/501

DESIGN OF A RELATIONAL DATA BASE AND A QUERY LANGUAGE

AND IMPLEMENTATION OF ITS DDL

A DISSERTATION SUBMITTED IN
PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF PHILOSOPHY
IN
COMPUTER AND SYSTEMS SCIENCES

BY
RAM NARESH SINGH

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

To

My Parents

DECLARATION

The work embodied in this dissertation contains the results of the research work carried out under the supervision of SR. G.V. Singh, Assistant professor School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi. The work is original and has not been submitted, in part or full, to any other university for the award of any other degree or diploma.

RNSingh
RAM NARESH SINGH
Student

G.V.Singh
Supervisor

Dean,
School Of Computer And
Systems Sciences,
Jawaharlal Nehru University,
New Delhi:110 067

ACKNOWLEDGEMENTS

I wish to express my deep sense of gratitude to my supervisor Shri G.V. Singh, for his guidence and the encouragement provided in carrying out this work.

I am indebted to Dr. N.P.Mukherjee, Dean of the school, Dr. D.K.Banerjee, Dr.R.Sadananda,Dr. A.K.Pujari and other teaching staff for their constant encouragement.

I acknowledge the invaluable help provided by Ms. Anjana Sahai during my project work.

I express my sincere thanks to Mr. Sunil K. Dhanawat for making available his word processor software for preparing the dissertation. I am also thankful to Mr. Hawa Singh, Mr. S.K. Sinha, Mr. Arun Prasad, Ms. Pampa Das and Mr. V.K.Singhal for their encourgement and help.

I am thankfull to Mr. Ajoy Roy and other technical staff for the help provided in using the computer and my special thanks are due to Mrs. Indu Singh.

I am also thankfull to Mr.C.A.Thakur and other school staff for their constant encouragement.

Finally, I express my gratitude to the JNU for the financial assistance.

RNSingh

CONTENTS

Page

CHAPTER ONE : Overview of database system

CHAPTER 2 : PROPOSED LIBRARY DATABASE MODEL

CHAPTER ONE

OVERVIEW OF A DATABASE SYSTEM

1.1 INTRODUCTION

The management of the present day organization depends heavily on computerized information systems. The use of these information systems have led to quantitative as well as qualitative developments in almost all walks of human activities. This development continuosly demands better & more sophisticated computerized management techniques. In this way computerized information systems have passed through several evolution phases & today they are familiar as data base management systems.

1.2 EVOLUTION OF DBMS : -

In the very begining data were handled with the help of sequential files & data processing systems were known as file processing systems. The files used in file processing systems had various anomalies like data redundancy & lack of data independence, security & integrity. In this way users were paying more cost to store these redundant data. The updating operations of the files containing redundant data could lead to the lack of data integrity, which could produce incorrect & conflicting results.

In file processing approach each system was designed & implemented almost independently. Around 1960's the concept of management information systems(MIS) came into existence. The MIS modules made great efforts to eliminate the deficiency of file processing systems. MIS had existence potenial & capability for day to day management activities & for the future planing & projections in an organization, but most of the MIS had certain limitatoins. The task of planning, designing & implementing an information system to be used by the management was often under estimated & was left to the unskilled persons in the area of management. It was not possible to establish priorities among the users. Due to above mentioned deficiency, management information systems could'nt remain in the field for too long. People cogitated & adopted a new approach & the concept of integrated data base management system started taking concrete shape.

Although the concept of database had been used since the end of 1960's to refer to almost integrated files, but due to lack of faster direct access devices this concept could not take birth. The concept of database took the concrete shape, as soon as faster direct access devices were invented. However, today every aspect of database technology is the topic of heated discussion.

GUIDE/SHARE defines the database : a named collection of data suffers from certain circularity. According to this definition - a database consists of all the record occurences, set occurences & areas, which are controlled by a specific schema. A schema is a complete description of the database. Another more precise definition of database is :

i) an organized, integrated collection of data

ii) a natural representation of the data with no imposed restriction or modification to suit a programmer and (iii) capable of use by all relevant applications without duplication of data.

Given the difficulty of defining a database management system & hence of deciding when a software product qualifies as such a system, one might expect the classifications of database software to be even more difficult task. Software is regarded as simply a piece of code providing the user with a number of specified facilities to ease the problems. Database management system can then be defined in terms of the facilities it can be expected to provide. These facilities are as follows:

a) DATA INDEPENDENCE :

Different applications will need different views of the same data. In an integrated data base environemt, the database management system provides each application program with its own view of the common data & implements various access operations of the data. In modern database management, the users are concerned with the information contents of the data & decreasingly concerned with its representation details. The representation of the given facts may change over time, without giving any information to user of this change. The general term for this trend, away from representation detail is referred as data independence

b) MINIMAL DATA REDUNDANCY:

In most current systems each application has its own private files. This can often lead to considerable redundancy in stored data, with resultant waste in storage space. One objective of the DBMS is to minimize data duplication. A database has some times been defined as a non-redundant collection of data items, but in reality some measure of redundancy exists in many data bases in order to give improved access times or simpler addresing methods. There is a trade off between non-redundancy & other desirable criteria. The data redundancy thus introduced is referred as controlled redundancy or minimal redundancy(7).

c) DATA REATABILITY :

The logical design of the database may model the real world relationships of the data. It should be sufficiently flexible to cater for at least a majority of application requirements. To fulfill these requirements sometimes complex file structures are necessary in the database. These can be provided through hardware pointers, inverted indexes or logical identifiers.

d) MINIMUM COST:

Cost is one of major factors, especially in terms of maintenance, other costs include the rental charges of the software and hardware. Some organizations cannot afford the raw material needed to support a database management system. Though it is not true for all database management systems, yet it is something that must be taken under consideration while evaluating a DBMS.

e) SECURITY :

DBMS provides facilities that only authorised users can access the database. As soon as a user wishes to access or update the database, DBMS checks the authorisation of that user. Invalid user will be given some message ("You are not a valid user") and process will be terminated.

Unathorized user say a student can destroy all the information about a book which was issued against his name. Thus in absence of security the library will be in loss.

f) INTEGRITY :

The integrity of a database system is a key factor in determining its success. No matter how advanced its other features, unless the output it produces is timely, consistent and correct, the system is of little practical use. In this way maintaining the integrity of the database is of prime concern to the systems designer.

Maintaining integrity of the database is to ensure that data in the database is accurate at all times. Basically it is protected against illegal alteration or destruction. For example in library database not more than one book should be issued on single library ticket. Also not more than one book should be of the same accession number & return date of a book should be after a fortnight of the date of issue.

The number of interactions that may take place & which effect the database integrity are as follows (4) :-

i) CONTENTION :

In multiple user system more than one user may attempt to updte the same data item & thus inconsistencies may arise in the data held in database. Thus update command issued by one user will be nullified by another user for a particular record, because the update action by second user will result in overwrite & thus first user's update will be lost.

ii) CONTENT CONSISTENCY :

It gives a 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. Thus we get a idea of integrity unit (sucess unit). An integrity unit is a sequence of computer processing at the begining & end of which database is in a consistent state, but during which it may be in an inconsistent state.

iii) TIME CONSISTENCY :

This is the ability to provide a snap shot of database which is being continually updated, so that information correct at a given instant of time can be produced from it.

iv) RESILIENCE :

This is the ability of the database to recover all informations without loss of stored data in the event of any failure of the system or database. Thus to provide resilience, back up & recovery facilities are required. The loss of integrity may cause by any of the following reasons:

1) hardware failure at any point in the system like at central processor, on a data channel, or at an input/output device.

2) Human error commited like computer operator or a terminal user.

3) Software error within the DBMS or the underlying operating systems.

4) Programming error in one of the database applications.

The situation of lost of integrity may occur even in the best regulated system, since one or another of these errors are bound to occur from time to time. Care is taken to detect & remove the error.

g) SYNCHRONIZATION :

In a database we may assume that programs accessing the database are run one at a time. However there are also numerous applications in which more than one program, or different executions of the same program, run concurently. As already stated the DBMS should provide protection against inconsistencies that result from two approximately simultaneous operations on a data item. An example is a library ticket issue system where several library sraff members are preparing library tickets for students. For simplicity if we take two library staff members, both will issue request to give ticket number to students. Each request results in execution of a progam which will examine the number currently assigned to students & will increment the counter to issue the library ticket number to next person. If the DBMS doesnot take care of concurrency then same ticket number may be given to more than one student. Thus two processes that read & change the value of the same data item should not be allowed to run concurently. However where only read requests are issued by many users, the concurrency is allowed so that the overall execution time is minimized.

1.3 COMPONENTS OF DATABASE MANAGEMENT SYSTEMS :

Any design of a database is generally implemented in two stages. In one stage we design physical database (PDB), logical database (LDB), & the applications programs (AP). At the second stage we design the interfaces to connect the PDB, LDB & AP'S. Fig 1.1 shows the organization of the database so designed.

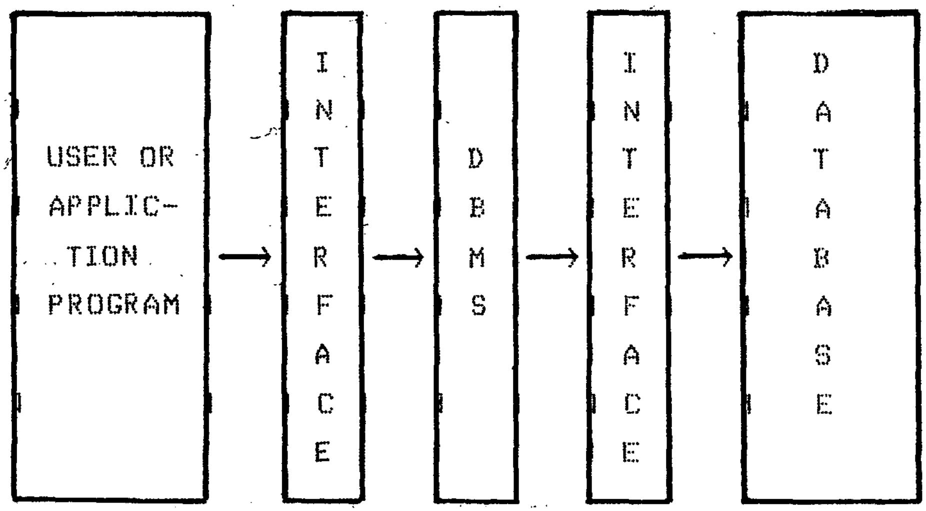


Figure 1.1

At the time of designing DBMS, the PDB,LDB & Application programs are logically separated from each other. In this way change in one component will not affect the other components. But change in one component will require the change in corresponding interface. Each of these components of database are briefly described below:

a). PHYSICAL DATABASE (PDB) OR INTERNAL SCHEMA :

It is the view of the database as seen by the system programmers & the system designers, who are concerned with organization & maintenance of data in the DBMS. The description of a PDB includes the description of :

1). Record types, file types used to store the data;

2) Addressing & searching techniques & pointers to pass & access the data.

3). And also the restore & recovery procedures incorporated in the system.

The selection of physical organization is determined by the need for operational efficiency, response time & cost optimization. The record formats & file types are largely fixed by the physical characteristics of the secondary storage devices. The techniques of high packing density consume more time for file addressing & searching. The techniques providing high flexibility for searching & facilitating easy real time insertion of new records, tend to use more space. The good data independence can be achieved only at the cost of machine performance. However resort & recovery procedures may have special physical storage requirements & need some amount of data redundancy.

b). LOGICAL DATABASE OR SCHEMA :-

As we know the most elemental piece of data is data item. A data item by itself is not of much use. It becomes useful only when it is associated with other data items. The association between different data item types is shown by a data model. A data model is the model of the real world. It does not resemble the real world too closely, but which is designed to be as useful a representation as possible to its users. These models are logical representation of data & are completly independent from their physical organization. The logical description of the data base is called the logical database (LDB).

Figure 1.2 describes a logical database record type in the schema. It has been drawn as blocks & the relationship between record types have been shown by lines drawn between record types. The association shown by solid lines shows the type of association where the complete information can only be obtained by properly connecting the relevent attributes from the record types involved in the relationship. Information about parts suplied by a supplier can be obtained by properly merging from A1 & A2 records. The relationships shown by dashed lines represent the cross refrence type of relationship.

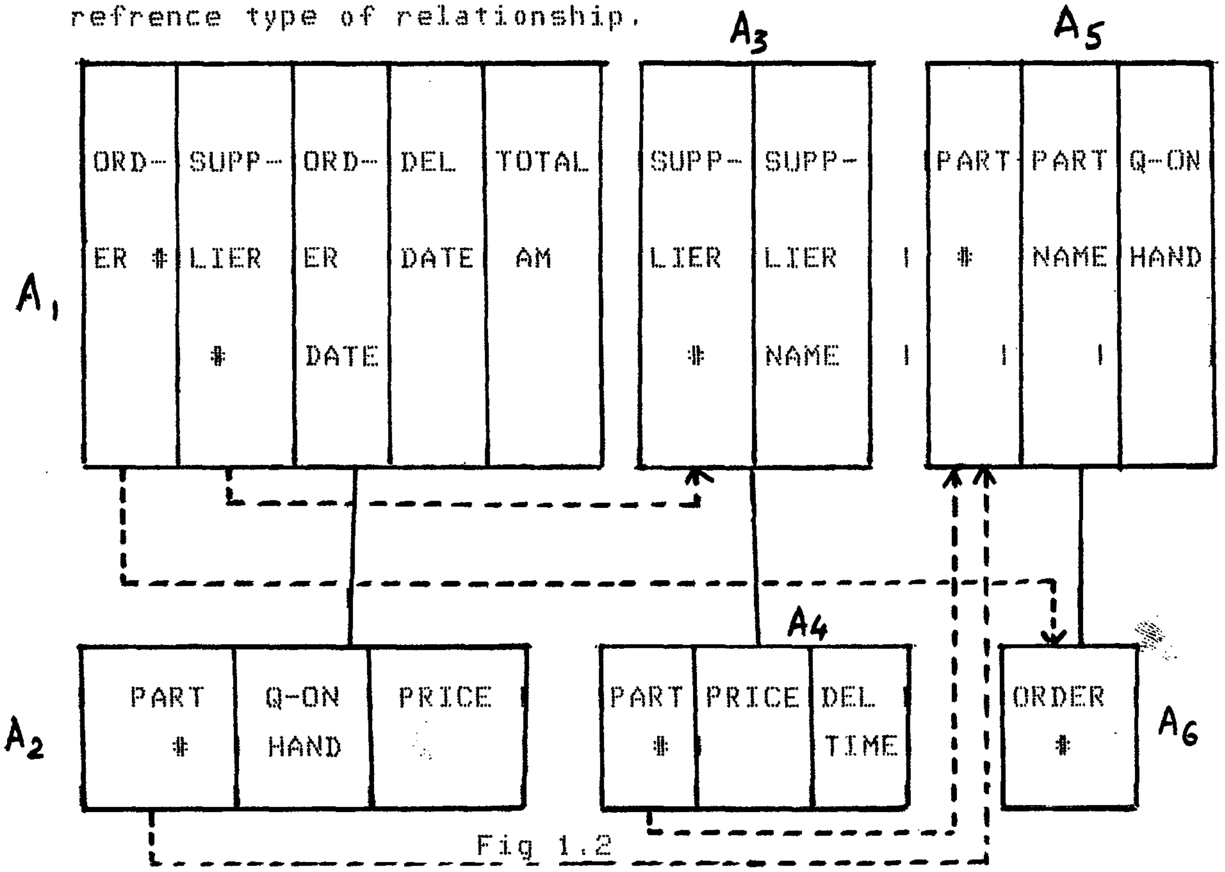


Fig 1.2

c). APPLICATION PROGRAMMER'S VIEW OR SUB SCHEMA:-

It represents the way in which data is viewed by individual user. This view is materialised from the data in the physical database under control of logical database. We may define a subschema over schema as follows :

For a given schema there may be derived several subschemas, which may overlap; or a subschema may consists of whole of the database; or a subschema may be restricted to even one of record type.

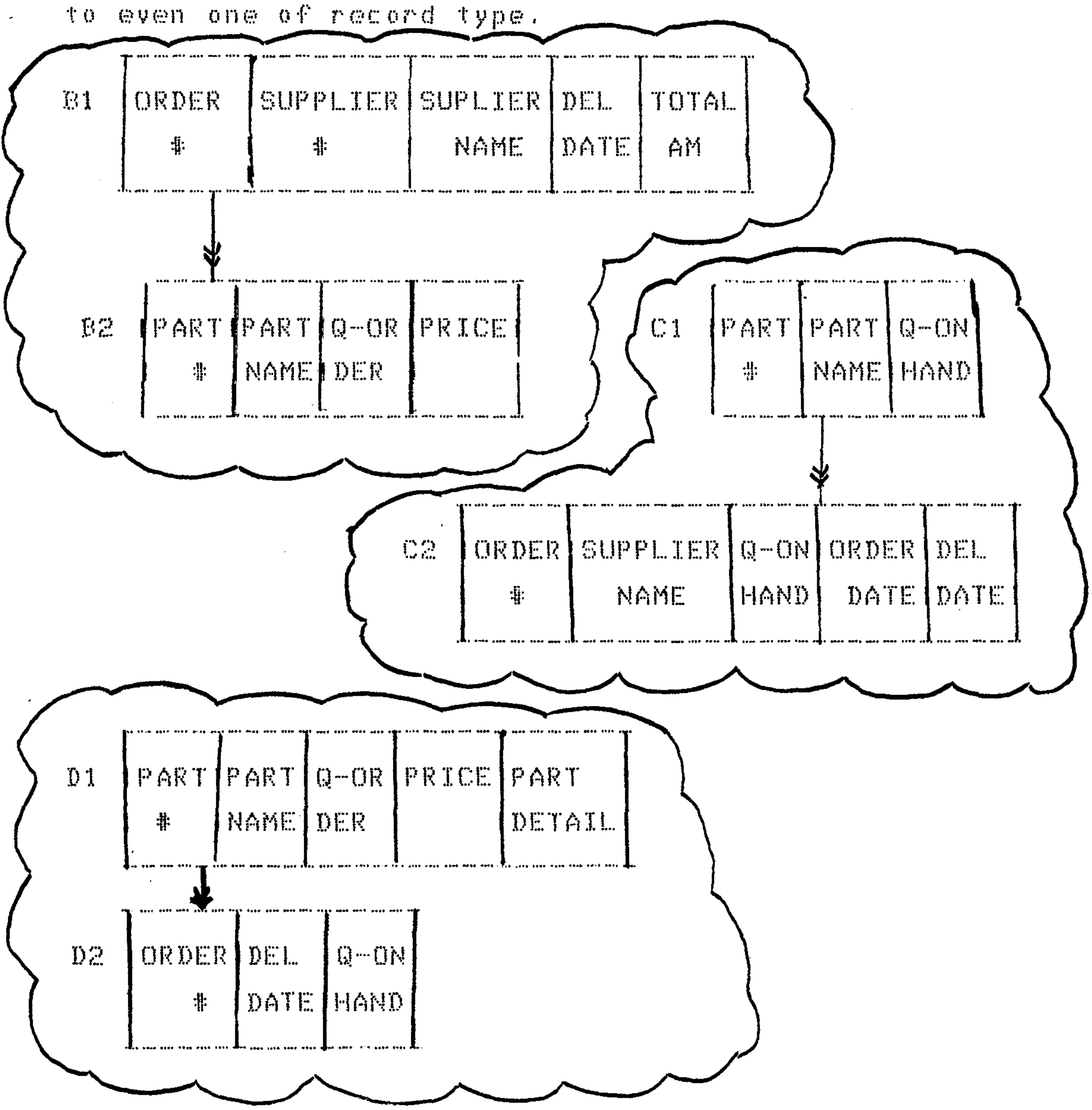


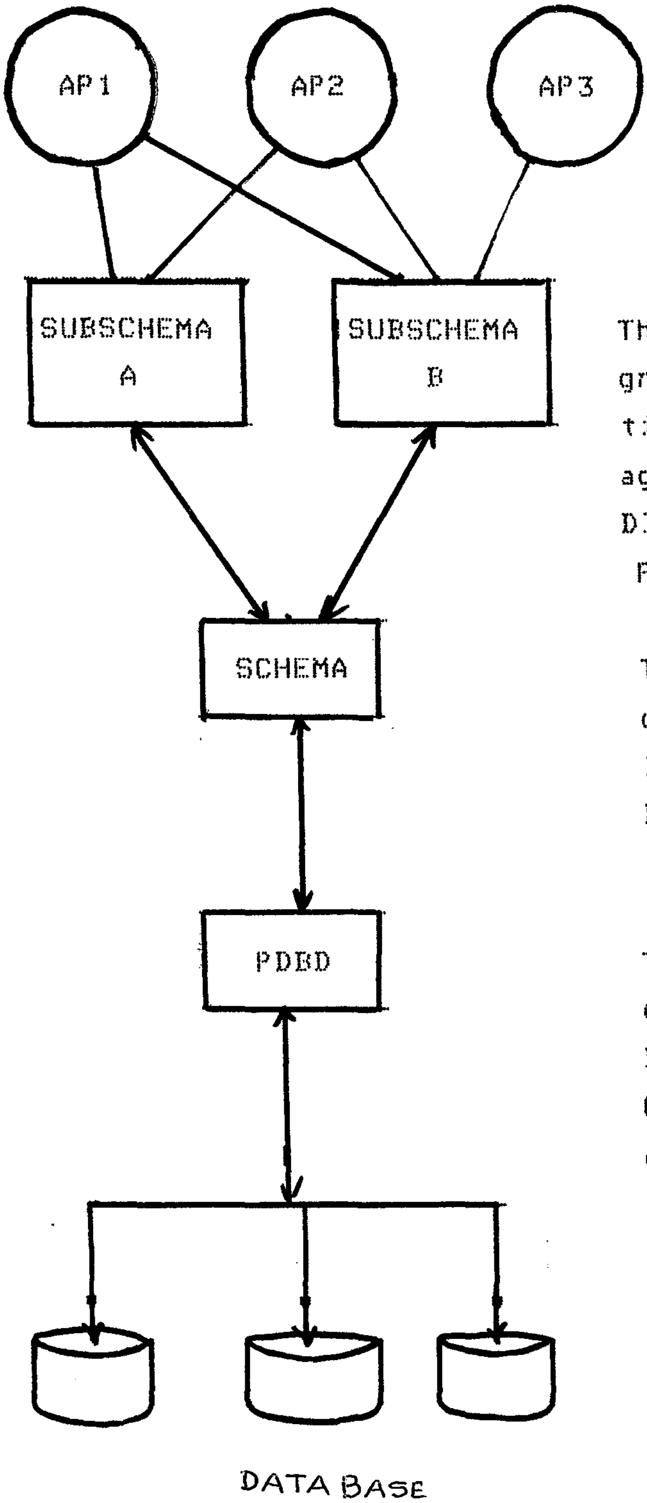
Figure 1.3

In the fig. 1.3 the subschema for three application programmers are shown. Here all three programmers have their own views which are completely different from each other but these views are derived from the same schema of fig. 1.2 The DBA must ensures that the subschemas, the programmer use are derivable from the schema. The data management software assembles the data described in the subschema from the data described in the schema automatically & gives it to the application programmer. The various study groups on database systems have referred three levels of data description by different names as we have shown in the table below:

	ANSI SPARC	CODASYL	IBM'S	OTHER TERM
PROGRAMMER'S VIEW	EXTERNAL SCHEMA	SUBSCHEMA	PSD(PRGORAM SPECIFICATION BLOCK)	SUB MODEL (LOGICAL VIEW)
OVERALL LOGICAL VIEW	CONCEPTUAL SCHEMA	SCHEMA	LOGICAL DBD	MODEL, CONCEPTUAL MODEL,ENTITY SET
PHYSICAL VIEW	INTERNAL SCHEMA	PHYSICAL DATA	PHYSICAL DBD	ENTITY RECORDS

d) PDB/LDB and LDB/Application programmes Interface:

The interface between LDB and PDB defines the mapping between schema and internal schema and the interface between LDB and AP defines the mapping between various subschemas & schema i.e. global logical view of the data. The mapping between schema & internal schema depends on the internal schema & if the internal structure of the records changes, the mapping between schema & internal schema also changes. The mapping between subschema & schema defines the correspondence between them as shown in fig. 1.4 on next page.



Application program (Programming language e.g.COBOL,DL/I)

The application programmer's data declaration (Subschema language e.g. COBOL DATA DIVISION OR IBM'S DL/I PSB)

The data base logical description (schema language e.g. CODASYL DDL, IBM's DL/I)

The physical data base description (e.g. IBM's DL/I, Physical DBD or CODASYL devices/media control language)

Figure 1.4

1.4 MAIN EVENTS IN PROCESSING A TYPICAL DBMS:

The schema & subschema are both used by the database management system, the primary function of the DBMS is to serve the application program by executing their data operations. The main events that occur when an application program reads a record by means of a DBMS as shown in fig. 1.5. The main events are as follows:

1). Application program A issues a call to DBMS to read a record.

2). The DBMS obtaines the subschema that is used by application program & looks up the description of the data.

3). The DBMS obtains the schema & determines which logical data type or types are needed.

4). The DBMS examines the physical database description & determines which physical record or records to be read.

5). The DBMS issues a command to the computer operating system, instructing it to read the requisite record (s).

6). The operating system interacts with the physical storage where the data are kept.

7). The required data are transferred between the storage & the system buffers.

8). Comparing the subschema & schema, the database management system derives from the data the logical record needed by the application program. Any data transformations between the data as declared in the subschema and the data as declared in the schema are made by the DBMS

9). The DBMS transfers the data from the system buffer to the work area of application program A.

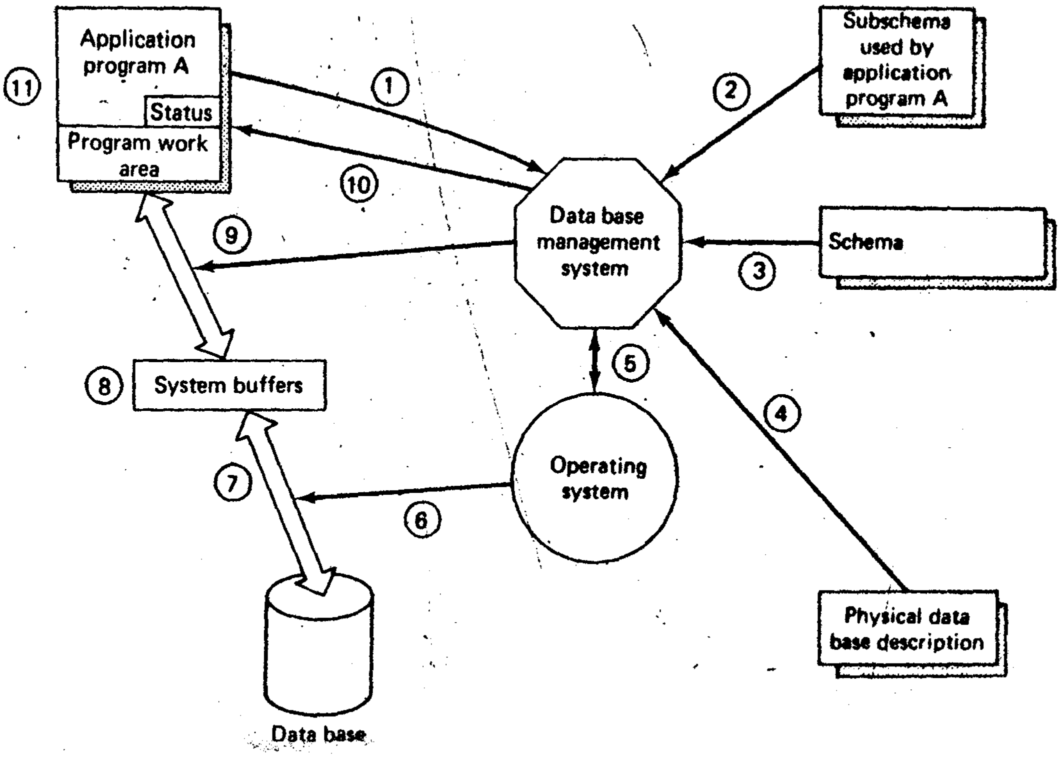

Application program A
Status
Program work area
11
1
10
9
Subschema used by application program A
2
Data base management system
Schema
3
5
4
8
System buffers
7
Operating system
6
Physical data base description
Data base

FIG. 1·5

10). The DBMS provides status information to the application program on the outcome of its call, including any error indications.

11). The application program then can operate with the data in its works area.

If the application program updates a record, the sequence of events is similar. It will normally read it first, modify it in program work area & then issue an instruction to the database management system to write back the modified data.

1.5 DATABASE MODELS :

A data model is a pattern to organise data logically. The data model used by DBMS can be distinguised mainly as to how they represent relationship among data. There are three main data models : The Hierarchical model, the network model & the relational model. These three models are discussed briefly in following paragraphs. Before describing Database models let us give some definitions.

a). ENTITIES :

An entity is a thing[8] that exists & is distinguishable i.e. we can tell one entity from another. For example each chair is an entity, so is each person & each automobile. We can't say an ant as an entity because we can't distinguish from each other.

b). ENTITY SET :

A group of all similar entities forms an entity set. For example, all persons, all automobiles, all emotions etc.

c). ATTRIBUTIES :

Entities have properties, called attributes, which associate a value from a domain of values for that attribute with each entity in an entity set. Usually, the domain for an attribute will be a set of integers, real numbers or character strings etc.

d). Tree :

A tree is a special form of directed graph with following properties:

(i). Either it has no vertices or has a distinguished vertex called the root or 'root vertex', which has no predeccessors and

(ii). Every vertex other than root vertex has a unique predecessor.

Vertices of a tree which have successors are called 'nonterminal vertices' while vertices that have no successors are called 'terminal vertices ' or leaves. Root of the tree is at highest level & leaves are at the lowest level in the hierarchy.

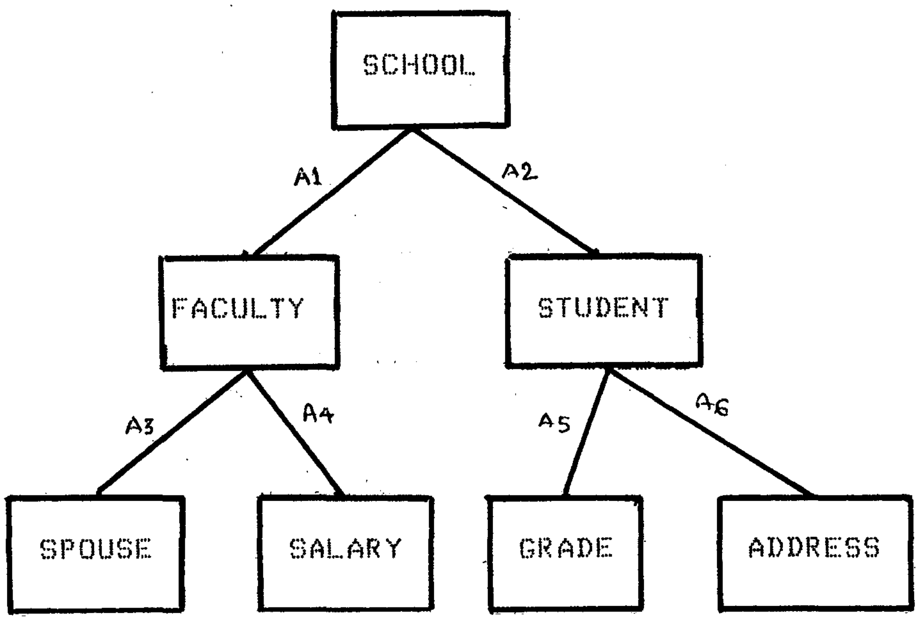


Figure 1.6

In a tree of fig 1.6 there are three non-terminal nodes i.e. SCHOOL, FACULTY, STUDENT. SPOUSE, SALARY, GRADE & ADDRESS are four terminal nodes which are at lowest level. Root node SCHOOL is at highest level in hierarchy. Root may have any number of lower level dependents & so on to any number of levels.

1.5.1 HIERARCHY MODEL:

In hierarchical model the data is represented by simple tree structure. Trees are natural data structure for any data objects whose components stand in a hierarchical relation to each other. The relationship between entities can be represented by graph in which each node indicates a record type, that represents in entity type & each link specifies a relationship between the entities.

For example we consider the tree (Hierarchical data model) of fig. 1.6. In this model there are seven record types i.e. SCHOOL, FACULTY, STUDENT, SPOUSE, SALARY, GRADE & ADDRESS. All record types are connected by link records i.e. record type FACULTY is connected with root node SCHOOL by link A1. Other links like A2, A3, A4, A5, A6 have been shown to connect the record type of our taken hierarchical model.

It is fundamental to the hierarchical view of data that any given record occurence has its own full significance only when seen in some context. In fact no dependent record occurence can ever exist without its superior. For example record type FACULTY has no significance without its superior record SCHOOL because there is no existence of FACULTY record type unless & until there exists a SCHOOL record type as its superior.

An information in tree structure can be found by the help of links from higher level to lower level starting at root & going towards leaves of the tree. Say to get a record at lowest level of tree we will have to traverse so many higher level nodes starting from root. For example to get record SALARY of a faculty member we will have to traverse two higher level node i.e. SCHOOL & FACULTY.

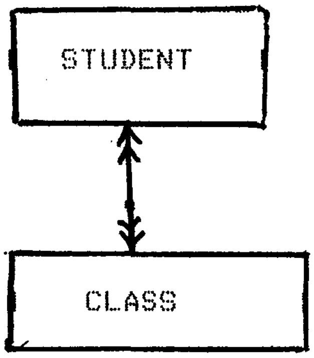
More record types may be introduced into the structure & the hierarchy becomes more complex & program becomes more complicated. For example to introduce CENTRE in SCHOOL record, we will have to change other links. Keeping previous information as it is, we have to make new hierarchical model meaningfull with some new inserted information. In this situation maintaining the appropriate link becomes very difficult.

Deletion of any record occurence automatically delets all dependent occurences too, so one has to pay great attention at the time of deleting any record.

1.5.2 NETWORK DATA MODEL :

A network model is composed of nodes & branches like a tree. It differs from a tree in that children can have multiple parents. That is both the relationship from child to parent & the relationship from parent to child can be one to many in a network.

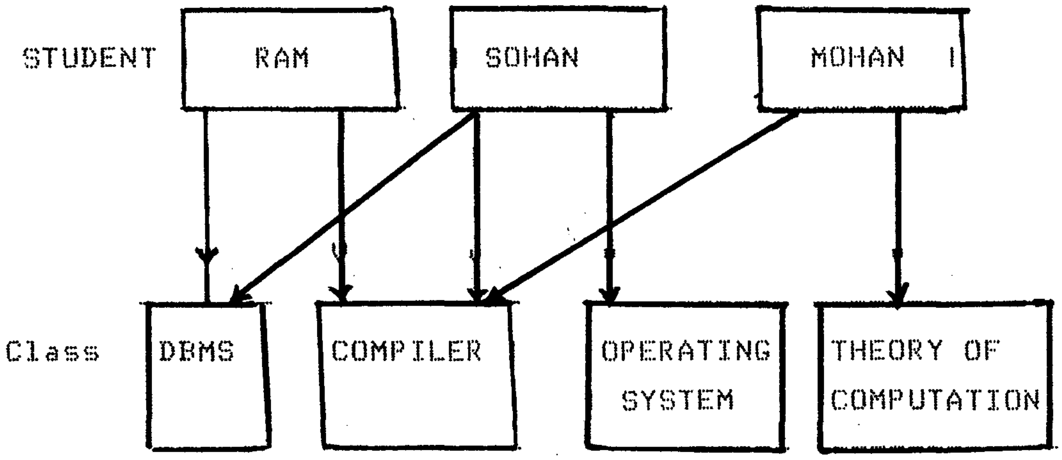
Fig. 1.7 shows an example of a network.



Student/Class network

Figure 1.7

Students may get registration for several classes & each class may contain many students. The relationship student to class is therefore one to many & the relationship class to student is also one to many.



Network occurence of student/class

Figure 1.8

In fig. 1.8 (shows a occurence of fig. 1.7) student sohan is registered for DBMS, compiler & operating system. Class compiler contains student RAM, MOHAN & SOHAN. From above example we see that the network database model is a more general structure than a hierarchical data model, because a given record type can have any number of immediate superiors as well as immediate dependents.

We draw a directed graph called a network, which is really a simplified entity relationship diagram to represent record types & their links. Nodes corresponds to record types. If there is link between two record types STUDENT & CLASS and the link is many to one from STUDENT to CLASS, then we can draw an arc from the node for STUDENT to CLASS, and we can say the link is from STUDENT to CLASS. Nodes & arcs are labled by the names of their record types and links.

Another type of record used to represent the network data model is known as connector. A connector occurence represents the association between parents node and their children nodes. For example all connector occurence for a given student may be placed on a chain starting at and returning to that student. Similarly all connector occurences for a given class may be placed on a chain starting at and returning to that class. Thus each connector occurence is on exactly two chains, one STUDENT chain one CLASS chain. It is to be noted that the searching techniques become ambiguous due to two chains of connectors. And it becomes difficult for user to decide which occurence to adopt?

1.5.3 RELATIONAL DATA MODEL :

Usually, relations when discussed are represented in the form of tables in which each row represents a tuple. In the tabular representation of the relation, the following properties must be observed (Nixom & Mc Govern, 1972).

a). No two rows are identical.

b). The ordering of the rows is not significant &

c). The ordering of the column is significant.

Any table that meets the above requirements is called a relation & a collection of such type of tables comprises a relational database.

When a relaton is represented as table, its degree is the number of columns & its cardinality is the number of rows. Each row of such table is termed as tuple or record & each column of the table are known as attribute. We have shown an EMPLOYEE relation in tabular from in fig. 1.9.

EMPLOYEE #	EMPLOYEE NAME	PROJECT #	SALARY	COMPLETION DATE
JNU01	Philips J.	JNUPHD01	2000.00	14.01.1984
JNU01	Philips J.	JNUPHD03	2000.00	12.08.1985
JNU01	Philips J.	JNUPHD09	2000.00	18.09.1985
JNU09	J.P.Singh	NIC01	1500.00	18.09.1986
JNU09	J.P.Singh	NIC09	1500.00	14.01.1984
JNU11	S.Mittal	CMC01	1600.00	10.11.1984
JNU11	S.Mittal	CMC09	1600.00	14.11.1984

Figure 1.9

Here in this table degree & cardinality are 5 & 7 respectively.

Another way to represent this relation in format shown below, is generally used for relational database model.

EMPLOYEE (EMPLOYEE #, EMPLOYEE NAME, PROJECT #, SALARY, COMPLETION DATE)

Next we give some definitions which are to be used in the successive description .

Candidate Key:-

The term candidate key is used to denote any minimal set of attributes that uniquely identifies a tuple in the relation.

Primary key : When a relation has more than one candidate key, then one of them is called primary key.

Foreign key : When an attribute of a set attribute in one relation corresponds to a key of another relation, this attribute will be called foreign key. A foreign key shouldn't be : (i) Key of its own relation and (ii) need not have the same attribute name as the corresponding key in the another relation.

Prime attribute : An attribute A in relation scheme R is a prime attribute if A is a member of atleast one candidate key for R.

Non prime attribute : If A is not a member of candidate key, then A is called non prime attribute.

Functional dependency : Non prime attribute B of a relation R is functionally dependent on prime attribute A of R, if at any instant of time, each value in A has no more than one value for B associated with it in relation R ( ).

Saying that B is functionally dependent on A is equivalent to saying that A identifies B & may be represented by

A -----> B

In other words, if at one instant of time the value of A is known, then the value of B can be determined.

Relation of fig 1.9 can be represented in following manner keeping functional dependencies in mind.

EMPLOYEE NAME ---> EMPLOYEE #

EMPLOYEE # ---> EMPLOYEE NAME

Either EMPLOYEE # or EMPLOYEE NAME ---->PROJECT #

EMPLOYEE # or EMPLOYEE NAME or PROJECT # ----> COMPLETION DATE

Either EMPLOYEE # or EMPLOYEE NAME----> SALARY

An attribute can be functionally dependent on a group of attribute rather a single attribute. Following relation shows an example .

FLIGHT (FLIGHT #, DATE, TIME ,PILOT, NO OF PASSENGERS)

Thus to know the name of pilot on flight number, concatenated key(FLIGHT #, DATE, TIME) only can give the whole information. Because there is daily flight of boeing 747 (may be more than one flight)

An attribute or a collection of attributes, B of a relation R can be said to be fully functionally dependent on another collection of attributes, A of relation R if B is functionally dependent on the whole of A but not on any subset of A. From previous relation of flight we see attribute NO. OF PASSENGERS & PILOT both are fully functionally dependent on concatenated key (FLIGHT NO, DATE, TIME) & not part of it. Only FLIGHT NO can't give about either number of passengers & pilot because there is daily flight of a particular airplane (May be more than one flight also) & pilot may not be same. So concatenated key only can uniquely identify the other attributes of relation.

1.6 NORMALISATION :-

When the real world with entities & their properties is expressed in the form of relations i.e as two dimensional table of n columns & a varying number of rows; these relations are in unnormalised form containing undesirable associations. But we need to normalize the relations before we can efficiently & effectively implement them. A normalised relation should possesses the following properties:-

i). Column homogeneity-In any particular column all items should be of the same type.

ii). All rows of the table should be distinct

iii). The ordering of rows is immaterial &

iv). If the distinct names are given to the columns the ordering of columns is immaterial.

Cod has defined three stages of normalisation known as the first normal form (1NF), second normal form (2NF), & the third normal form (3NF) corresponding to the three types of undesirable associations i.e. repeating groups, partial dependence & indirect dependence. The three stages of normalisation is shown in fig. 1.10

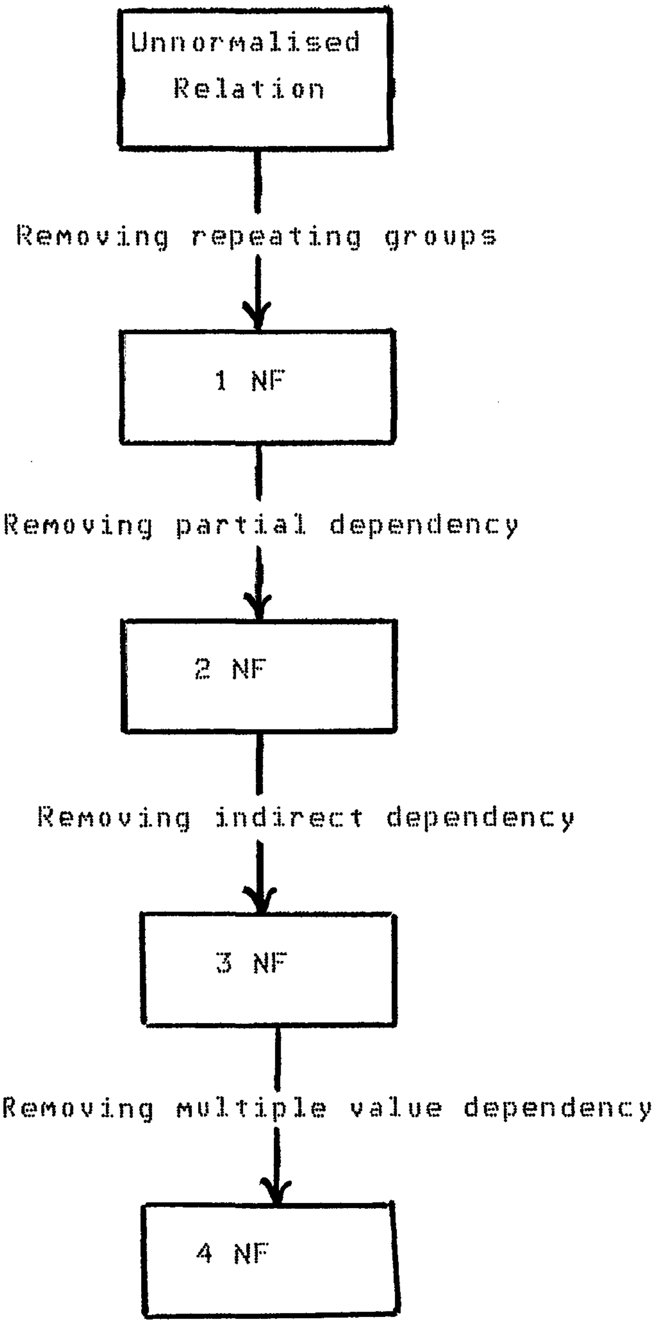

Unnormalised Relation
Removing repeating groups
1 NF
Removing partial dependency
2 NF
Removing indirect dependency
3 NF
Removing multiple value dependency
4 NF

Figure 1.10

1.6.1 FIRST NORMAL FORM :-

A relation R is in first normal form if and only if all underlying domain contains individual values, not sets or tuples. Array & repeating groups are not allowed in first normal form.

Fig. 1.9 shows a relation in 1NF. This relation will be unable to provide the following informations.

a). The completion date of project can't be obtained without assigning a project to the employee.

b). If an employee stops to work on his assigned project, then the deletion of the last segment containing his salary will also delete the completion date, but completion date may be needed in future.

c). There will be problems in updating the salary of an employee. In this case one has to search for every segment which contains that employee as part of key. If an employee is involved in many projects, much redundant updating of salary will be needed. The above mentioned problems can be removed by introducing 2NF.

1.6.2 SECOND NORMAL FORM :-

A relation R is said to be in 2NF if it is in first normal form & every non prime attribute of R is fully functionally dependent on each candidate key of R (6).

The above mentioned relation of EMPLOYEE has more than one candidate key as EMPLOYEE # & EMPLOYEE NAME. We will split the relation into two relations to make it in 2NF. As we see completion date is fully functionally dependent on concatenated key (EMPLOYEE #, PROJECT #), so all other attributes are removed to separate relation which has EMPLOYEE # only as its key.

EMPLOYEE A (EMPLOYEE #, PROJECT # , COMPLETION DATE)

EMPLOYEE B (EMPLOYEE # , EMPLOYEE NAME, SALARY)

Original relation can be obtained by taking an appropriate join of all splited (projected) relations, & original information is maintained. Any information that can be retrieved from the original structure can also be retrieved from the new structure where converse is not true. Sometimes new structure gives more information which are not available in original & the new structure becomes slightly more faithfull reflection of the real world. Anomalies as described in case of 1NF can occassionally occur in a relation which is in 2NF because of the transitive dependence in the relaton. To overcome the problem of transitive dependency in 2NF we represent the relation in third normal form.

Transitive dependence : -

In relation of EMPLOYEE as discussed earlier in fig 1.9 EMPLOYEE (EMPLOYEE #,EMPLOYEE NAME, PROJECT #, SALARY, COMPLETION DATE) we see

EMPLOYEE # ----> PROJECT #

PROJECT # ----> COMPLETION DATE

Here COMPLETION DATE is therefore transitively dependent on EMPLOYEE #. We can remove this transitivity only by representing the relation in third normal form.

1.6.3 THIRD NORMAL FORM :-

Relation R is in Third normal form if it is in 2NF & every non-prime attribute of R is non-transitively dependent on each candidate key of R.

A few problems might result from the relation not being in 3NF.

i) Before any employee is recruited the completion date of the project can't be recorded.

ii) If all the employees leave the project, the project has no employee until others will be recruited, all records containing the completion date will be deleted in this case. This may be thought an unlikely occurence, but on other files a similar danger of loss of information can be less improbable.

iii) If the completion date is changed it will be necessary to search for all records containing that completion date.

The above mentioned problems can be avoided by representing relations in 3NF. The relation EMPLOYEE can be converted to 3NF by spliting it as to avoid transitivity in following manner.

EMPLOYEE (EMPLOYEE #, EMPLOYEE NAME, SALARY,PROJECT #)

PROJECT (PROJECT # , COMPLETION DATE)

Now information about projects will be needed independently of employee information, & completion date can be independently known. Anomalies which were arising in 1NF & 2NF are removed completely in third normal form.

The advantage of representing relations in 3NF are as follows.

i). It frames the tabular structure & provides easy means for the representation of data relationships.

ii). It enhances the data access capability & hence reduces the need for restructuring the relations thereby providing the data independence and iii). It keeps relations away from undesirable update problems arising out of insertion, deletion & update.

1.6.4 FOURTH NORMAL FORM -

In the 3NF relations with multivalued dependency some problems will still exist as regards to update operation. Such relations are to be reduced to 4NF

A normalized relation R is said to be in fourth normal form (4NF)if and only if whenever there exists a multivalued dependency in R, say of attribute Y on attribute X.Thus a 3NF is not necessarily in 4NF but any 4NF relation is in 3NF.

This is most optimise form of relation. We consider the relation CTX (COURSE, TEACHER, TEXT).

CTX

COURSE	TEACHER	TEXT
COMPUTER	PROF. JOHN	COMPILER
COMPUTER	PROF. JOHN	DBMS
COMPUTER	PROF.BROWN	COMPILER
COMPUTER	PROF.BROWN	DBMS

Figure 1.11

A tuple <c,t,x,> appears in CTX if and only if course c is capable of being taught by teacher t uses text x as referene. Let us suppose that for a given course, there may exist any number of corresponding teachers & any number of corresponding texts moreover, we assume that teachers & texts are quite independent of each other. Fig. 1.11 shows a sample tabulation for CTX.

To add the information that the Computer Course uses a new text called Artificial intelligence, it is necessary to create two new tuples, one for each of the two teachers

It is clear that the difficulties are caused by the mutual independence of teachers & text, moreover that difficulties could be improved if CTX were replaced by its two projections CT (COURSE, TEACHER) & CX (COURSE,TEXT). In relation CTX of fig. 1.11 we say that there is a multivalued dependence of TEACHER on COURSE. In this way given course does not have a single corresponding teacher i.e. TEACHER is not functionally dependent on COURSE, and attribute TEXT of CTX is also multidependent on course. The problem with 3NF relations such as CTX is that they involve multivalued dependencies that are not also functional dependencies. The two projections CT (COURSE, TEACHER) & CX (COURSE,TEXT) donot involve any such dependencies, which is why they represent on improvement over the original relations. As we get from relation CTX that it is not in 4NF whereas the two projections CT & CX are each in 4NF. The example thus illustrates how reduction to fourth normal form can eliminate one final form of undesirable structure of 3NF.

1.7 COMPARISON OF DATA MODELS :-

We have stated in previous section about three well known data models. They are the relational model, in which data are assumed to be stored in the form of tables the hierarchical model in which data are assumed to be stored in the form of tree structures & the network model, in which data are assumed to be stored in the form of general graph structures. But there should be some criteria to select one of them to implement a database. We have taken relational data model to implement library database. The reason of selecting this model, is as follows:

Measuring EASE-OF-USE:- It is measured by considering two factors (i) Human factors methodology & (ii) Human factor methodology applied to query languages.

While implementing a database special care is taken about time spent by programmer. The goal of each and every organisation is to select a model which makes the programming job easier.

In case of hierarchy model user is forced to devote time & effort to solve problems that are introduced by the model and are not intrinsic to the questions being asked. When records are introduced into the structure, the hierarchy becomes more complex and so programs become more complicated. The debuging & maintenance of these programs will require more programmer time. The problems which we encountered in Hierarchical database model are also encountered in the Network model. In Network data models the data is stored in terms of records & links, so the structure is more complicated than in hierarchical database model. This complexity is reflected in the data sublanguages used for data definition & manipulation.

The above mentioned difficulties which are encountered in Hierarchical & Network models are removed in relation database. In case of relation database it is possible to define sublanguages which are easy for user. The beauty of relational model is that it does not have any structural complexity. It gives a purely logical view of data which is much easily understood. The ultimate result is that the relational data model appears to be more familiar model even to the unskilled programmers.

A relation may be stored & accesed in a variety of ways, without impacting user programs except in terms of their performance. Thus a relational database provides its users with a high degree of both physical & logical data independence.

The main goal to attain good performance of a data base is to choose a good physical representation for a relation & processing queries in ways that make good use of the given storage structure. Keeping all these mentioned advantages the relational database model has been selected.

1.8 RELATIONAL LANGUAGES :-

Since the introduction of the relational model of data by cod as a tool for general database management (7) there have been proposed several relational data languages intended for inexperience users.

Database models are broken into two parts. One part sometimes referred as the data definition language (DDL), which describes the structure of the database. The other part is called data manipulation language (DML), which describes the way the database is processed.

Retrieval of information from the database is perhaps the aspect of relational languages & the relational algebra provides the basic means of data retrieval & update in a relational data model. The relational algebra is based on the standard operations of the set theory & can be used to design a procedural data manipulation languages. Codd developed predicate calculus from relational algebra(6). This predicate calculus is called relational calculus & has been used by codd as a basis for a relatively non-procedural language called Data Sub-language (DSL)

Alpha. DSL Alpha is easier to use & retains all manipulative powers of relational Algebra. It consists simply of relational calculus in a syntactic form which more closely resembles that of a programming language.

A query in DSL Alpha has two parts: a target, which specifies the particular attributes of the particular relation which are to be returned & a qualification, which selects particular tuples form the target relation by giving a condition which these tuples must satisfy. The qualification part of DSL Alpha query, may be quite complex & may use the universal and existential quantifiers : 'for all'(∀) & 'there exists' (∃) various other languages based on relational calculus have been proposed. These languages are QUEL, COLARD, RIL.

A second major class of relational language is based on relational algebra. Relational Algebra operates on one or more relations & produces a new relation as a result. The operations can be used to retrieve information from one or more relations or to update a tuple of relation. The major operators of relational Algebra are:

(i) Projection, which returns only the specified columns of a specified relation, & eliminates duplicates from the result

(ii) Restriction - selects only those tuple of a relation which satisfy a given condition

(iii) Join, it takes two relations as arguments to produce a third relation

(iv) Division-operates on two input relations to produce a third relation

(v) Set theoretic-operators :- the union, intersection &

set difference which take two relations as operands, treating each as a set of tuples, & produce a single relation as a result.

The third class of relation languages are called Mapping oriented languages. These languages offer facilities equivalent to that of the relational calculus or relational algebra. These languages avoid mathematical concept such as quantifiers. The mapping oriented languages are SQUARE, a terse, APL like notation; SEQUEL, the structured -English Query Language (8) which is based on English Keywords & intended for use by non-specialist in data processing as well as by professional programmers and SLICK, a language intended for implementation on associative hardware. The basic building block of mapping oriented language is the mapping, which maps a known attribute or a set of attributes by means of some relation. The mapping may be used in the specification of another mapping oriented language to express queries of great complexity.

Our query language is a relational data language which is based on SEQUEL, and is intended for the inexperienced user. In this language english key word oriented set of facilities are available, which enables users making statements like query, processing, data definition, data manipulation & data control.

The words SELECT, FROM, WHERE, MODIFY, DELETE, etc. are key words used in most queries.

It is also a high level query languae. The language is used as a stand-alone language. All of its facilities are based on Consistent keyword oriented syntax. The syntax is given in Appendix [A]. This language accepts statements in free format.

The decision has also been enfluenced by the results reported in the reference (LOCH 77, LOCH 78) to adopt relational model and relational language to implement our library data base which is presented in following paragraph.

Lochovsky & Tsichrizis were concerned with the effect of data model on user performance (LOCH 77, LOCH 78). They performed several tests using different languages for different models. IMS language DL/I (IBM75), the DBTG COBOL DML (CODA 71), and Codd's ALPHA(Codd71) were taken for model hierarchical, network & relational respectively. The same information can be retrived with fewer statements in the relational language than in the record oriented (language for network & hierarchical) language. Lochovsky performed two experiments with several different kinds of applications. Result obtained from experiments were like:

Score of query correctness based on data model:-

	Hierar	Net	Relational
	chical	work	
Less experienced users			
before on line experience	23.1	28.0	72.9
After on line experience	44.6	64.0	84.3
More experienced person			
before on line experience	31.4	54.3	88.5
After on line experience	65.7	74.3	88.5

This data is adopted from LOCH78

Lochovsky concludes that the relational systems were more advantageous & lastly he pointed that it is difficult to differentiate the performance to either data model or the data language, since they were not separted. Data entry in above table is score obtained in different tests.

The proposed data base has been implemented on the HP 1000 system at the J.N.U. computer centre. The library data base was implemented jointly with mycolleague Ms. Anjana Sahai. Some portions of the project were commonly done while others are 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 analyser
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) MODIFY R SET A1 = [V1] WHERE A2 = [V2] AND
                          A3 = [V3] -- An = [Vn]  $
   (ii) INSRTN R WHERE A1 A2 -- An [V1] [V2] -- [Vn] $
  (iii) DELETE R WHERE A1 = [V1] AND A2 = [V2] --
                    AND An = [Vn] $
   (iv) DEFVEW R' ( A1' ( R1A1 ) A2' ( R1A2 ) --
                                            An' ( R1A1 ) ) $
    (v) RESTOR R $
   (vi) SUSPND R $
  (vii) GRANT P1 P2 -- Pn ON R TO [U1] [U2] -- [Un] $
 (viii) REVOKE P1 P2 -- Pn ON R FROM [U1] [U2] -- [Un] $
```

CHAPTER - 2

PROPOSED LIBRARY DATABASE MODEL

2.1 Why to use Computer based systems in libraries ?

There are several advantages of computer based systems in libraries (10). They are :

1) To manage a process more rapidly, more accurately, less expensively.

Many library processes can be reduced to clercial procedures of sorting, filing and sending notices. This tend to be routine and boring work which is subject to human error. Computer can be instructed to do these tasks and so the library staff can be released for more worthwhile work. The computer can process information much faster than humans and can therefore help to increase the flow of work through the library. Well designed computer based systems may reduce the operating and maintenance cost of library.

2). To help overcome increasing library work loads :

In many situations especially in end of financial year librarians are unable to recruit more trained staff to deal with the increase in work and so computer based systems are most suitable to reduce the work loads of libraries.

3) To offer new and improved services to users and library staff

Suppose users of a science library will wish to know about recently published articles and magazines. This type of need may be satisfied by a computer based current awareness system. Union catalogues as well as subset catalogues in particular subject areas will give more information to library staff and library users.

2.2 AN OVERVIEW OF COMPUTERBASED LIBRARY SYSTEMS :

To be more specific computers in library are those which are used in libraries to assist in the areas of housekeeping and information retrieval :

2.2.1 Housekeeping :

The housekeeping procedures in libraries are those which are necessary for the administration of the library and they are separated into four broad areas.

1) Acquisitions : In this procedure librarian is concerned with the selection, ordering and accessioning of items into the library's collection. Computers may be used to send order slips and reminders for unacknowledged or overdue orders to the booksellers, to produce list of books on order, to keep account of the money spent and the balance, to produce accessions list of recently acquired material and so-on.

2) Cataloguing :

It helps the librarian in recording and displaying details of the holdings of the library. Computers may be used in maintenance of catalogues.

3) CIRCULATION

Computers may be used to keep account of items that are on loan, to whom they are loaned, issue & due date of items and their status.

4). Serial Control : - Items which are published periodically or serially, such as journals, conference, proceedings, annuals or newsletters need to be processed in different manner by the library staff than items which are only published once. This process is time consuming and needs special attention. Computer can be used to produce faithful result in less time.

2.2.2 Information Retrieval :

Information retrieval is very important and frequently used process in a library. When some new books come to library, the library staff assigns some index or number. The computer can be used to generate index entries according to a prescribed set of rules which will be very efficient ,and accurate.

Computer can be used as current-awareness system to keep account of all recently published documents.

2.3 PROPOSED DATABASE MODEL FOR THE LIBRARY SYSTEM :

In this section we have described the various steps performed to design the proposed model of the database. The resulting schema has been called the <u>cannonical</u> schema. Before describing the actual database a few definitions in contexts of buble chart are given.

(i) <u>Cannonical schema</u> : - The cannonical schema is defined as 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 & using the data.

(ii) <u>Buble chart</u> : - It is graphical representation of user's view. It is a graph with point to point directed links between single data items, representing associations of the two types. A buble chart is shown in fig 2.1

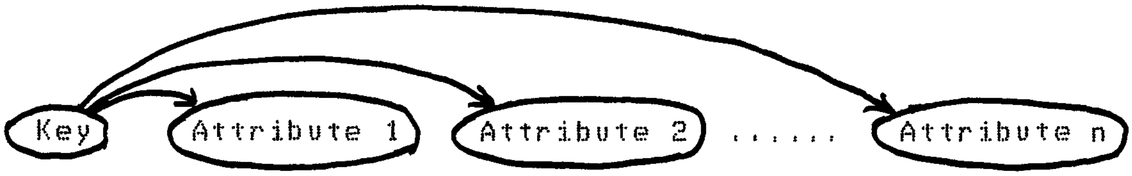


Figure 2.1

Note that the data item type is the same as the attributes, which is the name used in relational model. The buble chart is reduced to 3 NF by ensuring full functional dependency of the attributes on the key attribute (or all the attribute of the concatenated key) & by removing transitive dependencies between attributes. M:M relationships are removed by the method shown in fig 2.2

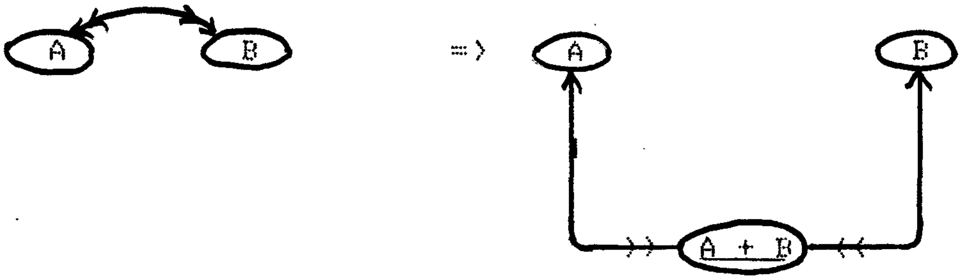


Figure 2.2

(iii) <u>Primary key</u> : - A primary key is a node with one or more single arrow leaving it.

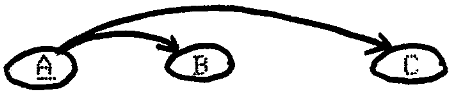


Figure 2.3

Primary key is underlined to make it unique from another keys.

(iv) Candidate key : - When more than one data item identifies the other data items in a tuple then each of them are known as candidate key. As shown in fig 2.4

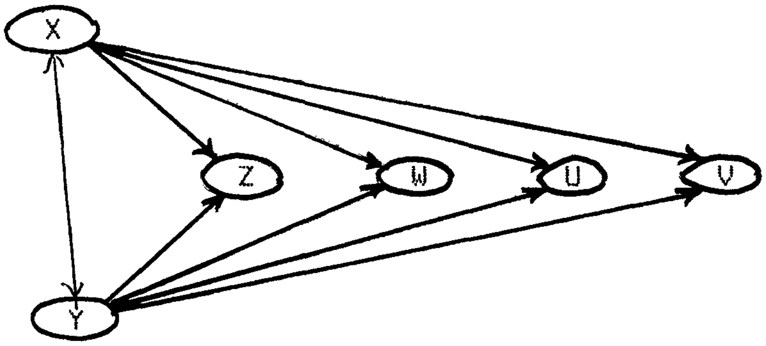


Figure 2.4

Here attributes Z, W, U, V are identified by both X & Y and so X & Y are candidate keys.

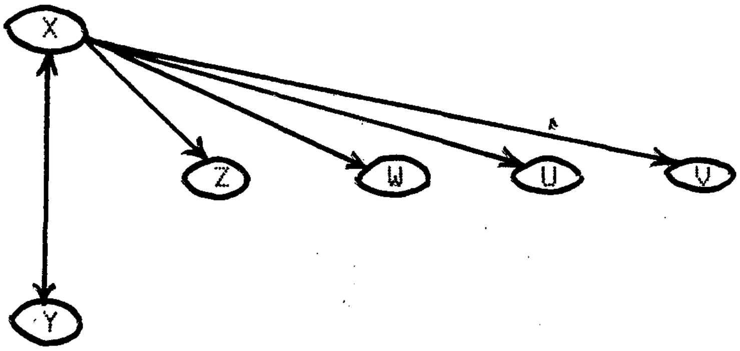


Figure 2.5

However as shown in fig 2.5 only one of these two attributes will be taken as prime key.

(v) Attribute : - An attribute is a node with no single arrows leaving it. For example in fig 2.5 Z, W, U, V are attributes.

(vi) Root key : - A root key is a primary key with no single arrow leaving it to another primary key.

(vii) Isolated attribute : - A node with no single arrow links entering or leaving it except double arrow links. It is not identified by primary key. There should be no isolated attributes in the canonical graph. It may be considered in one of the following way :-

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

(b) It may be treated as one data item record. It should however be noted that an isolated attribute often results from a wrong interpretation of the data, & so the meaning related to it should be carefully checked.

(viii) Intersecting attribute : It is an attribute with more than one single arrow link entering it. Intesecting attributes on the final canonical graph are removed. The methods of dealing with intersecting attribute are shown in fig 2.6 below.

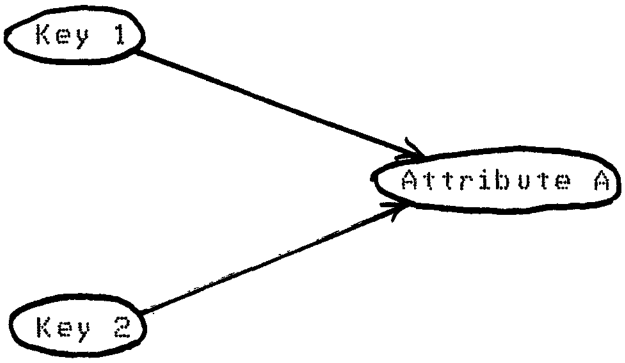


Fig 2.6.a

In fig 2.6.a Attribute A is intersecting attribute.

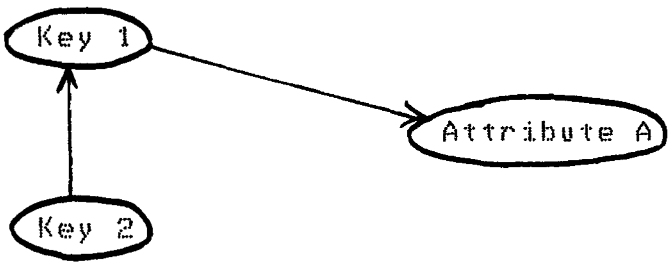


Fig 2.6.b

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

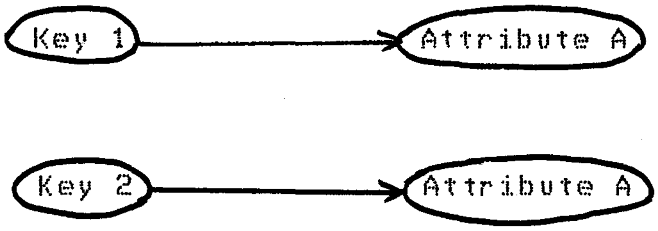
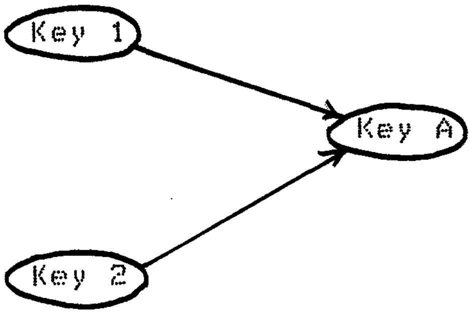


Fig 2.6.c

Redundant version of it may be connected to each associated key.



It may be made into a key with no attribute

Fig 2.6.d

(ix) Secondary Key : It is an attribute with either one or more double arrow links leaving it.

(x) Synonyms : Two data items with different names in different user views but having the same meaning are termed as synonyms.

(xi) Homonyms : Two data items having the same name in different users view but with different meanings are known as homonyms. Name of one item is generally changed to remove the homonyms.

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

(i) First user view is taken & drawn in the form of buble chart. All hidden transitive dependencies are removed before representing it in the 3NF.

Whenever a concatenated key is used, it is drawn as one buble component. Data items of the concatenated key are drawn as separate buble. See fig 2.7

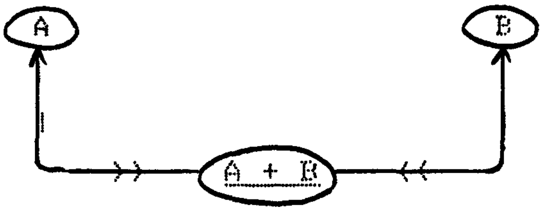


Fig 2.7

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 & not only on a part of it.

(ii) The next user view is taken and represented in above mentioned form of step (i). It is merged with the buble chart of previous view. A check is made for homonyms and synonyms & removed if any found.

(iii) In the resulting graph we distinguished between attribute nodes and primary key nodes. Primary key is underlined.

(iv) The inverse between key is added if it does not already exists. In case of M:M between keys, the inverse association is even to be used at any time in the future, it is replaced by adding an extra concatenated key as we have discussed in fig 2.2

(v) The associations are examined to identify the redundant ones among them. These are then carefully checked for their associated meaning & removed if found genuinely redundant.

(vi) thus steps (ii) to (v) are repeated until all user views are merged into the graphs.

(vii) The root keys are identified & the graph is rearranged with the root keys at top.

(viii) the graph is examined for presence of isolated attribute and if there exists, they are treated as repeating attribute in a variable length record & one data item record as discussed previously in part (vii) of section 2.3.

(ix) The graph is again arranged to avoid any intersecting attributes. The method to avoid intersecting attribute is mentioned in step (viii) of section 2.3.

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

(xi) All secondary keys are identified & the links drawn between boxes.

(xii) The canonical schema is finally converted into a relational schema. Each box is represented as a relation. Since all links and assciations 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 foriegn 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.

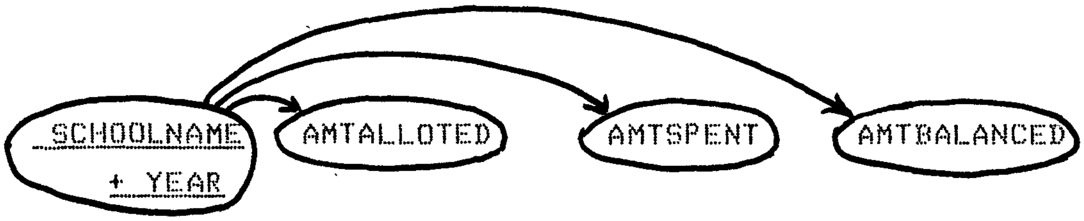
The next paragraph gives the actual conceptual schema designed for the library database.

Steps for merging few of the views for the design of the conceptual schema as explained in previous section are shown below.

1st view : Every year each school is alloted money to be spent on ordering literature for the library. Records are maintained for spent and balanced amount. Record used for this purpose is shown below.

SCHOOLNAME	YEAR	AMTALLOTED	AMTSPENT	AMTBALANCE

When represented as a buble chart it looks like :

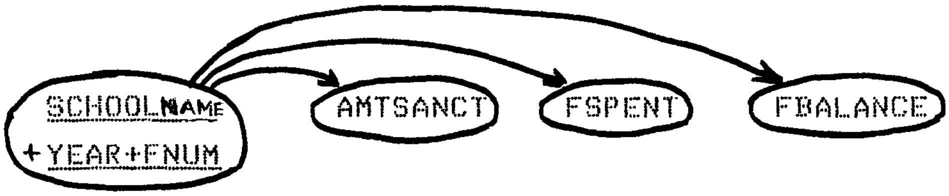


Full functional dependencies are ensured and no transitive dependency occurs.

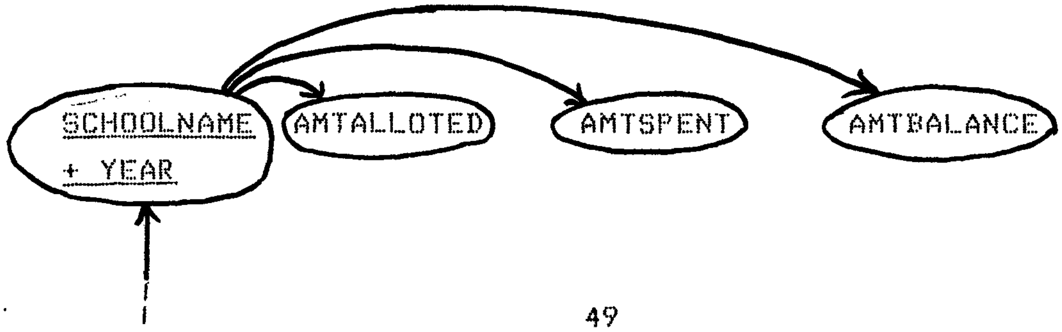
2nd view : Each faculty member of a school is sanctioned money every year for ordering of material for library. An information regarding amount spent and amount balanced is kept for each faculty member in following record.

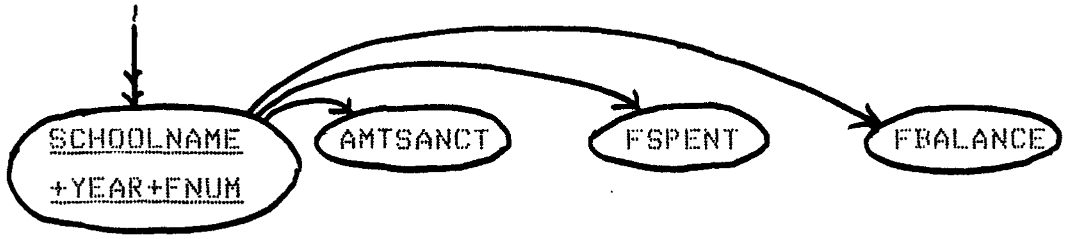
FNUM	SCHOOLNAME	YEAR	AMTSANCT	FSPENT	FBALANCE

When this record is represented as buble chart it looks like :



This view is then merged with the previous view & transitive dependencies if any, are removed.

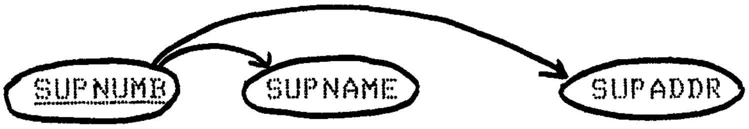




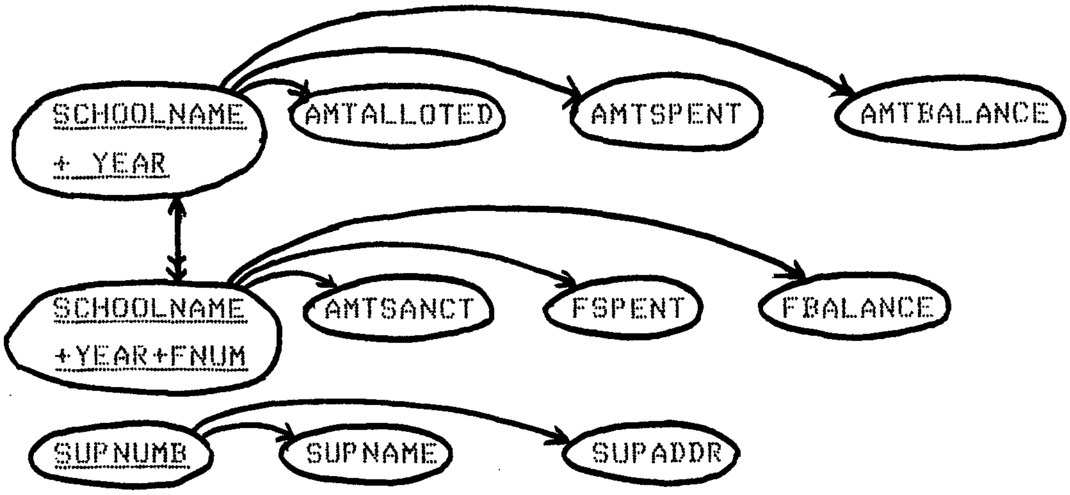
3rd view : Details of suppliers who supply material to the library are maintained in this view. Each supplier has a unique supplier number.

SUPNUMB	SUPNAME	SUPADDR

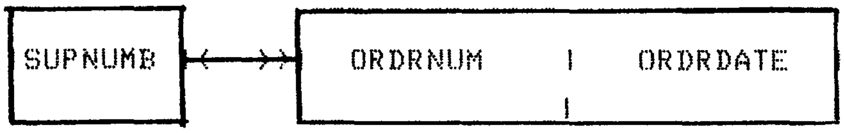
It is represented as a buble chart with no transitive dependencies as shown below.



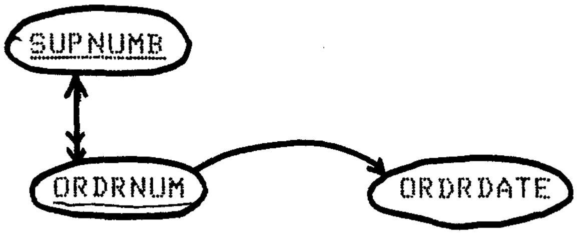
This view is then merged with previous views.



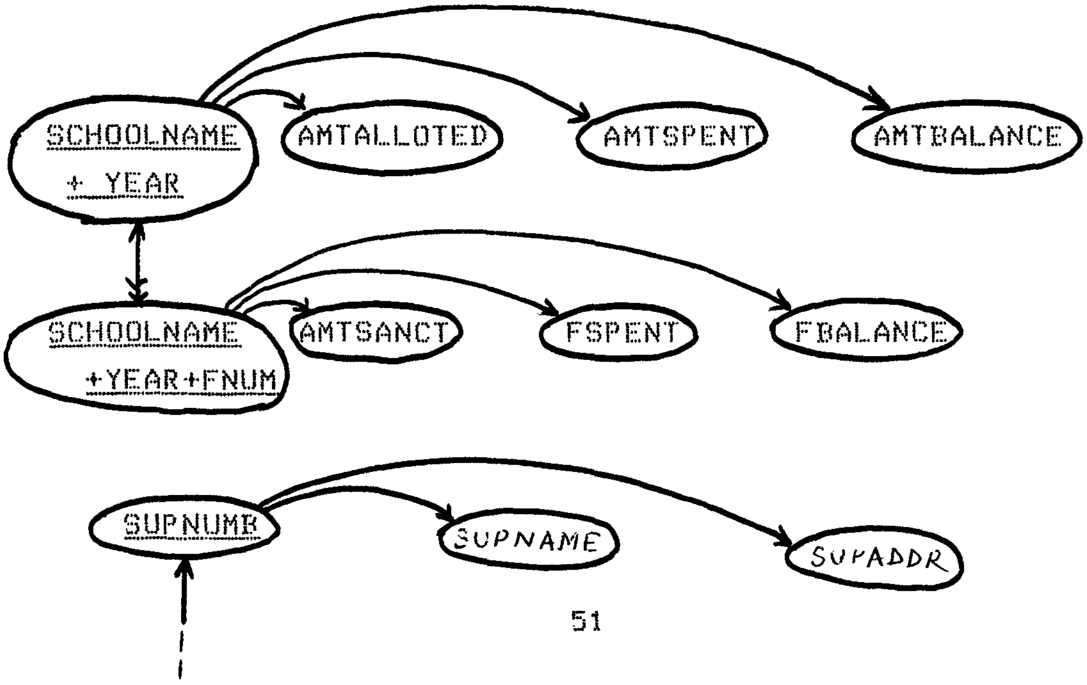
4th view : For each order placed with a supplier the order number and date of order are maintained by librarian for sending reminder letter in case of late service and for other purposes.

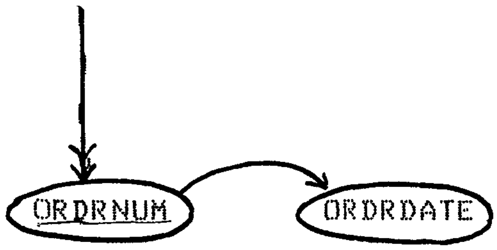


Buble chart of above record is drawn below :



After merging this view with previous views we get

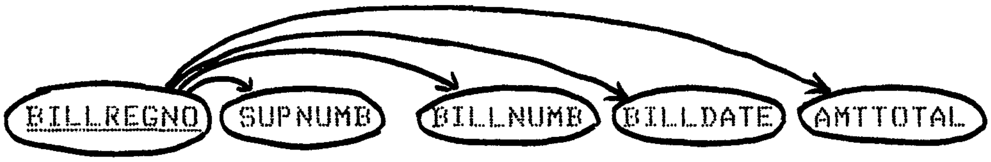




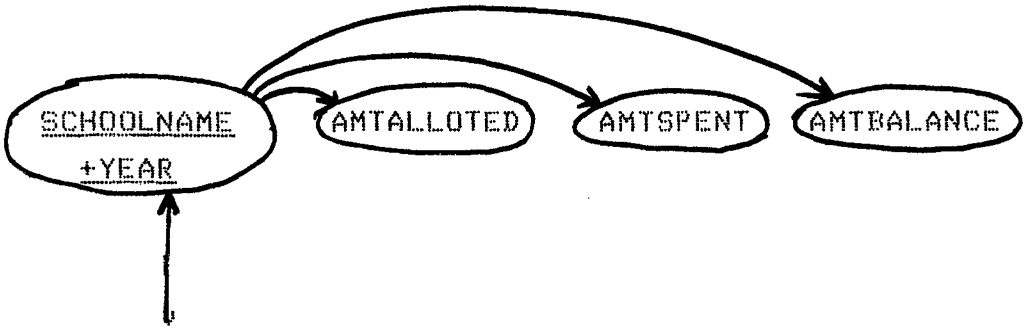
5th view : Suppliers send their bill with library items. Librarian checks the bill and gives registration number to each bill received by him. Other informations like bill number, the date of bill and the total amount of the bill are also maintained.

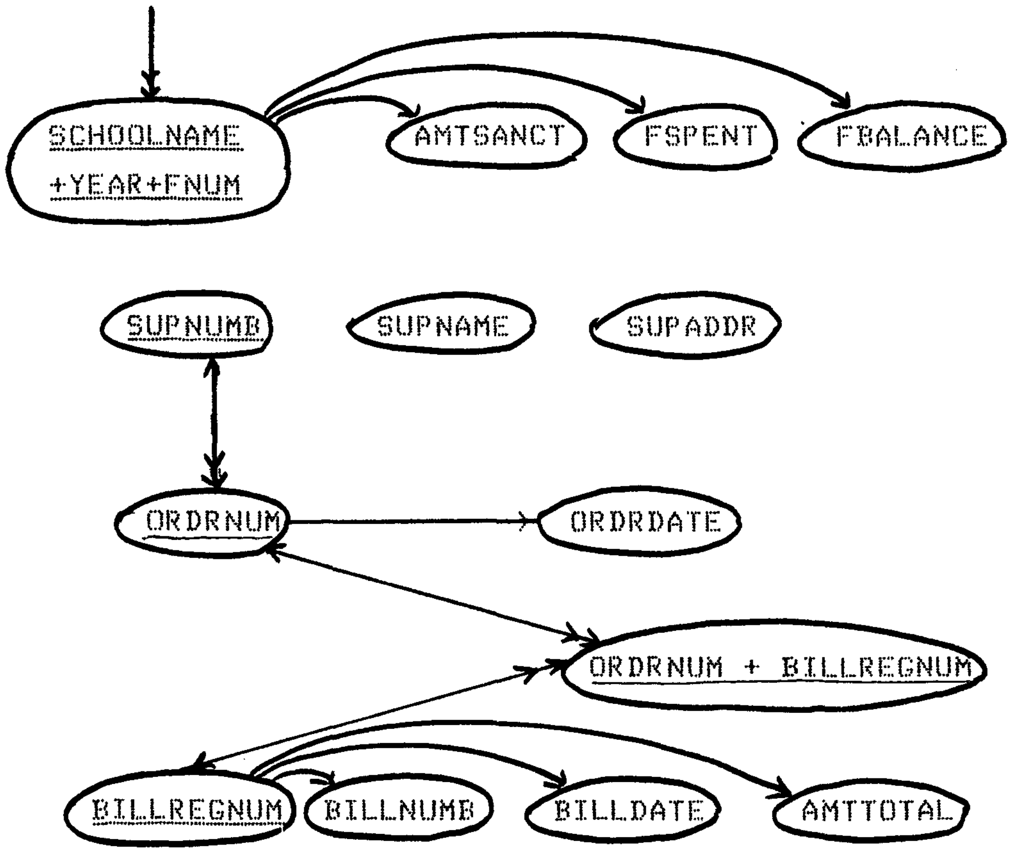
BILLREGNO	BILLNUMB	BILLDATE	SUPNUMB	AMTTOTAL

After representing this in buble chart we get :



After merging with previous views we will have :

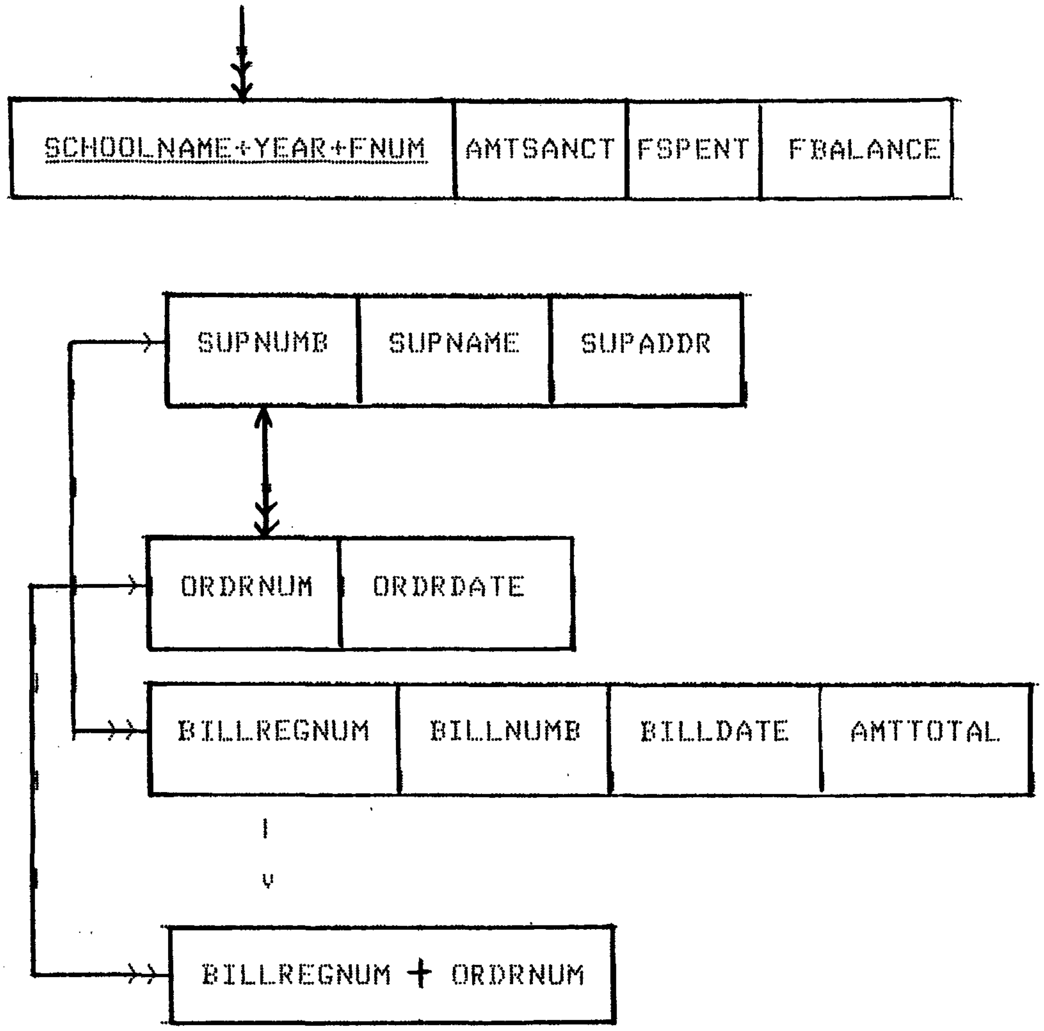




This process of merging of views is carried out for all other views. The final buble chart so obtained is a canonical structure and this is finally represented as a relational conceptual schema. Buble chart is first represented in the form of boxes and then converted into a relational conceptual schema.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

(a) The canonical structure is drawn below :-

SCHOOLNAME+YEAR	AMTALLOTED	AMTSPENT	AMTBALANCE



(b) To convert the above canonical structure 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.

SCHOOLNAME + YEAR	AMTALLOTED	AMTSPENT	AMTBALANCE

SCHOOLNAME + YEAR+FNUM	AMTSANCT	FSPENT	FBALANCE

SUPNUMB	SUPNAME	SUPADDR

ORDRNUM	ORDRDATE	SUPNUMB

SUPNUMB is foreign key in this relation.

BILLREGNUM	SUPNUMB	BILLNUMB	BILLDATE	AMTTOTAL

BILLREGNUM + ORDRNUM

2.5 RELATIONS USED IN LIBRARY DATABASE

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 3NF. The primary keys are represented by underlined attributes. The relationals are discussed below.

The govt. organisation (say U.G.C.) sanctions some fixed annual financial aid to purchase library items (books, journal, magazines etc) to universities according to certain rules & conditions. Again faculty members of an institute are sanctioned fixed amount to recomend books, journals etc. annually according to certain rule & conditions of universities and institutes.

Relations are as follows :

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

FACULTYDTA(YEAR, SCHOOLNAME, FNUM , AMTSANCT, AMTSPENT, FBALANCE)

Here attributes are explained in short :

AMTALLOTED : Amount alloted to a particular FNUM (faculty member) of institute

AMTSPENT : Amount exhausted in recommending library items by a faculty member.

AMTBALANCE : Amount remained balance against FNUM.

After getting recommendation of library items from faculty members, librarian puts in order to those items after proper verification. These items will be supplied by suppliers within time. Relations concerned are as follows :

(3) SUPPLIER (SUPNUMB , SUPNAME, SUPADDR)

(4) ORDER (ORDRNUM , ORDRDATE, SUPNUMB)

Relations (3) & (4) help librarian to keep all information of supplier of library like their name(SUPNAME), supplier number(SUPNUMB), supplier's address (SUPADDR) & order number(ORDRNUMB), order date (ORDRDATE).

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

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

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

Relation numbers (5), (6) & (7) keep information of recommended library items by faculty numbers (FNUM) of different schools. Attributes used in relations are disscused below :

YEAR : Year of publication of books or journal or confrence recommended.

RECOMCOPIS : Number of copies recommended

RECPERIOD : Period for which a particular library item is recommended.

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

(9) BILLORDER (BILLREGNUM + ORDRNUM)

Relations (8) & (9) give information about details of a bill received by librarian. It gives registration number (BILLREGNUM), supplier number, bill no. (BILLNUMB) date on which bill is prepared (BILLDATE) & total amount (AMTTOTAl) on that particular bill. It helps librarian for housekeeping information. Thus we can summarize that above mentioned relations (1 to 9) are used to maintain acquisitions systems in following way (10)

It helps to -

(i) Receive recommendation & establish that the items are not already on order.

(ii) Order the items & chase the bookseller if no action appears to be being taken.

(iii) Accession the items on arrival & keep statistics.

(iv) maintain a record of items on order or in process.

(v) maintain the accounts & to control of accounts so that expenditure can be more easily controlled and the current status of various bugdets made easily available.

(vi) production of list of recently acquired items.

(vii) notification of individuals when an item which they have recommended has been received.

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

Relation (10) keeps whole information about library items once they have been reached in library. It gives information about each book's TITLE, AUTHOR, VOLUME, EDITION, COPYNO, PUBNAME(Publisher name), PRICE, ARRIVDATE(Arrival date in library) , CALLNO, LIBDIV(Library division), STATUS(Whether in library, lost, issued or sent for binding), ABSTRCT (Abstact of that book) etc. It also helps to know about NAME (conference name), PLACE(Place of conference held) and DATE (date of conference) of library items related to conference.

(11) FIELDDETEL (<u>ACCNO + FIELD + SUBFIELD</u>)

This relation helps to know about field and subfield of each book and conference material available in library.

(12) CARDDETAIL (CARDNO , ISSUEENAME, ISSUEADDR,
NUMOFCARDS)

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

These two relations keep information about the library item on loan. It enables the library staff on counter to issue and return the library items. It also helps to maintain the integrity of library database. A library user can't get more than six books issued in his name. Librarian can send reminder to issuee if due date exceeds a fortnight. As soon as library item is returned to the library, all informations about carddetail file and issuedetail file are updated and status of that item is set to INLIB.

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

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

(16)JFLDSUBFLD(TITLE + VOLUME + NO + SUBFIELD)

(17)JFIELD (TITLE + VOLUME + FIELD)

These four relations are used to maintain information regarding journals (bound & separate) which have been received by library. Here bound journals means, journals received by library are bounded annually to make it one copy. Fields and subfields of each journal can also be known very easily.

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

(19)THESISFLD(TITLE + AUTHOR + FIELD + SUBFIELD)

These two relations give information about thesis of M.Phil

& Ph.D degrees. Here we can get TITLE of thesis, research scholar's name (AUTHOR), YEAR of submission, SUPERVISOR name under whom he has completed his thesis, name of INSTITUTE to which he belongs, field & subfield of his thesis and library division, where his dissertation is kept.

2.6 INTEGRITY CHECKS

The integrity constraints are discussed below:

LITTYPE must be checked before insert operation in the relation BOOKINLIB. Its value could either be ' B ' for books or 'C' for conference report.

At the time of issuing a library item the serial number of card in issuefile is compared with NUMOFCARDS in carddetail relation. The serial no. must be less than or equal to the NUMOFCARDS. This is to ensure that no authorized library member can get issued more library items than he is entitled to. At the time of issuing a material above check will have to be performed.

The value of STATUS in relations BOOKINLIB and BOUNDJURNAL should be INLIB if the item concerned is on shelf; ISSUED if it is issued to a member or LOST if it is not traced. The status of unbound journals is set to INLIB because they are not issued. Whenever, a material is issued the status of that item is set to ISSUED in file BOOKINLIB & corresponding entries are made in relation ISSUEFILE & CARDDETAILFILE.

2.7 DATA SECURITY

The security of data is maintained at two levels. At first level no unauthorized user can access the database, so the whole database is secured against invalid users. At the second level, security is maintained on individual relations for different operations to be performed on them. This is obtained by GRANT facilities in query language described in chapter 3 section 3.3.1. Only those users who have been granted the data manipulation facilities through grant command, can perform the manipulations. The grant options include all facilities of the language, namely the query, manipulation & control facilities. These options can be granted as well as revoked according to situation arises.

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

2.8 DIRECTORIES

In the process of implementing our library database we have created several directories to get good performance of our system. Our all directories are direct access type 2 files. Type 2 file has been discussed in appendix C. Following directories are maintained in our library data base

(1) DBMSRD : This directory maintains the relations used in library database and to keep status of those relations. We have adopted a hashing technique to physically keep the relation on file. The bucket size is taken as three in our implementation. we have also defined overflow area to get rid of collision. Our relation name may be of 10 characters in length. 11th-12th characters in front of each relation are used to keep status of relation in directory (00 for SUSPEND status & 01 for RESTORE status). Thus for each relation 12 characters are needed to maintain its existence in directory. Duplicate relation name has been not

taken in consideration. The format is shown below :

Relation name 1	00/01	RELATION name 2	00/01	-- --	Relation name n	00/01

1 10 11-12 13 22 23-24 ---> character position

(2) DBMS11 : It keeps information about attributes of each relation. It also maintains information regarding domain type, start position and length of each attributte. Each attribute is concatenated with relation name from which it belongs, to make it unique. Attribute may be of atmost 10 characters in length. Here 20 (10 for relation name + 10 for attribute) are reserved for each concatenated relation & attribute. 21st-22nd characters are used to represent the attribute type. Notations used to denote attribute type is as follows

01 - Alpha
02 - Alphanumeric
03 - Numeric

Type of attribute is checked especialy to maintain integrity of database. Character position 23rd - 26th in front of concatenated value of relation & attribute keeps start position of literal value of that attribute.

Character position 27th to 28th keeps character length of value of that attribute.

Character position 29th - 30th maintains bucket no.

Character postion 31st - 32nd keeps record no. This record no & bucket no are taken from directory DBMSRD where relations are stored on physical file. Bucket no and record no are considered to make a relaton name unique. some overflow area is also reserved to avoid collision more than three. format is shown below :

Concatenated relation & attribute	Attribute type	start position	attribute length	-- --
1 20	21 22	23 26	27 28	

bucket no of relation	Record no of rel	Concatenated rel + attr	Attribute type
29 30	31 32	33 52	53 54

(3) DBMS00 :

It is our view relation directory. View relation name is different from existing defined relation name in DBMSRD directory. It's length is almost 10 characters alpha. In character position 11th & 12th, status of relation (00 for SUSPEND & 01 for RESTORE) is maintained. View relation name is hashed and stored on the calculated recordno. Bucket size is five & some overflow area is also kept reserved to overcome the collision problem. Format is shown below :

View relation name 1	00/01	--- ----	View relation name n	00/01

1 10 11 12 ------> character position

(4) DBMS01 : It maintains view attribute's information defined on existing relations and attribute name. We can give same view attribute name as existing attribute in original relation in directory DBMS8. Since view attribute name is just synonyms for existing attributes, so all information with an attribute like its type, start position, length, bucket number & record number are copied from DBMS8. Two more informations are kept with view attributes to make it unique & to identify from other attributes of the same record number. These are record number, & bucket number of attributes on directory DBMS8 where actually they were stored physically. Bucket size of this directory is five & some overflow area is preserved to overcome the collision problem. Format is shown below:

R1'+A1'	BN	RN	TYPE	START POSTION	LENGTH	REL BUKET	REL RECRD	R1'+A2'	-- --

Where R1'+A1' is concatenated with view relation & view attribute.

B1 = Bucket number of attribute in directory.

RN = record number of attribute in directory.

REL BUKET = Bucket number of relation in directory.

REL RECRD = Record number of relation in directory.

(5) USRCD :- It is used for authorization check of users. It also helps to maintain integrity of database. In our database systems three types of users are authorized to operate the database, viz. X99711, Y88123 & Z01234. X99711 in our database is DBA & he has right to intertain all operation which are specified on our database. Y88123 can get information from database by using SELECT, & SECONT statements. He can Define his own view on existing relation & attribute. We have restricted the user to define view on only one relation name. He can also restore & suspend the relation name according to environment. Z01234 can also use the database like Y88123 except RESTOR & SUSPND option. Infront of each option we have set 01 in 7th & 8th characterposition. Directory format is shown below:

X99711	SELECT	01	SECONT	01	DEFVEW	01	RESTOR	01	-->

SUSPND	01	-------

Y88123	SELECT	01	SECONT	01	DEFVEW	01	SUSPND	01	RESTOR	01

Z01234	SELECT	01	SECONT	01	DEFVEW	01	Blank

All these operations are discussed in chapter 3.

(6) USRCOD :- It's entries come after executing GRANT command. DBA gives some right to Y88123 to use the database in different manner from what he has right to operate with it. DBA authorizes Y88123 to delete, Modify, INSERT the tuples or part of it on some relations. And DBA can also snatch the rights whatever he had given or part of it on some relation from Y88123. In case of revoke command entries (right & relation) which are revoked, are replaced by blanks. To create these entries, the option is placed on physical file against the name of grantee & relation name is also kept with option(on which it is granted). Bucket size is taken as 10 & some overflow area is also reserved to maintain more entries. To perform any operation first USRCD directory is referred. In case a user's right to perform that operation is not known from USRCD then this directory is referred to know the specified operation against this user's name. It helps to maintain the integrity of the database. Format is shown below.

Y88123	DELETE	R	01	MODIFY	R	01	INSRTN	-->

R	01

Here R may be any relation name on which grant right is issued.

CHAPTER — 3

QUERY STATEMENTS & QUERY PROCESSOR

It has already been mentioned that the query language provides the facilities as regards to ,Data definition, Data manipulation & data control in the same fashion as another relational language can provide. These facilities are discussed below:

3.1 Data Definition facilities:

It enables users to create relations, define alternative view and specify security locks on the data base. The data definition facilities of language describes the data structures provided by the system on which the language runs. Query statements for data definition are as follows:-

3.1.1 Create Statement:

This statement is designed to be used by DBA only. This statement creates a new relation (table) to be physically stored in the system. Null values are not permitted in this statement. The user specifies a relation name, domain name, its type and the length. Type & length are used for domains to signify the type of each attributes. Type of an attribute can take one of the following values, 'ALPHA' if attribute values are alpha; 'ALFNUM', if the attribute values are alphanumeric & 'NUMERIC', if the attribute values are numeric. Statement is as follows

```
CREATE R ( A1 Type ( S1 ) ( Length ) A2 Type ( S2 )
          ( length ) --- An Type ( S1 ) (Length ) ) $
```

S1, S2 --- Sn give the start position of attributes A1, A2, A3------ An respectively in the relation R. The word 'length' in the statement denotes the maximum length allowed for each attribute in terms of characters. Blank is also counted as a character in our HP 1000 systems. If the value specified for an attribute 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 two directories for this purpose. Directory named DBMSRD as discussed in chapter 2 maintains the name of relations & their status i.e. whether 'suspendend' or restored. Another directory DBMS8 as discussed in chapter 2 maintains the concatenated value of relations with each of their attributes & a corresponding definition of each of these.

3.1.2 DEFINE VIEW :

The define view statement defines a user's view over the existing relation or view. The view is defined here only in logical sense and no corresponding physical relation is created and stored in the data base. Here one can define a view on a single relation only. A view so defined used just like a physical relation. DML and control statements of the query operate on view just as they operate on physical relations. The format of the view statement is as follows :

```
DEFVEW R' ( A1' ( R1A1 ) A2' ( R1A2 ) -- An' ( R1An ) ) $
```

Through this statement, a user can define a view R' with attributes A1′, A2′, ------- An′ taking attributes A1, A2, ------- An from relation R1. Here R1 could be any existing relation name or view name in the database. The length of view name & attribute name must not be more than 10 characters. Each view must have a name different from any of the relations or view already defined. However attribute name may be same as used in these relations. A definition of the view is maintained in directories as that for existing relations. We have used two directories DBMS00 and DBMS01 for this purpose.

3.1.3 RESTORE STATEMENT :

A relation or view can only be used when it is in a Restore state. At the time of creation the relations are in Restore state. Any suspended relation may be restored using this statement. In the directory DBMSRD the entry 01 in columns 11 & 12 indicates the restored status & the entry 00 indicates suspended status (as explained in section 3.1.4)The statement is as follows :

RESTOR R $.

3.1.4 SUSPEND STATEMENT : A relation or a view is suspended using this statement. A relation in suspended state i.e. with the entry 00 in columns 11 & 12 of this directory is not accessible. Defined views are used as relations & hence they can also be suspended. As is done for relations, the restore bit in directory DBMS00 is put equal to zero. Note that when a relation is suspended, all views defined on that relations are also suspended. The statement format is as follows:

SUSPND R $

Thus R (either relation or view) is suspended by this statement.

3.2 DATA MANIPULATION FACILITIES:

Data manipulation facilities enable a user to access & update the relations & views in the data base. The update facilities are provided by the following 3 query language statements:

3.2.1(a) INSERT STATEMENT:-

This statement enables a legal user to insert a new tuple or set of tuples into a relation. Attributes that are not specified by the insertion statement are given blank values. The order of attributes as given in the query statement has no significance. The format of the Insert statement is as follows:

```
INSRTN R A1 A2 --- An [V1] [V2] -- [Vn] $
```

3.2.1(b) DELETE STATEMENT :- This statement allows a legal user to delete conditionally a tuple or a set of tuples from a relation or a view. The condition is specified in the where clause of the query statement. The format is as follows:-

```
DELETE  R  WHERE  A1 = [V1] AND A2 = [V2] -- An = [Vn] $
```

Note that relation name in the above statement could either be a relation of the conceptual schema or be a view, since views are treated in the same manner as relations.

3.2.1(c) MODIFY STATEMENT

This statement allows a legal user to modify conditionally an attribute or a set of attributes in tuple or a set of tuples in the specified relation or a view. The condition is specified in the where clause of the query statement. The format is as follows:

MODIFY R SET A1 = [V1] WHERE A2 = [V2] AND
A3 = [V3] ---- An = [Vn] $

The number of attributes to be modified as given in the SET clause was restricted to one in the actual implementation.

3.2.2 INFORMATION RETRIEVAL : This section describes query language statements for information retrieval. There are various format of select statement. Each one of them is designed for specific purpose. These statements are described below.

3.2.2(a). Format is as follows :

SELECT A1 A2 -- An FROM R WHERE Ai = [Vi] AND Aj = [Vj]
-- Az = [Vz] $.

This statement allows a legal user to obtain the values of the attribute A1, A2 -- An specified in the query statement from a tuple or set of tuples satisfying the condition in where clause of the statement. Ai, Ak --An are any attributes name in the relation & Vi --- Vn are the corresponding conditional values which are specified for the target tuples in the relation.

Note that the equality condition could have been replaced by the 'greater than' or 'less than' conditions, but since queries with such conditions are almost never asked from the library data base. Hence these conditions were not implemented. In actual implementation the condition in WHERE Clause was restricted to one.

3.2.2(b) A retrieval problem is very important for the library data base. Retrieval is always done by selecting 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 null, in which case the user will not know the real cause & will conclude that the information required does not exist in the database. This is very inconvenient & 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 relevent information e.g. Call No., STATUS, etc.about it. Obviously it is very difficult and inconvenient to remember the complete & exact titles. User might want to select information about books which contain say, words like 'database' or 'compiler design' etc. in their title & not be bothered with the preceeding or succeeding parts of the title. Sometimes user wants to retrieve information about library items by knowing only author name. Suppose a book has more than one author & user knows only one author name or part of its name. In short there are many instances when the user could like to retrieve information from the database knowing only 'part' of the complete value of the attribute by which retrieval is done. Such type of retrieval is possible & stated in query statements in succeeding two statement.

This statement exactly in the manner in which section 3.2.2(a). However in the condition Clause Vk is the part or full value for attribute Ak in the tuple.

SELECT A1 A2 ----- An FROM R WHERE PART Ak = [Vk] $

In actual implementation Where clause is restricted to only one attribute.

3.2.2(c) This statement needs to satisfy two domain names in full or partial both in specified relation name to retrieve the information STATEMENT is as follows:

SELECT A1 A2 -- An FROM R WHERE PART A6 AND
        A7 = [V6] AND [V7] $

Here conditions to be satisfied like A6 and A7 should have partial value. Where clause is restricted to only two attributes.

3.2.2(d) No. of tuples satisfying the WHERE Clause can be counted & retrievedby following format of query language. This statement selects & displays conditionally the values of the attribute A1 -- Ak, also provides the count of tuple satisfying the condition. The condition is specified in where clause. The format is as follows :

SECONT A1 A2 -- An FROM R WHERE As = [Vs]
        At = [Vt] -- Az = [Vz] $

Condition in WHERE Clause is restriced to one in the actual implementation.

3.2.2(e) This statement conditionally selects attributes specified in statement & are displayed in ascending order of the value of attribute coming after keyword ORDRBY. Statement is as follows :

SELECT A1 A2 -- An FROM R WHERE As = [Vs] AND
     At = [Vt] ORDRBY Ak $

The condition in where clause is restricted to one in actual implementation.

3.2.2(f)

This statement retrieves information about several domains where a particular attribute Ak is common in two relations R1 and R2. FORMAT is as follows:

SELECT A1 A2 -- An FROM R1 R2 WHERE R1,Ak = R2,Ak $

Here attribute Ak followed by relation name, may be any attribute.Relations R1 and R2 must contain all the attributes mentioned in the statement.

Note that in above all query statement the relation could either be a relation of the conceptual schema or a view defined by a user. Since views are used in the same way as relations.

3.3 DATA CONTROL FACILITIES : The data control facilities especially used by DBA are discussed in GRANT and REVOKE statements.

3.3.1 GRANT STATEMENT : A user like DBA can grant access to his relations to other users by means of this statement. Here in our statement grantor may not permit grantee to grant the listed privileges to other users. Statement is as follows :

```
GRANT P1 P2 -- Pn ON R1 R2 -- Rn  TO [U1] [U2] -- [Un] $.
```

In our implementation part we have restricted to one relation name & to one user. Here DBA can grant right P1, P2 -- Pn that could be SELECT, MODIFY, DELETE, DEFINEVIEW, INSERT, SUSPEND or RESTORE. The valus of U1, U2 -- Un are actually the user codes of the users. In the proposed model, there are three classes of the users. Each class of user has a unique usercode, which along with his privileges are maintained in directories USRCD & USRCOD which are discussed in chapter 2. The three classes of users & their corresponding codes are

	Class	Usercode
1	Data base Administrator	X99711
2	Assistant Administrator	Y88123
3	General user	Z01234

3.3.2 REVOKE STATEMENT : Once a privilege has been granted, it may be withdrawn through this statement. The named privileged are revoked from the grantee. Here grantor can revoke only the rights which he had granted on the relations. Statement is as follows :

REVOKE P1 P2 -- Pn ON R1 R2 -- Rn FROM [U1] [U2] -- [Un] $.

Thus we can revoke one or more than one right on one or more than one relations from one or more than one users. But in our actual implementation we have restricted to one relation name & to one user. These P1, P2 -- Pn are privileges such as DELETE, MODIFY, INSERT etc.

These data control facilities allow to maintain the security of database.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 operate upon the privileged right which are maintained in directies USRCD & USRCOD against his usercode.

3.4 THE QUERY LANGUAGE PROCESSOR : The user interogates the Library Database query language statements, specifying the data which must be retrieved & the conditions that must be satisfied by the desired data. It is the responsibility of the language processor to check syntactically the query statement and to call corresponding semantic routine provided query is found correct.The task of query processing is to determine the set of data to be checked & retrieved from the database, the proper order in which the data should be accessed, & the types of manipulations that must be performed on the data. This process is referred to by different authors as query translation or access path finding. We start with our database by RUN TOKN command. The DBMS asks for the usercode which is assigned to the user. If user fails to give his correct user code, the user is informed that the he is unauthorised user & query terminates. In case of correct user code user is asked to issue his query. User's statement is stored in a buffer of size 144 words. In our statement first query word is the operation user wants to perform. We have checked whether user has got right to perform operation, which he has specified in first query word in his statement or not. If he has right to perform that operation a flag is set to zero state otherwise flag is set to 1. Tokens are generated & stored in different tables.

3.4.1 **TABLE FORMATS : -** We have used three tables. They are (1) IDENTIFIER TABLE (2) LITRAL TABLE & (3) UNIFORM SYMBOL table.

《1》 IDENTIFIER TABLE :- This table keeps all identifiers occured in a query statement. As soon as an identifier comes in query statement the ID table is scanned by lexical analyser to check previous occurence of that identifier. Entry is made after ensuring that currently occured identifier has not occured previously. Format is as follows

IDENTIFIER	CODE	OCCURENCE NO.
Identifier 1	4	1
Identifier 2	4	2
|	|	|
|	|	|
Identifier n	4	n

(2) LITERAL TABLE :- All literals coming in query statement are recorded in this table with their code and occurence number. Entry of literal is made in in the same way as made for identifier in ID table. Format is as follows :

LITERAL	CODE	OCCURENCE NO
Literal 1	3	1
Literal 2	3	2
|	|	|
|	|	|
Literal n	3	n

(3) UNIFORM SYMBOL TABLE :- It keeps code of all query words in a query statement according to sequence they have occured. Say a query statement

DELETE R WHERE A1 = [V1] AND A2 = [V2] -- An = [Vn] $

Here in this statement DELETE, WHERE & AND are keywords. Equality sign ' = ' and square brackets are delimiters. Keywords and delimiters are kept in fixed table IARA. In above statement R, A1, A2 -- An are identifiers and V1, V2 -- Vn are literals. The uniform symbol table ISTB will keep all tokens generated by lexical analyser in above statement in the following form. In this table repeated keyword, delimiter, identifier and literal any one of these will get entry according to their occurence position.

Code	occurence position
1	6
4	1
1	3
4	2
2	26
3	1
1	8
4	3
2	26
3	2

3.4.2 LEXICAL ANALYSER :- Lexical analyser generates tokens. In our implementation tokens are separated by blank character. Codes used for tokens are as follows :-

1	Keyword
2	Delimiter
3	Literal
4	Identifier

3.4.3 SYNTAX CHECK :- As soon as tokens are generated by lexical analyser a corresponding syntax checking routine is called. This routine checks the tokens for their proper place of occurence in query statement. If token is in proper place in statement a flag is set to zero against that token otherwise flag is set to 1 to indicate the wrong occurence of token. In case flag is found 1 then error message is given according to situation.

3.4.4 SEMANTICS ROUTINE :- If all flags are zero then corresponding semantic routine is called to retrieve the information. This semantic routine retrieves information by taking appropriate informations from Identifier table & literal table.

APPENDIX A

Syntax is the description of ways in which words must be ordered to make structurally acceptable sentences in language. We have represented our syntax in BNF notation. Nonterminals are enclosed within angular brackets " < " & " > " .

< 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 > := <expression1> <comparison> <expression2>

<expression1> := <attribute name>
/ PART <attribute name>
/ <system entity name> , <attribute name>

<expression2> := < literal >/< system entity name>, <attribute name>
/ PART < literal >

<comparison> := = / -= / > / < / >= / =<

Here symbols have usual meaning like symbol ' = ' is equal & ' > ' is greater than.

< literal > := [< alpha >] / [<numeric>]
/ [< alphanumeric >]

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

< numeric > := < integer > / < real >

< alphanumeric > := < alphabet > [< alphabet >]$_0^n$
< digit > [< special character >]$_0^n$
[< alphabet >/< digit >]$_0^n$
/ < digit > [< digit >]$_0^n$
[<spl.character>]$_0^n$
< alphabet >
[< alphabet > / < digit >]$_0^n$

< special character > := . / , / - / : / /

< 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 > . < integer2 >

```
                             / [< sign >] . < integer2 >
                             / < integer >.
                             / < integer > E [<sign>] < integer1 >
                             / <integer>.<integer> E [<sign>]
                                         <integer1>
                                    / [<sign>].<integer2>E
                                     [< sign >] <integer1 >
                               / < integer > . E [<sign>]<integer1>
 <integer2 >          :=   < digit > [< digit >]$_0^n$
 < integer1 >              := < 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 list >]$_0^n$
 < option >                := INSRTN/DELETE/MODIFY/RESTORE
                             / SUSPND
```

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

< ddl statement > := < create-relation >
/ < define-view > / < suspnd>
/ < 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 > := < integer2 >

< length > := < integer2 >

< type > := ALPHA / NUMERIC / ALFNUM

< define-view > := DEFVEW < view name >
[< view field list >]

< view field list > := < attribute name >
(<relation name> <attribute
name >) [<attribute name>
(<relation name> <attrib-
ute name >)]$_0^n$

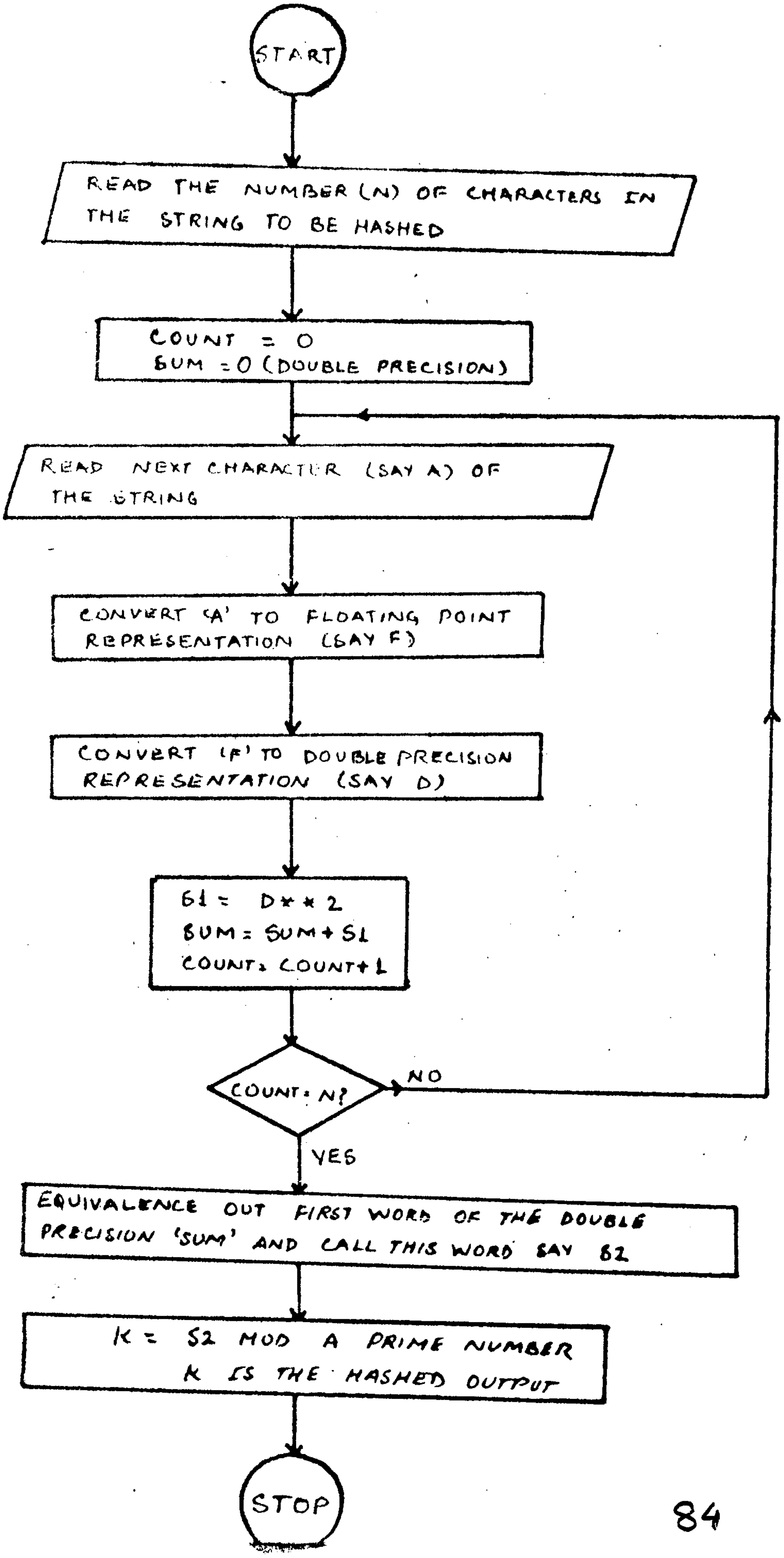
< view name > := < identifier >

< suspend > := SUSPND <system entity name>

< restore > := RESTOR <system entity name>

APPENDIX - B

HASHING TECHNIQUE FLOWCHART.


START
READ THE NUMBER (N) OF CHARACTERS IN THE STRING TO BE HASHED
COUNT = 0
SUM = 0 (DOUBLE PRECISION)
READ NEXT CHARACTER (SAY A) OF THE STRING
CONVERT 'A' TO FLOATING POINT REPRESENTATION (SAY F)
CONVERT 'F' TO DOUBLE PRECISION REPRESENTATION (SAY D)
S1 = D**2
SUM = SUM + S1
COUNT = COUNT + 1
COUNT = N?
NO
YES
EQUIVALENCE OUT FIRST WORD OF THE DOUBLE PRECISION 'SUM' AND CALL THIS WORD SAY S2
K = S2 MOD A PRIME NUMBER
K IS THE HASHED OUTPUT
STOP

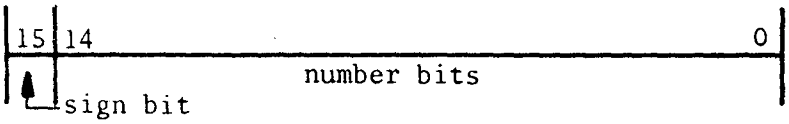
APPENDIX
DATA FORMAT IN MEMORY

The data format in memory of HP 1000 system for an Integer, Real and the 3 word double precision numbers are as follows

INTEGER FORMAT

PURPOSE: An integer datum is always an exact representation of a positive, negative or zero valued integer, occupies one 16-bit word and has a range of -2^{15} to $2^{15}-1$.

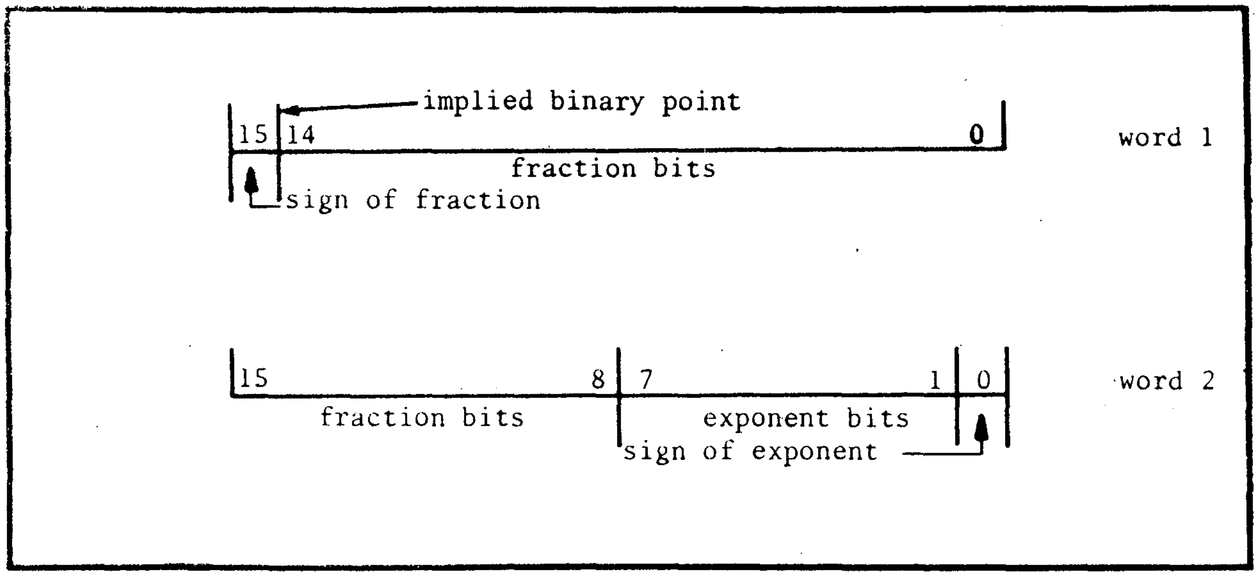
FORMAT:



REAL FORMAT

PURPOSE: A real datum is a processor approximation to the positive, negative or zero valued real number, occupies two consecutive 16-bit words in memory and has an approximate range of 10^{-38} to 10^{38}.

FORMAT:

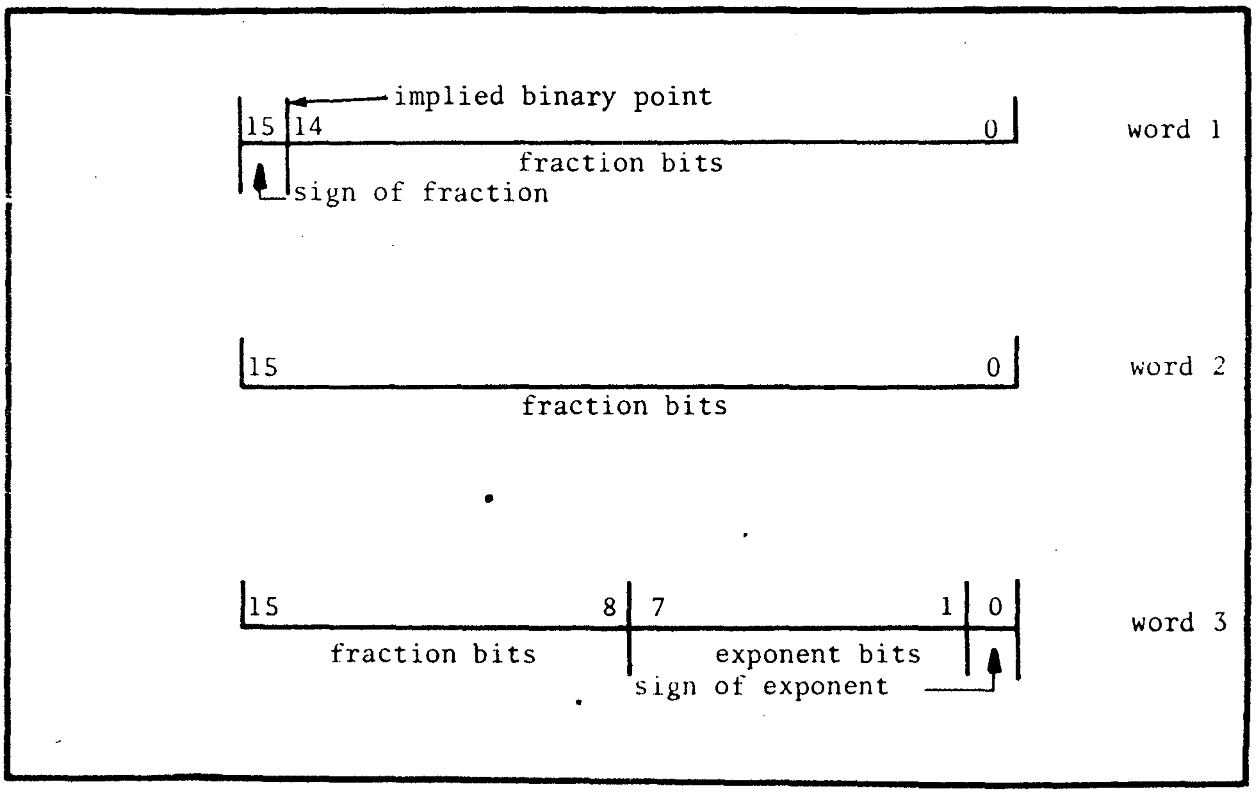


COMMENTS: A real number has a 23-bit fraction and a 7-bit exponent.

3 WORD DOUBLE PRECISION FORMAT

PURPOSE: A double precision datum is a processor approximation to a positive, negative or zero valued double precision number, occupies three consecutive 16-bit words in memory and has an approximate range of 10^{-38} to 10^{38}.

FORMAT:



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

APPENDIX — C

HP-1000 FILE MANAGEMENT SYSTEM

The data files in a database systems are organised in such a fashion that they can be used in several applications rather than a single application by DBMS. File management is supported by FMP Calls in our HP 1000 system. FMP Program calls enable us for interface between programs & file management utility routines. These calls help us to Create, Open, Close, Read, Write & to purge the files. All data used in database are stored on disc i.e. secondary storage of the computer. The capacity of secondary storage (disc) & main memory of our system is 20 M Byte & 128 K words respectively. The FMP Calls are mainly used for input to or output from disc files. The information about files is maiantained in directory created by FMP. These directories are: The FMP Cartridge directory which is a master index to all active FMP cartridges & the file directory, which contains information on each file on a particular cartridge. The Cartridge directory is maintained on the system disc, while file directory is maintained on each Cartridge.

C.1 FILE TYPE :- 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. File type has been categorised in three parts as shown below :

Category	Type	Description
Control	0	None disc device files
Fixed length random access non extendable	1	Fixed length 128 word record files
	2	Fixed user defined record length
Variable length sequential access	3	Variable length records any data type
	4	Source program file type ASCII
	5	Object program file relocatable binary
	6	Executable program file memory image code
	7	Absolute binary
	8 to 32767	User defined data format

All file types are discussed below :-

Type 0 files : These types are used to reference non-disc devices by name. Device independence can be measured by type 0 files where standard file commands are used to control the device. FMGR Command creates type 0 files. The record format of a type 0 file is achieved by the device type.

Type 1 file :-

These type of files contain fixed length records of 128 words. Type 1 files permit us for direct acces between disc & the user's buffer area in our program. File management package transfers data to & from disc in 128 word blocks. Due to this reason type 1 files are fastest in data transfer rate. Except type 0 files, all other file types may be opened & accessed as a type 1 file in order to make transfer rate faster. It is accessed randomly.

Type 2 files : Type 2 files are also of fixed length records but length is defined by the user at file creation time. Here in this type data transfer routed through data control block one logical record at a time and hence the transfer rate of type 2 & above file types become slower.

Type 3 files : These files are of variable length record & are extendable. Data transfer in these file types also takes place in the same way as in type 2 files.

Type 4 files : These are also of variable length records & same as type 3 only difference is that the system expects these files to contain ASCII data. Source programs are of type 4 files.

Type 5 files : These type are same as type 3 files, except the system assumes that type 5 files contain relocatable binary code. Typically object programs files are found type 5.

Type 6 file : These are also variable record length files. System assumes that type 6 file contains a program in memory-image format that is ready to run. These type files are created by save program (SP) Command. These files are always accessed by FMP as type 1 files.

Type 7 files : System assumes that type 7 files contain absolute binary code & these type files are variable record length.

Type > 7 files : These file types are same as type 3 files but the content is user defined. FMP recognises special processing based on file type for types greater than 7.

C.2 File Security : Each file has a security code. This code may be zero, negative or positive. A file with zero security can be opened & modified by any user. A file with positive security code can be opened to read by any user but it is write restricted. A negative code restricts all acess to the file.

C.3 FMP PROGRAM CALLS :- The FMP Program calls used in file management to assist our data base are discussed **below:-**

C.3.1 CREAT :- This call creates a disc file in terms of its name, size, type & its location. Creat call makes an entry in the file directory for the file & allocates disc space to the file. After executing CREAT Call, the file is left in the update mode for exclusive use of program performing the call. FORMAT is as follows :

CALL CREAT (IDCB, IERR, NAME, ISIZE, ITYPE, <u>ISECU</u> , <u>ICR</u> , <u>IDCBS</u>)

Where parameters have their own significance. Underlined parameters are, optional. Used parameters are discussed below:-

IDCB : Data control block is a block of words defined within our program that acts as an interface between the program & the file management package. It contains control information for the file including the file name, type, size & location on disc to avoid directory access. It also acts as a buffer for the physical transfer of data between a file & our program. It is also used to keep track of current record position in file. Each DCB is an array with a minimium of 144 words. The first 16 words are control block to provide all the file information required by the FMP Calls.

IERR : One word variable im which a negative error code is returned.

NAME :- NAME is used to specify the file name to be created. This is 3 word array containing file name in ASCII form.

ISIZE :- It specifies the file size. It is two word array. First word & second word contains number of blocks & record length respectively. Second word is used only in case of file type 2.

ITYPE : It is 1 word integer variable in range 1-32767.

ISECU :- It is used to protect the file from another user & is one word variable in range 0 through ±32767. +ve value is used for write protection and -ve for both read & write protection.

ICR :- It is cartridge reference parameter & is optional one word variable.

IDCBS :- Data control Buffer size is specified by this one word optional variable. It is set to number of words in DCB buffer if larger than 128.

C.3.2 OPEN :- This call opens an existing file which have been created prior to the open call. A file is opened for update or for standard sequential write. Files may be opened for exclusive use of the calling program or for non-exclusive use of upto seven programs. FORMAT is as follows :

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

Here all parameters except IOPTN is discussed previously in CREAT FMP.

IOPTN Parameter is discussed below :

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					X	A	K	V	M			F	T	U	E

FUNCTION CODE!
--> Type 0 options <---

To get more facilities of the file management we can set following bits as follows :

E (bit 0) = 0 File opened exclusively for this program
1 File may be shared by up to 7 programs

U (bit) = 0 File opened for standard(non-update)write
1 File is opened for update

T (bit) = 0 File type defined at creation
= 1 File type is forced to type 1

In update mode generally type 2 files are opened. Type 3 & above files are not generally used in update mode.

C.3.3 CLOSE:- This call is used to close a file after use in our program. The file remains in the system available to other programs once it is closed in one program. After Call close the data control block becomes free which can be used for another programs. A disc file opened for exclusive use of calling program may be truncated to its actual length. The format is as follows :

CALL CLOSE (IDCB, IERR , ITURN)

ITURN is discussed below:

ITURN - It is used to truncate a file to its actual length. It is 1 word variable containing integer number of blocks to be deleted from the file at closing time. If it is omitted or zero the file is closed without truncation.

C.3.4 FILE ACCESS :-

READF & WRITF routines are used to access the information from files. Type 1 & 2 are accesed randomly & type 3 and above are accessed sequentially. Calls to these routines are the same whether the file is a device (type 0) or a disc file type 1 & above. As we have discussed that the access mode of file type 3 & 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 & a new end of file is written immediately follwoing the record. This process of writting continues as we write a new record. But in file type 1 & 2 the end of file is written at the end of file according to the file size at creation. Since each record is a fixed length determined at creation, the file is easily positioned to particular record. Generally one record is written or read at a time although more may be transferred when accesing a type 1 file.

C.3.4(a) READF:- This routine reads a record from specified open file in program. One full record or a specified number of words is read. The format is as follows:

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

Here IDCB & IERR have as usual meaning.

IBUF :It is an array into which the record is read. It is also known as user buffer. It is kept large enough to adjust the record. In case IL is specified, it should be of length IL.

IL :- It is optional 1 word variable specifying number of words to be read. The effect of IL parameter in READF is shown below :

IL Value	File type 0	File type 1	File type > 1
IL > 0	Upto IL words are read if less than IL file defined record length is read	Exactly IL words are read IL may be more or less than 128 word record	Upto IL Words are read if less than IL actual record length is read
IL = 0	Zero length record is read,usually record is skipped & counted as read	No action	Record is skipped and counted as read
IL omitted	--- Do ---	128 word record is read	Actual record length is read

LEN :- The actual number of words transferred to the user buffer is returned in LEN upon completion of a read, number of words in LEN is set equal to IL. It is used to check [redacted] possible overflow of the user buffer. Following value is returned to inform the end of file.

Type 0 - Len is set to -1 when EOF is read

Type 1 & 2 - IERR is set to -12 indicating an error where access is not possible beyond EOF

Type 3 & up - LEN is set to -1 for the first EOF read.

NUM :- It is used only in case of file types 1 & type 2. It may be specified for other file types but it is ignored. It's value varies from 1 to 32767. By specifying this random access becomes very easy & we can read the record which is given in NUM parameter.

C.3.4(b) WRITF :- This routine is used to add a new record, delete a record or part of it & to modify a record in an open file. Type 1 files are written in blocks of 128 words. For type 2 files the exact record length specified at creation is written. FORMAT is as follows:

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

IDCB & IERR have usual meaning.

IBUF :- It is user defined buffer which contains record to be written.

IL :- It is length in word of the record to be written in file. If it is omitted, one record is written in case of type 1 & 2 files, zero length record to othertypes.

NUM :- It is also one word optional variable which specifies record number to be written in file.

APPENDIX-D

```
If you want to continue,please give your query
 Otherwise  give $ in first column to terminate query

INSRTN BOOKINLIB WHERE TITLE AUTHOR ACCNO [DATA PROCESSING] [MARTIN J.] [4123501 $
Query processed & entry is made in data file.

If you want to continue,please give your query
 Otherwise  give $ in first column to terminate query

 SELECT TITLE FROM BOOKINLIB WHERE ACCNO = [4123501 $

 TITLE :- DATA PROCESSING

 QUERY PROCESSED

 If you want to continue,please give your query
  Otherwise  give $ in first column to terminate query

 DELETE BOOKINLIB WHERE ACCNO = [4123501 $
 QUERY PROCESSED.

 If you want to continue,please give your query
  Otherwise  give $ in first column to terminate query

 SELECT TITLE FROM BOOKINLIB WHERE ACCNO = [4123501 $
 Data not found.

 If you want to continue,please give your query
  Otherwise  give $ in first column to terminate query
 $
 MODIFY BOOKINLIB SET STATUS = [INLIB] WHERE TITLE = [ AN INTRODUCTION TO DATA BASE SYSTEMS ] AND AUTHOR = [ DATE C.J.] $
Your Query is Syntactically Correct

  Tuple before update is :-
 AN INTRODUCTION TO DATA BASE SYSTEMS    DATE C.J.                          2 8 145.00    LOST  COMSC.  681.3.06D262IN2     115427COMPUTER S

 Updated record:-
 AN INTRODUCTION TO DATA BASE SYSTEMS    DATE C.J.                          2 8 145.00    INLIB COMSC.  681.3.06D262IN2     115427COMPUTER S

 QUERY PROCESSED


If you want to continue,please give your query
 Otherwise  give $ in first column to terminate query

 SUSPND BOOKINLIB $
QUERY IS PROCESSED


If you want to continue,please give your query
 Otherwise  give $ in first column to terminate query

 SELECT TITLE AUTHOR FROM BOOKINLIB WHERE ACCNO = [ 115427 ] $
RELATION IS SUSPENDED,QUERY CANNOT BE PROCESSED


If you want to continue,please give your query
 Otherwise  give $ in first column to terminate query

 RESTOR BOOKINLIB $
Query is syntactically correct
QUERY IS PROCESSED


If you want to continue,please give your query
 Otherwise  give $ in first column to terminate query

 SELECT STATUS FROM BOOKINLIB WHERE TITLE = [ AN INTRODUCTION TO DATA BASE SYSTEMS ] $

STATUS:-      INLIB

QUERY PROCESSED


If you want to continue,please give your query
 Otherwise  give $ in first column to terminate query
 DEFVEW JNULIB ( TYTLE ( BOOKINLIBTITLE ) AYTHOR ( BOOKINLIBAUTHOR )  ACCNO ( BOOKINLIBACCNO ) ) $
 Query is syntactically correct
Query is processed & entries in view relation
irectory & view attribute directories are made

 DEFVEW BOOK TYTLE ( BOOKINLIBTITLE ) AYTHOR ( BOOKINLIBAUTHOR )  ACCNO ( BOOKINLIBACCNO ) ) $
 Check ID for proper place
 Check for left & right parenthesis


If you want to continue,please give your query
 Otherwise  give $ in first column to terminate query

 DEFVEW BOOK ( TYTLE ( BOOKINLIBTITLE ) AYTHOR ( BOOKINLIBAUTHOR ) ACCNO ( BOOKINLIBACCNO ) ) $
 Query is syntactically correct
Please give some another view relation name
this view relation has been defined by some user


If you want to continue,please give your query
 Otherwise  give $ in first column to terminate query

 SELECT TYTLE AYTHOR FROM BOOK WHERE ACCNO = [ 115075 ] $
RELATION FOUND IN VIEW DIRECTORY
LITERAL DOES NOT EXIST IN DATA FILE

 $
If you want to continue,please give your query
 Otherwise  give $ in first column to terminate query

 SELECT TYTLE AYTHOR FROM BOOK WHERE ACCNO = [ 115427 ] $
RELATION FOUND IN VIEW DIRECTORY
LITERAL DOES NOT EXIST IN DATA FILE


If you want to continue,please give your query
 Otherwise  give $ in first column to terminate query

 SELECT ACCNO TYTLE FROM BOOK WHERE AYTHOR = [ DATE C.J. ] $
RELATION FOUND IN VIEW DIRECTORY
CORRECT ATTRIBUTE IN DIRECTORY NOT FOUND
```

```
 If you want to continue,please give your query
  Otherwise  give $ in first column to terminate query


  SELECT TYTLE AYTHOR FROM JNULIB WHERE ACCNO = [ 115427 ] $
 RELATION NOT FOUND


 If you want to continue,please give your query
  Otherwise  give $ in first column to terminate query

  SUSPND ISSUEFILE $
 QUERY IS PROCESSED


 If you want to continue,please give your query
  Otherwise  give $ in first column to terminate query

  DEFVEW JNULIB ( TYTLE ( ISSUEFILETITLE ) AYTHOR ( ISSUEFILEAUTHOR )  ADDRES ( ISSUEEADDR ) ) $
  Query is syntactically correct
 Relation name is suspended on which you are def. vew


 If you want to continue,please give your query
  Otherwise  give $ in first column to terminate query

  RESTOR ISSUEFILE $
 Query is syntactically correct
 QUERY IS PROCESSED


 If you want to continue,please give your query
  Otherwise  give $ in first column to terminate query

  DEFVEW JNULIB ( TYTLE ( BOOKINLIBTITLE ) AYTHOR ( ISSUEFILEAUTHOR ) ) $
  Query is syntactically correct
 You are defining view on different relations 1 0


 If you want to continue,please give your query
  Otherwise  give $ in first column to terminate query
```

```
If you want to continue,please give your query
 Otherwise  give $ in first column to terminate query

 SELECT TITLE AUTHOR FROM BOOKINLIB WHERE ACCNO = [ 115427 ] $
RELATION IS SUSPENDED,QUERY CANNOT BE PROCESSED

You are not a valid user



If you want to continue,please give your query
 Otherwise  give $ in first column to terminate query

 RESTOR BOOKINLIB $
Query is syntactically correct
QUERY IS PROCESSED


If you want to continue,please give your query
 Otherwise  give $ in first column to terminate query



 SELECT TITLE AUTHOR FROM BOOKINLIB WHERE ACCNO = [ 115427 ] $

TITLE:-       AN INTRODUCTION TO DATA BASE SYSTEMS

AUTHOR:-       DATE C.J.

QUERY PROCESSED


If you want to continue,please give your query
 Otherwise  give $ in first column to terminate query
```

```
  GRANT DELETE MODIFY INSRTN ON BOOKINLIB TO [ Z01234 ] $
Query is correct
  You can't give your grant right to user :- Z01234


 If you want to continue,please give your query
  Otherwise  give $ in first column to terminate query

  SUSPND ISSUEFILE $
 QUERY IS PROCESSED


 If you want to continue,please give your query
  Otherwise  give $ in first column to terminate query

  GRANT DELETE MODIFY ON ISSUEFILE TO [ Y88123 ] $
Query is correct
 ISSUEFILE  Relation name is suspended
 If you want to continue,please give your query
  Otherwise  give $ in first column to terminate query


  GRANT MODIFY ON THESIS TO [ Z01234 ] $
Query is correct
  You can't give your grant right to user :- Z01234


 If you want to continue,please give your query
  Otherwise  give $ in first column to terminate query

  GRANT MODIFY DELETE ON THESIS TO [ Y88123 ] $
Query is correct
 Query is processed & corresponding entries are made in directory


 If you want to continue,please give your query
  Otherwise  give $ in first column to terminate query

  REVOKE MODIFY DELETE ON THESIS FROM [Y88123 ] $
 Your Query is Syntactically correct
 Query is processed & corresponding entries are made in directory


 If you want to continue,please give your query
  Otherwise  give $ in first column to terminate query

  GRANT INSRTN MODIFY DELETE ON BOOKINLIB TO [ Y88123 ] $
Query is correct
 Query is processed & corresponding entries are made in directory


 If you want to continue,please give your query
  Otherwise  give $ in first column to terminate query
```

APPENDIX E

LIST OF SOURCE PROGRAM

```
FTN4,L
      PROGRAM TOKN,3,,,,,,,, THIS IS OUR MAIN PROGRAM
C    ****************************************************************
C     THIS IS OUR MAIN PROGRAM WHICH CALLS SEGMENT LEXCL
C    ****************************************************************
      INTEGER ISTB(50,2),ITAB1(30,22),ITAB(30,42),IPRS(50,3),KNAME(3)
     *,ICODE(20),ISTB1(20,8),IDCB(144),IBUF(144),NAM5(3),IBUF2(96),
     *ISECU(3)
      COMMON ITAB,ITAB1,ISTB,MM,M,JM,IPRS,IQRY,IF,ISTB1,KM,ICODE,JLVAR
     *,JFLAG,MFLG,IBUF2
      DATA ICODF,KNAME,NAM5/8,2HLE,2HXC,2HL ,2HUS,2HRC,2HD /
      DATA ISECU/2H-7,2H56,2H19/
      BLANK=020040B
C -----***     VALIDITY OF THE USER IS CHECKED                  ***---
      WRITE(1,1)
1     FORMAT(1H ,'GIVE YOUR USER CODE:_')
      READ(1,5) (ICODE(KI),KI=1,6)
5     FORMAT(6A1)
      CALL HASS(ICODE,6,7,NUMBR)
      CALL OPEN(IDCB,IERR,NAM5,1,ISECU)
      IF(IERR.NE.2) GO TO 50
      CALL SFILL(IBUF2,1,48,BLANK)
      CALL READF(IDCB,IERR,IBUF2,48,LEN,NUMBR)
      IF(IERR.LT.0) GO TO 50
      CALL CODE
      WRITE(IBUF,10)(ICODE(IK),IK=1,6)
10    FORMAT(6A1)
      IER=0
      IF(JSCOM(IBUF2,1,6,IBUF,1,IER)) 40,45,40
40    WRITE(20,41)
41    FORMAT(1H ,'You are not a valid user')
      STOP
50    WRITE(1,51)IERR
51    FORMAT(1H ,'FMGR ERROR **********',I4)
      STOP
C ------*** SEGMENT LEXCL IS CALLED                               ***----
45    CALL CLOSE(IDCB)
      CALL EXEC(ICODF,KNAME)
      END
C    *******************************************************************
      SUBROUTINE HASS(IBUFF,MAR,IDIV,NUMBR),SUBROUTINE USED FOR HASHING
      DIMENSION IBUFF(20), ALPH1(30),ISUM(3)
      DOUBLE PRECISION SUM,N ,ALPH2(30)
      EQUIVALENCE(SUM,ISUM(2))
      SUM=0.D00
      DO 10 J=1,MAR
      ALPH1(J) =FLOAT(IBUFF(J))
      ALPH2(J)=DBLE(ALPH1(J))
      N=(ALPH2(J))**2
      SUM=SUM+N
10    CONTINUE
      K=ISUM(2)
      NUMBR=MOD(K,IDIV)
      WRITE(1,20)NUMBR,IBUFF
20    FORMAT(1H ,I7,5X,30A1)
      RETURN
      END
C    *******************************************************************
      SUBROUTINE CONVR(NUMHAS,IB),CONVERSION FROM I FORMAT TO ASCII
      IF(NUMHAS.GT.9) GO TO 50
      IE=0
      CALL SDEA2(NUMHAS,1,2,IE)
      IB=NUMHAS
      GO TO 85
50    N=MOD(NUMHAS,10)
      M=NUMHAS/10
      IE=0
      CALL SDEA2(N,1,2,IE)
      CALL SDEA2(M,1,2,IE)
      CALL SMOVE(M,2,2,IB,1)
      CALL SMOVE(N,2,2,IB,2)
85    RETURN
      END
C **********************************************************************
C         THIS SEGMENT BREAKS THE QUERY INTO TOKENS
C ---*******************************************************************
      PROGRAM LEXCL,5
      INTEGER IBUF(256),IBUF1(40),IARA(8,34),ISTB(50,2),IDCB(144)
     *,NAM5(3),ITAB1(30,22),LL(30),ITAB(30,42),ICODE(20),ISECU(3),
     *BLANK,IBUF2(96),JNAME(3),IPRS(50,3),ISTB1(20,8),ICODH(20)
      COMMON ITAB,ITAB1,ISTB,MM,M,JM,IPRS,IQRY,IF,ISTB1,KM,ICODE,NUMBR
     *,JFLAG,MFLG,IBUF2
C ---***          THE KEYWORDS AND DELIMITERS USED IN THE QUERIES    ***
C       *         ARE GIVEN BELOW                                    *---
      DATA IARA,NAM5/1HS,1HE,1HL,1HE,1HC,1HT,1,1,
     *1HF,1HR,1HO,1HM,1H ,1H ,1,2,
     *1HW,1HH,1HE,1HR,1HE,1H ,1,3,
     *1HO,1HN,1H ,1H ,1H ,1H ,1,4,
     *1HO,1HR,1HD,1HR,1HB,1HY,1,5,
     *1HD,1HE,1HL,1HE,1HT,1HE,1,6,
     *1HI,1HN,1HS,1HR,1HT,1HN,1,7,
     *1HT,1HO,1H ,1H ,1H ,1H ,1,8,
     *1HR,1HE,1HA,1HD,1H ,1H ,1,9,
     *1HN,1HU,1HM,1HR,1HI,1HC,1,10,
     *1HC,1HR,1HE,1HA,1HT,1HE,1,11,
     *1HA,1HL,1HF,1HN,1HU,1HM,1,12,
     *1HD,1HE,1HF,1HV,1HE,1HW,1,13,
     *1HS,1HU,1HS,1HP,1HN,1HD,1,14,
     *1HR,1HE,1HV,1HO,1HK,1HE,1,15,
     *1HM,1HO,1HD,1HI,1HF,1HY,1,16,
     *1HS,1HE,1HT,1H ,1H ,1H ,1,17,
     *1HG,1HR,1HA,1HN,1HT,1H ,1,18,
     *1HS,1HE,1HC,1HO,1HU,1HN,1,19,
     *1HT,1HA,1HK,1HI,1HN,1HG,1,20,
     *1HW,1HI,1HT,1HH,1H ,1H ,1,21,
     *1HA,1HL,1HP,1HH,1HA,1H ,1,22,
     *1HP,1HA,1HR,1HT,1H ,1H ,1,23,
     *1HA,1HN,1HD,1H ,1H ,1H ,1,24,
     *1HR,1HE,1HS,1HT,1HO,1HR,1,25,
     *1H=,1H ,1H ,1H ,1H ,1H ,2,26,
     *1H>,1H ,1H ,1H ,1H ,1H ,2,27,
     *1H=,1H<,1H ,1H ,1H ,1H ,2,28,
     *1H#,1H ,1H ,1H ,1H ,1H ,2,29,
     *1H*,1H ,1H ,1H ,1H ,1H ,2,30,
     *1H<,1H ,1H ,1H ,1H ,1H ,2,31,
     *1H(,1H ,1H ,1H ,1H ,1H ,2,32,
     *1H),1H ,1H ,1H ,1H ,1H ,2,33,
     *1H,,1H ,1H ,1H ,1H ,1H ,2,34,
     *2HUS,2HRC,2HD /
      DATA ICODF,JNAME,ISECU/8,2HSY,2HTA,2HX ,2H-7,2H56,2H19/
      K=0
      KM=0
      L=0
      MM=0
      ICC=0
      KA=0
      JM=0
      IBB=0
      IC=0
      M=0
      N=0
      IKEY=0
      IL=0
      BLANK=020040B
      IF(MFLG.EQ.1)GO TO 45
C -----***       THE QUERY IS READ                                 ***-----
45    WRITE(1,1)
1     FORMAT(1H ,'PLEASE DO NOT BREAK A FULL WORD BY A BLANK OR ANY',
     */,'SPECIAL CHARACTER   ***********************')
      WRITE (6,2)
2     FORMAT(1H1,//////,' If you want to continue,please give your
     *query.',/,'  If you want to terminate,please give $ in first
     *   column ')
      CALL SFILL(IBUF,1,256,BLANK)
3     READ(1,4)IBUF
4     FORMAT(256A1)
```

```
      WRITE(20,8)IBUF
 8    FORMAT(1H ,256A1)
 5    K=K+1
      IF(IBUF(1).EQ.1H$) GO TO 905
      IF(K.EQ.256) GO TO 75
      DO 10 I=1,40
 10   IBUF1(I)=1H
      DO 20 I=K,256
      IF(IBUF(I).NE.1H ) GO TO 30
 20   CONTINUE
 30   J=0
      J=J+1
      IF(IBUF(I) .EQ.1H$) GO TO 900
      IF(IBUF(I).EQ.1H[) GO TO 300
      IBUF1(J)=IBUF(I)
      DO 50 K=I+1,256
      IF(IBUF(K).EQ.1H$) GO TO 900
      IF(IBUF(K).EQ.1H )GO TO 150
      J=J+1
      IBUF1(J) = IBUF(K)
 50   CONTINUE
 75   IF (IBUF(K-1).EQ.1H$)GO TO 900
      WRITE(20,760)
 760  FORMAT(1H ,'INPUT ERROR')
      STOP
 99   WRITE(20,100) IERR
 100  FORMAT(1H ,'**************** FMGR ERROR ******************',I4)
      STOP
 C *****************************************************************
 C    THIS PART CHECKS FOR IDENTIFIERS,KEYWORDS & DELIMITERS
 C    CODE FOR KEYWORDS =1
 C    CODE FOR DELIMITERS=2
 C    CODE FOR IDENTIFIERS=4
 C ----**************************************************************---
  150 LEN=J
 C -----*** Keyword & delimiters are checked      ***---
      DO 160 IJ=1,34
      DO 155 JJ=1,J
      IF(IBUF1(JJ).NE.IARA(JJ,IJ)) GO TO 160
 155  CONTINUE
      IKEY=IKEY+1
      JJ=JJ-1
      IF(JJ.NE.LEN) GO TO 160
 C----------------------------------------------------------------------
 C    THIS PART WRITES CODE FOR KEYWORD & DELIMS IN ISTB,THE SYMBOL
 C    TABLE
 C---------------------------------------------------------------------
      IF(IKEY.NE.1) GO TO 156
      CALL CODE
      WRITE(ICODH,210) (IBUF1(LK),LK=1,6)
```

```
210   FORMAT(6A1)
      DO 211 IVAR=1,11
      IBEG=7+(IVAR-1)*8
      IEND=13+(IVAR-1)*8
      IF(JSCOM(ICODH,1,6,IBUF2,IBEG,IER)) 211,212,211
211   CONTINUE
      STOP
C -----*** A flag is set to either zero or one depending   ***
C        * whether user has got right to operate on data   *
C        *-----------            base or not        ---------------*
212   CALL SMOVE(IBUF2,IEND,IEND+1,ILVAR,1)
      CALL CODE
      READ(ILVAR,213)LIVAR
213   FORMAT(I2)
      IF(LIVAR.EQ.1) GO TO 154
      JFLAG=0
      GO TO 156
154   JFLAG=1
156   N=7
      MM=MM+1
      NN=1
      ISTB(MM,NN) = IARA(N,IJ)
      ISTB(MM,NN+1)=IARA(N+1,IJ)
      IF((ISTB(1,1).EQ.1).AND.((ISTB(1,2).EQ.15).OR.(ISTB(1,2).EQ.18).
     *OR.(ISTB(1,2).EQ.11)))GO TO 189
      GO TO 5
160   CONTINUE
      M=M+1
      IAA=M
      IF(M.NE.1) GO TO 190
C -----*** Entry of IDENTIFIERS are made in table ITAB1      ***
C        *     without any duplication of identifiers         *
165   DO 170 KK=1,20
170   ITAB1(M,KK)=IBUF1(KK)
      KK=KK-1
      IL=IL+1
      LL(IL)=KK
      ITAB1(M,21)=4
      ITAB1(M,22)=M
      NANA=M
C -----*** Entry in UNIFORM SYMBOL table ISTB is made   ***---
185   MM=MM+1
      NN=1
      ISTB(MM,NN)=ITAB1(M,21)
      ISTB(MM,NN+1)=ITAB1(M,22)
      M=IAA
      GO TO 5
189   KM=KM+1
      DO 191 IU=1,6
191   ISTB1(KM,IU)=IBUF1(IU)
```

```
      ISTB1(KM,IU)=ISTB(MM,NN)
      ISTB1(KM,IU+1)=ISTB(MM,NN+1)
      GO TO 5
190   IK=0
      DO 200 IC=1,M-1
      IK=IK+1
      KK=LL(IK)
      DO 195 IB=1,KK
      IF(IBUF1(IB).NE.ITAB1(IC,IB)) GO TO 200
195   CONTINUE
      IB=IB-1
      IF(IB.NE.LEN) GO TO 200
      M=IC
      KK=IB
      IAA=IAA-1
      GO TO 185
200   CONTINUE
      GO TO 165
300   DO 430 IR=I,256
      KA=KA+1
      IBUF1(KA)=IBUF(IR)
      GO TO 312
305   DO 310 JK=KA-1,40
310   IBUF1(JK)=1H
      LI=1
      GO TO 322
312   IF(KA.EQ.1) GO TO 430
      IF(IBUF1(KA).EQ.1H]) GO TO 315
      IF(IBUF1(KA).EQ.1H$) GO TO 380
      IF(IBUF1(KA).EQ.1H ) GO TO 400
      IBB=0
      GO TO 430
315   IF(IBUF1(KA-1).EQ.1H[) GO TO 390
      IF(IBUF1(KA-1).EQ.1H ) GO TO 305
      LI=0
      DO 320 LK=KA,40
320   IBUF1(LK)=1H
322   DO 325 KL=1,KA-2
325   IBUF1(KL)=IBUF1(KL+1)
      IBUF1(KA-1)=1H
      KA=KA-(2+LI)
      IF(JM.EQ.0) GO TO 350
C -----*** Check of LITERAL is done in literal table ITAB    ***
C          * to avoid duplication of literal entries          *
      DO 340 IA=1,JM
      DO 330 IB=1,40
      IF(ITAB(IA,IB).NE.IBUF1(IB)) GO TO 340
330   CONTINUE
      TEMP=ICC
      N=40
      ICC=ITAB(IA,N+2)
      MM=MM+1
      NN=1
      GO TO 360
340    CONTINUE
C*************************************************************************
C              LITERAL TABLE ENTRY
C              CODE FOR LITERAL=3
C*************************************************************************
350   MM=MM+1
      JM=JM+1
      ICC=ICC+1
      TEMP=ICC
      NN=1
      N=40
      DO 352 IVAX=1,40
352   ITAB(JM,IVAX)=1H
      DO 355 JJ=1,N
355    ITAB(JM,JJ)=IBUF1(JJ)
      ITAB(JM,N+1)=3
      ITAB(JM,N+2)=JM
C*************************************************************************
C              SYMBOL TABLE ENTRY
C************************************************************************/
360   ISTB(MM,NN)=3
      ISTB(MM,NN+1)=ICC
C*************************************************************************
      KA=0
      ICC=TEMP
      K=IR
      GO TO 5
380   WRITE(20,385)
385    FORMAT(1H ,'] IS MISSING FOR LITERAL &YOUR QUERY TERMINATED')
396   STOP
390   WRITE(20,395)
395   FORMAT(1H ,'[ & ] IS PROPER BUT NO LIT. IN BETWEEN')
      STOP
400   IF(IBUF1(KA-1).EQ.1H[) GO TO 410
      IBB=IBB+1
      IF(IBB.EQ.1) GO TO 430
410   KA=KA-1
430   CONTINUE
      K=IR-1
      GO TO 5
900   CALL EXEC(ICODF,JNAME)
905   END
C -----*****************************************************************
C        FROM THIS SEGMENT A BRANCH IS MADE TO THE APPROPRIATE
C        SEGMENT FOR SYNTAX CHECKING
C     ******************************************************************
```

```
      PROGRAM SYTAX,5
      DIMENSION ISTB(50,2),IPRS(50,3),ITAB1(30,22),ITAB(30,42)
     *,LNAM1(3),LNAM2(3),LNAM3(3),LNAM4(3),LNAM5(3),LNAM6(3),LNAM7(3),
     *LNAM8(3),LNAM9(3),MNAM1(3)
      COMMON ITAB,ITAB1,ISTB,MM,MT,ML,IPRS,IQRY,IF
C -----***        NAMES OF SEGMENTS CALLED BY SYTAX ARE GIVEN    ***----
C        *
      DATA LNAM1,ICODF/2HCH,2HEC,2HK ,8/
      DATA LNAM2/2HDE,2HLT,2H  /
      DATA LNAM3/2HIN,2HSR,2H  /
      DATA LNAM4/2HKR,2HEA,2HT /
      DATA LNAM5/2HMO,2HDF,2HY /
      DATA LNAM6/2HGR,2HAN,2HT /
      DATA LNAM7/2HSS,2HPN,2HD /
      DATA LNAM8/2HRE,2HST,2HR /
      DATA LNAM9/2HRE,2HVO,2HK /
      DATA MNAM1/2HDE,2HFV,2HW /
      IF(ISTB(1,1).NE.1) GO TO 10
C -----****     CHECK IS MADE TO BRANCH TO APPROPRIATE SEGMENTS  ***-----
      IF((ISTB(1,2).EQ.1).OR.(ISTB(1,2).EQ.19)) GO TO 30
      IF(ISTB(1,2).EQ.11) GO TO 50
      IF(ISTB(1,2).EQ.6) GO TO 40
      IF(ISTB(1,2).EQ.7) GO TO 45
      IF(ISTB(1,2).EQ.16) GO TO 55
      IF(ISTB(1,2).EQ.18) GO TO 60
      IF(ISTB(1,2).EQ.14) GO TO 65
      IF(ISTB(1,2).EQ.15) GO TO 70
      IF(ISTB(1,2).EQ.13) GO TO 75
      IF(ISTB(1,2).EQ.25) GO TO 68
10    WRITE(1,2)
2     FORMAT(1H ,'SYNTAX ERROR,CHECK FIRST WORD OF YOUR QUERY ')
      STOP
C ----------      THE APPROPRIATE SEGMENT IS CALLED      ---------------
30    CALL EXEC(ICODF,LNAM1)
40    CALL EXEC(ICODF,LNAM2)
 45   CALL EXEC(ICODF,LNAM3)
50    CALL EXEC(ICODF,LNAM4)
55    CALL EXEC(ICODF,LNAM5)
 60   CALL EXEC(ICODF,LNAM6)
65    CALL EXEC(ICODF,LNAM7)
68    CALL EXEC(ICODF,LNAM8)
 70   CALL EXEC(ICODF,LNAM9)
75    CALL EXEC(ICODF,MNAM1)
      END
C***********************************************************************
C             THIS SEGMENT IS CALLED BY SEGMENT SYTAX
C***********************************************************************
      PROGRAM CHECK,5,,,,,,,, THIS IS THE PARSER FOR QUERY FACILITIES
      DIMENSION IA(50,2),IB(50,3),IE(7,3),JNAM1(3),ITAB1(30,22),ITAB(30
     *,42),KINM(3)
```

```
      COMMON ITAB,ITAB1,IA,MM,MT,ML,IB,IQRY,IF
      DATA JNAM1,KINM,ICODF/2HQR,2HYA,2H  ,2HCL,2HEA,2HR ,8/
      M1=1
      N1=1
      IX1=0
C----------   IB IS THE PARSED QUERY TABLE GENERATED    ------------------
C----------   ZERO OR ONE IS ENTERED IN IB DEPENDING ON  ------------------
C             WHETHER EACH WORD IN THE QUERY IS SYNTACTICALLY ----------
C             CORRECT OR NOT                ------------------------------
C   IX1 KEEPS  COUNT OF (1,2) IN ISTB
      IY1=0
C   IY1 KEEPS COUNT OF (1,3) IN ISTB
      IP1=0
C   IP1 KEEPS COUNT OF (2,34) IN ISTB
      IQ1=0
C   IQ1 KEEPS COUNT OF (2,26) IN ISTB
      IR1=0
C IR1 KEEPS COUNT OF (4,)AFTER (1,3) IN ISTB
      IZ1=0
C   IZ1=1 TELLS WHEN QUERY ENDS OR WHEN LITERAL COMES
      IS1=0
C   IS1 KEEPS COUNT OF (1,5) IN ISTB
      IR=0
C   IR KEEPS  COUNT OF (4,) AFTER (1,2) IN ISTB
      IQRY=0
C   IQRY TELLS THE TYPE OF SELECT QUERY
      IF=0
C   IF TELLS THE NO OF ATTRIBUTES AFTER SELECT
      DO 15 I=1,MM
      DO 15 J=1,2
      IB(I,J)=IA(I,J)
15    CONTINUE
      IF(IB(1,2).EQ.19) IQRY=4
10    IB(M1,3)=0
20    M1=M1+1
      IF(M1.GT.MM) GO TO 75
      IF(M1.NE.2) GO TO 40
C-----------   FIRST ID AFTER FIRST KEYWORD IS CHECKED   ---------------
      IF(IA(M1,N1).NE.4) GO TO 11
      IF=IF+1
      GO TO 10
11    WRITE(20,21)
30    IB(M1,3)=1
      GO TO 20
C-----------   CHECK FOR KEYWORD 'PART' IN THE QUERY    ------------------
40    IF((IY1.EQ.1).AND.(IR1.EQ.0).AND.(IA(M1,N1).EQ.1)) GO TO 130
C------------  SYNTAX CHECK AFTER KEYWORD 'WHERE'       ------------------
      IF(IY1.EQ.1) GO TO 95
C-----------   SYNTAX CHECK AFTER KEYWORD ' FROM'       ------------------
      IF((IX1.EQ.1).AND.(IA(M1,N1).EQ.4)) GO TO 25
```

```
C-----------    CHECK FOR IDENTIFIERS IN THE QUERY       --------------
      IF(IA(M1,N1).EQ.4) GO TO 46
45    IF(IA(M1,N1).NE.1) GO TO 70
      IF(IZ1.EQ.1) GO TO 115
      IF(IA(M1,N1+1).NE.2) GO TO 50
      IX1=IX1+1
      IF(IX1.GT.1) GO TO 30
      GO TO 10
46    IF=IF+1
      GO TO 10
25    IR=IR+1
      IF(IR.EQ.1)GO TO 41
      IF(IR.EQ.2)GO TO 42
      WRITE(20,32)
      GO TO 30
41    IF(IQRY.EQ.4) GO TO 10
      IQRY=1
      GO TO 10
42    IQRY=2
      GO TO 10
50    IF((IX1.EQ.1).AND.(IA(M1,N1+1).EQ.3).AND.(IR.GT.0)) GO TO 60
      WRITE(20,22)
      GO TO 30
60    IY1=IY1+1
      IF(IY1.GT.1) GO TO 30
      GO TO 10
70    IF((IY1.EQ.1).AND.(IA(M1,N1).EQ.2)) GO TO 80
      IF((IQ1.EQ.1).AND.(IA(M1,N1).EQ.3)) GO TO 110
75    IF((IZ1.EQ.1).AND.(M1.GT.MM)) GO TO 200
      IF((IZ1.EQ.2).AND.(M1.GT.MM)) GO TO 200
      IF((IZ1.EQ.1).AND.(IA(M1,N1).EQ.1)) GO TO 115
      IF(M1.GT.MM) GO TO 120
      WRITE(20,23)
      GO TO 30
C-----------    CHECK FOR '=' AFTER 'WHERE' IN QUERY     --------------
80    IF(IA(M1,N1+1).EQ.26) GO TO 90
C-------------    CHECK FOR ',' AFTER 'WHERE' IN QUERY   --------------
      IF(IA(M1,N1+1).EQ.34) GO TO 100
      WRITE(20,24)
      GO TO 30
90    IF((IR1.EQ.1).AND.(IQ1.EQ.0)) GO TO 85
      IF((IR1.EQ.2).AND.(IP1.EQ.1)) GO TO 85
      WRITE(20,23)
      GO TO 30
85    IQ1=IQ1+1
      IF(IQ1.EQ.1)GO TO 10
      WRITE(20,26)
      GO TO 30
95    IF(IA(M1,N1).NE.4) GO TO 45
      IR1=IR1+1
```

```
C-------------    CHECK THE 'WHERE CLAUSE' IN QUERY    ---------------
        IF(IR1.EQ.1) GO TO 10
        IF((IR1.EQ.2).AND.(IS1.EQ.1)) GO TO 110
        IF((IR1.EQ.3).AND.(IQ1.EQ.1)) GO TO 10
        IF((IR1.EQ.4).AND.(IP1.EQ.2)) GO TO 110
        IF((IR1.EQ.2).AND.(IP1.EQ.1)) GO TO 10
100     IF(IR1.EQ.1) GO TO 105
        IF((IR1.EQ.3).AND.(IQ1.EQ.1)) GO TO 105
        WRITE(20,27)
        GO TO 30
105     IP1=IP1+1
        GO TO 10
115     IF(IA(M1,N1+1).NE.5) GO TO 31
        IS1=IS1+1
        IQRY=3
        GO TO 10
31      WRITE(20,29)
        GO TO 30
110     IZ1=IZ1+1
        IF((IZ1.EQ.1).AND.(IP1.EQ.0)) GO TO 10
        IF((IZ1.EQ.1).AND.(IP1.EQ.2)) GO TO 10
        IF((IZ1.EQ.2).AND.(IR1.EQ.2)) GO TO 10
        WRITE(20,28)
        GO TO 30
130     IF(IA(M1,N1+1).EQ.23) GO TO 140
        WRITE(20,23)
        GO TO 30
140     IF((MM-M1).EQ.7) GO TO 141
        IF((MM-M1).EQ.3) GO TO 142
        GO TO 125
141     IQRY=5
        GO TO 126
142     IQRY=6
126     IB(M1,3)=0
        K1=M1
        DO 145 I=1,MM-K1
        M1=M1+1
        DO 145 J=1,2
145     IE(I,J)=IA(M1,J)
C-------     SUBROUTINE 'PRT' IS CALLED WHEN THE QUERY HAS 'PART'
C            IN ITS WHERE CLAUSE                                 ---------------
        CALL PRT(IE,MM-K1,3)
        M1=K1
        DO 150 I=1,MM-K1
        M1=M1+1
        IB(M1,3)=IE(I,3)
150     CONTINUE
        GO TO 200
21      FORMAT(1H ,'Identifier is missing after 1st KEYWORD')
22      FORMAT(1H ,'Wrong Keyword after 1st Keyword  OR  Identifier
       *missing OR Wrong Keyword after 2nd Keyword')
 23     FORMAT(1H ,'Either Identifier missing after 3rd Keyword OR
       *'' ='' is missing')
 24     FORMAT(1H ,'Either '' =''is  missing OR '','' is missing')
 29     FORMAT(1H ,'Correct Keyword missing')
 26     FORMAT(1H ,' ''=''  occurs more than once')
 27     FORMAT(1H ,'Identifier missing after'' = '' OR '','' is
       *missing after ''='' OR '','' after ''='' is in wrong place')
 28     FORMAT(1H ,'Literal in wrong place')
 32     FORMAT(1H ,'Wrong Number Of Identifiers After Keyword ''FROM'' ')
 120    WRITE(20,5)
 5      FORMAT(1H ,'QUERY NOT PROPERLY ENDED ')
        GO TO 200
 125    WRITE(20,6)
 6      FORMAT(1H ,'Query Not Proper.Check Last 7 Words Of Your Query')
 C--------------    THE PARSER TABLE IS CHECKED FOR WRONG SYNTAX  -------
 200    DO 210 IP=1,MM
        IF(IB(IP,3).EQ.1) GO TO 991
 210    CONTINUE
 C-----------      IF NO SYNTAX ERROR OCCURS, SEGMENT QRYA IS CALLED   ----
        CALL EXEC(ICODF,JNAM1)
 991    WRITE(20,995)
 995    FORMAT(1H ,'Sorry, Your Query is Wrong ')
        CALL EXEC (ICODF,KINM)
        END
 C  THIS IS SUBROUTINE 'PRT' CALLED BY SEGMENT 'CHECK'
        SUBROUTINE PRT (IX,M,N)
        DIMENSION IX(7,3)
        M1=0
        N1=1
 20     M1=M1+1
        IF(M1.GT.M) GO TO 100
        IF(M.EQ.7) GO TO 1
        GO TO 2
 1      GO TO(5,15,5,25,35,15,35),M1
 2      GO TO (5,25,35),M1
 5      IF(IX(M1,N1).EQ.4) GO TO 10
        WRITE(20,41)
        GO TO 30
 15     IF(IX(M1,N1).NE.1) GO TO 11
        IF(IX(M1,N1+1).EQ.24) GO TO 10
        WRITE(20,42)
        GO TO 30
 25     IF(IX(M1,N1).NE.2) GO TO 12
        IF(IX(M1,N1+1).EQ.26) GO TO 10
        WRITE(20,43)
        GO TO 30
 35     IF(IX(M1,N1).EQ.3) GO TO 10
        WRITE(20,44)
        GO TO 30
 10   IX(M1,3)=0
      GO TO 20
 11   WRITE(20,42)
      GO TO 30
 12   WRITE(20,43)
 30   IX(M1,3)=1
      GO TO 20
 41   FORMAT(1H ,'Identifier missing in query after Keyword  PART ')
 42   FORMAT(1H ,'Keyword ''AND'' missing')
 43   FORMAT(1H ,' ''='' is missing')
 44   FORMAT(1H ,' Literal missing')
 100  RETURN
      END
C     ************************************************************
C -----*** New segment starts here to create a new file        ***
C     ************************************************************
      PROGRAM KREAT,5
      DIMENSION IA(50,2),IB(50,3),ITAB1(30,22),ITAB(30,42),INAM(3),
     *ISTB1(20,8),KJNM(3)
      COMMON ITAB,ITAB1,IA,MM,M,JM,IB,IQRY,IF,ISTB1
      DATA ICODF,INAM,KJNM/8,2HKR,2HET,2H  ,2HCL,2HEA,2HR /
      ID=0
C  ID KEEPS COUNT OF IDENTIFIER AFTER ( IN ISTB
      IK=0
C  IK KEEPS COUNT OF KEYWORD AFTER ( IN ISTB
      IL=0
C   IL KEEPS COUNT OF LITERAL AFTER ( IN ISTB
C-----------    IB IS THE PARSED QUERY TABLE        ------------------
      DO 5 I=1,MM
      DO 5 J=1,2
 5    IB(I,J)=IA(I,J)
      L=1
      K=1
 10   IB(L,3)=0
 20   L=L+1
      IF(L.EQ.MM) GO TO 80
      IF(L.GT.MM) GO TO 100
      IF(L.NE.2) GO TO 40
C-----------    CHECK 2nd WORD OF QUERY TO BE AN IDENTIFIER   --------
C-----------    ENTRY I N IB FOR SYNTACTICALLY CORRECT WORD IS 0,
C               AND FOR A WRONG WORD IS 1            ------------------
      IF(IA(L,K).EQ.4) GO TO 10
      WRITE(20,51)
      GO TO 30
C-----------    CHECK FOR '(' AT THE 3rd QUERY WORD        -----------
 40   IF(L.NE.3) GO TO 50
      IF(IA(L,K).EQ.2) GO TO 45
      WRITE(20,52)
      GO TO 30
 45   IF(IA(L,K+1).EQ.32) GO TO 10
```

```
      WRITE(20,52)
      GO TO 30
C-----------    CHECK FOR IDENTIFIER AFTER '(' IN QUERY    ----------------
50    IF((IA(L,K).EQ.4).AND.(ID.EQ.0).AND.(IL.EQ.0)) GO TO 60
      IF((IA(L,K).EQ.1).AND.(ID.EQ.1)) GO TO 55
C-----------    CHECK FOR LITERAL AFTER '(' IN QUERY       ----------------
      IF((IA(L,K).EQ.3).AND.(IK.EQ.1)) GO TO 70
      WRITE(20,54)
      GO TO 30
C-----------    CHECK FOR THE 'TYPE' OF THE ATTRIBUTE      ----------------
55    IF((IA(L,K+1).EQ.10).OR.(IA(L,K+1).EQ.12).OR.(IA(L,K+1).EQ.22))
     *GO TO 65
      WRITE(20,53)
      GO TO 30
60    ID=ID+1
      GO TO 10
65    IK=IK+1
      GO TO 10
70    IL=IL+1
      IF(IL.EQ.1) GO TO 10
      IL=0
      ID=0
      IK=0
      GO TO 10
C-----------    CHECK FOR ')' AT THE END OF QUERY          ----------------
80    IF(IA(L,K).EQ.2) GO TO 85
      WRITE(20,59)
30    IB(L,3)=1
      GO TO 20
85    IF(IA(L,K+1).EQ.33) GO TO 10
      WRITE(20,59)
      GO TO 30
51    FORMAT(1H ,'identifier missing')
52    FORMAT(1H ,'Delimiter ( is missing')
53    FORMAT(1H ,'Wrong Keyword')
54    FORMAT(1H ,'Error after ( in your query')
59    FORMAT(1H ,' '')'' is missing at the end of your query')
100   DO 110 LK = 1,MM
      IF(IB(LK,3).EQ.1) GO TO 150
110   CONTINUE
C-----------    IF QUERY IS FOUND TO BE SYNTACTICALLY CORRECT,
C               THE SEGMENT 'KRET'IS CALLED                ----------------
      CALL EXEC(ICODF,INAM)
150   WRITE(20,155)
155   FORMAT(1H ,'Sorry, Your Query is Wrong')
      CALL EXEC(ICODF,KJNM)
      END
C -----*** New segment starts here to syntax check of INSRTN query ***
C        * ,which will help to insert new tuple in data file       *
      PROGRAM INSR,5
      DIMENSION IA(50,2),IB(50,3),ITAB1(30,22),ITAB(30,42),INAM(3),
     *LRNAM(3)
      COMMON  ITAB,ITAB1,IA,L,M,JM
      DATA ICODF,INAM,LRNAM/8,2HIN,2HST,2HN ,2HCL,2HEA,2HR /
      DO 8 I=1,L
      DO 8 J=1,2
 8    IB(I,J)=IA(I,J)
C-----***********************************************************
C ----*  VARIABLE ID IS USED TO DENOTE IDENTIFIER IN STATEMENT *
C ----*  VARIABLE LIT IS USED TO DENOTE LITERAL IN STATEMENT   *
C     *  VARIABLE IF  IS USED TO KEEP ACOUNT OF PROPER PLACE
C     *  FOR IDENTIFIER & LITERAL  IN QUERY STATEMENT          *
C     ***********************************************************
      ID=0
      LIT=0
      IF=0
      DO 140 I=1,L
      IF(I.GE.4) GO TO 36
      GO TO(10,20,30),I
C ----*********************************************************
C     * NEXT STATEMENT CHECKS FOR FIRST QUERY WORD INSRTN IN    *
C     *************   QUERY STATEMENT   *************************
 10   IF((IB(I,1).EQ.1).AND.(IB(I,2).EQ.7)) GO TO 15
 12   IB(I,3)=1
      GO TO 140
 15   IB(I,3)=0
      GO TO 140
C ----**** NEXT STATEMENT CHECKS IDENTIFIER IN QUERY STATEMENT  *****
 20   IF(IB(I,1).EQ.4) GO TO 15
      GO TO 12
 30   IF(IB(I,1).EQ.4) GO TO 35
      GO TO 12
 35   ID=ID+1
      GO TO 15
 36   IF(IB(I,1).EQ.4) GO TO 40
C ----*  LITERAL is checked in next statement  *
      IF(IB(I,1).EQ.3) GO TO 60
      GO TO 12
 40   ID=ID+1
C ----* Check is made in next five statements  *
C ----* for proper place of IDENTIFIER & LITERAL in query statement  **
      IF((ID.GT.LIT).AND.(I.EQ.L)) GO TO 48
      IF((ID.EQ.LIT).AND.(I.EQ.L).AND.(IF.EQ.0)) GO TO 50
      IF((ID.EQ.LIT).AND.(I.LT.L)) GO TO 51
      IF((ID.GT.LIT).AND.(I.LT.L).AND.(IF.EQ.0)) GO TO 15
      IF((ID.LT.LIT).AND.(I.EQ.L)) GO TO 48
      GO TO 12
 48   IB(I,3)=1
      WRITE(1,151)
      GO TO 170
```

```
   51 IF=IF+1
      GO TO 12
   50 IB(I,3)=0
      GO TO 169
   60 LIT=LIT+1
       IF((LIT.EQ.ID).AND.(I.EQ.L).AND.(IF.EQ.0)) GO TO 50
      IF((LIT.LT.ID).AND.(I.LT.L)) GO TO 15
      IF((LIT.LT.ID).AND.(I.EQ.L)) GO TO 48
      IF((LIT.GT.ID).AND.(I.LE.L)) GO TO 48
      IF((LIT.EQ.ID).AND.(I.LT.L)) GO TO 51
      IB(I,3)=1
  140 CONTINUE
C ----* Flags affected during syntax checking are checked  *
C     * for error possibility *****************************
  170 DO 149 I=1,L
      IF(IB(I,3).NE.1) GO TO 149
      GO TO 148
  149 CONTINUE
  169 WRITE(1,157)
      GO TO 162
  148 IF(IB(1,3).EQ.1) WRITE(1,153)
C ----* Error is checked & its diagnosis is given *
      IF(IB(2,3).EQ.1) WRITE(1,154)
      IF(IB(3,3).EQ.1) WRITE(1,155)
      IF(IF.GT.0) WRITE(1,152)
      CALL EXEC( ICODF,LRNAM)
  151 FORMAT(1H ,'No correspondence between Literal & Idetifier')
  152 FORMAT(1H ,'Literal has come before end of Idetifier in Query')
  153 FORMAT(1H ,'At your 1st Query Word no keyWord INSRTN ')
  154 FORMAT(1H ,'At your 2nd Query Word no Idetifier')
  155 FORMAT(1H ,'At your 3rd Query Word no Idetifier')
  157 FORMAT(1H ,'Your query is syntactically correct')
C ----*  Corresponding semantics routine is called provided query is cor
  162 CALL EXEC(ICODF,INAM)
      END
C ----************************************************************
C ----*** This segmnent checks the query statement to REVOKE the ***----
C     *** granted OPTIONS viz. TO MODIFY , TO DELETE , TO INSRTN ***
C     *** from authorised users                                  ***
C     ************************************************************
      PROGRAM REVOK,5
      DIMENSION IA(50,2),IB(50,3),ITAB1(30,22),ITAB(30,42),INAM(3)
     *,LRNAM(3)
      COMMON ITAB,ITAB1,IA,M,M1,JM,IB
      DATA ICODF,INAM,LRNAM/8,2HGR,2HNT,2HT ,2HCL,2HEA,2HR /
C     **************************************************************
C ----* INS keeps acount of key word INSRTN                          *
C     * IDEL keeps acount of key word DELETE                          *
C     * MODFY keeps acount of key word MODIFY                          *
C     ****************************************************************
```

```
      INS=0
      IDEL=0
      MODFY=0
C     ******************************************************************
C -----* Uniform symbol table from two dimensional array  IA is copied
C     * to array IB in next three statements
C     ******************************************************************
      DO 8 I=1,M
      DO 8 J=1,2
8     IB(I,J)=IA(I,J)
      DO 140 I=1,M
      IF(I.GE.3) GO TO 30
      GO TO (10,20),I
10    IF((IB(I,1).EQ.1).AND.(IB(I,2).EQ.15)) GO TO 15
12    IB(I,3)=1
      GO TO 140
15    IB(I,3)=0
      GO TO 140
C -----*** IN next two statements option rights which will be    ***
C     *** revoked from users are checked                          ***
20    IF(((IB(I,1).EQ.1).AND.(IB(I,2).EQ.7))
     *.OR.((IB(I,1).EQ.1).AND.(IB(I,2).EQ.6)).OR.((IB(
     *I,1).EQ.1).AND.(IB(I,2).EQ.16))) GO TO 25
      GO TO 12
30    IF(((IB(I,1).EQ.1).AND.(IB(I,2).EQ.7)).OR.
     *((IB(I,1).EQ.1).AND.(IB(I,2).EQ.6)).OR.((IB(I,1).
     *EQ.1).AND.(IB(I,2).EQ.16))) GO TO 31
C ----*** Key word ON is checked         ****
      IF((IB(I,1).EQ.1).AND.(IB(I,2).EQ.4)) GO TO 33
      IF(I.GT.3) GO TO 35
      GO TO 12
C -----*** In next statements repeatition of options are checked ***
25    IF(IB(I,2).EQ.6) GO TO 40
      IF(IB(I,2).EQ.7) GO TO 42
      IF(IB(I,2).EQ.16) GO TO 50
      GO TO 12
31    IF(IB(I-1,1).EQ.1) GO TO 25
      GO TO 12
33    IF((IB(I-1,1).EQ.1).AND.(IB(I+1,1).EQ.4)) GO TO 15
      GO TO 12
C -----*** Occurence of IDENTIFIER , KEYWORD & LITERAL are cheked ***
C       * for their proper place in query statement               *
35    IF((IB(I,1).EQ.4).AND.((IB(I-1,1).EQ.1).AND.(IB(I-1,2).EQ.4)).AND.
     *((IB(I+1,1).EQ.1).AND.(IB(I+1,2).EQ.2))) GO TO 15
      IF((IB(I,1).EQ.1).AND.(IB(I,2).EQ.2)) GO TO 36
      IF((IB(I,1).EQ.3).AND.((IB(I-1,1).EQ.1).AND.(IB(I-1,2).EQ.2)).O
     *R.(IB(I,1).EQ.3)) GO TO 15
      GO TO 12
36    IF((IB(I-1,1).EQ.4).AND.(IB(I+1,1).EQ.3)) GO TO 15
      GO TO 12
```

```
40     IDEL=IDEL+1
       IF(IDEL.GT.1) GO TO 12
       GO TO 15
42     INS=INS+1
       IF(INS.GT.1) GO TO 12
       GO TO 15
50     MOD=MOD+1
       IF(MOD.GT.1) GO TO 12
       GO TO 15
140    CONTINUE
       DO 141 I=1,M
       IF(IB(I,3).NE.1) GO TO 141
       GO TO 142
141    CONTINUE
       IF((INS.OR.IDEL.OR.MODFY).GT.1) GO TO 144
       WRITE(1,160)
       GO TO 143
C -----*** In next five statements error diagnosis is given ***
144    WRITE(1,166)
142    IF(IB(1,3).EQ.1) WRITE(1,162)
       IF(IB(2,3).EQ.1) WRITE(1,163)
       IF(IB(3,3).EQ.1) WRITE(1,164)
       IF(IB(4,3).EQ.1) WRITE(1,165)
       CALL EXEC( ICODF,LRNAM)
160    FORMAT(1H ,’Your Query is Syntactically correct’)
162    FORMAT(1H ,’At your 1st Query Word no proper keyWord REVOKE ’)
163    FORMAT(1H ,’At your 2nd Query Word no proper keyWord  DELETE
      * INSRTN MODIFY ’)
164    FORMAT(1H ,’Either  MODIFY INSRTN DELETE or ON MISSINGAT 3RD’)
165    FORMAT(1H ,’Either kyword MODIFY INSRTN DELETE or ON or
      *Idetier is missing at your 4th Query word ’)
166    FORMAT(1H ,’Repeated keyword INSRTN MODFY DELETE  in Query
      *  So your query is wrong’)
C -----*** In case correct query corresponding semantic routine is called
143    CALL EXEC(ICODF,INAM)
       END
C      **************************************************
C -----*   This segment checks syntax of query statement *
C      *   SUSPND . This query is to suspend the relation*
C      *  name ,so that any user can’t operate on it.    *
C      **************************************************
       PROGRAM SSPND,5
       DIMENSION IA(50,2),IB(50,3),ITAB1(30,22),ITAB(30,42),INAM(3)
      *,LRNAM(3)
       COMMON  ITAB,ITAB1,IA,M,M1,JM,IB
       DATA ICODF,INAM,LRNAM/8,2HRE,2HST,2HT ,2HCL,2HEA,2HR /
       DO 10 I=1,M
       DO 10 J=1,2
10     IB(I,J)=IA(I,J)
       DO 60 I=1,M
      IF(I.GT.2) GO TO 45
      GO TO (30,40),I
C -----* FIRST WORD OF QUERY STATEMENT IS CHECKED  *
30    IF((IB(I,1).EQ.1).AND.(IB(I,2).EQ.14)) GO TO 35
32    IB(I,3)=1
      GO TO 60
35    IB(I,3)=0
      GO TO 60
C -----* IDENTIFIER IS CHECKED IN NEXT STATEMENT *
40    IF(IB(I,1).EQ.4) GO TO 35
      GO TO 32
 45   WRITE(1,85)
      GO TO 68
60    CONTINUE
      DO 65 I=1,M
      IF(IB(I,3).NE.1) GO TO 65
      GO TO 66
65    CONTINUE
      WRITE(1,70)
      GO TO 90
C -----*    IN NEXT TWO STATEMENTS ERROR DIAGNOSIS IS MENTIONED *
66    IF(IB(1,3).EQ.1) WRITE(1,75)
      IF(IB(2,3).EQ.1) WRITE(1,80)
68    CALL EXEC(ICODF,LRNAM)
70    FORMAT(1H ,'Query is Syntactically Correct')
75    FORMAT(1H ,'At your 1st Query Word no correct keyWord SSPEND ')
80    FORMAT(1H ,'At your 2nd Query Word no Idetifier ')
85    FORMAT(1H ,'You are giving more than required Query ')
C     *******************************************
C ------* IN CASE OF CORRECT QUERY CORRESPONDING *
C     * SEMANTICS ROUTINE IS CALLED             *
C     *******************************************
90    CALL EXEC(ICODF,INAM)
      END
C -----* This segment checks syntax of query statement whose first query
C      ************    is GRANT     **************************************
      PROGRAM GRANT,5
      DIMENSION IA(50,2),IB(50,3),ITAB1(30,22),ITAB(30,42),INAM(3)
     *,LRNAM(3)
      COMMON ITAB,ITAB1,IA,L,M1,JM,IB
      DATA ICODF,INAM,LRNAM/8,2HGR,2HNT,2HT ,2HCL,2HEA,2HR /
C     *******************************************************************
C -----*  INS KEEPS ACOUNT OF GRANT OPTION INSRTN                       *
C     *  MOD keeps acount of grant option MODIFY                        *
C     *  IDEL keeps acount of grant option DELETE                       *
C     *  ITO  keeps acount of keyword TO                                *
C     *  ION  keeps acount of keyword ON                                *
C     *******************************************************************
      INS=0
      IDEL=0
      MOD=0
      ITO=0
      ION=0
      DO 8 I=1,L
      DO 8 J=1,2
8     IB(I,J)=IA(I,J)
      DO 140 I=1,L
C   *****     first keyword GRANT in query statement is checked    *****
      IF((IB(I,1).EQ.1).AND.(IB(I,2).EQ.18)) GO TO 90
C  ***grant OPTIONS viz. MODIFY DELETE INSRTN are checked in next statem
      IF((IB(I,1).EQ.1).AND.((IB(I,2).EQ.7).OR.(IB(I,2).EQ.6)
     *.OR.(IB(I,2).EQ.16))) GO TO 30
C ---*****   keyword " ON " is checked        *******
      IF((IB(I,1).EQ.1).AND.(IB(I,2).EQ.4)) GO TO 45
C -----****  IDENTIFIER is   checked in next statement  ****
      IF((IB(I,1).EQ.4).AND.(I.GT.3).AND.((IB(I-1,1).EQ.1).AND.(IB(I-1,2
     *).EQ.4)).AND.((IB(I+1,1).EQ.1).AND.(IB(I+1,2).EQ.8))) GO TO 90
C ----***** keyword " TO ", ' ON ' & literal are checked  ***---
      IF((IB(I,1).EQ.1).AND.(IB(I,2).EQ.8)) GO TO 50
      IF(IB(I,1).EQ.3) GO TO 60
      IF(((IB(I,1).EQ.1).AND.(IB(I,2).EQ.8)).AND.(IB(I-1,1).EQ.4).AND.(
     *IB(I+1,1).EQ.3)) GO TO 90
      GO TO 85
30    IF(IB(I,2).EQ.6) GO TO 35
      IF(IB(I,2).EQ.7) GO TO 36
      IF(IB(I,2).EQ.16) GO TO 40
      GO TO 85
35    IDEL=IDEL+1
      IF(IDEL.GT.1) GO TO 85
      GO TO 90
36    INS=INS+1
      IF(INS.GT.1) GO TO 85
      GO TO 90
40    MOD=MOD+1
      IF(MOD.GT.1) GO TO 85
      GO TO 90
45    ION=ION+1
      IF((IB(I+1,1).EQ.4).AND.(ION.EQ.1)) GO TO 90
      IC=1
      GO TO 85
50    ITO=ITO+1
      IF((IB(I-1,1).EQ.4).AND.((IB(I-2,1).EQ.1).AND.(IB(I-2,2).EQ.4))
     *.AND.(IB(I+1,1).EQ.3).AND.(ITO.EQ.1)) GO TO 90
      ID=1
      GO TO 85
60    IF((IB(I-1,1).EQ.3).OR.((IB(I-1,1).EQ.1).AND.(IB(I-1,2).EQ.8))
     *.AND.(IB(I-2,1).EQ.4)) GO TO 90
85    IB(I,3)=1
      GO TO 140
90    IB(I,3)=0
```

```
 140  CONTINUE
      DO 145 I=1,L
C---* Flags affected in syntax checking are checked for presence of erro
      IF(IB(I,3).EQ.0) GO TO 145
      GO TO 148
 145  CONTINUE
      WRITE(1,161)
      GO TO 150
C -----*** error diagnosis is given below *********
 148  IF((IDEL.GT.1).OR.(INS.GT.1).OR.(MOD.GT.1)) WRITE(1,162)
      IF((ION.GT.1).OR.(ITO.GT.1).OR.(IC.EQ.1).OR.(ID.EQ.1))WRITE(1,163)
      IF(IB(1,3).EQ.1) WRITE(1,165)
      IF(IB(2,3).EQ.1) WRITE(1,166)
      CALL EXEC(ICODF,LRNAM)
 161  FORMAT('Query is correct')
 162  FORMAT('Either of INSRTN  MODIFY DELETE is Repeated')
 163  FORMAT('Either TO or ON is repeated which is wrong')
 165  FORMAT('At your 1st Query Word GRANT is missing')
 166  FORMAT('either INSRTN or DELETE or MODIFY  missing at 2nd')
C -----* Corresponding semantic routine is called provided query is corre
  150 CALL EXEC(ICODF,INAM)
      END
C     ***************************************************************
C -----*    This segment checks syntaxof query statement whose first
C     *     query word is ' DELETE '
C     ***************************************************************
      PROGRAM DELT,5
      DIMENSION IA(50,2),IB(50,3),ITAB1(30,22),ITAB(30,42),JNAM2(3)
     *,LRNAM(3)
      COMMON ITAB,ITAB1,IA,M,L,JM,IB
      DATA ICODF,JNAM2,LRNAM /8,2HUP,2HDA,2HT ,2HCL,2HEA,2HR /
      DO 5 I=1,M
      DO 5 J=1,2
 5    IB(I,J)=IA(I,J)
      DO 100 I=1,M
      IF(I.GE.4) GO TO 40
      GO TO (10,20,30),I
C -----**** first query word   DELETE is checked        ******
 10   IF((IB(I,1).EQ.1).AND.(IB(I,2).EQ.6)) GO TO 15
 12   IB(I,3)=1
      GO TO 100
 15   IB(I,3)=0
      GO TO 100
C -----*  2nd query is checked for IDENTIFIER   *****
 20   IF(IB(I,1).EQ.4) GO TO 15
      GO TO 12
C -----*  3rd Query word is checked for Keyword WHERE   ****
 30   IF((IB(I,1).EQ.1).AND.(IB(I,2).EQ.3)) GO TO 15
      GO TO 12
C -----* WHERE clause is checked in next four statement ****
```

```
40    IF((IB(I,1).EQ.4).AND.(IB(I-1,1).EQ.1).AND.((IB(I+1,1).EQ.2).AND.
     *(IB(I+1,2).EQ.26))) GO TO 15
      IF(((IB(I,1).EQ.2).AND.(IB(I,2).EQ.26)).AND.(IB(I-1,1).EQ.4).AND.
     *(IB(I+1,1).EQ.3)) GO TO 15
      IF((IB(I,1).EQ.3).AND.((IB(I-1,1).EQ.2).AND.(IB(I-1,2).EQ.26)))
     *GO TO 15
      IF(((IB(I,1).EQ.1).AND.(IB(I,2).EQ.24)).AND.(IB(I-1,1).EQ.3).AND.
     *(IB(I+1,1).EQ.4)) GO TO 15
      GO TO 12
100   CONTINUE
C -----***  flag is checked for error     ****
      DO 99 J=1,M
      IF(IB(J,3).NE.1) GO TO 99
      GO TO 110
99    CONTINUE
      WRITE(1,109)
      GO TO 112
C -----** error diagnosis is given below   *****
110   IF(IB(1,3).EQ.1)WRITE(1,102)
      IF(IB(2,3).EQ.1)WRITE(1,103)
      IF(IB(3,3).EQ.1)WRITE(1,104)
      IF(IB(4,3).EQ.1)WRITE(1,105)
      IF(IB(5,3).EQ.1)WRITE(1,106)
      IF(IB(6,3).EQ.1)WRITE(1,107)
      CALL EXEC(ICODF,LRNAM)
102   FORMAT(1H ,'At Your 1st Query Word no proper Kyword DELETE')
103   FORMAT(1H ,'At Your 2nd Query Word no Identifier')
104   FORMAT(1H ,'At Your 3rd Query Word no proper Kyword WHERE')
105   FORMAT(1H ,'At Your 4th Query Word no Identifier')
106   FORMAT(1H ,'At Your 5th Query Word no Proper Delimiter = ')
107   FORMAT(1H ,'At Your 6th Query Word no Proper Literal ')
109   FORMAT(1H ,'Your Query is Syntactically correct')
C -----*** corresponding semantics routine is called in next statement **
112   CALL EXEC(ICODF,JNAM2)
      END
C     *********************************************************************
C -----*   This segment checks syntax of query statement whose first quer
C     *******************  is MODIFY          ***************************
      PROGRAM MODFY,5
      DIMENSION IA(50,2),IB(50,3),ITAB1(30,22),ITAB(30,42),JNAM3(3)
     *,LRNAM(3)
      COMMON ITAB,ITAB1,IA,M,L,JM,IB
      DATA ICODF,JNAM3,LRNAM/8,2HUP,2HDA,2HT ,2HCL,2HEA,2HR /
      DO 5 I=1,M
      DO 5 J=1,2
5     IB(I,J)=IA(I,J)
      DO 130 I=1,M
      IF(I.GE.8) GO TO 80
      GO TO (10,20,30,40,50,60,70),I
C -----******** First query word MODIFY is checked  in Query statement
```

```
 10     IF((IB(I,1).EQ.1).AND.(IB(I,2).EQ.16)) GO TO 15
 12     IB(I,3)=1
        GO TO 130
 15     IB(I,3)=0
        GO TO 130
C -----*****  2nd query word is checked for identifier         **********
 20     IF(IB(I,1).EQ.4) GO TO 15
        GO TO 12
C -----*** 3rd query word is checked for keyword SET   ************
 30     IF ((IB(I,1).EQ.1).AND.(IB(I,2).EQ.17)) GO TO 15
        GO TO 12
C -----*** 4th query word is checked for IDENTIFIER  again  ********
 40     IF(IB(I,1).EQ.4) GO TO 15
        GO TO 12
 50     IF((IB(I,1).EQ.2).AND.(IB(I,2).EQ.26)) GO TO 15
        GO TO 12
C -----*** 6th query word is checked for LITERAL     *******
 60     IF(IB(I,1).EQ.3) GO TO 15
        GO TO 12
 70     IF(((IB(I,1).EQ.1).AND.(IB(I,2).EQ.3)).AND.(IB(I+1,1).EQ.4))
       *GO TO 15
        GO TO 12
C -----*** Identifier , Delimiter ' = ' ,literal & keyword ' AND ' ***
C         *       is checked in next few statements                 *
 80     IF((IB(I,1).EQ.4).AND.((IB(I+1,1).EQ.2).AND.(IB(I+1,2).EQ.26)).AND
       *.(IB(I-1,1).EQ.1)) GO TO 15
        IF(((IB(I,1).EQ.2).AND.(IB(I,2).EQ.26)).AND.(IB(I-1,1).EQ.4).AN
       *D.(IB(I+1,1).EQ.3)) GO TO 15
        IF((IB(I,1).EQ.3).AND.((IB(I-1,1).EQ.2).AND.(IB(I-1,2).EQ.26)))
       *GO TO 15
        IF(((IB(I,1).EQ.1).AND.(IB(I,2).EQ.24)).AND.(IB(I+1,1).EQ.4).AN
       *D.(IB(I-1,1).EQ.3)) GO TO 15
        GO TO 12
 130    CONTINUE
C -----****     Flags affected during syntax checking are processed    **
        DO 135 I=1,M
        IF(IB(I,3).NE.1) GO TO 135
        GO TO 138
 135    CONTINUE
        WRITE(1,147)
        GO TO 139
C -----***  Error diagnosis is given in next few statements    ******
 138    IF(IB(1,3).EQ.1) WRITE(1,141)
        IF(IB(2,3).EQ.1) WRITE(1,142)
        IF(IB(3,3).EQ.1) WRITE(1,143)
        IF(IB(4,3).EQ.1) WRITE(1,144)
        IF(IB(5,3).EQ.1) WRITE(1,145)
        IF(IB(6,3).EQ.1) WRITE(1,146)
        IF(IB(7,3).EQ.1) WRITE(1,150)
        IF(IB(8,3).EQ.1) WRITE(1,151)
```

```
      IF(IB(9,3).EQ.1) WRITE(1,152)
      IF(IB(10,3).EQ.1) WRITE(1,153)
      CALL EXEC(ICODF,LRNAM)
141   FORMAT(1H ,'At your 1st Query Word no proper kyword MODIFY ')
142   FORMAT(1H ,'At your 2nd Query Word no idetifier')
143   FORMAT(1H ,'At Your 3rd Query word no proper Kyword SET ')
144   FORMAT(1H ,'At your 4th Query Word no Identifier')
145   FORMAT(1H ,'At Your 5th Query Word no proper Delimeter = ')
146   FORMAT(1H ,'At Your 6th Query Word no Literal ')
147   FORMAT(1H ,'Your Query is Syntactically Correct ')
148   FORMAT(1H ,'Your Query is Wrong Please Give Again ')
150   FORMAT(1H ,'At your 7th Query Word no proper KyWord WHERE ')
151   FORMAT(1H ,'At your 8th Query Word no Idetifier ')
152   FORMAT(1H ,'At your 9th Query Word no proper Delimeter = ')
153   FORMAT(1H ,'At your 10th Query Word no proper Literal ')
C -----*** Corresponding semantic routine is called in case      ******
C      *** of correct query statement by system utility routine ******
C      ******************  CALL EXEC (-------,------)  *****************
139   CALL EXEC(ICODF,JNAM3)
      END
C     *******************************************************************
C -----*   This segment checks syntax of query statement whose first
C      *   query word is RESSTOR . This will make the relation in state
C      *   of use which was previously suspended by some user
C     *******************************************************************
      PROGRAM RESTR,5
      DIMENSION IA(50,2),IB(50,3),ITAB1(30,22),ITAB(30,42),INAM(3)
     *,LRNAM(3)
      COMMON ITAB,ITAB1,IA,M,L,JM,IB
      DATA ICODF,INAM,LRNAM/8,2HRE,2HST,2HT ,2HCL,2HEA,2HR /
      DO 10 I=1,M
      DO 10 J=1,2
10    IB(I,J)=IA(I,J)
      DO 60 I=1,M
      IF(I.GT.2) GO TO 45
      GO TO (30,40),I
C -----**** First query word RESTOR is checked in query statement    ****
30    IF((IB(I,1).EQ.1).AND.(IB(I,2).EQ.25)) GO TO 35
32    IB(I,3)=1
      GO TO 60
35    IB(I,3)=0
      GO TO 60
C -----****  2nd query word is checked for IDENTIFIER       ********
40    IF(IB(I,1).EQ.4) GO TO 35
      GO TO 32
45    WRITE(1,85)
      GO TO 68
60    CONTINUE
C -----*** Flags are checked for presence of error    *******
      DO 65 I=1,M
      IF(IB(I,3).NE.1) GO TO 65
      GO TO 66
65    CONTINUE
      WRITE(20,70)
      GO TO 90
C -----***  Error diagnosis is given here      *****
66    IF(IB(1,3).EQ.1) WRITE(1,75)
      IF(IB(2,3).EQ.1) WRITE(1,80)
68    CALL EXEC(ICODF,LRNAM)
70    FORMAT(1H ,'Query is syntactically correct ')
75    FORMAT(1H ,'At your 1st query word keyword RESTOR is missing')
80    FORMAT(1H ,'No relation name after keyword')
85    FORMAT(1H ,'You are giving more infomation to restor ')
C -----***  Corresponding semantic routine is called in case  ****
C     ***  query statement is correct                        ****
90    CALL EXEC(ICODF,INAM)
      END
C     *******************************************************************
C -----*  This segment checks syntax of query statement whose first     *
C     *  query word is DEFVEW                                           *
C     *******************************************************************
      PROGRAM DEFVW,5
      DIMENSION IA(50,2),IB(50,3),ITAB1(30,22),ITAB(30,42),JNAM4(3)
     *,LRNAM(3)
      COMMON ITAB,ITAB1,IA,M,L,JM,IB
      DATA ICODF,JNAM4,LRNAM/8,2HDE,2HFV,2HE ,2HCL,2HEA,2HR /
      DO 5 I=1,M
      DO 5 J=1,2
5     IB(I,J)=IA(I,J)
      DO 100 I=1,M
      IF(I.EQ.M) GO TO 50
      IF(I.GT.6) GO TO 40
      GO TO (10,20,30,20,30,20),I
C -----*** First query word  DEFVEW is checked in query statement    ***
10    IF((IB(I,1).EQ.1).AND.(IB(I,2).EQ.13)) GO TO 25
15    IB(I,3)=1
      GO TO 100
C -----*** Identifier is checked        ***-----
20    IF(IB(I,1).EQ.4) GO TO 25
      GO TO 15
25    IB(I,3)=0
      GO TO 100
C -----*** Left parenthesis is checked      ***-----
30    IF((IB(I,1).EQ.2).AND.(IB(I,2).EQ.32)) GO TO 25
      GO TO 15
C -----*** Right paranthesis , Identifier , left paranthesis   ***---
C       *          are checked in next statements              *
40    IF(((IB(I,1).EQ.2).AND.(IB(I,2).EQ.33)).AND.(IB(I-1,1).EQ.4).AND.
     *((IB(I+1,1).EQ.4).OR.((IB(I+1,1).EQ.2).AND.(IB(I+1,2).EQ.33))))
     * GO TO 25
```

```
      IF((IB(I,1).EQ.4).AND.((IB(I-1,1).EQ.2).AND.((IB(I-1,2).EQ.33).OR
     *.(IB(I-1,2).EQ.32))) .AND. ((IB(I+1,1).EQ.2).AND.((IB(I+1,2).EQ.
     *33).OR.(IB(I+1,2).EQ.32)))) GO TO 25
        IF(((IB(I,1).EQ.2).AND.(IB(I,2).EQ.32)).AND.(IB(I+1,1).EQ.4).AND
     *.(IB(I-1,1).EQ.4).AND.((IB(I+2,1).EQ.2).AND.(IB(I+2,2).EQ.33)))
     * GO TO 25
      GO TO 15
 50   IF(((IB(I,1).EQ.2).AND.(IB(I,2).EQ.33)).AND.((IB(I-1,1).EQ.2).AND.
     *(IB(I-1,2).EQ.33))) GO TO 25
       GO TO 15
 100  CONTINUE
C -----***  Flags are checked for presence of error  ***-----
      DO 110 I=1,M
      IF(IB(I,3).NE.1) GO TO 110
      GO TO 120
 110  CONTINUE
      WRITE(20,150)
      GO TO 130
C -----***  Error diagnosis is given in next few statements   ***------
 120  IF((IB(2,3).EQ.1).OR.(IB(4,3).EQ.1).OR.(IB(6,3).EQ.1))
     * WRITE(1,140)
      IF((IB(3,3).EQ.1).OR.(IB(7,3).EQ.1)) WRITE(1,145)
      CALL EXEC(ICODF,LRNAM)
 140  FORMAT(1H ," Check ID for proper place")
 145  FORMAT(1H ," Check for left & right parenthesis")
 150  FORMAT(1H ," Query is syntactically correct")
C -----***  Corresponding semantic routine is called    ***-----
C -----***  provided query statement is correct         ***-----
 130  CALL EXEC(ICODF,JNAM4)
      END
C -----*** A query processor starts here  for SELECT query  ***
      PROGRAM QRYA,5
      COMMON ITAB,ITAB1,ISTB,MM,MT,ML,IPRS,IQRY,IF
      INTEGER IPRS(50,3),IDCB(144),NAME(3),NAM1(3),NAM2(3),NAM3(3),
     *NAM4(3),NAM5(3),IBUF(160),IBUF1(10),IBUF2(256),ITAB(30,42),
     *ITAB1(30,22),IB(128),IC(80),ID(10),BLANK,IBUF3(20),IE(128),
     *ISTRT(10),ILEN(10),IRHS(10),ISECU(3),KF(2),IDCB1(144),NF(2),
     *IBUFL(41),IBUFF(10),NAM6(3),IBUFC(40),IBUFP(40),IBUFX(40),
     *ISTB(50,2),NAM7(3),NAM8(3),NAM9(3),NAM10(3),NAM11(3),
     *NAM12(3),IDCB2(272),NE(256),ICNAM(3)
      DATA NAME,NAM1,NAM2,NAM3,NAM5,NAM4,NAM6/2HDB,2HMS,2HRD,
     *2HDB,2HMS,2H8 ,2HDB,2HMS,2H00,2HDB,2HMS,2H11,2HIS,2HUF,2HYL,
     *2HCA,2HRD,2HAT,2HDB,2HMS,2H01/
      DATA NAM7,NAM8,NAM9,NAM10,NAM11,NAM12/2HJU,2HRN,2HAL,2HJU,2HRF,
     *2HLD,2HCN,2HFR,2HNC,2HCN,2HFF,2HLD,2HTH,2HSI,2HS ,2HAB,2HST,
     *2HRC/
      DATA BLANK/000040B/
      DATA ISECU/2H-7,2H56,2H19/
      DATA ICOE,ICNAM /8,2HCL,2HEA,2HR /
C -----*************************************************************
```

```
C      *  In body of our program following systems subroutines have     *
C      *  been used :-                                                   *
C      * 1. WRITF :- It writes a new record or modified record in data  *
C      *                              file                               *
C      * 2. READF :- It reads a record from data files                  *
C      * 3. SMOVE :- It helps in character manipulation                 *
C      * 4. SFILL :- It fills a buffer by any character , it also       *
C      *                    helps in character manipulation              *
C      * 5. JSCOM:-  It helps to compare to arrays of strings , it      *
C      *                      also helps in character manipulation.      *
C      * 6. SDEA2:- It converts decimal to ASCII                        *
C      * 7. MOD   :- It helps to get remainder in arithmetic function   *
C      * 8. OPEN :- It helps to open an existing file for accessing     *
C      * 9. CLOSE:- It helps to close a file to prevent its further use *
C      *                   in that program                               *
C      *                                                                 *
C      *******************************************************************
       M=0
       N=0
       KOUNT=0
       KONT=0
       IFL=0
       KA=0
       KB=0
       KC=0
       IVEW=0
       ICONT=0
C     ICONT IS THE COUNT FOR SECOUN
       IN=0
       IFLGT=0
       DO 5 I=1,MM
       DO 5 J=1,2
       IF(IPRS(I,1).NE.1) GO TO 5
       IF(IPRS(I,2).NE.2) GO TO 5
       GO TO 10
5      CONTINUE
       WRITE(20,1)
1      FORMAT(1H ,'FROM MISSING')
       STOP
C -----***      IDENTIFIERS IN 'FROM CLAUSE' ARE RECOVERED FROM     ***---
C        *      ID TABLE ITAB1                                      *
10     DO 15 II=1,MT
       IF(IPRS(I+1,1).NE.ITAB1(II,21)) GO TO 15
       IF(IPRS(I+1,2).NE.ITAB1(II,22)) GO TO 15
       GO TO 20
15     CONTINUE
20     CALL SFILL(IBUF1,1,20,BLANK)
       DO 25 IJ=1,10
       IF(ITAB1(II,IJ).EQ.1H )GO TO 52
       IBUF1(IJ)=ITAB1(II,IJ)
```

```
25    KOUNT=KOUNT+1
C-----------      CHECK WHETHER THE RELATION GIVEN IN QUERY EXISTS
C                 IN THE RELATION DIRECTORY      ------------------------
52    CALL OPEN(IDCB,IERR,NAME,1,ISECU)
      IF(IERR.NE.2) GO TO 101
      IF(KOUNT.LT.10)GO TO 53
      GO TO 54
53    CALL SFILL(IBUF1,19,20,BLANK)
54    CALL HASS(IBUF1,10,6,NUMBR)
      IF(NUMBR.EQ.0)NUMBR=6
      N1=NUMBR
      CALL CODE
      WRITE(IBUF,26)(IBUF1(IX),IX=1,10)
26    FORMAT(10A1)
      CALL SFILL(IC,1,160,BLANK)
      CALL READF(IDCB,IERR,IC,80,LEN,NUMBR)
      IBUC=1
      DO 50 IBUC=1,4
      INO=1+(IBUC-1)*12
      IBEG1=6+(IBUC-1)*6
      IER=0
      IF(JSCOM(IBUF,1,10,IC,INO,IER))50,55,50
50    CONTINUE
      DO 62 N2=7,10
      CALL SFILL(IC,1,160,BLANK)
      CALL READF(IDCB,IERR,IC,80,LEN,N2)
      IF(IERR.LT.0)GO TO 101
      IBUC=1
      DO 62 IBUC=1,4
      INO=1+(IBUC-1)*12
      IBEG1=6+(IBUC-1)*6
      IER=0
      IF(JSCOM(IBUF,1,10,IC,INO,IER))62,55,62
62    CONTINUE
      CALL CLOSE(IDCB)
C-----------       IF RELATION NOT FOUND IN RELATION DIRECTORY,
C                  CHECK IN THE VIEW DIRECTORY   ------------------------
      CALL OPEN(IDCB,IERR,NAM2,1,ISECU)
      IF(IERR.NE.2)GO TO 101
      CALL HASS(IBUF1,KOUNT,19,NUMBR)
      IF(NUMBR.EQ.0)NUMBR=19
      CALL SFILL(IC,1,160,BLANK)
      CALL READF(IDCB,IERR,IC,80,LEN,NUMBR)
      IF(IERR.LT.0)GO TO 101
      DO 85 IBUC=1,5
      INO=1+(IBUC-1)*16
      IBEG=6+(IBUC-1)*8
      ITYP=13+(IBUC-1)*16
      IER=0
      IF(JSCOM(IBUF,1,2*KOUNT,IC,INO,IER))85,88,85
```

```
85    CONTINUE
      DO 86 N2=20,22
      CALL SFILL(IC,1,160,BLANK)
      CALL READF(IDCB,IERR,IC,80,LEN,N2)
      IF(IERR.LT.0)GO TO 101
      IBUC=1
      DO 86 IBUC=1,5
      INO=1+(IBUC-1)*16
      IBEG=6+(IBUC-1)*8
      ITYP=13+(IBUC-1)*16
      IER=0
      IF(JSCOM(IBUF,1,2*KOUNT,IC,INO,IER))86,88,86
86    CONTINUE
      WRITE(20,82)
82    FORMAT(1H ,'RELATION NOT FOUND')
      CALL CLOSE(IDCB)
      STOP
88    CALL SMOVE(IC,ITYP,ITYP+3,NF,1)
      CALL CODE
      READ(NF,90)N1,IBUC1
90    FORMAT(2I2)
      CALL SMOVE(IC,2*IBEG-1,2*IBEG,IZ,1)
      CALL CODE
      READ(IZ,89)IU
89    FORMAT(I2)
C------------    CHECK WHETHER RELATION IS SUSPENDED       ---------------
      IF(IU.EQ.0)GO TO 58
      WRITE(20,83)
83    FORMAT(1H ,'RELATION FOUND IN VIEW DIRECTORY')
      IVEW=1
      CALL CLOSE(IDCB)
      GO TO 500
55    IBUC1=IBUC
      CALL SMOVE(IC,2*IBEG1-1,2*IBEG1,IZ,1)
      CALL CODE
      READ(IZ,63)IU
63    FORMAT(I2)
      IF(IU.EQ.0)GO TO 58
      WRITE(1,56)
56    FORMAT(1H ,"RELATION FOUND")
      GO TO 59
58    WRITE(20,51)
51    FORMAT(1H ,'RELATION IS SUSPENDED,QUERY CANNOT BE PROCESSED')
      STOP
59    CALL CLOSE(IDCB)
      CALL SFILL(IBUF3,1,40,BLANK)
      CALL SMOVE(IBUF1,1,2*KOUNT,IBUF3,1)
      CALL OPEN(IDCB,IERR,NAM1,1,ISECU)
      IF(IERR.NE.2)GO TO 101
C-------------     ATTRIBUTES IN SELECT AND WHERE CLAUSES ARE
```

```
C                    RETRIEVED FROM ITAB1          ------------------------------
57     DO 60 KI=1,MT
       IF(IPRS(I-1,1).NE.ITAB1(KI,21))GO TO 60
       IF(IPRS(I-1,2).NE.ITAB1(KI,22))GO TO 60
       M=M+1
       GO TO 65
60     CONTINUE
       WRITE(20,61)
61     FORMAT(1H ,"ATTRIBUTE NOT IN ITAB1")
       STOP
65     CALL SFILL(IBUF1,1,20,BLANK)
       KONT=0
       DO 66 IG=1,10
       IF(ITAB1(KI,IG).EQ.1H )GO TO 67
       IBUF1(IG)=ITAB1(KI,IG)
66     KONT=KONT+1
67     CALL SFILL(IBUF3,2*KOUNT+1,40,BLANK)
       CALL SMOVE(IBUF1,1,2*KONT,IBUF3,2*KOUNT+1)
       IF((KOUNT+KONT).LT.20)GO TO 64
       GO TO 68
C---------------    CHECK WHETHER THE ATTRIBUTES GIVEN IN QUERY
C                   EXIST IN THE DIRECTORY          ------------------------------
64     CALL SFILL(IBUF3,2*(KOUNT+KONT)+1,40,BLANK)
68     CALL HASS(IBUF3,20,31,NUMBR)
       IF(NUMBR.EQ.0)NUMBR=31
       CALL CODE
       WRITE(IBUFF,69)(IBUF3(IX),IX=1,20)
69     FORMAT(20A1)
       CALL SFILL(IE,1,256,BLANK)
       CALL READF(IDCB,IERR,IE,128,LEN,NUMBR)
       IF(IERR.LT.0)GO TO 101
       KKO=KOUNT+KONT
       IER=0
       IBUC=1
       DO 72 IBUC=1,3
       INO=1+(IBUC-1)*32
       IF(JSCOM(IBUFF,1,KKO,IE,INO,IER))72,73,72
72     CONTINUE
       DO 79 N2=32,35
       CALL SFILL(IE,1,256,BLANK)
       CALL READF(IDCB,IERR,IE,128,LEN,N2)
       IF(IERR.LT.0)GO TO 101
       IBUC=1
       DO 79 IBUC=1,3
       INO=1+(IBUC-1)*32
       IER=0
       IF(JSCOM(IBUFF,1,KKO,IE,INO,IER))79,73,79
79     CONTINUE
       WRITE(20,77)
77     FORMAT(1H ,'ATTRIBUTE NOT IN DIRECTORY')
```

```
      STOP
73    WRITE(1,76)
76    FORMAT(1H ,'Attribute found')
      KKB=23+(IBUC-1)*32
      CALL SMOVE(IE,KKB,KKB+3,KF,1)
      CALL CODE
      READ(KF,75)K2
75    FORMAT(I4)
      KA=KA+1
      ISTRT(KA)=K2
      CALL SMOVE(IE,KKB+4,KKB+5,KG,1)
      CALL CODE
      READ(KG,78)K3
78    FORMAT(I2)
      KB=KB+1
      ILEN(KB)=K3
      IF(N.EQ.1)GO TO 200
      IF(M.EQ.IF)GO TO 150
      I=I-1
      KI=1
      GO TO 57
150   N=1
      I=1
      M=0
      DO 155 I=1,MM
      DO 155 J=1,2
      IF(IPRS(I,1).NE.2)GO TO 155
      IF(IPRS(I,2).NE.26)GO TO 155
      KI=1
      GO TO 57
155   CONTINUE
      WRITE(20,156)
156   FORMAT(1H ,"= IS MISSING")
      STOP
C-----------       THE LITERALS GIVEN IN QUERY ARE RETRIEVED
C                  FROM THE LITERAL TABLE ITAB     ---------------------
200   DO 220 IK=1,ML
      IF(IPRS(I+1,1).NE.ITAB(IK,41))GO TO 220
      IF(IPRS(I+1,2).NE.ITAB(IK,42))GO TO 220
      GO TO 225
220   CONTINUE
      WRITE(20,221)
221   FORMAT(1H ,"LITERAL NOT IN ITAB")
      STOP
225   CALL SFILL(IBUFL,1,80,BLANK)
      DO 230 L=1,40
230   IBUFL(L)=ITAB(IK,L)
      CALL CLOSE(IDCB)
C -----***         Appropriate data files are opened & retrieval    ***
C        *         done according to query                         *
```

```
      IF((N1.EQ.1).AND.(IBUC1.EQ.1))GO TO 235
      IF((N1.EQ.2).AND.(IBUC1.EQ.1))GO TO 236
      IF((N1.EQ.2).AND.(IBUC1.EQ.2))GO TO 237
      IF((N1.EQ.4).AND.(IBUC1.EQ.1))GO TO 238
      IF((N1.EQ.4).AND.(IBUC1.EQ.2))GO TO 239
      IF((N1.EQ.5).AND.(IBUC1.EQ.1))GO TO 240
      IF((N1.EQ.6).AND.(IBUC1.EQ.1))GO TO 241
      IF((N1.EQ.5).AND.(IBUC1.EQ.2))GO TO 242
      WRITE(20,243)
  243 FORMAT(1H ,'WRONG RELATION NAME')
  235 CALL OPEN(IDCB,IERR,NAM3)
      IF(IERR.NE.2)GO TO 101
      GO TO 248
  236 CALL OPEN(IDCB,IERR,NAM4)
      IF(IERR.NE.2)GO TO 101
      GO TO 248
  237 CALL OPEN(IDCB,IERR,NAM7,1,ISECU)
      IF(IERR.NE.2)GO TO 101
      GO TO 248
  238 CALL OPEN(IDCB,IERR,NAM5)
      IF(IERR.NE.2)GO TO 101
  239 CALL OPEN(IDCB,IERR,NAM11,1,ISECU)
      IF(IERR.NE.2)GO TO 101
      GO TO 248
  240 CALL OPEN(IDCB,IERR,NAM8,1,ISECU)
      IF(IERR.NE.2)GO TO 101
      GO TO 248
  241 CALL OPEN(IDCB,IERR,NAM9,1,ISECU)
      IF(IERR.NE.2)GO TO 101
      GO TO 248
  242 CALL OPEN(IDCB,IERR,NAM10,1,ISECU)
      IF(IERR.NE.2) GO TO 101
  248 IF(IQRY.NE.6)GO TO 250
  342 KUNT=0
      LO=0
      LP=1
      DO 345 LP=1,40
      IF(IBUFL(LP).EQ.1H )GO TO 344
      LO=LO+1
      IBUFC(LO)=IBUFL(LP)
      KUNT=KUNT+1
  345 CONTINUE
  344 IF(IFLGT.EQ.1)GO TO 346
      IF(IFLAG.EQ.1)GO TO 300
  250 DO 300 IN=1,10
      ITB=0
      CALL SFILL(IB,1,256,BLANK)
      CALL READF(IDCB,IERR,IB,128,LEN,IN)
      IF(IERR.LT.0) GO TO 101
      CALL SFILL(IBUF2,1,512,BLANK)
      CALL CODE
      READ(IB,232)(IBUF2(IX),IX=1,256)
  232 FORMAT(256A1)
      IER=0
      IST=ISTRT(KA)
      ISTT=2*IST-1
      ILT=ILEN(KA)
      LITL=2*(IST+ILT-1)
  346 IF(IQRY.NE.6)GO TO 292
      CALL SMOVE(IBUF2,ISTT,LITL,IBUFP,1)
      LH=0
      IF(ITB.EQ.1)GO TO 350
      LF=1
      DO 350 LF=1,40
      LH=LH+1
      IF(IBUFP(LF).EQ.1H )GO TO 352
      IBUFX(LH)=IBUFP(LF)
      ITB=1
  350 CONTINUE
  352 IF(JSCOM(IBUFC,1,2*KUNT,IBUFX,1,IER))353,354,353
  353 CALL SFILL(IBUFX,1,80,BLANK)
      LH=0
      IF(LF.GT.40)GO TO 300
      IF(IBUFP(LF+1).EQ.1H )GO TO 300
      GO TO 350
  354 CALL SFILL(IBUFX,1,80,BLANK)
      CALL SFILL (IBUFC,1,80,BLANK)
      IFLGT=1
      IF(IBUFL(LP+1).EQ.1H )GO TO 245
      IF(LP+1.GT.40)GO TO 245
      KUNT=0
      LO=0
      GO TO 345
  292 IF(JSCOM(IBUF2,ISTT,LITL,IBUFL,1,IER))300,245,300
  300 CONTINUE
      IF(IFLAG.EQ.1)GO TO 910
      GO TO 900
  245 IFLAG=1
      IFLGT=0
      ICONT=ICONT+1
C---------------     RESULTS OF RETRIEVAL ARE PRINTED       -------------------
      DO 251 LS=KA-1,1,-1
      GO TO 301
  251 CONTINUE
      IF(IQRY.EQ.6)GO TO 342
      GO TO 300
  301 IF((ISTRT(LS).EQ.1).AND.((N1.EQ.6).OR.((N1.EQ.6).AND.(IBUC1.EQ.2))
     *))GO TO 382
      IF(ISTRT(LS).EQ.1)GO TO 371
      IF((ISTRT(LS).EQ.41).AND.((N1.EQ.1).OR.(N1.EQ.2).OR.(N1.EQ.4)))
     *GO TO 377
      IF((ISTRT(LS).EQ.41).AND.(((N1.EQ.5).AND.(IBUC1.EQ.1)).OR.((N1.EQ
     *.2).AND.(IBUC1.EQ.2))))GO TO 385
      IF(ISTRT(LS).EQ.61)GO TO 386
      IF((ISTRT(LS).EQ.67).AND.(N1.EQ.4).AND.(IBUC1.EQ.2))GO TO 387
      IF(ISTRT(LS).EQ.67)GO TO 373
      IF((ISTRT(LS).EQ.101).AND.(N1.EQ.2))GO TO 390
      IF(ISTRT(LS).EQ.101)GO TO 391
      IF((ISTRT(LS).EQ.81).AND.(N1.EQ.4))GO TO 392
      IF(ISTRT(LS).EQ.81)GO TO 393
      IF(ISTRT(LS).EQ.69)GO TO 380
      IF(ISTRT(LS).EQ.111)GO TO 394
      IF((ISTRT(LS).EQ.85).AND.((N1.EQ.5).OR.((N1.EQ.2).AND.(IBUC1.EQ.2
     *))))GO TO 396
      IF(ISTRT(LS).EQ.77)GO TO 397
      IF((ISTRT(LS).EQ.73).AND.(N1.EQ.1))GO TO 376
      IF(ISTRT(LS).EQ.73)GO TO 375
      IF(ISTRT(LS).EQ.113)GO TO 398
      IF(ISTRT(LS).EQ.95)GO TO 400
      IF(ISTRT(LS).EQ.83)GO TO 370
      IF((ISTRT(LS).EQ.85).AND.(N1.EQ.2).AND.(IBUC1.EQ.1))GO TO 381
      IF((ISTRT(LS).EQ.85).AND.(N1.EQ.4))GO TO 383
      IF((ISTRT(LS).EQ.115).AND.(N1.EQ.6))GO TO 401
      IF((ISTRT(LS).EQ.115).AND.(N1.EQ.5))GO TO 402
      IF((ISTRT(LS).EQ.97).AND.(N1.EQ.1))GO TO 384
      IF(ISTRT(LS).EQ.97)GO TO 403
      IF(ISTRT(LS).EQ.107)GO TO 404
      IF(ISTRT(LS).EQ.89)GO TO 378
      IF((ISTRT(LS).EQ.135).AND.(N1.EQ.6))GO TO 405
      IF((ISTRT(LS).EQ.99).AND.(N1.EQ.2))GO TO 407
      IF((ISTRT(LS).EQ.135).AND.(N1.EQ.5))GO TO 406
      IF((ISTRT(LS).EQ.99).AND.(N1.EQ.5))GO TO 408
      IF(ISTRT(LS).EQ.147)GO TO 409
      IF(ISTRT(LS).EQ.123)GO TO 410
      IF(ISTRT(LS).EQ.155)GO TO 411
      IF(ISTRT(LS).EQ.105)GO TO 412
      IF(ISTRT(LS).EQ.119)GO TO 413
      IF(ISTRT(LS).EQ.167)GO TO 414
      IF(ISTRT(LS).EQ.143)GO TO 415
      IF(ISTRT(LS).EQ.161)GO TO 416
      IF(ISTRT(LS).EQ.125)GO TO 417
      IF(ISTRT(LS).EQ.183)GO TO 418
      WRITE(20,265)
  265 FORMAT(1H ,'CORRECT ATTRIBUTE IN DIRECTORY NOT FOUND')
      STOP
  370 WRITE(20,420)(IBUF2(IX),IX=83,88)
  420 FORMAT(1H ,/,' STATUS:- ',5X,6A1)
      GO TO 395
  371 WRITE(20,421)(IBUF2(IX),IX=1,40)
  421 FORMAT(1H ,/,' TITLE:- ' ,5X,40A1)
```

```
      GO TO 395
372   WRITE(20,422)(IBUF2(IX),IX=71,72)
422   FORMAT(1H ,/,' COPYNO:- ' ,5X,2A1)
      GO TO 395
373   WRITE(20,423)(IBUF2(IX),IX=67,68)
423   FORMAT(1H ,/,' VOLUME:- ' ,5X,2A1)
      GO TO 395
375   WRITE(20,425)(IBUF2(IX),IX=73,84)
425   FORMAT(1H ,/,' CARDNO:- ' ,5X,12A1)
      GO TO 395
376   WRITE(20,426)(IBUF2(IX),IX=73,80)
426   FORMAT(1H ,/,' PRICE:- ' ,5X,8A1)
      GO TO 395
377   WRITE(20,427)(IBUF2(IX),IX=41,66)
427   FORMAT(1H ,/,' AUTHOR:- ' ,5X,26A1)
      GO TO 395
378   WRITE(20,428)(IBUF2(IX),IX=89,96)
428   FORMAT(1H ,/,' LIBDIV/- ' ,5X,8A1)
      GO TO 395
380   WRITE(20,430)(IBUF2(IX),IX=69,70)
430   FORMAT(1H ,/,' EDITION:- ' ,5X,2A1)
      GO TO 395
381   WRITE(20,431)(IBUF2(IX),IX=85,110)
431   FORMAT(1H ,/,' ISUEENAME:- ' ,5X,16A1)
      GO TO 395
382   WRITE(20,432)(IBUF2(IX),IX=1,60)
432   FORMAT(1H ,/,' NAME:- ' ,5X,60A1)
      GO TO 395
383   WRITE(20,433)(IBUF2(IX),IX=85,94)
433   FORMAT(1H ,/,' DUEDATE:- ' ,5X,10A1)
      GO TO 395
384   WRITE(20,434)(IBUF2(IX),IX=97,116)
434   FORMAT(1H ,/,' CALLNO:- ' ,5X,20A1)
      GO TO 395
385   WRITE(20,437)(IBUF2(IX),IX=41,80)
437   FORMAT(1H ,/,' PUBNAME:- ' ,5X,40A1)
      GO TO 395
386   WRITE(20,438)(IBUF2(IX),IX=61,100)
438   FORMAT(1H ,/,' PLACE:- ' ,5X,40A1)
      GO TO 395
387   WRITE(20,439)(IBUF2(IX),IX=67,76)
439   FORMAT(1H ,/,' DEGREE:- ' ,5X,10A1)
      GO TO 395
390   WRITE(20,436)(IBUF2(IX),IX=101,130)
436   FORMAT(1H ,/,' ISSUEADDR:- ' ,5X,30A1)
      GO TO 395
391   WRITE(20,440)(IBUF2(IX),IX=101,110)
440   FORMAT(1H ,/,' DATE:- ' ,5X,10A1)
      GO TO 395
392   WRITE(20,441)(IBUF2(IX),IX=81,106)
```

```
441   FORMAT(1H ,/,' SUPERVISOR:- ' ,5X,26A1)
      GO TO 395
393   WRITE(20,442)(IBUF2(IX),IX=81,84)
442   FORMAT(1H ,/,' YEAR:- ' ,5X,4A1)
      GO TO 395
394   WRITE(20,423)(IBUF2(IX),IX=111,112)
      GO TO 395
396   WRITE(20,444)(IBUF2(IX),IX=85,94)
444   FORMAT(1H ,/,' MONTH:- ' ,5X,10A1)
      GO TO 395
397   WRITE(20,442)(IBUF2(IX),IX=77,80)
      GO TO 395
398   WRITE(20,446)(IBUF2(IX),IX=113,114)
446   FORMAT(1H ,/,' NUMBER:- ' ,5X,2A1)
      GO TO 395
400   WRITE(20,423)(IBUF2(IX),IX=95,96)
      GO TO 395
401   WRITE(20,448)(IBUF2(IX),IX=115,134)
448   FORMAT(1H ,/,' EDITOR:- ' ,5X,20A1)
      GO TO 395
402   WRITE(20,449)(IBUF2(IX),IX=115,134)
449   FORMAT(1H ,/,' FIELD:- ' ,5X,20A1)
      GO TO 395
403   WRITE(20,450)(IBUF2(IX),IX=97,98)
450   FORMAT(1H ,/,' COPYNO:- ' ,5X,2A1)
      GO TO 395
404   WRITE(20,451)(IBUF2(IX),IX=107,146)
451   FORMAT(1H ;/,' INSTITUTE:- ' ,40A1)
      GO TO 395
405   WRITE(20,434)(IBUF2(IX),IX=135,154)
      GO TO 395
406   WRITE(20,453)(IBUF2(IX),IX=135,174)
453   FORMAT(1H ,/,' SUBFIELD:- ' ,5X,40A1)
      GO TO 395
407   WRITE(20,435)(IBUF2(IX),IX=99,104)
435   FORMAT(1H ,/,' ACCNO:- ' ,5X,6A1)
      GO TO 395
408   WRITE(20,449)(IBUF2(IX),IX=99,118)
      GO TO 395
409   WRITE(20,449)(IBUF2(IX),IX=147,166)
      GO TO 395
410   WRITE(20,449)(IBUF2(IX),IX=123,142)
      GO TO 395
411   WRITE(20,435)(IBUF2(IX),IX=155,160)
      GO TO 395
412   WRITE(20,434)(IBUF2(IX),IX=105,124)
      GO TO 395
413   WRITE(20,453)(IBUF2(IX),IX=119,158)
      GO TO 395
414   WRITE(20,453)(IBUF2(IX),IX=167,206)
```

```
      GO TO 395
415   WRITE(20,453)(IBUF2(IX),IX=143,182)
      GO TO 395
416   WRITE(20,420)(IBUF2(IX),IX=161,166)
      GO TO 395
417   WRITE(20,420)(IBUF2(IX),IX=125,130)
      GO TO 395
C -----***         IF THE QUERY IS ON A VIEW THE CORRESPONDING       ***
C        *         CHECKS ARE MADE ON VIEW DIRECTORIES AND RETRIEVAL *
C        *         IS DONE ACCORDINGLY                               *
418   CALL SMOVE(IB,183,186,NF,1)
      CALL CODE
      READ(NF,419)NG
419   FORMAT(I4)
      CALL SFILL(NE,1,512,BLANK)
      CALL OPEN(IDCB2,IERR,NAM12,1,ISECU)
      IF(IERR.NE.2)GO TO 101
      CALL READF(IDCB2,IERR,NE,256,LEN,NG)
      IF(IERR.LT.0)GO TO 101
      WRITE(20,465)(NE(IX),IX=1,256)
465   FORMAT(1H ,/,' ABSTRACT:- ' ,5X,256A2)
      CALL CLOSE(IDCB2)
395   IF(IVEW.EQ.1)GO TO 720
      GO TO 251
500   CALL SFILL(IBUF3,1,40,BLANK)
      CALL SMOVE(IBUF1,1,2*KOUNT,IBUF3,1)
      CALL OPEN(IDCB,IERR,NAM6,1,ISECU)
      IF(IERR.NE.2)GO TO 101
      IF(IQRY.EQ.6)GO TO 502
      DO 505 KI=1,MT
      IF(IPRS(I+3,2).NE.ITAB1(KI,22))GO TO 505
      GO TO 506
505   CONTINUE
503   WRITE(20,501)
501   FORMAT(1H ,'ATTRIBUTE NOT IN ITAB1')
      STOP
502   DO 509 KI=1,MT
      IF(IPRS(I+4,2).NE.ITAB1(KI,22))GO TO 509
      GO TO 506
509   CONTINUE
      GO TO 503
506   KONT=0
      CALL SFILL(IBUF1,1,20,BLANK)
      DO 510 IG=1,10
      IF(ITAB1(KI,IG).EQ.1H )GO TO 507
      IBUF1(IG)=ITAB1(KI,IG)
510   KONT=KONT+1
507   CALL SFILL(IBUF3,2*KOUNT+1,40,BLANK)
      CALL SMOVE(IBUF1,1,2*KONT,IBUF3,2*KOUNT+1)
      KKO=KOUNT+KONT
      CALL HASS(IBUF3,KKO,31,NUMBR)
      IF(NUMBR.EQ.0)NUMBR=31
      CALL SFILL(IE,1,256,BLANK)
      CALL READF(IDCB,IERR,IE,128,LEN,NUMBR)
      IF(IERR.LT.0)GO TO 101
      CALL CODE
      WRITE(IBUFF,508)(IBUF3(IX),IX=1,20)
508   FORMAT(20A1)
      IBUC=1
      DO 515 IBUC=1,4
      INO=1+(IBUC-1)*36
      IER=0
      IF(JSCOM(IBUFF,1,KKO,IE,INO,IER))515,520,515
515   CONTINUE
      DO 525 N3=32,34
      CALL SFILL(IBUFF,1,20,BLANK)
      CALL SFILL(IE,1,256,BLANK)
      CALL READF(IDCB,IERR,IE,128,LEN,N3)
      IF(IERR.LT.0)GO TO 101
      IBUC=1
      DO 525 IBUC=1,4
      INO=1+(IBUC-1)*36
      IF(JSCOM(IBUFF,1,KKO,IE,INO,IER))525,520,525
525   CONTINUE
      WRITE(20,526)
526   FORMAT(1H ,'ATTRIBUTE NOT IN VIEW DIRECTORY')
      STOP
520   WRITE(1,527)
527   FORMAT(1H ,'Attribute found in VIEW')
      KKB=27+(IBUC-1)*36
      CALL SMOVE(IE,KKB,KKB+3,KF,1)
      CALL CODE
      READ(KF,528)K2
528   FORMAT(I4)
      KA=KA+1
      ISTRT(KA)=K2
      CALL SMOVE(IE,KKB+4,KKB+5,KG,1)
      CALL CODE
      READ(KG,529)K3
529   FORMAT(I2)
      KB=KB+1
      ILEN(KB)=K3
      CALL SMOVE(IE,KKB+6,KKB+7,KG,1)
      CALL CODE
      READ(KG,530)K4
530   FORMAT(I2)
      KC=KC+1
      IRHS(KC)=K4
      IF(IFL.EQ.1)GO TO 615
      K5=IRHS(1)
      GO TO (600,605,237,606),K5
600   CALL OPEN(IDCB1,IERR,NAM3)
      IF(IERR.NE.2)GO TO 101
      GO TO 610
605   CALL OPEN(IDCB1,IERR,NAM4)
      IF(IERR.NE.2)GO TO 101
      GO TO 610
606   CALL OPEN(IDCB1,IERR,NAM5)
      IF(IERR.NE.2)GO TO 101
610   IFL=1
      IF(IQRY.EQ.6)GO TO 622
      IF(IPRS(I+5,2).NE.ITAB(1,42))GO TO 620
611   DO 625 L=1,40
625   IBUFL(L)=ITAB(1,L)
615   I=I-1
      IF(I.EQ.1)GO TO 700
      KI=1
      DO 630 KI=1,MT
      IF(IPRS(I,2).NE.ITAB1(KI,22))GO TO 630
      IG=1
      GO TO 506
630   CONTINUE
      WRITE(20,631)
631   FORMAT(1H ,'VIEW ATTRIBUTE NOT IN ITAB1')
      STOP
620   WRITE(20,621)
621   FORMAT(1H ,'VIEW LITERAL NOT IN ITAB')
      STOP
622   IF(IPRS(I+6,2).NE.ITAB(1,42))GO TO 620
      GO TO 611
700   IST=ISTRT(1)
      ISTT=2*IST-1
      IFLGL=0
      ILT=ILEN(1)
      LITL=2*(IST+ILT-1)
      IF(IQRY.NE.6)GO TO 702
699   KUNT=0
      LO=0
      LP=1
      DO 703 LP=1,40
      IF(IBUFL(LP).EQ.1H )GO TO 704
      LO=LO+1
      IBUFC(LO)=IBUFL(LP)
      KUNT=KUNT+1
703   CONTINUE
704   IF(IFLGT.EQ.1)GO TO 706
      IF(IFLGL.EQ.1)GO TO 710
702   DO 710 IN=1,10
      ITB=0
      CALL SFILL(IB,1,256,BLANK)
      CALL SFILL(IBUF2,1,256,BLANK)
      CALL READF(IDCB1,IERR,IB,128,LEN,IN)
      IF(IERR.LT.0)GO TO 101
      CALL CODE
      READ(IB,701)(IBUF2(IX),IX=1,256)
701   FORMAT(256A1)
      IER=0
706   IF(IQRY.NE.6)GO TO 705
      CALL SMOVE(IBUF2,ISTT,LITL,IBUFP,1)
      LH=0
      IF(ITB.EQ.1)GO TO 707
      LF=1
      DO 707 LF=1,40
      LH=LH+1
      IF(IBUFP(LF).EQ.1H )GO TO 708
      IBUFX(LH)=IBUFP(LF)
      ITB=1
707   CONTINUE
708   IF(JSCOM(IBUFC,1,2*KUNT,IBUFX,1,IER))709,711,709
709   CALL SFILL(IBUFX,1,80,BLANK)
      LH=0
      IF(LF.GT.40)GO TO 710
      IF(IBUFP(LF+1).EQ.1H )GO TO 710
      GO TO 707
711   CALL SFILL(IBUFX,1,80,BLANK)
      CALL SFILL(IBUFC,1,80,BLANK)
      IFLGT=1
      IF(IBUFL(LP+1).EQ.1H )GO TO 715
      IF(LP+1.GT.40)GO TO 715
      KUNT=0
      LO=0
      GO TO 703
705   IF(JSCOM(IBUF2,ISTT,LITL,IBUFL,I,IER))710,715,710
710   CONTINUE
      IF(IFLGL.EQ.1)GO TO 910
      GO TO 900
715   IFLGL=1
      IFLGT=0
      ICONT=ICONT+1
      DO 720 LS=KA,2,-1
      GO TO 301
720   CONTINUE
      IF(IQRY.EQ.6)GO TO 699
      GO TO 710
101   WRITE(20,912)IERR
912   FORMAT(1H ,'IERR=',I2)
      STOP
900   WRITE(20,911)
911   FORMAT(1H ,'LITERAL DOES NOT EXIST IN DATA FILE')
      STOP
```

```
 910  CALL CLOSE(IDCB)
      IF(IVEW.NE.1)GO TO 950
      CALL CLOSE(IDCB1)
 950  IF(IQRY.NE.4)GO TO 1000
      WRITE(20,960)ICONT
 960  FORMAT(1H ,'COUNT=',I4)
1000  WRITE(20,1001)
1001  FORMAT(1H ,/,' QUERY PROCESSED')
      CALL EXEC(ICOE,ICNAM)
      END
C -----*** Query processor for CREATE Query statement starts here ***
      PROGRAM KRET,5
      INTEGER BLANK,ITAB1(30,22),ITAB(30,42),IPRS(50,3),ISTB1(20,8)
     *,IBUF(20),IDCB(144),NAM1(3),NAM2(3),IA(20),ID(20),IG(20)
     *,ISECU(3),IB(128),ISTB(50,2),KOMN(3)
      COMMON ITAB,ITAB1,ISTB,MM,MT,ML,IPRS,IQRY,IF,ISTB1
      DATA NAM1,NAM2/2HTE,2HST,2H1 ,2HTE,2HST,2H2 /
      DATA ISECU/2H-7,2H56,2H19/
      DATA BLANK/000040B/
      DATA ICDA,KOMN/8,2HCL,2HEA,2HR /
      N=1
      KON=0
      KOUNT=0
      KONT=0
      NT=0
      DO 5 K=1,20
      IF(ITAB1(1,K).EQ.1H )GO TO 6
      IBUF(K)=ITAB1(1,K)
 5    KOUNT=KOUNT+1
 6    CALL CODE
      WRITE(IA,7)(IBUF(IX),IX=1,20)
 7    FORMAT(20A1)
      CALL HASS(IBUF,10,6,NUMBR)
      IF(NUMBR.EQ.0)NUMBR=6
      N1=NUMBR
      CALL SFILL(IG,1,40,BLANK)
      CALL SMOVE(IBUF,1,2*KOUNT,IG,1)
      CALL OPEN(IDCB,IERR,NAM1,2,ISECU)
      IF(IERR.NE.2)GO TO 101
      CALL SFILL(IB,1,256,BLANK)
      CALL READF(IDCB,IERR,IB,128,LEN,N1)
      IF(IERR.LT.0)GO TO 101
      IBUC=1
      CALL SFILL(ID,1,40,BLANK)
      DO 10 IBUC=1,4
      INO=1+(IBUC-1)*12
      IEND=10+(IBUC-1)*12
      IER=0
      IF(JSCOM(IB,INO,IEND,IA,1,IER))15,100,15
 15   IF(JSCOM(IB,INO,IEND,ID,1,IER))10,30,10
```

```
10    CONTINUE
      DO 20 IR=7,10
      CALL SFILL(IB,1,256,BLANK)
      CALL READF(IDCB,IERR,IB,128,LEN,IR)
      IF(IERR.LT.0)GO TO 101
      IBUC=1
      DO 20 IBUC=1,4
      INO=1+(IBUC-1)*12
      IEND=10+(IBUC-1)*12
      IER=0
      IF(JSCOM(IB,INO,IEND,IA,1,IER))16,100,16
16    IF(JSCOM(IB,INO,IEND,ID,1,IER))20,29,20
20    CONTINUE
21    WRITE(20,22)
22    FORMAT(1H ,'NO PLACE IN DIRECTORY')
      STOP
29    N1=IR
30    IBUCF=IBUC
      CALL SMOVE(IA,1,KOUNT,IB,INO)
      IM=1
      IE1=0
      CALL SDEA2(IM,1,2,IE1)
      CALL SMOVE(IM,1,2,IB,IEND+1)
      CALL WRITF(IDCB,IERR,IB,0,N1)
      IF(IERR.LT.0)GO TO 101
      CALL SFILL(IB,1,256,BLANK)
      CALL READF(IDCB,IERR,IB,128,LEN,N1)
      IF(IERR.LT.0)GO TO 101
      CALL CLOSE(IDCB)
      DO 35 IP=1,20
      IF(ISTB1(IP,7).EQ.2)GO TO 31
      IF(ISTB1(IP,7).NE.1)GO TO 35
      IF(ISTB1(IP,8).EQ.10)GO TO 37
      IF(ISTB1(IP,8).EQ.12)GO TO 38
      IF(ISTB1(IP,8).EQ.22)GO TO 39
31    IF(ISTB1(IP,8).EQ.33)GO TO 500
35    CONTINUE
      WRITE(20,36)
36    FORMAT(1H ,'TYPE MISSING')
      STOP
37    ITYP=3
      GO TO 40
38    ITYP=2
      GO TO 40
39    ITYP=1
40    CALL SFILL(IBUF,1,40,BLANK)
      N=N+1
      KONT=0
      DO 45 IS=1,10
      IF(ITAB1(N,IS).EQ.1H )GO TO 46
      IBUF(IS)=ITAB1(N,IS)
45    KONT=KONT+1
46    CALL SMOVE(IBUF,1,2*KONT,IG,2*KOUNT+1)
      CALL HASS(IG,20,31,NUMBR)
      IF(NUMBR.EQ.0)NUMBR=31
      N2=NUMBR
      CALL OPEN(IDCB,IERR,NAM2,2,ISECU)
      IF(IERR.NE.2)GO TO 101
      CALL SFILL(IB,1,256,BLANK)
      CALL READF(IDCB,IERR,IB,128,LEN,N2)
      IF(IERR.LT.0)GO TO 101
      IBUC=1
      DO 50 IBUC=1,3
      INO=1+(IBUC-1)*32
      IEND=20+(IBUC-1)*32
      IER=0
      IF(JSCOM(IB,INO,IEND,ID,1,IER))50,55,50
50    CONTINUE
      DO 51 IR1=32,35
      CALL SFILL(IB,1,256,BLANK)
      CALL READF(IDCB,IERR,IB,128,LEN,IR1)
      IF(IERR.LT.0)GO TO 101
      IBUC=1
      DO 51 IBUC=1,3
      INO=1+(IBUC-1)*32
      IEND=20+(IBUC-1)*32
      IER=0
      IF(JSCOM(IB,INO,IEND,ID,1,IER))51,54,51
51    CONTINUE
      GO TO 21
54    N2=IR1
55    CALL SFILL(IA,1,40,BLANK)
      CALL CODE
      WRITE(IA,56)(IG(IX),IX=1,20)
56    FORMAT(20A1)
      CALL SMOVE(IA,1,KOUNT+KONT,IB,INO)
      IE1=0
      CALL SDEA2(ITYP,1,2,IE1)
      CALL SMOVE(ITYP,1,2,IB,IEND+1)
      NT=NT+1
      IF(NT.GT.ML)GO TO 500
      CALL SFILL(IBUF,1,40,BLANK)
      DO 60 LT=1,20
60    IBUF(LT)=ITAB(NT,LT)
      CALL SFILL(IA,1,40,BLANK)
      CALL CODE
      WRITE(IA,61)(IBUF(IX),IX=1,20)
61    FORMAT(20A1)
      CALL SMOVE(IA,1,4,IB,IEND+3)
      NT=NT+1
      CALL SFILL(IBUF,1,40,BLANK)
      DO 62 LT1=1,20
62    IBUF(LT1)=ITAB(NT,LT1)
      CALL SFILL(IA,1,40,BLANK)
      CALL CODE
      WRITE(IA,63)(IBUF(IX),IX=1,20)
63    FORMAT(20A1)
      CALL SMOVE(IA,1,2,IB,IEND+7)
      IE1=0
      IF(KON.EQ.1)GO TO 70
      CALL SDEA2(IBUCF,1,2,IE1)
      KON=1
      IF(N1.LT.10)GO TO 65
      K1=MOD(N1,10)
      K2=N1/10
      IE1=0
      CALL SDEA2(K2,1,2,IE1)
      CALL SDEA2(K1,1,2,IE1)
      CALL SMOVE(K2,2,2,NA,1)
      CALL SMOVE(K1,2,2,NA,2)
      GO TO 70
65    IE1=0
      CALL SDEA2(N1,1,2,IE1)
      CALL SMOVE(N1,1,2,NA,1)
70    CALL SMOVE(NA,1,2,IB,IEND+9)
      CALL SMOVE(IBUCF,1,2,IB,IEND+11)
      CALL WRITF(IDCB,IERR,IB,0,N2)
      IF(IERR.LT.0)GO TO 101
      GO TO 35
100   WRITE(20,103)
103   FORMAT(1H ,'RELATION ALREADY CREATED BEFORE,GIVE ANOTHER NAME')
      GO TO 500
101   WRITE(20,102)IERR
102   FORMAT(1H ,'IERR=',I4)
      STOP
500   CALL CLOSE(IDCB)
      CALL EXEC(ICDA,KOMN)
      END
C ----*************************************************************
C     * This segment helps to MODIFY tuples or to DELETE tuples  *
C     * from data file according to specifications .              *
C     *************************************************************
      PROGRAM UPDAT,5
      INTEGER IDCB(400),NAM7(3),NAM8(3),BLANK,ITAB1(30,22),IBUF(40),
     *IBUF1(144),IBUF2(128),NAM9(3),ITAB(30,42),IB(40),IDCB2
     *(400),NAM1(3),NAM2(3),IBUFL(20),ISECU(3),IBFR(5),NAM3(3),NAM4(3)
     *,ISTB(50,2),NAM5(3),NAM6(3),NAM10(3),NAM11(3),NAM12(3),NAM13(3)
     *,ISTB1(20,8),ICODE(20),IPARS(50,3),LLNAM(3),NAM14(3)
      COMMON ITAB,ITAB1,ISTB,MM,NANA,IVAR5,IPARS,IQRY,IF,ISTB1,KMM,ICODE
     *,JJNUM,JFLAG
```

```
      DATA NAM7,NAM8,NAM9,NAM1,NAM2,BLANK/2HDB,2HMS,2HRD,2HDB,2HMS,2H8 ,
     *2HDB,2HMS,2H11,2HCA,2HRD,2HAT,2HIS,2HUF,2HYL,000040B/
      DATA ISECU,NAM3,NAM4/2H-7,2H56,2H19,2HDB,2HMS,2H00,2HDB,2HMS,2H01/
      DATA NAM5,NAM6,NAM10,NAM11,NAM12/2HJU,2HRN,2HAL,2HTH,2HSI,2HS ,
     *2HJU,2HRF,2HLD,2HCO,2HNF,2HFD,2HCO,2HNF,2HNC/
      DATA NAM13,LLNAM,ICODX/2HUS,2HRC,2HOD,2HCL,2HEA,2HR ,8/
      DATA NAM14 / 2HAB,2HST,2HRC/
      ICHAR=0
      NUM=0
      IFLAG=0
      ITEST=0
      LVAL=0
C -----***  Contents of first row of array ITAB1 is transffered to IBUF *
C -----***  & characters are also counted                               *
C -----***  ITAB1 contains all IDENTIFIERS in query statement           *
      CALL SFILL(IBUF,1,40,BLANK)
      DO 2 KP=1,10
      IF(ITAB1(1,KP).EQ.1H ) GO TO 30
      ICHAR=ICHAR+1
2     IBUF(KP)=ITAB1(1,KP)
30    CALL HASS(IBUF,10,6,NUMB)
      IF(NUMB.EQ.0) NUMB=6
      CALL SFILL( IBUFL,1,40,BLANK)
C -----*** Conversion from A1 format to A2 format  ***-----
      CALL CODE
      WRITE(IBUFL,33) (IBUF(LK),LK=1,10)
33    FORMAT(10A1)
      IF ( JFLAG.EQ.1 ) GO TO 22
      CALL SFILL (IB,1,40,BLANK)
      CALL SFILL (ICODE,1,40,BLANK)
      DO 43 IJH=1,6
43    ICODE(IJH)=ISTB1(1,IJH)
C -----*** Conversion from A1 format to A2 Format  ***-----
      CALL CODE
      WRITE(IB,4) (ICODE(IG),IG=1,6)
4     FORMAT(6A1)
      CALL SMOVE(IBUFL,1,ICHAR,IB,7)
C -----*** File  USRCOD is opened to check the right of update for user *
      CALL OPEN(IDCB,IERR,NAM13,1,ISECU)
      IF(IERR.NE.2) GO TO 705
      CALL SFILL ( IBUF1,1,200,BLANK)
      CALL READF(IDCB,IERR,IBUF1,100,LEN,JJNUM)
      IF(IERR.LT.0) GO TO 705
      DO 5 JJJ= 1,10
      KSTRT=7+(JJJ-1)*18
      IF(JSCOM(IB,1,16,IBUF1,KSTRT,IER)) 5,6,5
5     CONTINUE
      GO TO 710
6     CALL CLOSE(IDCB)
C -----*** File DBMSRD is opened to check the relation name specified  **
```

```
C -----***----------------         by user       -------------------------------**
 22     CALL OPEN(IDCB,IERR,NAM7,1,ISECU)
        IF(IERR.NE.2) GO TO 705
        IRSIZ=30
 34     CALL SFILL(IBUF2,1,64,BLANK)
        CALL READF(IDCB,IERR,IBUF2,IRSIZ,LEN,NUMB)
        IF(IERR.LT.0) GO TO 705
        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
C -----*** Flag is set to 1 to indicate that relation name is   ***-----
C -----***--------          not in original directory         ---------***-----
        IFLAG=1
        CALL CLOSE(IDCB)
C -----*** File DBMS00 is opened to check the view relation name  ***-----
C -----***-----------               in VIEW directory           ---------***-----
        CALL OPEN(IDCB,IERR,NAM3,1,ISECU)
        IF(IERR.NE.2) GO TO 705
        IRSIZ=32
        CALL HASS(IBUF,ICHAR,19,NUMB)
        IF(NUMB.EQ.0) NUMB=19
        GO TO 34
 35     DO 36 I=1,5
        LSTRT=1+(I-1)*16
        NFLAG=7+(I-1)*7
        IF(JSCOM(IBUFL,1,ICHAR,IBUF2,LSTRT,1,IER)) 36,38,36
 36     CONTINUE
        WRITE(20,37) IBUFL
 37     FORMAT(1H ,"Relation name :-  ",2X,5A2,2X,/, "neither in  actual
       *directory nor in view directory ")
        GO TO 710
 38     CALL SMOVE(IBUF2,(2*NFLAG)-1,2*NFLAG,JVAR,1)
C -----***         Conversion from A2 format to I2 format         ***-----
C -----***         Status of view relation name is checked        ***-----
C -----***                in following few statements             ***-----
        CALL CODE
        READ(JVAR,16) JVAR1
 16     FORMAT(I2)
        IF(JVAR1.EQ.1) GO TO 14
        WRITE(20,15) IBUFL
 15     FORMAT(1H ,"View relation name suspended ",2X,5A2)
        GO TO 710
 14     CALL CLOSE(IDCB)
C -----*** File DBMS01 is opened to provided view attributes   ***-----
C -----***------     have ocurred in query statement       -----------***-----
        CALL OPEN(IDCB,IERR,NAM4,1,ISECU)
```

```
      IF(IERR.NE.2) GO TO 705
      IRSIZE=85
      GO TO 39
29    CALL HASS(IBUF,ICONT,31,NUMBR)
      IF(NUMBR.EQ.0) NUMBR=31
      GO TO 31
45    CALL SMOVE(IBUF2,(2*NFLAG)-1,2*NFLAG,JVAR,1)
C -----*** Conversion from A2 format to I2 format & status of    ***-----
C -----***            relation name is checked                  ***-----
      CALL CODE
      READ(JVAR,27) JVAR1
27    FORMAT(I2)
      IF(JVAR1.EQ.1) GO TO 47
      WRITE(20,28) IBUFL
28    FORMAT(1H ,5A2,2X,"RELATION NAME SUSPENDED")
      GO TO 710
47    CALL CLOSE(IDCB)
      ITEST=1
C -----*** A flag is set to 1 to indicate the presence of relation ***---
C -----*** in DBMSRD directory & file DBMS8 is opened . Attributes ***---
C -----*** are checked which have occured in query statement.  -----***---
      CALL OPEN(IDCB,IERR,NAM8,1,ISECU)
      IF(IERR.NE.2) GO TO 705
      IRSIZE=45
39    IBEG=2*ICHAR+1
      DO 55 IV=NANA,2,-1
      KAUNT=0
      CALL SFILL(IBUF2,1,20,BLANK)
      DO 46 IU=1,10
      IF(ITAB1(IV,IU).EQ.1H ) GO TO 44
      KAUNT=KAUNT+1
46    IBUF2(IU)=ITAB1(IV,IU)
44    CALL SMOVE(IBUF2,1,2*KAUNT,IBUF,IBEG)
      ICONT=KAUNT+ICHAR
      IF(ICONT.LT.20) CALL SFILL(IBUF,(2*ICONT)+1,40,BLANK)
      IF(ITEST.EQ.0) GO TO 29
      CALL HASS(IBUF,20,31,NUMBR)
31    CALL CODE
      WRITE(IBUFL,32) (IBUF(JK),JK=1,20)
32    FORMAT(20A1)
      CALL SFILL(IBUF2,1,170,BLANK)
      CALL READF(IDCB,IERR,IBUF2,IRSIZE,LEN,NUMBR)
      IF(IERR.LT.0) GO TO 705
      IF(ITEST.EQ.0) GO TO 90
      DO 48 IBUC=1,3
      JBEG=1+(IBUC-1)*32
      JEND=20+(IBUC-1)*32
      IF(JSCOM(IBUF2,JBEG,JEND,IBUFL,1,IER)) 48,50,48
48    CONTINUE
      WRITE(20,53)
```

```
 53    FORMAT(1H ,'ATTRIBUTE NOT IN BUCKET')
       GO TO 710
 50    LVAL=LVAL+1
       ISTRT=23+(IBUC-1)*32
       IEND= 28+(IBUC-1)*32
       CALL SMOVE(IBUF2,ISTRT,IEND,IBFR,1)
       CALL CODE
       READ(IBFR,83) ISTART,LENTH
 83    FORMAT(I4,I2)
 C -----*** According to relation names different data files are  ***----
 C -----***----------           opened for specific use         ---------***----
 84    IF((NUMB.EQ.1).AND.(NVAR.EQ.1)) GO TO 600
       IF((NUMB.EQ.2).AND.(NVAR.EQ.1)) GO TO 700
       IF((NUMB.EQ.2).AND.(NVAR.EQ.2)) GO TO 708
       IF((NUMB.EQ.4).AND.(NVAR.EQ.1)) GO TO 800
       IF((NUMB.EQ.4).AND.(NVAR.EQ.2)) GO TO 810
       IF((NUMB.EQ.5).AND.(NVAR.EQ.1)) GO TO 820
       IF((NUMB.EQ.5).AND.(NVAR.EQ.2)) GO TO 830
       IF((NUMB.EQ.6).AND.(NVAR.EQ.1)) GO TO 835
       WRITE(20,85)
 85    FORMAT(1H ,"RELATION NAME IS WRONG")
       GO TO 710
 90    DO 92 IL=1,4
       JSTRT=1+(IL-1)*36
       JEND=36+(IL-1)*36
       IF(JSCOM(IBUFL,1,ICONT,IBUF2,JSTRT,IER)) 92,94,92
 92    CONTINUE
       WRITE(20,93) IBUFL
 93    FORMAT(1H ,"Attribute concatenated with view relation ",2X,
      */,20A2,2X, " doesnot exists in directory ")
       GO TO 710
 94    LVAL=LVAL+1
       KSTRT=27+(IL-1)*36
       KEND=36+(IL-1)*36
       CALL SFILL(IBFR,1,10,BLANK)
       CALL SMOVE(IBUF2,KSTRT,KEND,IBFR,1)
 C -----***         Conversion from A format to I format           ***----
 C -----***                  in next few statements                ***----
       CALL CODE
       READ(IBFR,96)ISTART,LENTH,NUMB,NVAR
 96    FORMAT(I4,I2,I2,I2)
       IF(LVAL.EQ.1) GO TO 84
       GO TO 500
 600   CALL OPEN(IDCB2,IERR,NAM9,2)
       IF(IERR.NE.2) GO TO 705
       IRSIZ=128
       GO TO 500
 700   CALL OPEN(IDCB2,IERR,NAM1,2)
       IF(IERR.NE.2) GO TO 705
       IRSIZ=66
      GO TO 500
708   CALL OPEN(IDCB2,IERR,NAM5,2,ISECU)
      IF(IERR.NE.2) GO TO 705
      IRSIZ=128
      GO TO 500
800   CALL OPEN(IDCB2,IERR,NAM2,2)
      IF(IERR.NE.2) GO TO 705
      IRSIZ=50
      GO TO 500
810   CALL OPEN(IDCB2,IERR,NAM6,2,ISECU)
      IF(IERR.NE.2) GO TO 705
      IRSIZ=128
      GO TO 500
820   CALL OPEN(IDCB2,IERR,NAM10,2,ISECU)
      IF(IERR.NE.2) GO TO 705
      IRSIZ=100
      GO TO 500
830   CALL OPEN(IDCB2,IERR,NAM11,2,ISECU)
      IF(IERR.NE.2) GO TO 705
      IRSIZ=128
      GO TO 500
835   CALL OPEN(IDCB2,IERR,NAM12,2,ISECU)
      IF(IERR.NE.2) GO TO 705
      IRSIZ = 128
500   CALL SFILL(IB,1,80,BLANK)
      DO 601 ID=1,40
601   IB(ID)=ITAB(IV-1,ID)
19    CALL SFILL(IBUFL,1,40,BLANK)
C -----*** Conversion from A1 format to A2 format  ***------
      CALL CODE
      WRITE(IBUFL,18) (IB(L),L=1,40)
18    FORMAT(40A1)
      DO 690 IH=1,10
      IF(NUM.GE.1) GO TO 696
      IREC=IH
C -----*** Record size of data files are assigned here according  ***-----
C -----***     to relation name specified in query statement     ***-----
5705  CALL SFILL ( IBUF2,1,2*IRSIZ,BLANK)
      CALL READF(IDCB2,IERR,IBUF2,IRSIZ,LEN,IREC)
      IF(IERR.LT.0) GO TO 705
      IF(JSCOM(IBUF2,ISTART,ISTART+(LENTH-1),IBUFL,1,IER)) 690,692,690
690   CONTINUE
      IF((NUM.EQ.IVAR5-1).AND.((ISTB(1,1).EQ.1).AND.(ISTB(1,2).EQ.16
     *))) GO TO 698
617   WRITE(20,691) (IBUFL,I=1,20)
691   FORMAT(1H ,'No record corresponds to literal :-',2X,20A2)
      GO TO 710
692   NUM=NUM+1
55    CONTINUE
      IF((NUM.EQ.IVAR5).AND.((ISTB(1,1).EQ.1).AND.(ISTB(1,2).EQ.16)
     *)) GO TO 694
      IF((NUM.EQ.IVAR5).AND.((ISTB(1,1).EQ.1).AND.(ISTB(1,2).EQ.6)
     *)) GO TO 695
698   WRITE(20,976) ( IBUF2(IF),IF=1,IRSIZ)
976   FORMAT(1H ," Tuple before update is :- ",/,2X,128A2)
      CALL SMOVE(IBUFL,1,LENTH,IBUF2,ISTART)
      WRITE(20,977) (IBUF2(IF),IF=1,IRSIZ)
977   FORMAT(1H ,"Updated record",/,2X,128A2)
678   CALL WRITF(IDCB2,IERR,IBUF2,0,IREC)
      IF(IERR.LT.0) GO TO 705
      IF((ISTB(1,1).EQ.1).AND.(ISTB(1,2).EQ.16)) GO TO 694
      M=10-IREC
      IF(M.EQ.0) GO TO 694
      DO 680 J=1,M
      IP=J+IREC
      IQ=IP-1
      CALL SFILL(IBUF2,1,IRSIZ,BLANK)
      CALL READF(IDCB2,IERR,IBUF2,IRSIZ,LEN,IP)
      IF(IERR.LT.0) GO TO 705
      CALL WRITF(IDCB2,IERR,IBUF2,0,IQ)
      IF(IERR.LT.0) GO TO 705
680   CONTINUE
      CALL SFILL(IBUF2,1,IRSIZ,BLANK)
      CALL WRITF(IDCB2,IERR,IBUF2,0,IP)
      IF(IERR.LT.0) GO TO 705
694   CALL CLOSE(IDCB2)
      WRITE(20,677)
677   FORMAT(1H ,"QUERY PROCESSED ")
      GO TO 710
695   WRITE(20,701) (IBUF2(IRR),IRR=1,IRSIZ)
701   FORMAT(1H , " Tuple to be deleted is :-",/,2X,128A2)
      CALL SFILL( IBUF2,1,2*IRSIZ,BLANK)
      GO TO 678
696   IREC=IREC
      GO TO 5705
C -----*** File manager errors are issued to user if he tries to ****-----
C        * access wrong relation name or data file                *
705     WRITE(1,709) IERR
709     FORMAT(1H ," FMGR ERROR ",2X,I4)
710     CALL EXEC( ICODX,LLNAM)
      END
C -----*** This segment is to define new views on existing relation  ***-
C       ***-------           name & its attributes            ----------***--
      PROGRAM DEFVE,5
      INTEGER IDCB(400),NAM8(3),IBUFR(40),IBUF(128),IBUF1(20),BLANK,
     *IHOST(128),IPARS(50,3),ITAB1(30,22),ITEM(5),IBUK(5),ISECU(3),
     *IBLNK(20),ISTOR(6,30),NAMR(3),ICC(5),NAM5(3),ITAB(30,42)
     *,IBUFL(10),LVAR(2),IDCB1(144),NAM1(3),ISTB(50,2),LRNAM(3)
      COMMON ITAB,ITAB1,ISTB,N,NANA,IVAR5,IPARS
      DATA NAM8,NAMR,ISECU,NAM5/2HDB,2HMS,2H00,2HDB,2HMS,2H01,2H-7,2H56,
     *2H19,2HDB,2HMS,2H8 /
      DATA NAM1,LRNAM,ICODC /2HDB,2HMS,2HRD,2HCL,2HEA,2HR ,8/
      ICAR=0
      INUM=0
      KAUNT=0
      IDAT=0
      IBEG=0
      IVAR3=0
      IVAR9=0
      IXZ=0
      IVEW=0
      IER=0
      IBAR=0
      INAM=0
      BLANK=000040B
      DO 10 I=1,N
C -----*** IDENTIFIERS ( relation name concatenated with its      ***-----
C     ***                attributes ) are checked                  ***
      IF((IPARS(I,1).EQ.4).AND.((IPARS(I-1,1).EQ.2).AND.(IPARS(I-1,2).
     *EQ.32)).AND.((IPARS(I+1,1).EQ.2).AND.(IPARS(I+1,2).EQ.33)))
     * GO TO 110
10    CONTINUE
      IF(INUM.EQ.IDAT) GO TO 135
      WRITE(20,20)  I
20    FORMAT(1H ,"No record corresponds to identfier:-",2X,I2)
      GO TO 288
110   IDAT=IDAT+1
      CALL SFILL(IBUFR,1,40,BLANK)
C -----*** NO. of characters are counted in IDENTIFIERS .          ***----
C     ***  Table ITAB1 contains all identifiers coming in a         ***
C     ***  query statement. Contents of ITAB1 are transferred to    ***
C     ***          a buffer IBUFR                                   ***
      DO 115 II=1,20
      IF(ITAB1(IPARS(I,2),II).EQ.1H ) GO TO 116
      IBUFR(II)=ITAB1(IPARS(I,2),II)
115   ICAR=ICAR+1
116   IF(ICAR.NE.20) CALL SFILL(IBUFR,(2*ICAR)+1,40,BLANK)
      CALL HASS(IBUFR,20,31,NUMBR)
C -----*** File DBMS8 is opened to check the concatenated relation with *
C     ***-------------               its attributes .          ----------**
      CALL OPEN(IDCB,IERR,NAM5,1,ISECU)
      IF(IERR.NE.2) GO TO 299
      CALL SFILL(IBUF,1,256,BLANK)
      CALL READF(IDCB,IERR,IBUF,45,LEN,NUMBR)
      IF(IERR.LT.0) GO TO 299
C -----*** Conversion from A1 format to A2 format  ***----
      CALL CODE
      WRITE(IBLNK,113) (IBUFR(IX),IX=1,20)
113   FORMAT(20A1)
C -----*** Concatenated relation & attribute is searched in bucket ***---
```

```
      DO 120 IBUC=1,3
      JBEG=1+(IBUC-1)*32
      JEND=20+(IBUC-1)*32
      IF(JSCOM(IBUF,JBEG,JEND,IBLNK,1,IER)) 120,130,120
120   CONTINUE
      WRITE(20,122) IBLNK
122   FORMAT(1H ," You are giving Wrong RELATION & ATTRIBUTE  in ",
     */,1H ,"Hashed FORM:_",2X,10A2)
      GO TO 288
130   INUM=INUM+1
      CALL SFILL(IBUF1,1,60,BLANK)
      MBEG=21+(IBUC-1)*32
      MEND=32+(IBUC-1)*32
      CALL CONVR(NUMBR,IHASH)
      CALL CONVR(IBUC,IBUKET)
      IBUF1(1)=IHASH
      IBUF1(2)=IBUKET
      CALL SMOVE(IBUF,MBEG,MEND,IBUF1,5)
      CALL SMOVE(IBUF1,13,16,LVAR,1)
C -----*** Conversion from A format to I format ***-----
      CALL CODE
      READ(LVAR,128) JFILE,JBUKET
128   FORMAT(I2,I2)
      IBUFL(INUM)=JFILE
      IF(INUM.EQ.1) GO TO 235
      IF(INUM.GT.1) GO TO 133
129   DO 131 IVAR=1,8
131   ISTOR(IDAT,IVAR)=IBUF1(IVAR)
      GO TO 10
133   IF(IBUFL(INUM).EQ.IBUFL(INUM-1)) GO TO 129
      WRITE(20,134) (IBUFL(KO),KO=1,INUM)
134   FORMAT(1H ,"You are defining view on different relations",5I2)
      GO TO 288
C -----*** File DBMSRD is opened to check the relation name on which   ***
C     ***-----        user is going to define his own view          -----***
235   CALL OPEN(IDCB1,IERR,NAM1,1,ISECU)
      IF(IERR.NE.2) GO TO 299
      KSTRT=11+(JBUKET-1)*12
      CALL SFILL ( IBUF,1,60,BLANK)
      CALL READF(IDCB1,IERR,IBUF,30,LEN,JFILE)
      IF(IERR.LT.0) GO TO 299
C -----*** Relation's status is checked in few next statements whether  *
C     ***-----         relation is suspended or restored               -----*
      CALL SMOVE(IBUF,KSTRT,KSTRT+1,MVAR,1)
      CALL CODE
      READ(MVAR,236) MVAL
236   FORMAT(I2)
      IF(MVAL.EQ.1) GO TO 129
      WRITE(20,337)
337   FORMAT(1H ,"RELATION NAME IS SUSPENDED ON WHICH YOU ARE DEF. VEW")
```

```
      GO TO 288
135   CALL SMOVE(IBUF1,13,16,LVAR,1)
      CALL CLOSE(IDCB)
C -----*** File DBMS00 is opened in which user will check the existence *
C     *** of view relation name & entry will be given with its status   *
C     ***---   provided it is not existing in this directory        ----**
      CALL OPEN(IDCB,IERR,NAM8,2,ISECU)
      IF(IERR.NE.2) GO TO 299
      CALL SFILL(IBUFR,1,40,BLANK)
      CALL SFILL(IBLNK,1,40,BLANK)
      DO 136 IM=1,10
      IF(ITAB1(1,IM).EQ.1H ) GO TO 147
      IVAR3=IVAR3+1
136   IBUFR(IM)=ITAB1(1,IM)
147   CALL HASS(IBUFR,IVAR3,19,NUMB)
      IF(NUMB.EQ.0) NUMB=19
      CALL SFILL(IBUF,1,128,BLANK)
      CALL READF(IDCB,IERR,IBUF,32,LEN,NUMB)
C -----***    conversion from A1 format to A2 format        ***------
      CALL CODE
      WRITE(IBUF1,137) (IBUFR(IT),IT=1,10)
137   FORMAT(10A1)
      DO 138 IBUC=1,5
      ISTRT=1+(IBUC-1)*16
      IEND=10+(IBUC-1)*16
      IF(IVAR3.LT.10) CALL SFILL(IBUF1,IVAR3+1,10,BLANK)
      IF(JSCOM(IBUF,ISTRT,IEND,IBUF1,1,IER)) 148,139,148
148   IF(JSCOM(IBUF,ISTRT,IEND,IBLNK,1,IER)) 138,159,138
159   IF(IXZ.EQ.1) GO TO 138
      IXZ=IXZ+1
      ITERM=NUMB
      IBAK=IBUC
138   CONTINUE
      DO 151 IVAX1=20,22
      CALL SFILL(IBUF,1,64,BLANK)
      CALL READF(IDCB,IERR,IBUF,32,LEN,IVAX1)
      IF(IERR.LT.0) GO TO 299
      DO 151 IVAX=1,5
      ISTRT=1+(IVAX-1)*16
      IEND=10+(IVAX-1)*16
      IF(JSCOM(IBUF,ISTRT,IEND,IBUF1,1,IER)) 154,139,154
154   IF(JSCOM(IBUF,ISTRT,IEND,IBLNK,1,IER)) 151,155,151
155   IF(IXZ.EQ.1) GO TO 151
      IXZ=IXZ+1
      ITERM=IVAX1
      IBAK=IVAX
151   CONTINUE
      IF(IXZ.EQ.1) GO TO 160
      WRITE(20,150)
150   FORMAT(1H ,"Please give some another view relation name",
        */,"all place of bucket in hashed record no. & overflow area",
     ,  */,"is full ")
        GO TO 288
  160   CALL SFILL(IBUF,1,256,BLANK)
        CALL READF(IDCB,IERR,IBUF,128,LEN,ITERM)
        LBEG=1+(IBAK-1)*16
        LEND=16+(IBAK-1)*16
C -----*** View relation name with its restored status is given   ***----
C       ***-------------         in DBMS00 directory        ----------------***
        NVAR=1
        CALL CONVR(NVAR,NVAR1)
        CALL SMOVE(NVAR1,1,2,IBUF1,11)
        CALL SMOVE(LVAR,1,4,IBUF1,13)
        CALL SMOVE(IBUF1,1,16,IBUF,LBEG)
        CALL WRITF(IDCB,IERR,IBUF,0,ITERM)
        IF(IERR.LT.0) GO TO 299
        GO TO 237
  139   KAUNT=1
  237   CALL CLOSE(IDCB)
C -----*** File DBMS01 is opened where existence of attributes will be **
C       ***------------------------------------          checked .       -------***-
        CALL OPEN(IDCB,IERR,NAMR,2,ISECU)
        IF(IERR.NE.2) GO TO 299
        DO 280 IS=2,NANA,2
        IYZ=0
        KOUNT=0
        IVAR9=IVAR9+1
        CALL SFILL(IBUF1,1,40,BLANK)
        DO 240 ISS=1,10
        IF(ITAB1(IS,ISS).EQ.1H ) GO TO 245
        IBUF1(ISS)=ITAB1(IS,ISS)
  240   KOUNT=KOUNT+1
  245   CALL SMOVE(IBUF1,1,2*KOUNT,IBUFR,(2*IVAR3)+1)
        IVAR4=IVAR3+KOUNT
        CALL SFILL(IBUFR,(2*IVAR4)+1,40,BLANK)
        CALL HASS(IBUFR,IVAR4,31,NUMBR)
        IF(NUMBR.EQ.0) NUMBR=31
        ICC(IVAR9)=NUMBR
        IF(IVAR9.GT.1) GO TO 230
  246   CALL SFILL(IBLNK,1,40,BLANK)
        CALL SFILL(IBUF1,1,120,BLANK)
        CALL CODE
        WRITE(IBUF1,249) (IBUFR(IU),IU=1,20)
  249   FORMAT(20A1)
        CALL SMOVE(IBUF1,1,20,IBUF,1)
        CALL SFILL(IBLNK,1,40,BLANK)
C -----***    Attributes are searched in bucket of DBMS8 directory   ***-
        DO 247 IMM=1,8
  247   IBLNK(IMM)=ISTOR(IVAR9,IMM)
        CALL SMOVE(IBLNK,1,16,IBUF,21)
        CALL SFILL(IHOST,1,256,BLANK)
        CALL READF(IDCB,IERR,IHOST,85,LEN,NUMBR)
        IF(IERR.LT.0) GO TO 299
        CALL SFILL(IBLNK,1,40,BLANK)
        DO 258 IVAX2=1,4
        NBEG=1+(IVAX2-1)*36
        NEND=36+(IVAX2-1)*36
        IF(JSCOM(IHOST,NBEG,NEND,IBUF,1,IER)) 252,270,252
  252   IF(JSCOM(IHOST,NBEG,NEND,IBLNK,1,IER)) 258,255,258
  255   IF(IYZ.EQ.1) GO TO 258
        IYZ=IYZ+1
        ITEM(IVAR9)=NUMBR
        IBUK(IVAR9)=IVAX2
  258   CONTINUE
  C -----*** Attributes are searched in overflow area of directory ***-----
        DO 260 IVAX1=32,34
        CALL SFILL(IHOST,1,256,BLANK)
        CALL READF(IDCB,IERR,IHOST,85,LEN,IVAX1)
        IF(IERR.LT.0) GO TO 299
        DO 260 IBHAR=1,4
        NBEG=1+(IBHAR-1)*36
        NEND=36+(IBHAR-1)*36
        IF(JSCOM(IHOST,NBEG,NEND,IBUF,1,IER)) 262,270,262
  262   IF(JSCOM(IHOST,NBEG,NEND,IBLNK,1,IER)) 260,265,260
  265   IF(IYZ.EQ.1) GO TO 260
        IYZ=IYZ+1
        ITEM(IVAR9)=IVAX1
        IBUK(IVAR9)=IBHAR
  260   CONTINUE
        IF(IYZ.EQ.1) GO TO 273
  261   WRITE(20,271)
  271   FORMAT(1H ,"Please give some another view relation name ",
       */,1H ,"this view relation has been defined by some user ")
        GO TO 288
  230   IF(ICC(IVAR9).EQ.ICC(IVAR9-1)) GO TO 261
        GO TO 246
  273   IBAR=IBAR+1
        DO 272 IVAR8=1,18
  272   ISTOR(IBAR,IVAR8)=IBUF(IVAR8)
        GO TO 280
  270   IVEW=IVEW+1
  280   CONTINUE
        IF((IVEW.EQ.IVAR9).AND.(KAUNT.EQ.1)) GO TO 290
        IF((KAUNT.EQ.0).AND.(IVEW.EQ.0)) GO TO 275
        GO TO 261
  275   DO 285 IW=1,IVAR9
        CALL SFILL(IHOST,1,256,BLANK)
        ITERM=ITEM(IW)
        IBHAR=IBUK(IW)
        CALL READF(IDCB,IERR,IHOST,128,LEN,ITERM)
```

```
      IF(IERR.LT.0) GO TO 299
      LBEG=1+(IBHAR-1)*36
      LEND=36+(IBHAR-1)*36
      CALL SFILL(IBUF1,1,48,BLANK)
      DO 276 IE=1,18
276   IBUF1(IE)=ISTOR(IW,IE)
      CALL SMOVE(IBUF1,1,36,IHOST,LBEG)
      CALL WRITF(IDCB,IERR,IHOST,0,ITERM)
      IF(IERR.LT.0) GO TO 299
285   CONTINUE
      CALL CLOSE(IDCB)
286   WRITE(20,287)
287   FORMAT(1H ,"Query is processed & entries in view relation ",
     */,"directory & view attribute directories are made  ")
288   CALL EXEC(ICODC,LRNAM)
290   WRITE(20,291)
291   FORMAT(1H ,"Your defined.view exists in directory")
      GO TO 288
C -----***  File manager errors are given to user in case of accessing **
C     ***  wrong directory or beyond the directory                      **
299   WRITE(1,303) IERR
303   FORMAT(1H ,"** FMGR ERROR **:",2X,I5)
      END
C -----*** This segment either RESTORES or SUSPENDS a relation   ***-----
C --------*------          name as needed by user              ---------***-----
      PROGRAM RESTT,5
      INTEGER IDCB(144),IBUFR(20),IBUFL(20),NAME(3),LNAM(3),ISECU(3),
     *BLANK,ITAB1(30,22),LBUFR(85),ISTB(50,2),ITAB(30,42),IBB(50,3),
     *LLNAM(3),ICODE(20),ISTB1(20,8),LRNAM(3)
      COMMON ITAB,ITAB1,ISTB,LKM,JNN,JJVAR,IBB,IQRY,IF,ISTB1,KMM,ICODE
     *,JJNUM,JFLAG
      DATA NAME,LNAM,ISECU/2HDB,2HMS,2HRD,2HDB,2HMS,2H00,2H-7,2H56,2H19/
      DATA LLNAM,LRNAM,ICODC/2HUS,2HRC,2HOD,2HCL,2HEA,2HR ,8/
      BLANK=000040B
      KOUNT=0
C     ***  characters are counted  .                                     ***
      DO 10 I=1,10
      IF(ITAB1(1,I).EQ.1H ) GO TO 15
      IBUFR(I)=ITAB1(1,I)
10    KOUNT=KOUNT+1
15    IF(KOUNT.LT.10) CALL SFILL(IBUFR,(2*KOUNT)+1,20,BLANK)
C -----*** Relation name is hashed & an address is obtained      ***-----
      CALL HASS(IBUFR,10,6,NUMB)
      IF(NUMB.EQ.0) NUMB=6
C -----*** Conversion from A1 format to A2 format ***-----
      CALL CODE
      WRITE(IBUFL,20) (IBUFR(J),J=1,10)
20    FORMAT(10A1)
      IF(JFLAG.EQ.1) GO TO 28
      CALL CODE
      WRITE(IBUFR,25) (ISTB1(1,L),L=1,6)
25    FORMAT(6A1)
      CALL SMOVE(IBUFL,1,KOUNT,IBUFR,7)
C -----*** File USRCOD is opened to check the right of user ( either   **
C     *** to restore or suspend the relation name or VIEW relation name
      CALL OPEN(IDCB,IERR,LLNAM,1,ISECU)
      IF(IERR.NE.2) GO TO 45
      CALL SFILL(LBUFR,1,170,BLANK)
      CALL READF(IDCB,IERR,LBUFR,85,LEN,JJNUM)
      IF(IERR.LT.0) GO TO 45
      DO 24 I=1,10
      KSTRT=7+(I-1)*18
      IF(JSCOM(IBUFR,1,6+KOUNT,LBUFR,KSTRT,IER)) 24,26,24
24    CONTINUE
      WRITE(20,29)
29    FORMAT(1H ,"You can't do so  ")
      GO TO 65
26    CALL CLOSE(IDCB)
C -----*** File DBMSRD is opened to check the relation name which user **
C     ***           wants to either suspend or restore               ***-
28    CALL OPEN(IDCB,IERR,NAME,2,ISECU)
      IF(IERR.NE.2) GO TO 45
      CALL SFILL(LBUFR,1,60,BLANK)
      CALL READF(IDCB,IERR,LBUFR,30,LEN,NUMB)
      IF(IERR.LT.0) GO TO 45
      IER=0
      DO 30 IVAR=1,4
      LSTRT=1+(IVAR-1)*12
      LEND=10+(IVAR-1)*12
      IF(JSCOM(IBUFL,1,10,LBUFR,LSTRT,IER)) 30,35,30
30    CONTINUE
      GO TO 50
35    IF((ISTB(1,1).EQ.1).AND.(ISTB(1,2).EQ.14)) GO TO 40
C -----*** This portion of program restores the relation name      ***---
      N=1
36    CALL CONVR(N,M)
      CALL SMOVE(M,1,2,LBUFR,LEND+1)
      CALL WRITF(IDCB,IERR,LBUFR,0,NUMB)
      IF(IERR.LT.0) GO TO 45
      CALL CLOSE(IDCB)
      GO TO 63
C -----*** This portion of program suspends the relation name       ***--
40    N=0
      GO TO 36
C -----*** File manager error is given to user if he tries to acess ***--
C -----*** the wrong directory or beyond the directory size         ***--
45    WRITE(1,48) IERR
48    FORMAT(1H ,"**** FMGR ERROR ******",2X,I4)
      GO TO 65
49    NVAR=0
```

```
      GO TO 61
50    CALL CLOSE(IDCB)
C -----*** File DBMS00 is opened to check the view relation name if   ***
C     ***   relation name is not found in DBMSRD directory             ***
      CALL OPEN(IDCB,IERR,LNAM,2,ISECU)
      IF(IERR.NE.2) GO TO 45
      CALL HASS(IBUFR,KOUNT,19,NUMBR)
      IF(NUMBR.EQ.0) NUMBR=19
      CALL SFILL(LBUFR,1,170,BLANK)
      CALL READF(IDCB,IERR,LBUFR,85,LEN,NUMBR)
      IF(IERR.LT.0) GO TO 45
C -----***         View relation name is searched in bucket         ***---
      DO 55 IVAR=1,5
      LSTRT=1+(IVAR-1)*16
      LEND=10+(IVAR-1)*16
      IF(JSCOM(IBUFL,1,KOUNT,LBUFR,LSTRT,IER)) 55,60,55
55    CONTINUE
      WRITE(20,58) IBUFL
58    FORMAT(1H ,"Relation name :-",2X,5A2,2X,"Neither in actual,/,
     * directory nor in view directory ")
      GO TO 65
C-----*** Next statement checks whether user wants to either  suspend   *
C -----***          or restore the view relation name.              ***-
60    IF((ISTB(1,1).EQ.1).AND.(ISTB(1,2).EQ.14)) GO TO 49
C -----*** View relation name is restored in next few statements    ***--
      NVAR=1
61    CALL CONVR(NVAR,LVAR)
      CALL SMOVE(LVAR,1,2,LBUFR,LEND+1)
      CALL WRITF(IDCB,IERR,LBUFR,0,NUMBR)
      CALL CLOSE(IDCB)
63    WRITE(20,64)
64    FORMAT(1H ,"QUERY IS PROCESSED ")
65    CALL EXEC(ICODC,LRNAM)
      END
C     ***************************************************************
C     * This segment helps the DATA BASE ADMINISTRATOR to grant some  *
C     * grant opitions ( like TO MODIFY , TO DELETE , TO INSRTN )     *
C     * to some of his responsible person.                            *
C     ***************************************************************
      PROGRAM GRNTT,5
      INTEGER IDCB(400),NAM7(3),NAM8(3),BLANK,ITAB1(30,22),IBUF(40),
     *IBUF1(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,IVAR5,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
      LSTRT=1+(I-1)*14
      NFLAG=7+(I-1)*7
      IF(JSCOM(IBUFL,1,ICHAR,IBUF2,LSTRT,IER)) 36,38,36
36    CONTINUE
      WRITE(20,37) IBUFL
37    FORMAT(1H ,"Relation name :-",2X,5A2,2X,is neither ",/,
     *in view nor in actual directory  ")
      GO TO 105
C -----*** Status of view relation name is checked whether it is  ***-----
C     *** suspended or restored on which DBA is going to grant     ***
C     *** some grant options.                                      ***
```

```
38    CALL SMOVE(IBUF2,(2*NFLAG)-1,2*NFLAG,JVAR,1)
      CALL CODE
      READ(JVAR,16) JVAR1
16    FORMAT(I2)
      IF(JVAR1.EQ.1) GO TO 47
      WRITE(20,15) IBUFL
15    FORMAT(1H ,"View relation name is suspended:-",2X,5A2)
      GO TO 105
C -----*** Status of relation name is checked in next few statements ***-
45    CALL SMOVE(IBUF2,(2*NFLAG)-1,2*NFLAG,JVAR,1)
      CALL CODE
      READ(JVAR,27) JVAR1
27    FORMAT(I2)
      IF(JVAR1.EQ.1) GO TO 47
      WRITE(20,28) IBUFL
28    FORMAT(1H ,5A2,2X,"Relation name is suspended")
      GO TO 105
47    CALL CLOSE(IDCB)
C -----*** File USRCD is opened to check the validity of user ***
      CALL OPEN(IDCB,IERR,NAM8,2,ISECU)
      IF(IERR.NE.2) GO TO 99
      CALL SFILL(IBUF,1,40,BLANK)
      DO 48 I=1,6
48    IBUF(I)=ITAB(1,I)
      CALL HASS(IBUF,6,7,NUMBR)
      IF(NUMBR.EQ.2) GO TO 72
      CALL CODE
C     ***  Conversion from A1 format to A2 format in next few statements
      WRITE(IB,50) (IBUF(II),II=1,6)
50    FORMAT(6A1)
      IF (NUMBR.EQ.3) GO TO 96
      CALL SFILL(IBUF2,1,90,BLANK)
      CALL READF(IDCB,IERR,IBUF2,45,LEN,NUMBR)
      IF(IERR.LT.0) GO TO 99
      IF(JSCOM(IBUF2,1,6,IB,1,IER)) 55,60,55
55    WRITE(20,58) (IB(IJ),IJ=1,3)
58    FORMAT(1H ,3A2,2X,"Not authorised user ")
      STOP
60    CALL SFILL(IBUF,1,40,BLANK)
C     *** File USRCOD is opened where granted opitions will be entred   *
C       *           against grantor name                                *
      CALL OPEN(IDCB1,IERR,NAM1,2,ISECU)
      IF(IERR.NE.2) GO TO 99
      CALL SFILL(IBUF1,1,256,BLANK)
      CALL READF(IDCB1,IERR,IBUF1,128,LEN,NUMBR)
      IF(IERR.LT.0) GO TO 99
C -----*** Conversion from A1 format to A2 format        ***
      CALL CODE
      WRITE(IBUFZ,59) (ICODE(L),L=1,6)
59    FORMAT(6A1)
```

```
       DO 120 IJ=1,10
120    IBUFF(IJ)=ITAB1(NANA,IJ)
       CALL CODE
       WRITE(IBUFL,121) (IBUFF(IK),IK=1,10)
121    FORMAT(10A1)
64     DO 90 IVAR=2,KM-2
       CALL SFILL(IBUFF,1,40,BLANK)
       DO 65 IVAR1 = 1,6
65     IBUFX(IVAR1)=ISTB1(IVAR,IVAR1)
       CALL CODE
       WRITE(IBUFY,62) (IBUFX(KP),KP=1,6)
62     FORMAT(6A1)
       CALL SMOVE(IBUFY,1,6,IBUFF,1)
       CALL SMOVE(IBUFL,1,10,IBUFF,7)
       DO 70 IVAR2=1,10
       ISTRT=7+(IVAR2-1)*8
       IEND=12+(IVAR2-1)*8
       IF(JSCOM(IBUF2,ISTRT,IEND,IBUFY,1,IER)) 70,75,70
70     CONTINUE
       WRITE(20,71)IBUFY
71     FORMAT(1H ,"Key word not available is:",2X,3A2)
       GO TO 105
72     WRITE(20,73)
73     FORMAT(1H ,"You are giving GRANT right to yourself")
       GO TO 105
96     WRITE(20,97) (IB(LLL),LLL=1,3)
97     FORMAT(1H ," You can´t give your grant right to user :- ",3A2)
       GO TO 105
75     IF((ISTB(1,1).EQ.1).AND.(ISTB(1,2).EQ.15)) GO TO 74
       M=1
78     CALL CONVR(M,N)
       CALL SMOVE(N,1,2,IBUFF,17)
74     DO 77 IVAR3=1,10
       JSTRT=7+(IVAR3-1)*18
       JEND=22+(IVAR3-1)*18
       IF((ISTB(1,1).EQ.1).AND.(ISTB(1,2).EQ.15)) GO TO 85
       IF(JSCOM(IBUF1,JSTRT,JEND,IBUFF,1,IER)) 76,82,76
76     IF(JSCOM(IBUF1,JSTRT,JEND,IBUF,1,IER)) 77,79,77
79     CALL SMOVE(IBUFF,1,18,IBUF1,JSTRT)
       CALL SMOVE(IB,1,6,IBUF1,1)
       GO TO 90
85     IF(JSCOM(IBUF1,JSTRT,JEND,IBUFF,1,IER)) 77,92,77
86     WRITE(20,89)
89     FORMAT(1H,"You can revoke only what you have granted ")
       GO TO 105
92     CALL SFILL(IBUF1,JSTRT,JEND+2,BLANK)
       GO TO 90
77     CONTINUE
       IF((ISTB(1,1).EQ.1).AND.(ISTB(1,2).EQ.15)) GO TO 86
       WRITE(20,81)
```

```
81    FORMAT(1H ,"NO PLACE")
      GO TO 105
82    CALL SMOVE(N,1,2,IBUF1,JEND+1)
90    CONTINUE
C     *** Entry is made in USRCOD directory for granted rights    ***
      CALL WRITF(IDCB1,IERR,IBUF1,0,NUMBR)
      IF(IERR.LT.0) GO TO 99
      CALL CLOSE(IDCB)
      CALL CLOSE(IDCB1)
      WRITE(20,93)
93    FORMAT(1H ,"Query is processed & corresponding entries are made in
     * directory ")
      GO TO 105
C     *** FMGR errors are issued to user in case he must have tried   **
C       * to access wrong relation name or beyond the directory size  **
99    WRITE(1,100) IERR
100   FORMAT(1H ,"FMGR ERROR *******",2X,I4)
105   CALL EXEC(ICODC,LRNAM)
      END
C ----*************************************************************
C ----*** This segment is to insert new tuple in data file    ***----
C    **************************************************************
      PROGRAM INSTN,5
      INTEGER IDCB(400),NAM7(3),NAM8(3),BLANK,ITAB1(30,22),IBUF(40),
     *IBUF1(144),IBUF2(128),NAM9(3),ITAB(30,42),IB(40),IDCB2(400)
     *,NAM1(3),NAM2(3),IBUFL(20),ISECU(3),IBFR(5),NAM3(3),NAM4(3)
     *,ISTB(50,2),TEMP(20),TEMP1(20),IPARS(50,3),NAM5(3),NAM6(3),NAM10
     *(3),NAM11(3),NAM12(3),NAM13(3),LLNAM(3),ISTB1(20,8),ICODE(20)
     *,NAM14(3)
      COMMON ITAB,ITAB1,ISTB,MM,NANA,IVAR5,IPARS,IQRY,IF,ISTB1,KKM,ICODE
     *,JJNUM,JFLAG
      DATA NAM7,NAM8,NAM9,NAM1,NAM2,BLANK/2HDB,2HMS,2HRD,2HDB,2HMS,2H8 ,
     *2HDB,2HMS,2H11,2HCA,2HRD,2HAT,2HIS,2HUF,2HYL,000040B/
      DATA ISECU,NAM3,NAM4/2H-7,2H56,2H19,2HDB,2HMS,2H00,2HDB,2HMS,2H01/
      DATA NAM13,LLNAM,ICODX /2HUS,2HRC,2HOD,2HCL,2HEA,2HR ,8/
      DATA NAM5,NAM6,NAM10,NAM11,NAM12,NAM14 /2HJU,2HRN,2HAL,2HTH,2HSI
     *,2HS ,2HJU,2HRF,2HLD,2HCO,2HNF,2HLD,2HCO,2HNF,2HRC,2HAB,2HST,2HRC/
      ICHAR=0
      NUM=0
      IFLAG=0
      ITEST=0
C ----*****************************************************************
C     * Array ITAB1 keeps IDENTIFIERS occuring in  query statement
C     * Number of characters are counted in first row of ITAB1 &
C     * transfered to a buffer IBUF
C ----*****************************************************************
      DO 2 KP=1,10
      IF(ITAB1(1,KP).EQ.1H ) GO TO 30
      ICHAR=ICHAR+1
2     IBUF(KP)=ITAB1(1,KP)
```

```
30    IRSIZ=30
      CALL HASS(IBUF,10,6,NUMB)
      IF(NUMB.EQ.0) NUMB=6
C -----*** Conversion from A1 format to A2 format   ***-----
      CALL CODE
      WRITE(IBUFL,33) (IBUF(LK),LK=1,10)
33    FORMAT(10A1)
      IF(JFLAG.EQ.1) GO TO 26
C -----***  Conversion from A1 format to A2 format   ***-----
      CALL CODE
      WRITE(IBUF2,11) (ISTB1(1,L),L=1,6)
11    FORMAT(6A1)
      CALL SMOVE(IBUFL,1,ICHAR,IBUF2,7)
C -----*** file USRCOD is opened to checked the right of   ***-----
C -----*** INSERTION of the user                           ***-----
      CALL OPEN(IDCB,IERR,NAM13,1,ISECU)
      IF(IERR.NE.2) GO TO 575
      CALL SFILL(IBUF1,1,200,BLANK)
      CALL READF(IDCB,IERR,IBUF1,100,LEN,JJNUM)
      IF(IERR.LT.0) GO TO 575
      DO 12 I=1,10
      KSTRT=7+(I-1)*18
      IF(JSCOM(IBUF2,1,6+ICHAR,IBUF1,KSTRT,IER)) 12,15,12
12    CONTINUE
      WRITE(20,13)
13    FORMAT(1H ,"You can't INSERT ")
      GO TO 585
15    CALL CLOSE(IDCB)
C -----*** File DBMSRD is opened in case user has right to insert   ***--
C -----*** new tuple. Relation name is checked on which user wants  ***--
C -----*** to operate & also status of realation whether suspended  ***--
C -----*** or restore is checked.                                   ***--
26    CALL OPEN(IDCB,IERR,NAM7,1,ISECU)
      IF(IERR.NE.2) GO TO 575
34    CALL SFILL(IBUF2,1,64,BLANK)
      CALL READF(IDCB,IERR,IBUF2,IRSIZ,LEN,NUMB)
      IF(IERR.LT.0) GO TO 575
      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 if relation name is not found in  ***----
C -----*** DBMSRD directory.This file keeps information of all    ***-----
C -----*** defined view relation names against existing relation  ***-----
C -----*** name in directory DBMSRD.                              ***-----
```

```
      CALL OPEN(IDCB,IERR,NAM3,1,ISECU)
      IF(IERR.NE.2) GO TO 575
      IRSIZ=32
      CALL HASS(IBUF,ICHAR,19,NUMB)
      IF(NUMB.EQ.0) NUMB=19
      GO TO 34
35    DO 36 I=1,5
      LSTRT=1+(I-1)*16
      NFLAG=7+(I-1)*7
      IF(JSCOM(IBUFL,1,ICHAR,IBUF2,LSTRT,IER)) 36,38,36
36    CONTINUE
      WRITE(20,37) IBUFL
37    FORMAT(1H ,"Relation name:-",2X,5A2,2X, "neither in actual nor"
     *,/," in  view directory ")
       GO TO 585
C -----*** Status of view relation name whether suspended or restore is *
C -----*** checked in next statements                                   *
38    CALL SMOVE(IBUF2,(2*NFLAG)-1,2*NFLAG,JVAR,1)
      CALL CODE
      READ(JVAR,16) JVAR1
16    FORMAT(I2)
      IF(JVAR1.EQ.1) GO TO 14
      WRITE(20,17) IBUFL
17    FORMAT(1H ,"View relation name is in suspend state: ",2X,5A2)
      GO TO 585
14    CALL CLOSE(IDCB)
C -----*** File DBMS01 , which keeps information of attributes of each **
C -----*** relation is opened. It also keeps information about         **
C -----*** attribute's type length start location  & original existing **
C -----*** attributes on which it is defined                           **
      CALL OPEN(IDCB,IERR,NAM4,1,ISECU)
      IF(IERR.NE.2) GO TO 575
      IRSIZE=85
      GO TO 39
29    CALL HASS(IBUF,ICONT,31,NUMBR)
      IF(NUMBR.EQ.0) NUMBR=31
      GO TO 31
C -----*** Status of relation name is checked  ***-----
45    CALL SMOVE(IBUF2,(2*NFLAG)-1,2*NFLAG,JVAR,1)
C-----***  Conversion from A format to I format  ***-----
      CALL CODE
      READ(JVAR,27) JVAR1
27    FORMAT(I2)
      IF(JVAR1.EQ.1) GO TO 47
      WRITE(20,28) IBUFL
28    FORMAT(1H ,5A2,2X,"Relation name is in suspend state ")
      GO TO 585
47    CALL CLOSE(IDCB)
      ITEST=1
      CALL OPEN(IDCB,IERR,NAM8,1,ISECU)
      IF(IERR.NE.2) GO TO 575
      IRSIZE=45
 39   IBEG=2*ICHAR+1
      LVAR=0
      DO 55 IV=2,NANA
      KAUNT=0
      CALL SFILL(IBUF2,1,20,BLANK)
      DO 46 IU=1,10
      IF(ITAB1(IV,IU).EQ.1H ) GO TO 44
      KAUNT=KAUNT+1
 46   IBUF2(IU)=ITAB1(IV,IU)
 44   CALL SMOVE(IBUF2,1,2*KAUNT,IBUF,IBEG)
      ICONT=KAUNT+ICHAR
      IF(ICONT.LT.20) CALL SFILL(IBUF,(2*ICONT)+1,40,BLANK)
      IF(ITEST.EQ.0) GO TO 29
      CALL HASS(IBUF,20,31,NUMBR)
C -----*** Conversion from A1 format to A2 format  ***-----
 31   CALL CODE
      WRITE(IBUFL,32) (IBUF(JK),JK=1,20)
 32   FORMAT(20A1)
C -----CALL SFILL(IBUFL,ICONT+1,20,BLANK)
      CALL SFILL(IBUF2,1,170,BLANK)
      CALL READF(IDCB,IERR,IBUF2,IRSIZE,LEN,NUMBR)
      IF(IERR.LT.0) GO TO 575
      IF(ITEST.EQ.0) GO TO 90
      DO 48 IBUC=1,3
      JBEG=1+(IBUC-1)*32
      JEND=20+(IBUC-1)*32
      IF(JSCOM(IBUF2,JBEG,JEND,IBUFL,1,IER)) 48,50,48
 48   CONTINUE
      WRITE(20,53)
 53   FORMAT(1H ,"Attribute not in bucket ")
      GO TO 585
 50   ISTRT=23+(IBUC-1)*32
      IEND= 28+(IBUC-1)*32
      CALL SMOVE(IBUF2,ISTRT,IEND,IBFR,1)
C -----*** Start location & length are noted for attributes   ***-----
      CALL CODE
      READ(IBFR,83) ISTART,LENTH,LLL,JJJ
 83   FORMAT(I4,3I2)
C -----***  Start location & length of all attributes are stored    ***--
C -----***------              in buffer TEMP                    -------***--
      LVAR=LVAR+1
      TEMP(LVAR)=ISTART
      TEMP1(LVAR)=LENTH
      GO TO 55
 84   CALL CLOSE(IDCB2)
C -----***  According to specific relation name differnt data        ***--
C -----***------           files are opened                     -------***--
      IF((NUMB.EQ.1).AND.(NVAR.EQ.1)) GO TO 600
      IF((NUMB.EQ.2).AND.(NVAR.EQ.1)) GO TO 700
      IF((NUMB.EQ.2).AND.(NVAR.EQ.2)) GO TO 710
      IF((NUMB.EQ.4).AND.(NVAR.EQ.1)) GO TO 800
      IF((NUMB.EQ.4).AND.(NVAR.EQ.2)) GO TO 810
      IF((NUMB.EQ.5).AND.(NVAR.EQ.1)) GO TO 815
      IF((NUMB.EQ.5).AND.(NVAR.EQ.2)) GO TO 820
      IF((NUMB.EQ.6).AND.(NVAR.EQ.1)) GO TO 830
      WRITE(20,85)
85    FORMAT(1H ,"RELATION NAME IS WRONG")
      GO TO 585
90    DO 92 IL=1,4
      JSTRT=1+(IL-1)*36
      JEND=36+(IL-1)*36
      IF(JSCOM(IBUFL,1,ICONT,IBUF2,JSTRT,IER)) 92,94,92
92    CONTINUE
      WRITE(20,93) IBUFL
93    FORMAT(1H ,"Attribute concatenated with VIEW relation "
     *,4X,20A2,/,"    does not exists exists in directory ")
      GO TO 585
94    KSTRT=27+(IL-1)*36
      KEND=36+(IL-1)*36
      CALL SFILL(IBFR,1,10,BLANK)
      CALL SMOVE(IBUF2,KSTRT,KEND,IBFR,1)
      CALL CODE
      READ(IBFR,96)ISTART,LENTH,IFILE,NVAR
96    FORMAT(I4,I2,I2,I2)
      LVAR=LVAR+1
      TEMP(LVAR)=ISTART
      TEMP1(LVAR)=LENTH
      IF(LVAR.EQ.1) JFILE = IFILE
 55   CONTINUE
      IF(IFLAG.EQ.1) NUMB = JFILE
      GO TO 84
600   CALL OPEN(IDCB2,IERR,NAM9,2)
      IF(IERR.NE.2) GO TO 575
      IRSIZ=128
      GO TO 500
700   CALL OPEN(IDCB2,IERR,NAM1,2)
      IF(IERR.NE.2) GO TO 575
      IRSIZ=66
      GO TO 500
710   CALL OPEN(IDCB2,IERR,NAM5,2,ISECU)
      IF(IERR.NE.2) GO TO 575
      IRSIZ=128
      GO TO 500
800   CALL OPEN(IDCB2,IERR,NAM2,2)
      IF(IERR.NE.2) GO TO 575
      IRSIZ=50
      GO TO 500
810   CALL OPEN(IDCB2,IERR,NAM6,2,ISECU)
```

```
        IF(IERR.NE.2) GO TO 575
        IRSIZ=128
        GO TO 500
  815   CALL OPEN(IDCB2,IERR,NAM10,2,ISECU)
        IF(IERR.NE.2) GO TO 575
        IRSIZ=100
        GO TO 500
  820   CALL OPEN(IDCB2,IERR,NAM11,2,ISECU)
        IF(IERR.NE.2) GO TO 575
        IRSIZ=128
        GO TO 500
  830   CALL OPEN(IDCB2,IERR,NAM12,2,ISECU)
        IF(IERR.NE.2) GO TO 575
        IRSIZ=128
  500   CALL SFILL(IB,1,40,BLANK)
        CALL SFILL(IBUF1,1,2*IRSIZ,BLANK)
        DO 450 IZ=1,10
        CALL SFILL(IBUF2,1,2*IRSIZ,BLANK)
        CALL READF(IDCB2,IERR,IBUF2,IRSIZ,LEN,IZ)
        IF(IERR.LT.0) GO TO 575
        IF(JSCOM(IBUF2,1,2*IRSIZ,IBUF1,1,IER)) 450,550,450
  450   CONTINUE
        WRITE(20,451)
  451   FORMAT(1H ,"No place on file ")
        STOP
  550   NUMBR=IZ
        DO 570 IBB=1,IVAR5
        DO 560 IBC=1,40
  560   IBUF(IBC)=ITAB(IBB,IBC)
        CALL CODE
        WRITE(IB,562) (IBUF(LJ),LJ=1,40)
  562   FORMAT(40A1)
        CALL SMOVE(IB,1,TEMP1(IBB);IBUF1,TEMP(IBB))
  570   CONTINUE
        WRITE(20,571) IBUF1
  571   FORMAT(1H ,/,"   Tuple to be inserted is :- ",/,4X,128A2)
        CALL WRITF(IDCB2,IERR,IBUF1,0,NUMBR)
        IF(IERR.LT.0) GO TO 575
        CALL CLOSE(IDCB2)
        WRITE(20,572)
  572   FORMAT(1H ,"Query is processed & new tuple is inserted")
        GO TO 585
  C -----***  File manager errors are given to user if he tries to     ***--
  C -----***---        access the illeagle file or its status       ---***----
  575    WRITE(1,580) IERR
  580   FORMAT(1H ," *** FMGR ERROR ******",2X,I5)
  585   CALL EXEC(ICODX,LLNAM)
        END
  C -----*** This segment clears all buffers & tables entries  used ***
  C        * in previous query statement                             *
```

```
      PROGRAM CLEAR , 5 , , , , , , , , THIS CLEARS BUFF&CALLS LEXCL
      COMMON ITAB,ITAB1,ISTB,MM,M,JM,IPRS,IQRY,IF,ISTB1,KM,ICO,JLV,JFL
     *,MFLG
      INTEGER ISTB(50,2),ITAB1(30,22),ITAB(30,42),IPRS(50,3),ICO(20),
     *ISTB1(20,8),KNAM(3)
      DATA ICOE,KNAM /8,2HLE,2HXC,2HL /
      DO 20 I=1,50
      DO 5 J=1,2
5     ISTB(I,J)=1H
      DO 10 K=1,3
10    IPRS(I,K)=1H
20    CONTINUE
      DO 40 I1=1,30
      DO 25 J1=1,22
25    ITAB1(I1,J1)=1H
      DO 35 K1=1,42
35    ITAB(I1,K1)=1H
40    CONTINUE
      DO 50 I2=1,20
      DO 50 J2=1,8
50    ISTB1(I2,J2)=1H
      DO 55 I3=1,20
55    ICO(I3)=1H
      MFLG=1
C-----------   SEGMENT LEXCL IS CALLED . NOW THE USER CAN GIVE ANOTHER
C              QUERY IF DESIRED                                 ------------------------
      CALL EXEC(ICOE,KNAM)
      END
```

BIBLIOGRAPHY

1. ACM Computing surveys Vol 13 1981
2. INFOTECH STATE OF THE ART REPORT DATA BASE Technology Vol 1
3. INFOTECH STATE OF THE ART REPORT DATA BASE Technology vol 3
4. ACM Transactions on data base systems Vol 6 1981
5. Communication of ACM Oct 1974 Vol 17 No 10
6. Computer Data Base Organisation 2nd edition Martin James
7. An Introduction to data base systems Date C.J. 2nd edition
8. Data Base Systems J.D. Ulman
9. Data Base Management S.P.GHOSH
10. An Introduction to Computer Based Library Systems L.A.TEEDD
11. MANUALS OF HP 1000 SYSTEM
12. ON LINE DATA BASE 1 ANALYSIS AND BIBLIOGRAPHY
13. ON LINE DATA BASE 2 INVITED PAPERS
14. ENCYCLOPIDIA OF COMPUTER SCIENCE
15. SECURITY, ACCURACY AND PRIVACY IN COMPUTER SYSTEMS MARTIN J.
16. DATA BASE PROCESSING - DAVID KROENKE
17. DATA BASE DESIGN - GIO WIEDERHOLD
18. ON LINE REVOLUTION IN LIBRARY - PROCEEDINGS OF THE 1977 CONFERENCE IN PITTSBURGH, PENNSYLVANIA