **
1462   299    WRITE(1,303) IERR
1463   303    FORMAT(1H ,"** FMGR ERROR **:",2X,I5)
1464          END
1465          PROGRAM RESTT,5
1466   C ------*** This segment either RESTORES or SUSPENDS a relation    ***------
1467   C -------*-------          name as needed by user                 -------***------
1468          INTEGER IDCB(144),IBUFR(20),IBUFL(20),NAME(3),LNAM(3),ISECU(3),
1469         *BLANK,ITAB1(30,22),LBUFR(85),ISTB(50,2),ITAB(30,42),IBB(50,3),
1470         *LLNAM(3),ICODE(20),ISTB1(20,8),LRNAM(3)
1471          COMMON ITAB,ITAB1,ISTB,LKM,JNN,JJVAR,IBB,IQRY,IF,ISTB1,KMM,ICODE
1472         *,JJNUM,JFLAG
1473          DATA NAME,LNAM,ISECU/2HDB,2HMS,2HRD,2HDB,2HMS,2H00,2H-7,2H56,2H19/
1474          DATA LLNAM,LRNAM,ICODC/2HUS,2HRC,2HOD,2HCL,2HEA,2HR ,8/
1475          BLANK=000040B
1476          KOUNT=0
1477   C      ***  characters are counted .                                    ***
1478          DO 10 I=1,10
1479          IF(ITAB1(1,I).EQ.1H ) GO TO 15
1480          IBUFR(I)=ITAB1(1,I)
1481   10     KOUNT=KOUNT+1
1482   15     IF(KOUNT.LT.10) CALL SFILL(IBUFR,(2*KOUNT)+1,20,BLANK)
1483   C ------*** Relation name is hashed & an address is obtained      ***------
1484          CALL HASS(IBUFR,10,6,NUMB)
1485          IF(NUMB.EQ.0) NUMB=6
1486   C ------*** Conversion from A1 format to A2 format ***------
```

```
1487          CALL CODE
1488          WRITE(IBUFL,20) (IBUFR(J),J=1,10)
1489    20    FORMAT(10A1)
1490          IF(JFLAG.EQ.1) GO TO 28
1491          CALL CODE
1492          WRITE(IBUFR,25) (ISTB1(1,L),L=1,6)
1493    25    FORMAT(6A1)
1494          CALL SMOVE(IBUFL,1,KOUNT,IBUFR,7)
1495    C -----*** File USRCOD is opened to check the right of user ( either   *>
1496    C     *** to restore or suspend the relation name or VIEW relation name
1497          CALL OPEN(IDCB,IERR,LLNAM,1,ISECU)
1498          IF(IERR.NE.2) GO TO 45
1499          CALL SFILL(LBUFR,1,170,BLANK)
1500          CALL READF(IDCB,IERR,LBUFR,85,LEN,JJNUM)
1501          IF(IERR.LT.0) GO TO 45
1502          DO 24 I=1,10
1503          KSTRT=7+(I-1)*18
1504          IF(JSCOM(IBUFR,1,6+KOUNT,LBUFR,KSTRT,IER)) 24,26,24
1505    24    CONTINUE
1506          WRITE(1,29)
1507    29    FORMAT(1H ,"YOU CANNOT DO SO ")
1508          GO TO 65
1509    26    CALL CLOSE(IDCB)
1510    C -----*** File DBMSRD is opened to check the relation name which user *>
1511    C     ***          wants to either suspend or restore                ***-
1512    28    CALL OPEN(IDCB,IERR,NAME,2,ISECU)
1513          IF(IERR.NE.2) GO TO 45
1514          CALL SFILL(LBUFR,1,60,BLANK)
1515          CALL READF(IDCB,IERR,LBUFR,30,LEN,NUMB)
1516          IF(IERR.LT.0) GO TO 45
1517          IER=0
1518          DO 30 IVAR=1,4
1519          LSTRT=1+(IVAR-1)*12
1520          LEND=10+(IVAR-1)*12
1521          IF(JSCOM(IBUFL,1,10,LBUFR,LSTRT,IER)) 30,35,30
1522    30    CONTINUE
1523          GO TO 50
1524    35    IF((ISTB(1,1).EQ.1).AND.(ISTB(1,2).EQ.14)) GO TO 40
1525    C -----*** This portion of program restores the relation name      ***---
1526          N=1
1527    36    CALL CONVR(N,M)
1528          CALL SMOVE(M,1,2,LBUFR,LEND+1)
1529          CALL WRITF(IDCB,IERR,LBUFR,0,NUMB)
1530          IF(IERR.LT.0) GO TO 45
1531          CALL CLOSE(IDCB)
1532          GO TO 63
1533    C -----*** This portion of program suspends the relation name       ***--
1534    40    N=0
```

```
1535          GO TO 36
1536    C -----*** File manager error is given to user if he tries to acess ***--
1537    C -----*** the wrong directory or beyond the directory size          ***--
1538    45    WRITE(1,48) IERR
1539    48    FORMAT(1H ,"**** FMGR ERROR ******",2X,I4)
1540          GO TO 65
1541    49    NVAR=0
1542          GO TO 61
1543    50    CALL CLOSE(IDCB)
1544    C -----*** File DBMS00 is opened to check the view relation name if    ***
1545    C     ***   relation name is not found in DBMSRD directory            ***
1546          CALL OPEN(IDCB,IERR,LNAM,2,ISECU)
1547          IF(IERR.NE.2) GO TO 45
1548          CALL HASS(IBUFR,KOUNT,19,NUMBR)
1549          IF(NUMBR.EQ.0) NUMBR=19
1550          CALL SFILL(LBUFR,1,170,BLANK)
1551          CALL READF(IDCB,IERR,LBUFR,85,LEN,NUMBR)
1552          IF(IERR.LT.0) GO TO 45
1553    C -----***          View relation name is searched in bucket        ***---
1554          DO 55 IVAR=1,5
1555          LSTRT=1+(IVAR-1)*16
1556          LEND=10+(IVAR-1)*16
1557          IF(JSCOM(IBUFL,1,KOUNT,LBUFR,LSTRT,IER)) 55,60,55
1558    55    CONTINUE
1559          WRITE(1,58) IBUFL
1560    58    FORMAT(1H ,"RELATION NAME ",2X,5A2,2X,"NEITHER IN ACTUAL NOR VIEW
1561         * DIRECTORY")
1562          GO TO 65
1563    C-----*** Next statement checks whether user wants to either  suspend   *
1564    C -----***          or restore the view relation name.                 ***-
1565    60    IF((ISTB(1,1).EQ.1).AND.(ISTB(1,2).EQ.14)) GO TO 49
1566    C -----*** View relation name is restored in next few statements    ***--
1567          NVAR=1
1568    61    CALL CONVR(NVAR,LVAR)
1569          CALL SMOVE(LVAR,1,2,LBUFR,LEND+1)
1570          CALL WRITF(IDCB,IERR,LBUFR,0,NUMBR)
1571          CALL CLOSE(IDCB)
1572    63    WRITE(6,64)
1573    64    FORMAT(1H ,"QUERY IS PROCESSED ")
1574    65    CALL EXEC(I"DC,LRNAM)
1575          END
1576          PROGRAM GRNTT,5
1577    C     ************************************************************************
1578    C     * This segment helps the DATA BASE ADMINISTRATOR to grant some  *
1579    C     * grant opitions ( like TO MODIFY , TO DELETE , TO INSRTN )     *
1580    C     * to some of his responsible person.                             *
1581    C     ************************************************************************
1582          INTEGER IDCB(400),NAM7(3),NAM8(3),BLANK,ITAB1(30,22),IBUF(40),
```

```
      BUF1(128),IBUF2(85),ITAB(30,42),IB(40),IDCB1(128),IPARS(50,3)
     NAM1(3),IBUFL(20),ISECU(3),NAM3(3),IBUFF(20),ISTB1(20,8)
     ,ISTB(50,2),ICODE(20),IBUFX(6),IBUFY(6),IBUFZ(6),LRNAM(3)
      COMMON  ITAB,ITAB1,ISTB,MM,NANA,IVARS,IPARS,IQRY,IF,ISTB1,KM,ICODE
      DATA NAM7,NAM8,BLANK/2HDB,2HMS,2HRD,2HUS,2HRC,2HD ,000040B/
      DATA ISECU,NAM3/2H-7,2H56,2H19,2HDB,2HMS,2H00/
      DATA NAM1,LRNAM,ICODC/2HUS,2HRC,2HOD,2HCL,2HEA,2HR ,8/
      ICHAR=0
      NUM=0
      IFLAG=0
      ITEST=0
      MVAR=0
      CALL HASS (ICODE,6,7,NUMR)
      DO 2 KP=1,10
      IF(ITAB1(NANA,KP).EQ.1H ) GO TO 30
      ICHAR=ICHAR+1
2     IBUF(KP)=ITAB1(NANA,KP)
C -----*** File DBMSRD is opened to check the relation name on     ***-----
C     *** which DBA is going to grant some grant options           ***
30    CALL OPEN(IDCB,IERR,NAM7,1,ISECU)
      IF(IERR.NE.2) GO TO 99
      IRSIZ=30
      CALL HASS(IBUF,10,6,NUMB)
      IF(NUMB.EQ.0) NUMB=6
      CALL CODE
      WRITE(IBUFL,33) (IBUF(LK),LK=1,10)
33    FORMAT(10A1)
34    CALL SFILL(IBUF2,1,64,BLANK)
      CALL READF(IDCB,IERR,IBUF2,IRSIZ,LEN,NUMB)
      IF(IERR.LT.0) GO TO 99
      IF(IFLAG.EQ.1) GO TO 35
      IER=0
      DO 40 NVAR=1,4
      NSTRT=1+(NVAR-1)*12
      NFLAG=6+(NVAR-1)*6
      IF(JSCOM(IBUFL,1,ICHAR,IBUF2,NSTRT,IER)) 40,45,40
40    CONTINUE
      IFLAG=1
      CALL CLOSE(IDCB)
C -----*** File DBMS00 is opened to check the relation name if it is ***-
C     *** not found in actual relation directory DBMSRD              ***
      CALL OPEN(IDCB,IERR,NAM3,1,ISECU)
      IF(IERR.NE.2) GO TO 99
      IRSIZ=32
      CALL HASS(IBUF,ICHAR,19,NUMB)
      IF(NUMB.EQ.0) NUMB=19
      GO TO 34
35    DO 36 I=1,5
```

```
1631          LSTRT=1+(I-1)*14
1632          NFLAG=7+(I-1)*7
1633          IF(JSCOM(IBUFL,1,ICHAR,IBUF2,LSTRT,IER)) 36,38,36
1634    36    CONTINUE
1635          WRITE(1,37) IBUFL
1636    37    FORMAT(1H ,"RELATION NAME ","***",1X,5A2,1X,"***","NEITHER IN
1637         *VIEW",/,"NOR IN ACTUAL DIRECTORY ")
1638          GO TO 105
1639    C -----*** Status of view relation name is checked whether it is  ***----
1640    C     *** suspended or restored on which DBA is going to grant   ***
1641    C     *** some grant options.                                     ***
1642    38    CALL SMOVE(IBUF2,(2*NFLAG)-1,2*NFLAG,JVAR,1)
1643          CALL CODE
1644          READ(JVAR,16) JVAR1
1645    16    FORMAT(I2)
1646          IF(JVAR1.EQ.1) GO TO 47
1647          WRITE(1,15) IBUFL
1648    15    FORMAT(1H ,"VIEW RELATION NAME SUSPENDED",2X,5A2)
1649          GO TO 105
1650    C -----*** Status of relation name is checked in next few statements ***
1651    45    CALL SMOVE(IBUF2,(2*NFLAG)-1,2*NFLAG,JVAR,1)
1652          CALL CODE
1653          READ(JVAR,27) JVAR1
1654    27    FORMAT(I2)
1655          IF(JVAR1.EQ.1) GO TO 47
1656          WRITE(1,28) IBUFL
1657    28    FORMAT(1H ,5A2,2X,"RELATION NAME SUSPENDED")
1658          GO TO 105
1659    47    CALL CLOSE(IDCB)
1660    C -----*** File USRCD is opened to check the validity of user ***
1661          CALL OPEN(IDCB,IERR,NAM8,2,ISECU)
1662          IF(IERR.NE.2) GO TO 99
1663          CALL SFILL(IBUF,1,40,BLANK)
1664          DO 48 I=1,6
1665    48    IBUF(I)=ITAB(1,I)
1666          CALL HASS(IBUF,6,7,NUMBR)
1667          IF(NUMBR.EQ.2) GO TO 72
1668          CALL CODE
1669    C     ***  Conversion from A1 format to A2 format in next few statement
1670          WRITE(IB,50) (IBUF(II),II=1,6)
1671    50    FORMAT(6A1)
1672          IF (NUMBR.EQ.3) GO TO 96
1673          CALL SFILL(IBUF2,1,90,BLANK)
1674          CALL READF(IDCB,IERR,IBUF2,45,LEN,NUMBR)
1675          IF(IERR.LT.0) GO TO 99
1676          IF(JSCOM(IBUF2,1,6,IB,1,IER)) 55,60,55
1677    55    WRITE(1,58) (IB(IJ),IJ=1,3)
1678    58    FORMAT(1H ,3A2,2X,"NOT AUTHORISED USER ")
```

```
1679          STOP
1680    60    CALL SFILL(IBUF,1,40,BLANK)
1681    C     *** File USRCOD is opened where granted opitions will be entred  *
1682    C       *           against grantor name                                *
1683          CALL OPEN(IDCB1,IERR,NAM1,2,ISECU)
1684          IF(IERR.NE.2) GO TO 99
1685          CALL SFILL(IBUF1,1,256,BLANK)
1686          CALL READF(IDCB1,IERR,IBUF1,128,LEN,NUMBR)
1687          IF(IERR.LT.0) GO TO 99
1688    C -----*** Conversion from A1 format to A2 format       ***
1689          CALL CODE
1690          WRITE(IBUFZ,59) (ICODE(L),L=1,6)
1691    59    FORMAT(6A1)
1692          DO 120 IJ=1,10
1693    120   IBUFF(IJ)=ITAB1(NANA,IJ)
1694          CALL CODE
1695          WRITE(IBUFL,121) (IBUFF(IK),IK=1,10)
1696    121   FORMAT(10A1)
1697    64    DO 90 IVAR=2,KM-2
1698          CALL SFILL(IBUFF,1,40,BLANK)
1699          DO 65 IVAR1 = 1,6
1700    65    IBUFX(IVAR1)=ISTB1(IVAR,IVAR1)
1701          CALL CODE
1702          WRITE(IBUFY,62) (IBUFX(KP),KP=1,6)
1703    62    FORMAT(6A1)
1704          CALL SMOVE(IBUFY,1,6,IBUFF,1)
1705          CALL SMOVE(IBUFL,1,10,IBUFF,7)
1706          DO 70 IVAR2=1,10
1707          ISTRT=7+(IVAR2-1)*8
1708          IEND=12+(IVAR2-1)*8
1709          IF(JSCOM(IBUF2,ISTRT,IEND,IBUFY,1,IER)) 70,75,70
1710    70    CONTINUE
1711          WRITE(1,71)IBUFY
1712    71    FORMAT(1H ,"KEY WORD NOT AVAILABLE IS ",2X,3A2)
1713          GO TO 105
1714    72    WRITE(1,73)
1715    73    FORMAT(1H ,"You are giving GRANT right to yourself")
1716          GO TO 105
1717    96    WRITE(6,97) (IB(LLL),LLL=1,3)
1718    97    FORMAT(1H ," You can't give your grant right to user :- ",3A2)
1719          GO TO 105
1720    75    IF((ISTB(1,1).EQ.1).AND.(ISTB(1,2).EQ.15)) GO TO 74
1721          M=1
1722    78    CALL CONVR(M,N)
1723          CALL SMOVE(N,1,2,IBUFF,17)
1724    74    DO 77 IVAR3=1,10
1725          JSTRT=7+(IVAR3-1)*18
1726          JEND=22+(IVAR3-1)*18
```

```
1727          IF((ISTB(1,1).EQ.1).AND.(ISTB(1,2).EQ.15)) GO TO 85
1728          IF(JSCOM(IBUF1,JSTRT,JEND,IBUFF,1,IER)) 76,82,76
1729    76    IF(JSCOM(IBUF1,JSTRT,JEND,IBUF,1,IER)) 77,79,77
1730    79    CALL SMOVE(IBUFF,1,18,IBUF1,JSTRT)
1731          CALL SMOVE(IB,1,6,IBUF1,1)
1732          GO TO 90
1733    85    IF(JSCOM(IBUF1,JSTRT,JEND,IBUFF,1,IER)) 77,92,77
1734    86    WRITE(1,89)
1735    89    FORMAT(1H,"You can revoke only what you have granted ")
1736          GO TO 105
1737    92    CALL SFILL(IBUF1,JSTRT,JEND+2,BLANK)
1738          GO TO 90
1739    77    CONTINUE
1740          IF((ISTB(1,1).EQ.1).AND.(ISTB(1,2).EQ.15)) GO TO 86
1741          WRITE(1,81)
1742    81    FORMAT(1H ,"NO PLACE")
1743          GO TO 105
1744    82    CALL SMOVE(N,1,2,IBUF1,JEND+1)
1745    90    CONTINUE
1746    C     *** Entry is made in USRCOD directory for granted rights    ***
1747          CALL WRITF(IDCB1,IERR,IBUF1,0,NUMBR)
1748          IF(IERR.LT.0) GO TO 99
1749          CALL CLOSE(IDCB)
1750          CALL CLOSE(IDCB1)
1751          WRITE(6,93)
1752    93    FORMAT(1H ,"QUERY PROCESSED ")
1753          GO TO 105
1754    C     *** FMGR errors are issued to user in case he must have tried   **
1755    C       * to access wrong relation name or beyond the directory size  **
1756    99    WRITE(1,100) IERR
1757    100   FORMAT(1H ,"FMGR ERROR ********",2X,I4)
1758    105   CALL EXEC(ICODC,LRNAM)
1759          END
1760          PROGRAM INSTN,5
1761    C ----*** This segment is to insert new tuple in data file    ***----
1762           INTEGER IDCB(400),NAM7(3),NAM8(3),BLANK,ITAB1(30,22),IBUF(40),
1763          *IBUF1(144),IBUF2(85),NAM9(3),ITAB(30,42),IB(40),IDCB1(400),IDCB2
1764          *(400),NAM1(3),NAM2(3),IBUFL(20),ISECU(3),IBFR(5),NAM3(3),NAM4(3)
1765          *,ISTB(50,2),TEMP(20),TEMP1(20),IPARS(50,3),NAM5(3),NAM6(3),NAM10
1766          *(3),NAM11(3),NAM12(3),NAM13(3),LLNAM(3),ISTB1(20,8),ICODE(20)
1767           COMMON ITAB,ITAB1,ISTB,MM,NANA,IVAR5,IPARS,IQRY,IF,ISTB1,KKM,ICODE
1768          *,JJNUM,JFLAG
1769           DATA NAM7,NAM8,NAM9,NAM1,NAM2,BLANK/2HDB,2HMS,2HRD,2HDB,2HMS,2H8 ,
1770          *2HDB,2HMS,2H11,2HCA,2HRD,2HAT,2HIS,2HUF,2HYL,000040B/
1771           DATA ISECU,NAM3,NAM4/2H-7,2H56,2H19,2HDB,2HMS,2H00,2HDB,2HMS,2H01/
1772           DATA NAM13,LLNAM,ICODX /2HUS,2HRC,2HOD,2HCL,2HEA,2HR  ,8/
1773           ICHAR=0
1774           NUM=0
```

```
1775          IFLAG=0
1776          ITEST=0
1777    C -----*************************************************************
1778    C     * Array ITAB1 keeps IDENTIFIERS occuring in  query statement
1779    C     * Number of characters are counted in first row of ITAB1 &
1780    C     * transfered to a buffer IBUF
1781    C -----*************************************************************
1782          DO 2 KP=1,10
1783          IF(ITAB1(1,KP).EQ.1H ) GO TO 30
1784          ICHAR=ICHAR+1
1785    2     IBUF(KP)=ITAB1(1,KP)
1786    30    IRSIZ=30
1787          CALL HASS(IBUF,10,6,NUMB)
1788          IF(NUMB.EQ.0) NUMB=6
1789    C -----*** Conversion from A1 format to A2 format   ***-----
1790          CALL CODE
1791          WRITE(IBUFL,33) (IBUF(LK),LK=1,10)
1792    33    FORMAT(10A1)
1793          IF(JFLAG.EQ.1) GO TO 26
1794    C -----***  Conversion from A1 format to A2 format   ***-----
1795          CALL CODE
1796          WRITE(IBUF2,11) (ISTB1(1,L),L=1,6)
1797    11    FORMAT(6A1)
1798          CALL SMOVE(IBUFL,1,ICHAR,IBUF2,7)
1799    C -----*** file USRCOD is opened to checked the right of   ***-----
1800    C -----*** INSERTION of the user                          ***-----
1801          CALL OPEN(IDCB,IERR,NAM13,1,ISECU)
1802          IF(IERR.NE.2) GO TO 575
1803          CALL SFILL(IBUF1,1,200,BLANK)
1804          CALL READF(IDCB,IERR,IBUF1,100,LEN,JJNUM)
1805          IF(IERR.LT.0) GO TO 575
1806          DO 12 I=1,10
1807          KSTRT=7+(I-1)*18
1808          IF(JSCOM(IBUF2,1,6+ICHAR,IBUF1,KSTRT,IER)) 12,15,12
1809    12    CONTINUE
1810          WRITE(1,13)
1811    13    FORMAT(1H ,"YOU CAN'T INSERT ")
1812          GO TO 585
1813    15    CALL CLOSE(IDCB)
1814    C -----*** File DBMSRD is opened in case user has right to insert   ***--
1815    C -----*** new tuple. Relation name is checked on which user wants  ***--
1816    C -----*** to operate & also status of realation whether suspended  ***--
1817    C -----*** or restore is checked.                                   ***--
1818    26    CALL OPEN(IDCB,IERR,NAM7,1,ISECU)
1819          IF(IERR.NE.2) GO TO 575
1820    34    CALL SFILL(IBUF2,1,64,BLANK)
1821          CALL READF(IDCB,IERR,IBUF2,IRSIZ,LEN,NUMB)
1822          IF(IERR.LT.0) GO TO 575
```

```
1823          IF(IFLAG.EQ.1) GO TO 35
1824          IER=0
1825          DO 40 NVAR=1,4
1826          NSTRT=1+(NVAR-1)*12
1827          NFLAG=6+(NVAR-1)*6
1828          IF(JSCOM(IBUFL,1,ICHAR,IBUF2,NSTRT,IER)) 40,45,40
1829    40    CONTINUE
1830          IFLAG=1
1831          CALL CLOSE(IDCB)
1832    C -----*** File DBMS00 is opened if relation name is not found in  ***----
1833    C -----*** DBMSRD directory.This file keeps information of all     ***-----
1834    C -----*** defined view relation names against existing relation   ***-----
1835    C -----*** name in directory DBMSRD.                               ***-----
1836          CALL OPEN(IDCB,IERR,NAM3,1,ISECU)
1837          IF(IERR.NE.2) GO TO 575
1838          IRSIZ=32
1839          CALL HASS(IBUF,ICHAR,19,NUMB)
1840          IF(NUMB.EQ.0) NUMB=19
1841          GO TO 34
1842    35    DO 36 I=1,5
1843          LSTRT=1+(I-1)*16
1844          NFLAG=7+(I-1)*7
1845          IF(JSCOM(IBUFL,1,ICHAR,IBUF2,LSTRT,IER)) 36,38,36
1846    36    CONTINUE
1847          WRITE(6,37) IBUFL
1848    37    FORMAT(1H ,"RELATION NAME ","***",1X,5A2,1X,"***","NEITHER IN
1849         *VIEW",/,"NOR IN ACTUAL DIRECTORY ")
1850           GO TO 585
1851    C -----*** Status of view relation name whether suspended or restore is *
1852    C -----*** checked in next statements                                    *
1853    38    CALL SMOVE(IBUF2,(2*NFLAG)-1,2*NFLAG,JVAR,1)
1854          CALL CODE
1855          READ(JVAR,16) JVAR1
1856    16    FORMAT(I2)
1857          IF(JVAR1.EQ.1) GO TO 14
1858          WRITE(1,17) IBUFL
1859    17    FORMAT(1H ,"VIEW RELATION NAME SUSPENDED",2X,5A2)
1860          GO TO 585
1861    14    CALL CLOSE(IDCB)
1862    C -----*** File DBMS01 , which keeps information of attributes of each **
1863    C -----*** relation is opened. It also keeps information about          **
1864    C -----*** attribute's type length start location  & original existing **
1865    C -----*** attributes on which it is defined                            **
1866          CALL OPEN(IDCB,IERR,NAM4,1,ISECU)
1867          IF(IERR.NE.2) GO TO 575
1868          IRSIZE=85
1869          GO TO 39
1870    29    CALL HASS(IBUF,ICONT,31,NUMBR)
```

```
1871          IF(NUMBR.EQ.0) NUMBR=31
1872          GO TO 31
1873    C -----*** Status of relation name is checked  ***-----
1874     45   CALL SMOVE(IBUF2,(2*NFLAG)-1,2*NFLAG,JVAR,1)
1875    C-----***  Conversion from A format to I format  ***-----
1876          CALL CODE
1877          READ(JVAR,27) JVAR1
1878     27   FORMAT(I2)
1879          IF(JVAR1.EQ.1) GO TO 47
1880          WRITE(1,28) IBUFL
1881     28   FORMAT(1H ,5A2,2X,"RELATION NAME SUSPENDED")
1882          GO TO 585
1883     47   CALL CLOSE(IDCB)
1884          ITEST=1
1885          CALL OPEN(IDCB,IERR,NAM8,1,ISECU)
1886          IF(IERR.NE.2) GO TO 575
1887          IRSIZE=45
1888     39   IBEG=2*ICHAR+1
1889          LVAR=0
1890          DO 55 IV=2,NANA
1891          KAUNT=0
1892          CALL SFILL(IBUF2,1,20,BLANK)
1893          DO 46 IU=1,10
1894          IF(ITAB1(IV,IU).EQ.1H ) GO TO 44
1895          KAUNT=KAUNT+1
1896     46   IBUF2(IU)=ITAB1(IV,IU)
1897     44   CALL SMOVE(IBUF2,1,2*KAUNT,IBUF,IBEG)
1898          ICONT=KAUNT+ICHAR
1899          IF(ICONT.LT.20) CALL SFILL(IBUF,(2*ICONT)+1,40,BLANK)
1900          IF(ITEST.EQ.0) GO TO 29
1901          CALL HASS(IBUF,20,31,NUMBR)
1902    C -----*** Conversion from A1 format to A2 format  ***-----
1903     31   CALL CODE
1904          WRITE(IBUFL,32) (IBUF(JK),JK=1,20)
1905     32   FORMAT(20A1)
1906    C -----CALL SFILL(IBUFL,ICONT+1,20,BLANK)
1907          CALL SFILL(IBUF2,1,170,BLANK)
1908          CALL READF(IDCB,IERR,IBUF2,IRSIZE,LEN,NUMBR)
1909          IF(IERR.LT.0) GO TO 575
1910          IF(ITEST.EQ.0) GO TO 90
1911          DO 48 IBUC=1,3
1912          JBEG=1+(IBUC-1)*32
1913          JEND=20+(IBUC-1)*32
1914          IF(JSCOM(IBUF2,JBEG,JEND,IBUFL,1,IER)) 48,50,48
1915     48   CONTINUE
1916          WRITE(6,53)
1917     53   FORMAT(1H ,'ATTRIBUTE NOT IN BUCKET')
1918          GO TO 585
```

```
1919   50     ISTRT=23+(IBUC-1)*32
1920          IEND= 28+(IBUC-1)*32
1921          CALL SMOVE(IBUF2,ISTRT,IEND,IBFR,1)
1922   C -----*** Start location & length are noted for attributes   ***-----
1923          CALL CODE
1924          READ(IBFR,83) ISTART,LENTH,LLL,JJJ
1925   83     FORMAT(I4,3I2)
1926   C -----***  Start location & length of all attributes are stored    ***--
1927   C -----***------              in buffer TEMP                  ------***--
1928          LVAR=LVAR+1
1929          TEMP(LVAR)=ISTART
1930          TEMP1(LVAR)=LENTH
1931          GO TO 55
1932   84     CALL CLOSE(IDCB2)
1933   C -----***  According to specific relation name differnt data        ***--
1934   C -----***------            files are opened                  ------***--
1935          IF((NUMB.EQ.1).AND.(NVAR.EQ.1)) GO TO 600
1936          IF((NUMB.EQ.2).AND.(NVAR.EQ.1)) GO TO 700
1937          IF((NUMB.EQ.2).AND.(NVAR.EQ.2)) GO TO 710
1938          IF((NUMB.EQ.4).AND.(NVAR.EQ.1)) GO TO 800
1939          IF((NUMB.EQ.4).AND.(NVAR.EQ.2)) GO TO 810
1940          IF((NUMB.EQ.5).AND.(NVAR.EQ.1)) GO TO 815
1941          IF((NUMB.EQ.5).AND.(NVAR.EQ.2)) GO TO 820
1942          IF((NUMB.EQ.6).AND.(NVAR.EQ.1)) GO TO 830
1943          WRITE(6,85)
1944   85     FORMAT(1H ,"RELATION NAME IS WRONG")
1945          GO TO 585
1946   90     DO 92 IL=1,4
1947          JSTRT=1+(IL-1)*36
1948          JEND=36+(IL-1)*36
1949          IF(JSCOM(IBUFL,1,ICONT,IBUF2,JSTRT,IER)) 92,94,92
1950   92     CONTINUE
1951          WRITE(6,93) IBUFL
1952   93     FORMAT(1H ,"Attribute concatenated with VIEW relation "
1953         *,4X,20A2,/,"    does not exists exists in directory ")
1954          GO TO 585
1955   94     KSTRT=27+(IL-1)*36
1956          KEND=36+(IL-1)*36
1957          CALL SFILL(IBFR,1,10,BLANK)
1958          CALL SMOVE(IBUF2,KSTRT,KEND,IBFR,1)
1959          CALL CODE
1960          READ(IBFR,96)ISTART,LENTH,IFILE,NVAR
1961   96     FORMAT(I4,I2,I2,I2)
1962          LVAR=LVAR+1
1963          TEMP(LVAR)=ISTART
1964          TEMP1(LVAR)=LENTH
1965          IF(LVAR.EQ.1) JFILE = IFILE
1966    55    CONTINUE
```

```
1967            IF(IFLAG.EQ.1) NUMB = JFILE
1968            GO TO 84
1969   600      CALL OPEN(IDCB2,IERR,NAM9,2)
1970            IF(IERR.NE.2) GO TO 575
1971            GO TO 500
1972   700      CALL OPEN(IDCB2,IERR,NAM1,2)
1973            IF(IERR.NE.2) GO TO 575
1974            GO TO 500
1975   710      CALL OPEN(IDCB2,IERR,NAM5,2,ISECU)
1976            IF(IERR.NE.2) GO TO 575
1977            GO TO 500
1978   800      CALL OPEN(IDCB2,IERR,NAM2,2)
1979            IF(IERR.NE.2) GO TO 575
1980            GO TO 500
1981   810      CALL OPEN(IDCB2,IERR,NAM6,2,ISECU)
1982            IF(IERR.NE.2) GO TO 575
1983            GO TO 500
1984   815      CALL OPEN(IDCB2,IERR,NAM10,2,ISECU)
1985            IF(IERR.NE.2) GO TO 575
1986            GO TO 500
1987   820      CALL OPEN(IDCB2,IERR,NAM11,2,ISECU)
1988            IF(IERR.NE.2) GO TO 575
1989            GO TO 500
1990   830      CALL OPEN(IDCB2,IERR,NAM12,2,ISECU)
1991            IF(IERR.NE.2) GO TO 575
1992   500      CALL SFILL(IB,1,40,BLANK)
1993   C -----***  Record sizeof different data files are assigned here  ***-----
1994   C -----***-----          according to relation name          -----***-----
1995            IF(NUMB.EQ.1) IRSIZ=62
1996            IF(NUMB.EQ.2) IRSIZ=65
1997            IF(NUMB.EQ.4) IRSIZ=48
1998            CALL SFILL(IBUF1,1,2*IRSIZ,BLANK)
1999            DO 450 IZ=1,10
2000            CALL SFILL(IBUF2,1,2*IRSIZ,BLANK)
2001            CALL READF(IDCB2,IERR,IBUF2,IRSIZ,LEN,IZ)
2002            IF(IERR.LT.0) GO TO 575
2003            IF(JSCOM(IBUF2,1,2*IRSIZ,IBUF1,1,IER)) 450,550,450
2004   450      CONTINUE
2005            WRITE(1,451)
2006   451      FORMAT(1H ,"NO PLACE ON FILE ")
2007            STOP
2008   550      NUMBR=IZ
2009   C        DO 570 IBB=1,4
2010            DO 570 IBB=1,IVAR5
2011            DO 560 IBC=1,40
2012   560      IBUF(IBC)=ITAB(IBB,IBC)
2013            CALL CODE
2014            WRITE(IB,562) (IBUF(LJ),LJ=1,40)
```

```
2015  562    FORMAT(40A1)
2016         CALL SMOVE(IB,1,TEMP1(IBB),IBUF1,TEMP(IBB))
2017  570    CONTINUE
2018         WRITE(6,571) IBUF1
2019  571    FORMAT(1H ,/,"   Tuple to be inserted is :- ",/,4X,128A2)
2020         CALL WRITF(IDCB2,IERR,IBUF1,0,NUMBR)
2021         IF(IERR.LT.0) GO TO 575
2022         CALL CLOSE(IDCB2)
2023         WRITE(6,572)
2024  572    FORMAT(1H ,"QUERY IS PROCESSED ")
2025         GO TO 585
2026  C ------***  File manager errors are given to user if he tries to    ***--
2027  C ------***------      access the illeagle file or its status      ------***------
2028  575     WRITE(1,580) IERR
2029  580    FORMAT(1H ," *** FMGR ERROR ******",2X,I5)
2030  585    CALL EXEC(ICODX,LLNAM)
2031         END
2032         PROGRAM CLEAR , 5 , , , , , , , , , THIS CLEARS BUFF&CALLS LEXCL
2033         COMMON ITAB,ITAB1,ISTB,MM,M,JM,IPRS,IQRY,IF,ISTB1,KM,ICO,JLV,JFL
2034        *,MFLG
2035         INTEGER ISTB(50,2),ITAB1(30,22),ITAB(30,42),IPRS(50,3),ICO(20),
2036        *ISTB1(20,8),KNAM(3)
2037         DATA ICOE,KNAM /8,2HLE,2HXC,2HL /
2038         DO 20 I=1,50
2039         DO 5 J=1,2
2040  5      ISTB(I,J)=1H
2041         DO 10 K=1,3
2042  10     IPRS(I,K)=1H
2043  20     CONTINUE
2044         DO 40 I1=1,30
2045         DO 25 J1=1,22
2046  25     ITAB1(I1,J1)=1H
2047         DO 35 K1=1,42
2048  35     ITAB(I1,K1)=1H
2049  40     CONTINUE
2050         DO 50 I2=1,20
2051         DO 50 J2=1,8
2052  50     ISTB1(I2,J2)=1H
2053         DO 55 I3=1,20
2054  55     ICO(I3)=1H
2055         MFLG=1
2056  C------------   SEGMENT LEXCL IS CALLED . NOW THE USER CAN GIVE ANOTHER
2057  C              QUERY IF DESIRED ----------------------------------------
2058         CALL EXEC(ICOE,KNAM)
2059         END
```