DESIGN OF A RELATIONAL DATABASE FOR LOAN LEDGER ACCOUNTING SYSTEM OF A FINANCIAL ORGANIZATION

A Project
by

MD. NESAR UDDIN BHUIYAN

Submitted to the Department of Computer Science and Engineering of Bangladesh University of Engineering and Technology, Dhaka in partial fulfilment of the requirements for the degree

POST-GRADUATE DIPLOMA IN COMPUTER SCIENCE AND ENGINEERING

under

AIT - BUET PROGRAMME.

DESIGN OF A RELATIONAL DATABASE FOR LOAN LEDGER ACCOUNTING SYSTEM OF A FINANCIAL ORGANIZATION

A Project
by

MD. NESAR UDDIN BHUIYAN

Accepted as satisfactory as to style and contents for partial fulfilment of the requirements for the degree of Post-Graduate Diploma in Computer Science and Engineering under AIT - BUET Programme on 07-01-91.

[illegible] 7/1/91

DR. SYED MAHBUBUR RAHMAN
Associate Professor and Head,
Computer Science and Engineering Dept.,
Bangladesh University of Engineering and
Technology, Dhaka, Bangladesh.

Chairman
and
Supervisor

[illegible]

DR. MD. SHAMSUL ALAM
Associate Professor,
Computer Science and Engineering Dept.,
Bangladesh University of Engineering and
Technology, Dhaka, Bangladesh.

Member

[illegible]

DR. JAMILUR REZA CHOUDHURY
Director, Computer Centre, and
Professor of Civil Engineering Dept.,
Bangladesh University of Engineering and
Technology, Dhaka, Bangladesh.

Member

ABSTRACT

This project presents the design procedures of a relational database. The LOAN LEDGER ACCOUNTING SYSTEM of Bangladesh House Building Finance Corporation, a leading financial Organization in Bangladesh is selected for the present case study.

A database design methodology is defined for the design of large relational database. The methodology produces logical database designs as well as physical database. It reduces the number of data dependencies that must be analyzed, using the entity - Relationalship model conceptualization, and maintains data integrity through normalization. This approach can be implemented manually or in a simple software packages as long as a "good" solution is acceptable and absolute optimality is not required.

Desing methodologies for Relational Database are used the overall logical view on the conceptual schema is developed. The schema has all the meanings or semantics of the data. The details of descriptions such as properties of entity sets, attribute sets, attribute domains, tuple relations etc. are illustrated in course of model development. So modified database modeling techniques are implemented in the development of conceptual models, logical and physical models.

The program of the Accounting System is developed by INFORMIX-4GL under UNIX-V operating system environment and implemented in NCR TOWER-700 mini-computer. The program processes the queries by the embedded SQL commands which is convenient for large databases. Reduction of paper works to a considerable degree, expedite the overall works, reduction of the number of staff and thus minimizing the expenditure to a large extent is also a major attraction of the system.

ACKNOWLEDGEMENTS

The author wishes to express his gratitude to his adviser **Dr. Syed Mahbubur Rahman** for patiently guiding and fully supporti-ng him through all stages of work.

He also likes to express his heartfelt thanks to the following who have helped him during the production and writing of this work.

To Dr. Jamilur Reza Choudhury and Dr. Md. Shamsul Alam for being the members of the board of examiners and for their valuable suggestions and recommendations.

To Mostafa Fazle Rabbi, Computer Cell of Bangladesh House Building Finance Corporation (BHBFC), Samarjit Roy of BHBFC, and Md. Faizur Rahman of BHBFC for their help and cooperation.

To all his friends at AIT, BUET and BHBFC for their encouragement and well wish.

To the UNDP for providing him with the financial support without which his studies would not be possible.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER-1

INTRODUCTION

1.1 EVALUATION OF RDBMS

From a historical perspective, the relational data model is relatively new. The first database systems were based on either the hierarchical model or the network model. These two older models are tied more closely to the underlying implementation of the database than is the relational model.

The relational model, although not the data model used in the first database management systems, has grown slowly in importance since its exposition by E. Codd in 1970, to the point where it is generally the model of choice for the implementation of new databases[1]. Perhaps the most important reason for the model's popularity is the way it supports powerful, yet simple and declarative languages with which operations on data are expressed. We may trace these capabilities to the fact that,unlike competing models, the relational model is value oriented. That fact, in turn,leads to our ability to define operations on relations whose results are themselves relations. These operations can be combined and cascaded easily, using an algebraic notation called "relational algebra".

The behavioural analysis of data concerns itself with life history of entities[2]. When these entities exist, they change in a discrete or continuous manner until such time that their existence is lost. Things in the real world change over the time and therefore, a database system must change accordingly. It is the need to keep track of the dynamic nature of an entity which is often the prime motivation for including it in a data processing system.

The entity-relationship, ER, has the purpose of representing the real world in a pure and natural way so that it is independent of storage and efficiency considerations. The key idea of RDBMS is to concentrate on the design of the conceptual schema, which is an intermediate stage in the logical design of a Relational Database.

1.2 DIFFERENT TYPES OF DATA MODELS

A *data model* is a mathematical formalism with two parts :

- A notation for describing data, and
- A set of operations used to manipulate that data.

There are mainly three types *data model*, such as

- Hierarchical
- Network
- Relational

The major salient features of different data models are as follows :

Hierarchical Data Model

* Use tree structure
* First level has only root node
* Other level always have parents
* Each node has one parent

Network Data Model

* Use plex structure
* Each node may have several parents
* Relationship is expressed as a set type

Relational Data Model

* A new approach in DBMS based on mathematical theory
* Provide ease of use for search, retrieve, insert, and update
* Data is represented as two - dimensional tables
* Each entity has a corresponding relation which consists of a set of attributes
* Each occurrence of the entity is a column in the relation
* Each relation has a key

1.3 COMPARISON BETWEEN DATA MODELS

TABLE - 1. Data Model Comparisons[3].

	HIERARCHICAL	NETWORK	RELATIONAL
Representative system	DL/I (IBM)	IDMS (Cullinet)	SQL - the DML for IBM's DB2

Data building blocks	Field
	Segment
	Physical data base
		Data item
		Record
		Set
			Attribute (column)
			Tuple(row)
			Relation(table)

Representation of logical structures			
- Trees	Directly	Decomposition into sets	Decomposition into tables
- Simple networks	Unidirectional logical	Decomposition into sets	Decomposition into tables (table)
- Complex networks	Bidirectional logical relationship	Decomposition into sets using intersection data	Decomposition into tables using intersection data
Data independence			
- Path	No	No	Yes
- Sequence	No	No	Yes
Means of Data base navigation	Through hierarchical path	Through sets	Through the value of the attributes
Navigator	Experienced, trained programmer	Experienced, trained programmer	End user
DML commands			
- Retrieval	GU,GHU,GN,GHN GNP,GHNP	FIND,GET	SELECT
- Data alteration	REPL	MODIFY	UPDATE
- Data addition	ISRT	STORE	INSERT

TABLE - 1 (Contd.)

	HIERARCHICAL	NETWORK	RELATIONAL
- Data deletion - Miscellaneous	DLET	ERASE READY, FINISH, CONNECT, DISCONNECT	DELETE
Performance	High - with well-defined access paths Low - with unstructured access paths	High - with well-defined access paths Low - with unstructured access paths	High - with unstructured access paths Low (in comparison to hierarchical and network data models)- with well-defined access paths
Additional pointers available to improve performance	Yes-retrieval	Yes-retrieval	Yes-but SQL may choose not to use them
Security	Defined in subschema	Defined in subschema	Defined in subschema but may be modified at any time including during on-line execution
Security officer	DBA or equivalent	DBA or equivalent	DBA or equivalent-may be delegated
Modification of the data structure	Redefined structure, reload new structure	Redefined structure, reload new structure	Restructure at any time, including operation in an on-line environment

A brief comparative study between the different data models is shown in Table - 1 according to different features. A short description of different features are explained in the following paragraphs.

Representative System : The most widely used implementation of the hierarchical data model is DL/I, marketed by IBM. IDMS, marketed by Cullinet, is based on the 1971 DBTG report and represents the network data model. SQL, the data manipulation language for DATABASE 2 (DB2), marketed by IBM, is representative of the relational data model.

Data Building Blocks : The smallest unit of data defined in DL/I is a field, which is comparable to a data item in the DBTG model and a column in the relational data model. The unit of data transferred to an application program is a segment in DL/I and a record in the DBTG model. The relational model offers two different units of data transfer : (1) Application programs, written in third generation programming languages, currently offer no means for a unit of data transfer of multiple records; therefore, SQL operates in piped mode,which permits the transfer of one row of data at a time through a cursor. (2) When data is operated upon through an on-line terminal in SQL, the unit of data transfer is a table.The next larger data building block contains data about a subject and is called a physical data base in DL/I, a set in the network model, and a relation or table in the relational data model.

Representation of Logical Data Structures : DL/I represents tree structure directly. Tree structure cannot be represented directly in the network model; instead, they are decomposed into two-level structures called sets. The relational model also cannot represent tree structures directly; instead, each segment type in a tree structure is represented as a table. The tables which represent dependent segments include the key of the parent segment. The reproduced keys permit the unique identification of each row and contain values which permit the relationship between two differe-nt tables to be established.

Simple networks are not permitted in DL/I. Data is represented as if it were duplicated. However, a unidirectional logical relatio-nship implemented through logical pointers eliminates redundant data. The application program views the simple network as a tree structure. In the network model, simple networks are represented in the same manner as tree structures.Each parent-child relation-ship is represented as a set. In the relational model,each record type is represented in a table. The key of each parent record type is duplicated in the table that represents the child record type.

The network model cannot represent complex networks directly. Instead, a record type is created which contains intersection data for the records involved in the many-to-many relationship. The intersection record is a member in each of two sets in which the related records are owners. The relational model represents complex networks in much the same manner as the network model. A table is created which contains intersection data, including the primary key of each of the record types in the many-to-many rela-tionship. Each of the records in the complex network is stored in its own table, and the values of the reproduced keys in the table containing intersection data relate them.

Data Independence : DL/I makes parameters available in the DBD to allow the logical sequenceing of segments and permit multiple secondary indexes to be created for additional sequencing. While this provides great flexibility to the application program, it does cause sequence dependencies, since the correct path must be traversed to obtain the desired segments.

IDMS, like DL/I,provides parameters for the logical sequencing of records and requires a knowledge of the paths to be traversed to obtain the desired record. It is therefore subject to the same sequence and path dependencies.

SQL maintains relationships between tables through the values of the stored data; pointers are not used. Therefore, SQL is free of path dependence. SQL is also free of any sequence dependencies, since one of the basic rules of the relational data model is that columns and rows may be presented in any sequence.

DML Commands : DL/I has six different commands, each with a different function to make data available to the application program: GU,GN,GNP,GHU,GHN, and GHNP. The network model has one command, GET, to make data available to the application, program, but it must be preceded by a FIND command which locates the data. IDMS reduces the number of interactions between the application program and itself by offering an OBTAIN command, which combines the functions of lthe FIND and GET commands. SQL has only the SELECT command for data retrieval.

Each of the sample data base management system has one command to modify existing data in the data base: REPL in DL/I, MODIFY in IDMS, and UPDATE in SQL. One command is available in each system to add new data to the data base:ISRT in DL/I, STORE in IDMS, and INSERT in SQL.

Each system also has only one DML command to remove exiting data from a data base: DLET in DL/I,ERASE in IDMS, and DELETE in SQL. IDMS has four additional DML commands. The CONNECT command causes a record existing in the data base to become a member of a set when insertion is manual.CONNECT connot be used when insertion is automatic. The DISCONNECT command causes a recoed to be removed from an occurrence of a set without removing it from the data base. DISCONNECT can be used when retention is optional, but can not be used with any other retention status. The program notifies IDMS that it will access a data base through the READY command, and that no further accesses will occur through a FINISH commnad.

Means of Navigation : Navigation of hierarchical structure is through the traversal of the tree structure. The navigator must know the current location in the data base and the direction of the desired segment and may have to traverse many unwanted segme-nts to obtain the desired segment.

Navigation in the network model in through the traversal of sets. Travel in a complex network structure is especially descriptive of the navigation process for the navigator must follow the path from the owner to a member record containing intersection data in one set type, then to the owner of the same menber record in another set type.

Since paths through the tables do not exist in the relational model. navigation as such is not performed. Desired records are obtained based on the value of the attributes within relations. It is not necessary to direct the relational data base management system in how to find the data, but only in what data is desried.

Navigator : Since data is obtained through predefined paths in the hierarchical model and the navigator must konw both the curr-ent location and the final destination the navigator is an appli-cation programmer who has had a high degree of training and expe-rience to hone navigational skills.

IDMS also uses predefined paths through occurrences of record types to obtain the desired data. Once again, the navigator requ-ires a high degree of education and training.

Since navigation is not performed in the relational model, there is no need for highly trained programmer. Commands like those of SQL specify what is to be accomplished, not how to do it, and so end users (with a little training) perform data manipulation themselves.

Performance : With a well designed data base, a well-defined access path, and a proficient navigator,DL/I can be tuned to provide rapid response. The specific paths through the data base defined by the pointers permit rapid retrieval of the desired data.IDMS,because of direct access paths of the data can also be tuned to provide a very responsive system when used with specific access paths. However, neither IDMS nor DL/I is well-suited to an unstructured environment where data relationships can be changed while the user is accessing the data base.

The relational medol, as illustrated by SQL, has greater memory requirements and requires more resources than either DL/I or IDMS. But SQL is not as responsive as either DL/I or IDMS and may require more secondary storage space to store data, unless DL/I or IDMS use many pointers to store the data. SQL is powerful. It performs operation on many records in singal query, whereas DL/I and IDMS operate on one record or segment at a time and may require many commands to perform the work accomplished by one SQL command.

Additional Pointer Available to Improve Performance : Both DL/I and IDMS have default pointers. Both also have optional pointers to improve performance. While these optional pointers do improve performance in data retrieval operations, performance is often reduced by them during operations which add or delete records or update fields on which the additional pointers exist, because of the pointer maintenance required.

SQL permits the creation of indexes by the user to improve retrieval performance. Performance is reduced during the creation of the index and during the maintenance of a table on which indexes have been created.

Security : In all three systems, security is defined through the subschema. With DL/I and IDMS, the subschema definition is translated before the execution of the application program and stored for use during execution. SQL differs from the other two in that the subschema definition may be modified at any time, even during on-line execution.

Security Officer : All three systems provide the highest degree of security when controlled by the data base administrator or by the security officer of the enterprise. However, it is sometimes difficult to determine which user have a legitimate need to access data.SQL offers an additional option to the security officer: to delegate access to the data base. SQL may permit the user to in turn grant access to other users. This enables a responsible user to grant access to the individuals who have a legitimate need to access and update the data base.

Responsibility for Adding Data to the Data Base : The DL/I and SQL data base models have the responsibility to add data to the data base, based on schema and subschema definitions, when requested to do so by the programmer or the end user.

IDBMS has the same responsibility. However, when a record is added, it may or may not become a member of a set. This responsibility is defined in the schema. When the same record type participates in more than one set type, IDMS may be responsible for connecting it in one set type and the application program responsible for connecting it in another set type.

Modification of the Data Structure : Since both DL/I and IDMS have fixed paths to segments and and records, modification of the data structure requires a working copy of the data base(s) to be made, the new data structure to be defined, and the data base to be recreated in the new structure from the working copy. While the restructuring is occurring, programs may not access or modify the existing data and retain data integrity.

Since SQL does not have path dependencies, the data structure may be modified at any time. New columns may be added or new tables created while the system is executing. There is no need for the user to perform the copying of data as is done in DL/I and IDMS; database definitions take effect immediately.This is advantageous when new systems are developed, for it permits a prototype of the new system to be developed and modified during development. Because of this flexibility, design errors are easily corrected and the impact of design changes is reduced.

1.4 ADVANTAGES OF RELATIONAL DATA MODEL

The relational data model is a new approach in DBMS based on mathematical theory and data is represented as two - dimensional tables. It Provides ease of use for search, retrieve, insert, and update. The relational model offers two different units of data transfer : (1) Application programs, written in third generation programming languages, currently offer no means for a unit of data transfer of multiple records; therefore, SQL operates in piped mode,which permits the transfer of one row of data at a time through a cursor. (2) When data is operated upon through an on-line terminal in SQL, the unit of data transfer is a table.

SQL maintains relationships between tables through the values of the stored data; pointers are not used. Therefore, SQL is free of path dependence. SQL is also free of any sequence dependencies, since one of the basic rules of the relational data model is that columns and rows may be presented in any sequence.

The relational model, as illustrated by SQL, has greater memory requirements and requires more resources than either DL/I or IDMS. But SQL is not as responsive as either DL/I or IDMS and may require more secondary storage space to store data, unless DL/I or IDMS use many pointers to store the data. SQL is powerful. It performs operation on many records in singal query. SQL permits the creation of indexes by the user to improve retrieval performance.

SQL offers an additional option to the security officer: to delegate access to the data base. SQL may permit the user to in turn grant access to other users. This enables a responsible user to grant access to the individuals who have a legitimate need to access and update the data base. Since SQL does not have path dependencies, the data structure may be modified at any time. New columns may be added or new tables created while the system is executing.

1.5 PRESENT STATE AND FUTURE PROSPECT :

The Relational Database Model is favourite to the academic community, due to its operational simplicity and power which permit users with little training to execute DML (Data Manipulation Language) commands. However, the power of the relational model initially presented problems in data access performance, causing the model to be largely confined to the research environment through most of the 1970s, relational data base management systems are being used in large scale data processing applications[4].

The relational model is founded on mathematical principles, and much of the early terminology associated with the model was mathematical. The DML for the model uses mathematics as its basis,and relational calculus and relational algebra data manipulation languages have been developed. Thus, a problem exists, for outside of academia, only a small percentage of users have an adequate background to understand the mathematics of the DML.

Fortunately, an understanding of the mathematics involved is not necessary in order to use the model.Instead,the data manipulation operations can be viewed as a series cutting and pasting operations on the data within a table. Unlike the models previously presented, a user can obtain specific data from the relational model via a nonprocedural language which allows the user to describe the data required, rather than the navigation required. The data base management system then selects the data, based on the description provided by the user rather than by a navigational path thruógh the data base.

The Relational Data Model represents data relationships as tables. The relational model is couched in complex mathematical notation, and this terminology is unfamiliar to the average data base management system student.

The traditional data manipulation language of the relational model was relational algebra and relational calculus. However, the current trend is to provide entries in a table using English-like statements and to eliminate mathematical procedures for manipulation. SQL, the data manipulation language reflects this trend.Because of its power and ease of use, the relational database management system is likely to be the database management system of the future.

1.6 Objective of study

In Bangladesh, many large databases such as Management Information System (MIS), Accounting System, Inventory Control etc. are becoming computerized. Most of the databases are using either hierarchical model or network model due to the lack of knowledge of design procedure of relational database model.

Loan Ledger Accounting System is a large database which is used manually in different financial organizations in Bangladesh. So, for getting frequent informations, a well-structured database and query formulation is required for this database.

Bangladesh House Building Finance Corporation is one of the well -known financial organization which is engaged to lend Loan for building public houses. It has more than 80,000 borrowers now. BHBFC has been taken as a case study in this project. Loan Ledger Accounting System is a most important system in this Organization.

In this project, an attempt is made to develop a relational database models for the construction of Loan Ledger Accounting System database.

1.7 Methodology

Different steps required for implementation of the Loan Ledger Accounting System database are :

Step 1: First, all the information required for database is to be collected from the Accounting Section of the Organization for the Information Requirement Analysis (IRA) and then the informal strategy of the problem for the database is to be defined. Proper Information will be collected through exchange of views of the concern authority. This procedure is called *Information Requirement Analysis* .

Step 2: Identify all the Entities which are objects or concepts that exist and are distinguishable from each other, and relationships between them from the informal strategy of the problem.

Step 3: The *entity - relationship (ER) diagram* is most successful as a tool for communication between the designer and the end user during the Information Requirements Analysis and conceptual design phases because of its ease of understanding and its convenience in representation. The data requirements are analyzed and modeled using ER diagram that includes semantics for relationships between entities. Processing requirements are assumed to be specified using natural language expressions, along with the frequency of occurrence. Logical views from multiple sources are integrated into a common global view of the entire database.

Step 4: Determine the tables required for the database from ER Diagram by applying some logical rules.

Step 5: *Normalization of Relations.* Functional dependencies (FDs) are derived from ER diagram to represent the dependencies among data elements that are keys of entities. Additional FDs and multivalued dependencies (MVDs), which represent the dependencies among key and nonkey attributes within entities,are derived from the requirements specification. Redundancies that occur in normalized relations are then analyzed further for possible elimination, with the constraint that data integrity must be preserved.

Step 6: Implementation. Lastly, the logical design of the system is to be implemented in consistency with the Hardware and Software available resources.

The Block Diagram of the overall methodology is shown in figure 1:

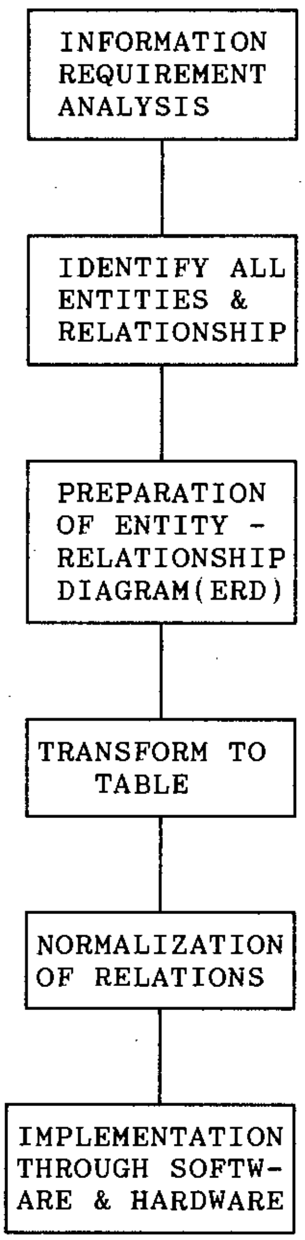

INFORMATION REQUIREMENT ANALYSIS
IDENTIFY ALL ENTITIES & RELATIONSHIP
PREPARATION OF ENTITY - RELATIONSHIP DIAGRAM(ERD)
TRANSFORM TO TABLE
NORMALIZATION OF RELATIONS
IMPLEMENTATION THROUGH SOFTW-ARE & HARDWARE

FIGURE - 1. Block Diagram of Methodology.

CHAPTER-2

LOGICAL DATA BASE DESIGN AND FORMAL QUERY LANGUAGES

A Logical Database design methodology is defined for the design of large relational databases. First, the data requirements are conceptualised using an Entity - Relationship (ER) model. The methodology produces logical database designs that are not accurate representations of reality,but flexible enough to accommodate future processing requirements[5]. It reduces the number of data dependencies that must be analysed and maintains data integrity through normalization. During logical design of a database, knowledge on formal query language is required that optimizes the queries.

2.1 ENTITY - RELATIONSHIP MODEL

The purpose of the entity - relationship model is to allow the description of the conceptual schema of an enterprise to be written down without the attention to efficiency or physical database design that is expected in most other models. It is normally assumed that the "entity - relationship diagram" thus constructed will be turned later into a conceptual scheme in one of the other models, e.g., the relational model, upon which real database systems are built.

The entity - relationship (ER) model has been most successful as a tool for communication between the designer and the end user during the Requirements Analysis and conceptual design phases because of its ease of understanding and its convenience in representation[6]. One of the reasons for its effectiveness is that it is a top-down approach using the concept of abstraction. The number of entities (i.e.,the objects that collect information about) in a database is typically an order of magnitude less than the number of data elements.

Therefore, using entities as an abstraction for data elements and focusing on the interentity relationships greatly reduces the number of objects under consideration and simplifies the analysis. Although it is still necessary to represent data elements by attributes of entities at the conceptual level, their dependencies are normally confined to the other attributes within the entity or, in some cases, to those attributes associated with other entities that have a direct relationship to their entity.

To obtain the conceptual scheme that offers the most efficiency can be quite difficult and requires an understanding of design issues, such as entities, entity sets, attributes, relationship, different types of keys and entity-relationship diagram in the target model.

2.1.1 Entities, Entity Sets, and Attributes

The entity - relationship, ER, has the purpose of representing the real world in a pure and natural way. The key idea of ER model is to concentrate on the design of the conceptual schema, which is an intermediate stage in the logical design of a database. The ER model is constructed based on the idea that the perception of the real world is usually expressed in terms of entities, entity-set, attributes, and relationship.

Entities : An entity is an object that exists and is distinguishable from other objects, that is, one entity from another. For example, each person is an entity, and each automobile is an entity. The notion of distinguishability of entities is very close to object identity.

Entity-Sets : A group consisting of all "similar" entities forms an *entity sets.* Examples of entity sets are :

- All persons
- All red - haired persons
- All automobiles

Attributes : Entity sets have properties, called *attributes*, which associate with each entity in the set a value from a domain of values for that attribute. Usually, the domain for an attribute will be a set of integers, real numbers, or character strings, but we donot rule out other types of values. For example, the entity set of persons may be declared to have attributes such as name (a character string), height (a real number), and so on.

Relationship and Relationship Sets : A Relationship is an association among several entities. A particular entity sets may appear more than once on the list. This list of entity sets is the scheme - level notion of a relationship. If there is a relationship R among entity sets $E_1, E_2, \ldots E_k$, then the current instance of R is a set of k-tuples, such a set is called a *relationship set.* So, a relationship set is a set of relationships of the same type. Each k-tuple $(e_1, e_2, \ldots, e_k)$ in relationship set R implies that entities $e_1, e_2, \ldots, e_k$ where e_1 is in set E_1, e_2 is in E_2, and so on, stand in relationship R to each other as a group.

2.1.2 Different types of Keys

An attribute or set of attributes whose values uniquely identify each entity is an entity set is called a key for that entity set.

The *super key* is a set of one or more attributes, which taken collectively, allow to identify uniquely an entity in the entity set. For example, the *social_security_no* attribute of the entity set *customer* is sufficient to distinguish one *customer* entity from another. Thus, *social_security_no* is a *super key.*

Similarly,the combination of *customer_name* and *social_security_no* is a superkey for the entity set *customer.* The *customer_name* attribute of customer is not a *super key*, as several people might have the same name.

A *super key* may contain extraneous attributes. *Smallest* possible super keys is preferable. That is, in *super keys* for which no proper subset is a super key. Such *minimal* super keys are ***Candidate keys.*** The term ***Primary key*** to denote a *candidate key* that is chosen by the database designer as the principal means of identifying entities within an entity set.

A ***foreign key*** is a column (or group of columns) in a table which is a key in some other table.

2.1.3 Entity - Relationship Diagram :

The overall logical structure of a database can be expressed graphically by *Entity - Relationship Diagram* which consists of the following components :

* Rectangles, which represent entity sets.
* Ellipses, which represent attributes.
* Diamonds, which represent relationship sets.
* Lines, which link attributes to entity sets and entity sets to relationship sets.

2.1.4 Functionality of Relationships

To model the real world adequately, it is often necessary to classify relationships according to how many entities from one entity set can be associated with how many entities of another entity set.

For a binary relationship set R between entity sets A and B, the mapping cardinality must be one of the following :

* One - to - one : The simplest and rarest form of relationship on two sets is *one - to - one*, meaning that for each entity in either in either set there is at most one associated member of the other set. An entity in A is associated with at most one entity in B, and an entity in B is associated with at most one entity in A.

* One - to - **many** : An entity in A is associated with any number of entities in B. An entity in B, however,can be associated with at most one entity in A.

* **Many** - to - one : An entity in A is associated with at most one entity in B. An entity in B, however, can be associated with any number of entities in A.

* **Many** - to - **many** : An entity in A is associated with any number of entities in B and an entity in B is associated with any number of entities in A.

2.2 Logical rules for determination of number of tables in database :

For One - to - one relation

Rule no.	Condition	No. of Table	Remark
1	Both side obligatory	1	
2	One side obligatory	2	Key attributes from non obligatory side are included in the obligatory side.
3	Both side Non-obligatory	3	
For One - to - many or Many - to - one relation			
4	M - side obligatory	2	
5	M - side non-obligatory	3	
For Many - to - many relation			
6	M - side obligatory or non-obligatory	3	Obligatory side will not be considered.

2.3 DESIGN ANOMALIES IN DATABASE

Some database schemes might present some anomalies and problems. To make a good database scheme, these anomalies should be removed from the database scheme. The following example is to combine the relations SUPPLIERS and SUPPLIES into one relation SUP_INFO, with scheme:

SUP_INFO (SNAME,SADDR,ITEM,PRICE)

that included all the information about suppliers. Several problems and anomalies of the scheme are as follows :

1. *Redundancy.* The address of the supplier is repeated once for each item supplied.

2. *Potential inconsistency (update anomalies).* As a consequence of the redundancy, updating the address for a supplier in one tuple is to be done, while leaving it fixed in another. Thus, a unique address would not be produced.

3. *Insertion anomalies.* An address can not be recorded for a supplier if that supplier does not currently supply at least one item. Null values might put in the ITEM and PRICE components of a tuple for that supplier. ITEM and SNAME together form a key for the relation, and it might be impossible to look up tuples through a primary index, if there were null values in the key field ITEM.

4. *Deletion anomalies.* The inverse to problem (3) is that all of the items supplied by one supplier will be deleted, that results the track of the supplier's address will be lost unintentionally.

As entities are mapped to records in the logical data base design, the designer must ensure that the design will satisfy the test of time. The records developed must support not only systems currently under development,but also systems to be developed in the future whose requirements are not completely known at the time of the logical data base design .

2.4 NORMAL FORM

Four anomalies which occur using the relational model are identified. Anomalies are unexpected results during data base maintenance. *Deletion anomalies* are experienced when a value for one attribute about which keep is unexpectedly removed when a value for another attribute is deleted . *Insertion anomalies* are experienced when store a value for one attribute but cannot because the value of another attribute is unknown.

These anomalies do not occur as a result of problems with the relational model but come about due to the ever-changing nature of a data base. Data bases are designed to contain attributes for specific applications as applications are developed. New attributes are added as a result of adding new systems which require new data relationships. The anomalies are not restricted to relational data bases; they can occur with any data model. If the anomalies are understood, today's data bases can be designed such that redesign to add future systems will be minimized or eliminated. Thus, the process of *normalization* provides stability of data structures over time.

Anomalies of data relationships are identified in three normal forms. Actually, other normal forms exist, but Codd's three forms are the ones most likely to be encountered.

A table is said to be in *first normal form* if and only if no row has multiple entries in any column. Some authors do not even consider a table to be relational unless the first normal form criterion is satisfied. A table is said to be in *second normal form* if it is in first normal form and every nonprime attribute is fully dependent on every key for the table.

An attribute A (column) of a table is said to be *transitively dependent* upon a subset X of the columns of the table if there exists a subset Y of the columns of the table , such that :

- All of the columns in the subset Y are functionally dependent on the subset X.
- Not all of the columns in the subset X are functionally depe- ndent on the set of columns in Y.
- A is functionally dependent on the set of columns in Y.
- A is not in either of the subsets X or Y.

A table is said to be in *third normal form* if :

- It is in first normal form
- No nonprime attribute is transitively dependent upon a key of the table .

A table is said to be in *Boyce-Codd normal form* if it meets the following criteria:

- It is in first normal form
- No attribute of the table is transitively dependent on a key of the table.

Notice that the only difference between Boyce - Codd normal form and third normal form is that the attribute which is transitively dependent in the second above can be prime. It is easy to see that, if a table is in Boyce-Codd normal form ,then it is in third normal form.

Data is placed in a higher normal form by placing certain restrictions on the way in which attributes are grouped within records.

Thus, data stored in the second normal form is a subset of data stored in the first normal form, and data stored in the third normal form is a subset of the data stored in the second normal form.

2.4.1 Example of Normalization

The normal form in which the data is stored identifies certain anomalies which may occur. The following Example illustrates anomalies of the first normal form.

JOB	EMPNAME	EMP-ADDRESS	EMPNO	DEPTNO	DEPTNAME	DEPTHEAD
MANAGER	KARIM	123, NURBEG	786	001	MECHANICAL	WASIM
CLERK	FARID	23,DANMONDI	123	001	MECHANICAL	WASIM
FOREMAN	AKBAR	10,BASHABOO	666	002	ELECTRICAL	MINHAZ
MECHANIC	KAMAL	40,BANANI	777	100	PRODUCTION	ZABIR
HELPER	NAHID	88,MIRPUR	999	007	ACCOUNTING	MALEK
CLERK	NADIR	45,BASHABOO	133	001	MECHANICAL	WASIM
CLERK	TARIK	101,GINGIRA	780	001	MECHANICAL	WASIM
MECHANIC	FARID	23,DANMONDI	123	001	MECHANICAL	WASIM

Figure 2.1 First normal form.

The figure 2.1 depicts the employee relation for a company. The relation maintains information about each employee and the department in which the employee works. The *primary key* of this relation is employee number, department number, and job worked. It has no repeating attributes and no duplicate tuples, each attribute is named, and the primary key field is not null. Because this relation adheres to the rules for storing data in relations, it is said to be in the first normal form.

Let's examine the insertion and deletion anomalies which occur with relations in the first normal form. Using the relation in figure 2.1 as an example, if an employee is hired, data cannot be stored for the employee until he has worked his first job, since the value of DEPTNO, which is one of the attributes of the composite primary key, is unknown. If a new job is created, data about the job cannot be entered until someone works in that position, since EMPNO, which is also one of the attributes of the composite primary key, is unknown. These are all insertion anomalies.

Data stored in the first normal form also had deletion anomalies. If the last employee in a department is deleted,information about the department is also removed . If a department is affected by seasonal variations in employment ,information about the departm-ent is lost. In Figure 2.1,if the employee Nahid is deleted,not only is the data for the employee removed, but the data for the Accounting Department is also lost. These anomalies occur because data is stored in the first normal form.

In Figure 2.1,if the value of EMPNO is known,the value of EMPNAME can be determined. When the value of EMPNO is 123, the value of EMPNAME is Farid. Thus, EMPNAME is functionally dependent on EMPNO,that is,the value of EMPNO determines the value of EMPNAME. Likewise , EMP-ADDRESS is functionally dependent upon EMPNO: When the value of EMPNO is known, the value of EMP-ADDRESS can be determined. DEPTNAME is functionally dependent on DEPTNO: When the value of DEPTNO is known, the value of DEPTNAME can be determined.

The last definitions needed to understand the anomalies of the first normal form are those of prime and non-prime attributes. Simply stated, a *prime attribute* is an attribute which is part of the primary key of a relation. *A non-prime attribute* is an attribute that is not a part of the primary key of a relation.

The anomalies of the first normal form occur because functional dependencies exist that are not based on the full key of the relation. For example, the attributes EMPNAME and EMP-ADDRESS in Figure 2.1 depend on only part of the key EMPNO. The attributes DEPTNAME and DEPTHEAD depend upon only part of the DEPTNO. These anomalies can be corrected by altering the way in which attributes are stored in a relation. This is accomplished by storing the data in the second normal form . All relations in the second normal form are also in the first normal form, but all nonprime attributes in the second normal form are fully functionally dependent,that is, all non-prime attributes depend on all of the key, not just a part of the key.

The data in the relation presented in Figure 2.1 is regrouped in the following Figure 2.2 to satisfy the rule of the second normal form. In Figure 2.2, three relations have been created to store the department-employee data.They are : No data or data relationships are lost in this transformation. The original relation can be obtained by a join in the relational model, by following owner-member paths in the network model, and by following parent-child paths in the hierarchical model.

Data stored in the second normal form removes the anomalies existing in the first normal form. All non-prime attributes are fully functionally dependent on all of the primary key. For example,in Figur 3, all non-prime attributes depend upon all of the primary key. EMPNAME and EMP-ADDREESS depend upon EMPNO. DEPTNAME and

DEPTHEAD depend upon DEPTNO. The JOBS-WORKED relation has no non-prime attributes; all of the attributes are part of the key.

Example in figure 2.2 shows anomalies which occur with data stored in the second normal form.

In Figure 2.2, all non-prime attributes of the DEPARTMENT relation are fully functionally dependent upon all of the primary key. The DEPARTMENT relation contains a dependency that does not involve the key. DEPTNO determines the value of DEPTNAME and DEPTNAME, determines the value of DEPTHEAD, but DEPTHEAD does not determine the value of DEPTNO. Therefore, DEPTNO indirectly determines the value of DEPTHEAD-a dependency exists which does not involve the key field. Dependencies not involving the key field are called *Transitive dependencies.*

Relations in the second normal form experience insertion and deletion anomalies. For example, if a department is phased out of operation and deleted from the relation, not only is the DEPTNO and DEPTNAME removed from the relation, but the name of the DEPTHEAD is also removed. Thus more information is removed from the relation than is desired. Or, if an individual is named the head of a new department, information about the DEPTHEAD cannot be recorded until the DEPTNO is determined and added to the relation.

Anomalies of the second normal form can be eliminated by placing the relation in the third normal form. All relations in the third normal form are also in the second normal form, but the third normal form does not contain any transitive dependencies. Figure 2.2 shows the division of the DEPARTMENT relation into two relations in the third normal form, DEPARTMENT and MANAGER. Every attribute in each relation is fully functionally dependent on the entire key, and all transitive dependencies have been eliminated.

The purpose of normalization is to design data structures in such a manner as to eliminate the need to redesign them when new applications are added. By storing data in the third normal form, structures designed today can be integrated with new applications in the future with little, if any, redesign effort.

EMPLOYEE

EMPNAME	EMP-ADDRESS	EMPNO
KARIM	123, NURBEG	123
FARID	23,DANMONDI	786
AKBAR	10,BASHABOO	666
KAMAL	40,BANANI	777
NAHID	88,MIRPUR	999
NADIR	45,BASHABOO	133
TARIK	101,GINGIRA	780

JOB WORKED

DEPTNO	EMPNO	JOB
001	786	MANAGER
001	123	CLERK
002	666	FOREMAN
100	777	MECHANIC
007	999	HELPER
001	123	MECHANIC
001	786	CLERK

DEPARTMENT

DEPTNO	DEPTNAME	DEPTHEAD
001	MECHANICAL	WASIM
002	ELECTRICAL	MINHAZ
100	PRODUCTION	ZABIR
007	ACCOUNTING	MALEK
001	MECHANICAL	WASIM

EMPLOYEE(EMPNAME,EMP-ADDRESS, <u>EMPNO</u>)

JOB-WORKED(<u>DEPTNO</u>,<u>EMPNO</u>,<u>JOB</u>)

DEPT(<u>DEPTNO</u>,DEPTNAME,DEPTHEAD)

Figure 2.2 Second Normal form

EMPLOYEE

EMPNAME	EMP-ADDRESS	EMPNO
KARIM	123, NURBEG	123
FARID	23,DANMONDI	786
AKBAR	10,BASHABOO	666
KAMAL	40,BANANI	777
NAHID	88,MIRPUR	999
NADIR	45,BASHABOO	133
TARIK	101,GINGIRA	780

JOB WORKED

DEPTNO	EMPNO	JOB
001	786	MANAGER
001	123	CLERK
002	666	FOREMAN
100	777	MECHANIC
007	999	HELPER
001	123	MECHANIC
001	786	CLERK

DEPARTMENT

DEPTNO	DEPTNAME
001	MECHANICAL
002	ELECTRICAL
100	PRODUCTION
007	ACCOUNTING

MANAGER

DEPTNO	DEPTHEAD
001	WASIM
002	MINHAZ
100	ZABIR
007	MALEK

FIGURE 2.3 Third Normal Form

2.5 FORMAL QUERY LANGUAGES

A *query language* is a language in which a user requests information from the database. These languages are typically higher-level languages than standard programming languages[7]. Query languages can be categorized as being either *procedural or nonprocedural.* In a procedural language, the user instructs the system to perform a sequence of operations on the database to compute the desired result. In a nonprocedural language, the user describes the information desired without giving a specific procedure for obtaining that information. Most commercial relational database systems offer a query language that includes elements of both the procedural and the nonprocedural approaches. The relational algebra is a procedural query language.

2.5.1 THE RELATIONAL ALGEBRA

A relation is a set of k-tuples for some k, called the arity of the relation. In general, attributes to the components of tuples, although some of the operations mentioned below, such as union, difference, product, and intersection, do not depend on the attributes of the components.

The operands of relational algebra are either constant relations or variables denoting relations of a fixed arity. The arity associated with a variable will be mentioned only when it is important. There are five basic operations that serve to define *relational algebra.*

a. **Union.** The union of relations R and S, denoted R U S, is the set of tuples that are in R or S or both. Applying the union operator to relations of the same arity, all tuples in the result have the same number of components. An example of *Union* is shown in figure 2.5a.

b. **Set difference.** The difference of relations R and S, denoted R - S, is the set of tuples in R but not in S. R and S of the same arity is required. An example of *Difference* is shown in figure 2.5b.

c. **Cartesian product.** Let R and S be relations of arity k_1 and k_2, respectively.Then R X S,the Cartesian product of R and S, is the set of all possible $(k_1 + k_2)$ - tuples whose first k_1 components form a tuple in R and whose last k_2 components form a tuple in S. An example of *Cartesian product* is shown in figure 2.5c.

d. **Projection.** The idea behind this operation is that a relation R is taken, remove some of the components(attributes) and /or rearrange some of the remaining components.If R is a relation of arity k, we let $\pi_{i_1,i_2,...,i_m}(R)$, where the i_j's are distinct integers in the range 1 to k, denote the projection of R onto components $i_1,i_2,...,i_m$, that is, the set of m - tuples $a_1a_2...a_m$ such that there is some k-tuple $b_1b_2...b_k$ in R for which $a_j = b_{i_j}$ for j = 1,2,...,m. For example, $\pi_{3,1}(R)$ is computed by taking each tuple μ in R and forming a 2-tuple from the third and first components of μ, in that order. If R has attributes labeling its columns, then we may substitute attribute names for component numbers,and we may use the same attribute names in the projected relation. Thus, if relation R is R(A,B,C,D), then $\pi_{C,A}(R)$ is the same as $\pi_{3,1}(R)$, and the resulting relation has attribute C naming its first column and attribute A naming its second column. An example of *Projection* is shown in figure 2.5d.

5. **Selection.** Let F be a formula involving
   i) Operands that are constants or component numbers; component i is represented by \$i,

ii) The arithmetic comparison operators $<$, $=$, $>$, $\leq$, $\geq$, and $<>$, and

iii) The logical operators and , or , and not.

Then $\sigma_F(R)$ is the set of tuples μ for any occurrences of $\$i$ in formula F, the formula F becomes true. For example, $r_{\$2>\$3}(R)$ denotes the set of tuples μ in R such that the second component of μ exceeds its third component, while $\sigma_{\$1='Smith' \text{ or } \$1='Jones'}(R)$ is the set of tuples in R whose first components have the value 'Smith' or 'Jones'. As with projection, if a relation has named columns, then the formula in a selection can refer to columns by name instead of by number. An example of *Selection* is shown in figure 6e.

A	B	C
a	b	c
d	a	f
c	b	d

(a) Relation R

D	E	F
b	g	a
d	a	f

(b) Relation S

FIGURE - 2.4 Two relations R and S

a	b	c
d	a	f
c	b	d
b	g	a

(a) R U S

a	b	c
c	b	d

(b) R - S

A	B	C	D	E	F
a	b	c	b	g	a
a	b	c	d	a	f
d	a	f	b	g	a
d	a	f	d	a	f
c	b	d	b	g	a
c	b	d	d	a	f

(c) R X S

A	C
a	c
d	f
c	d

(d) $\pi_{A,C}$ (R)

A	B	C
a	b	c
c	b	d

(e) $\sigma_{B=b}$ (R)

FIGURE - 2.5 Results of some relational algebra operations.

The relational algebra is a procedural query language. There are five fundumental operations in the relational algebra which are mentioned above. These operations are *select*, *project*, *cartesian-product*, *union*,and *set-difference*.All of these operations produce a new relation as their result.

In addition to the five fundamental operations, several other operations, namely, *set intersection*, *theta join*, *natural join, and division* can be introduced. These operations will be defined in terms of the fundamental operations.

2.5.2 FUNDAMENTAL OPERATION

The select and project operations are called *unary* operations, since they operate on one relation. The other three relartions operate on pairs of relations and are, therefor called *binary* operations.

The *select* operation selects tuples that satisfy a given predicate. Lowercase Greek letter sigma (σ) is used to denote selection. The predicate appears as a subscript to σ.The argument relation is given in parentheses following the σ . Thus to select those tuples of the *borrow* relation where the branch is " Perryr-idge", can be written as

$$\sigma_{\text{branch-name = "Perryridge"}}(\text{borrow}) \quad \text{------} \quad (2.1)$$

For a borrow relation as shown in figure 7a, a relation that results from equ.(2.1) is shown in figure 7b.

All tuples in which the amount borrowed is more than $1200 by writing

$$\sigma_{\text{mount} > 1200}(\text{borrow}) \quad \text{------} \quad (2.2)$$

In general, comparisons $=, <>, <, \leq, >, \geq$ in the selection predicate are used. Furthermore, several predicates may be combined into a larger predicate using the connectives ***and*** (^) and ***or*** (v). Thus, to find those tuples pertaining to loans of more than $1200 made by the Perryridge branch, can be written as

$\sigma_{\text{ranch-name = "Perryridge" ^ amount>1200}}(\text{borrow})$ --- (2.3)

The selection predicate may include comparisons between two attributes as illustrated in the relation scheme shown in equ.(2.4).

Client-scheme = (customer-name,employee-name) ---- (2.4)

The equ.(2.4) indicates that the employee is the "personal banker" of the customer. The relation *client (Client scheme)* is shown in Figure 7c. All those customers who have the same name as their personal banker may be found by writing

$\sigma_{\text{customer-name = employee-name}}(\text{client})$ --- (2.5)

If the client relation is as given in Figure 7c, the answer is the relation shown in Figure 7d.

In the above example, a relation (Figure 7d) on *(Customer-name, employee-name)* in which *t[customer-name] = t[emplyee-name]* for all tuples t is obtained. It seems redundant to list the person's name twice.

One attribute relation on *(customer-name)* which lists all those who have the same name as their personal banker is preferred. The *project* operation allows to produce this relation. The project operation is a unary operation that copies its argument relation, with certain columns left out. Projection is denoted by the

Greek pi (π). Those attributes that appeared in the result as a subscript to π. The argument relation follows π in parentheses. Some examples for project are given belows :

i) For a relation showing customers and the branches from which they borrow, independent of the amount of the loan or the loan number, the query may be expressed as in equ.(2.6).

$$\pi_{\text{branch-name, customer-name}} (\text{borrow}) \qquad \text{---} \ (2.6)$$

ii) For the query "Find those customers who have the same name as their personal banker", the equation is shown in (2.7).

$$\pi_{\text{customer-name}} (\sigma_{\text{customer-name = employee-name}} (\text{client})) \qquad \text{---} \ (2.7)$$

It may be noted that the operations discussed so far allow to extract information from only one relation at a time. One operation that allows to combine information, is known as the *cartesian product* operation, denoted by a cross (x). This operation is a binary operation. The cartesian product of relations r1 and r2 can be written as r1 X r2.

branch-name	loan-number	customer-name	amount
Downtown	17	Jones	1000
Redwood	23	Smith	2000
Perryridge	15	Hayes	1500
Downtown	14	Jackson	1500
Mianus	93	Curry	500
Round Hill	11	Turner	900
Perryridge	25	Glenn	2500

(a) The borrow relation.

branch-name	loan-number	customer-name	amount
Perryridge	15	Hayes	1500
Perryridge	25	Glenn	2500

(b) Result of query for equ. (2.1)

customer-name	employee-name
Turner	Johnson
Hayes	Jones
Johnson	Johnson

(c) Client relation table.

customer-name	employee-name
Johnson	Johnson

(d) Result of query for equ. (2.5)

Figure 2.6 Some relations and relational results.

2.5.3 STRUCTURED QUERY LANGUAGE (SQL)

SQL (Structured Query Language) for relational database management systems is becoming very popular DBMS. The major strength of SQL is that it deals with *sets* of data. In fact, SQL is defined by relational mathematics - the very base of relational database. It therefore needs no new constructs to solve any database management problem. Also, SQL offers a standard (as defined by ANSI and IBM) method to query very large databases and exchange data with mainframes.

SQL provides four Data Manipulation Language (DML) statements :

- **SELECT, UPDATE, DELETE, INSERT.**

and some Data Definition Language (DDL) commands, such as, creating tables, views, indexes and adding columns to existing tables.

2.5.4 QUERY OPTIMIZATION

There are a large number of possible strategies for processing a query, especially if the query is complex. Strategy selection can be done using information available in main memory, with little or no disk accesses. The actual execution of the query will involve many accesses to disk. Since the transfer of data from disk is slow relative to the speed of main memory and the central processor of the computer system, it is advantageous to spend a considerable amount of processing to save disk accesses.

Given a query, there are generally a variety of methods for computing the answer. It is the responsibility of the system to transform the query as entered by the user into an equivalent query which can be computed more efficiently. The "optimizing", or, more accurately, improving of the strategy for processing a query is called ***query optimization.***

The first action the system must take on a query is to translate the query into its internal form which (for relational database systems) is usually based on the relational algebra.

Each relational algebra expression represents a particular sequence of operations. The first step in selecting a query-processing strategy is to find a relational algebra expression that is equivalent to the given expression and is efficient to execute. There are a number of different rules for transforming relational algebra queries, including :

* Perform selection operations as early as possible.
* Perform projection early.

The strategy choose for a query depends upon the size of each relation and the distribution of values within columns. In order to be able to choose a strategy based on reliable information, database systems may store statistics for each relation. These statistics include :

* The number of tuples in the relation .
* The size of a record (tuple) of relation in bytes (for fixed length records).
* The number of distinct values that appear in the relation for a particular.

The first two statistics allow us to estimate how many tuples satisfy a simple selection predicate.

Statistical information about relations is particular useful when several indices are available to assist in the processing of a query. The presence of these structures has a significant influence on the choice of a query-processing strategy.

CHAPTER-3

LOGICAL APPROACH TO DESIGN OF LOAN LEDGER ACCOUNTING SYSTEM

3.1 INFORMAL STRATEGY OF LOAN LEDGER PROBLEM

Major features of Bangladesh House Building Finance Corporation is to provide loan to public for building their houses. The Authority of the corporation considers their loan application after threadbare examination of all related papers and documents and sanctions loans according to ceiling, category, type of the applicant. Then the person getting the loan becomes a bonafide borrower of the corporation.

After sanctioning the loan, the amount is divided into some equal instalments for disbursement to the borrower. The instalments are then paid to the borrowers through investigation of the progress of works. The borrower is liable to complete the construction of the as per estimate submitted to the corporation. After completion of the house the borrower is required to pay back the loan amount instalments as fixed by the corporation. Every repayment is posted in the loan ledger in the borrower account. After full repayment of the borrowed amount with interest, all papers and documents of the borrower is released[8].

3.2 DESCRIPTION OF DETAIL LOAN LEDGER PROBLEM

There are two types of loan, such as

1. General loan
2. Multi Storied loan

3.2.1 General Loan

Highest Loan Amount : For one unit house in the city/town viz. DHAKA, CHITTAGONG, SHILET, highest ceiling is limited to TK. 600,000.00. For all other districts of the country, this ceiling is limited to TK.400,000.00. For some particular Sadar Upazillas area this ceiling is limited to TK. 200,000.00 for 1000 sq.ft house (without stair). This loan is not be applicable in the name of Co-operative society.

Interest Rate : The rate of interest for amount of first Tk. 400,000.00 is fixed at 2.5 % above Bank Rate but not less than 13 % (simple). For the rest TK. 200,000.00 the rate of interest will be 16 %(simple). In special case during the repayment period of loan if the bank rate increases then the rate of interest of loan is increased and the borrower will reimburse the loan at increased monthly instalment. Other than divisional/district headquater the loans are distributed at a rate of 5 % interest (simple) for Upazillas and others rural areas.

Period of Reimbursement : Decided monthly instalment including principal, interest, and insurance is reimbursable as under :

1. Upazilla loan at a 5 % interest	30 yrs

2. Loan for House(Pacca House)	25 yrs
3. Loan for Semi - house(Semi Pacca)	20 yrs

Primary Investment : Initially minimum 10 % of loan ceiling and entitlement will be required to be invested by the borrower.

3.2.2 Multi Storied Loan

Highest Loan Amount : For the city / town of Dhaka, Narayangonj, Khulna and Chittagong for 1000 sq. ft. four storied house (except stair and balcony, including boundary wall, garage, car spaces etc), the highest ceiling is TK. 800,000.00 and for other districts fulfiling some conditions for three storied house the loans are sanctioned as required . This ceiling is not applicable in the name of Co-operative society. In this case the amount is sanctioned considering site, importance of the house and the reimbursement capacity of the borrower etc.

Interest Rate : Rate of interest will be at prevailing bank rate but not less than 10.5 % (simple) subject to the condition that if the bank rate increases, the borrower will reimburse the increased monthly instalment.

Reimbursement Period : Decided monthly instalment is reimbursable in 30 yrs. including principal, interest, and insurance.

Primary Investment : Of the total loan entitlement for eight hundreds sq.ft. house the borrower will invest minimum 2.5 % (of the estimate) and for above that is for 1000 sq.ft. house the borrower will invest minimum 10 % of his entitlement.

3.2.3 Method of Loan Application

Only one member of every family of Bangladesh can apply for loan of HOUSE BUILDING FINANCE CORPORATION. Unless the loan once taken in the name of one member of a family is not completely reimbursed, any other member of the family (family member means husband,wife, or dependent children) will not be eligible to apply for such loan. Every applicant will have to apply for loan in the specified application form and as per specified rules of the H.B.F.C .

3.2.4 Issue of Disbursement Instalment

After sanctioning of loan, the sanctioned amount is disbursed in some instalments keeping in consistency of the construction works and subject to proper submission of all papers and documents. The difference between construction expenditure (as per estimate) and sanctioned amount of loan is required to be invested by the applicant as initial investment before receiving of the first instalment of loan. The cheque of the first instalment loan sanctioned is issued to the applicant subject to submission of registered mortgage deeds. The cheque for every next instalment is required to receive by the borrower within next three months of the first previous cheque received subject to considering the construction works.

After sanctioning of loan all cheques are issued from related zonal office subject to considering required investment. Interest will be charged from the date of the rest of that month in which the cheque is received.

A case is considered "incomplete" until the repayment date starts.

Repayment period counting will be considered from repayment date.

Lump interest will be decided by calculation of interest of every month and adding the same to the interest balance of previous month as usual upto fifth month of the last cheque received (However, advanced interest will not be posted into the ledger. The actual interest which will come at the end of month will be posted in the ledger). Lump interest is divided into 12 equal instalment and it is realizable 12 equal monthly instalment from the 7th month of the last cheque received.

In case of link case with category no. 08, the last cheque of category no. 08 will be treated as last cheque.

3.2.5 Method of Loan Repayment

Unless refixation or special indication in normal cases, the repayment date will be started from the first day of the 7th month of the last cheque received and monthly instalment will be repayable. Rate of monthly instalment of loan repayment is mentioned in the sanctioned letter. Except if every borrower is noticed to repay loan along with deposit book after receiving the last cheque by him. If any borrower does not receive the notice in time, then he may enquiry in the related office of the corporation.

3.2.6 Calculation of instalment

a. Formula for Principal Instalment :

$$\frac{SA}{RY \times 12} + SA \times 0.0001$$

b. Formula for Interest Instalment :

$$\frac{SA \times RI \times \{(RY \times 12) + 1\}}{RY \times 28800}$$

Where

SA = Sanctioned Amount

RY = Repayment Year

RI = Rate of Interest

If any borrower doesnot repay instalment of a month in time, then the decided higher rate of interest on principal amount of the defaulted instalment is charged. This additional rate of interest is limited to maximum 17.5 % at present. In this case the decided rate of interest will be applicable to be charged on the balance of principal amount on the basis of days of the month.

The case in which penal interest will be charged that will require to be done in accordance with the particular penal code and that is to be kept in a separate head. The penal interest is chargable from the date of repayment.

Formula for Penal Interest (IDIP) :

$$BOD \times \frac{IP}{100} \times \frac{D}{365}$$

Where

BOD = Balance Outstanding on Defaulting instalment of principal

IP = Penal rate of interest for no. of Principal instalment default

D = No. of days in a month

Formula for charged normal interest :

$$BOP \times \frac{IN}{100} \times \frac{D}{365}$$

Where

BOP = Balance Outstanding on Principal

IN = Normal rate of interest

D = No. of days in a month

If the amount of instalment is deposited within the 7th day of a month that deposit will be treated as a deposited in the first day of that month and accordingly as per rule interest and principal are adjusted in the account and monthly interest and penal interest are required to be calculated. If an amount of instalment is deposited after 7th day of a month, in that case normal interest at a decided rate of interest will be calculated and deposited amount will be adjusted including interest /principal as per rule and for the rest of the time, the rest of the balance of the principal will be taken to charge interest for posting.

If the amount of instalment deposited by a borrower in a particular month towards repayment of instalment exceeds repayment of defaulting lump amount then from the rest of the amount the defaulting interest and repayment of instalment for that period will be adjusted. If the amount of deposit is not consumed still then principal instalment default, principal instalment, and total balance will be adjusted accordingly.

Repayment period will be counted from repayment date.

Time expired cases will be marked taking into account the repayment date of the case and that requires to be marked in the ledger.

As per information supplied the words "ABANDONED" and "SECOND MORTGAGED" are required to be marked in the ledger.

If a deposit is given before starting repayment date that will be credited to principal account.

In cases of cases other than Time expired/incomplete cases if an amount is deposited then the same will be adjusted as under :

1. Lump interest or upto date instalment of lump interest.

2. Balance of complete penal instalment.

3. Upto date interest instalment.

4. Instalment of principal account.

In case of time expired cases the deposited amount will be adjusted as under:

1. whole balance of interest amount.
2. whole balance of principal amount.

After posting of deposited amount, if it shows credit balance in any category of any account then it requires to be shown in the interest side is nill or insufficient then the rest of the amount or the whole credit balance requires to be transferred to other category account of that account number.

If the category mentioned in the payment statement/voucher statement are absent in the ledger account then the amount may be posted to any category of the account number.

If any category of account number are not mentioned in a ledger then the amount along with information , memo / voucher/ disbursement will be shown in the suspense list as debit / credit.

After posting/adjustment of the deposited amount the date of lump interest repayment and covering date and principal,interest instalment requires to be mentioned in the ledger.

The defaulting instalment of principal/interest are to be decided on the basis of covering date and that will be mentioned in the ledger; overdue amount will be prepared taking defaulting lump interest and whole balance of penal interest and defaulting instalment of interest/principal.

Amount of voucher statement will be posted in the mentioned head and as per requirement covering date will be changed .

There will be a provision for correction the borrower/borrowers name, address, interest rate of account, repayment period, penal interest rate etc through correction slip.

Every year in the month of february, yearly insurance premium charge and list will be supplied to the insurance company on the basis of sanctioned amount.

If an amount of deposit is posted in a month after the month of actual deposit in that case on the amount which will be posted in the principal head interest benefit will be calculated at a decided interest rate upto previous day of posting for onward credit to interest account.

If a borrower fails to repay the regular monthly instalment then on the defaulted instalment of principal amount interest are charged of a decided higher rate. This additional higher rate of interest is maximum 17.5 % at present.

3.2.7 Some Formulas used in Ledger :

1. $I = \dfrac{0.90 \times SA}{1000}$

   Where , I = Insurance Premium
   SA = Sanctioned Amount

2. Principal Credit = Principal instalment X given instalment no. in that year.

3. Interest default total = IDIP X Interest Instalment of the month + Previous total interest default.

4. Interest side total Balance = IDIP + Normal Interest charged + Previous Total Balance.

5. Total Principal Balance = Previous Principal + New charged amount.

6. Expiry Date = Repayment date + Period of loan.

3.2.8 Coding of Type, Category, and Purpose code of Loan

Type of loan :

General	1
Multi Storied	2

Category of Account :

Original	=	00	Additional	=	01	2nd Additional	= 02
3rd Additional	=	03	Special	=	04	Defferential	= 05
Old multi	=	06				Old multi defferential	= 07
16 % General	=	08					

Purpose code

Memo no.	= M	Insurance	= I	Low charge	= L
Advertisement cost	= A	Voucher adjustment	= V		
Lump Interest	= LI	Refund by cheque	= R		
Cheque	= CV	IDIP	= PI		

3.2.9 Design / Modification of a House

Without the recommendation of the proper authority a design can not be changed and without prior permission of the corporation, a house constructed by loan from corporation can not be changed in any way.

3.2.10 Transfer of mortgaged property

Besides Corporation, any kind of transfer (sale, mortgage, heba etc.) is illegal. After fulfilment of required conditions a house can be transferred through deed. In such cases all expenditures incurred are to be brone by the saler. Even a portion of a mortgaged property is not released before receipt of full payment of the loan.

3.3 DATA FLOW DIAGRAM

The *Data flow diagram (DFD)* is used as a graphical tool to depict information flow. The representation of information flow is one element of the requirements analysis activity that call information domain analysis[9]. DFD represents the *Software Requirements Specification*. The Data Flow Diagram of Loan Ledger Accounting System is shown in figure - 8 and sequence of information flow is presented according to the serial number mentioned in the figure. The informations and their sequence number shown in figure - 3.1 are as follows :

Sequence no.	Information
1	Application for loan to the Sanctioning Authority.
2	Inform the applicant about application after details examination.
3	After sanctioning loan, give information to concern authority.

4	Give information to Disbursement Section.
5	After sanctioning loan, submits the documents and performs mortgage deeds.
6	After threadbare examination of documents, applicant is enlisted as a borrower.
7	Give permission for disbursement.
8	In consistency with the construction, disburse the amount.
9	After total disbursement, reimburse the repayment amount.
10	Issue yearly loan ledger statement.
11	After total repayment, give information.
12	Release all documents and give full ownership to the borrower.
13	Inform about incomplete expiry cases.
14	Reimburse the total amount by selling the house through auction and release all documents and give ownership to the new owner.

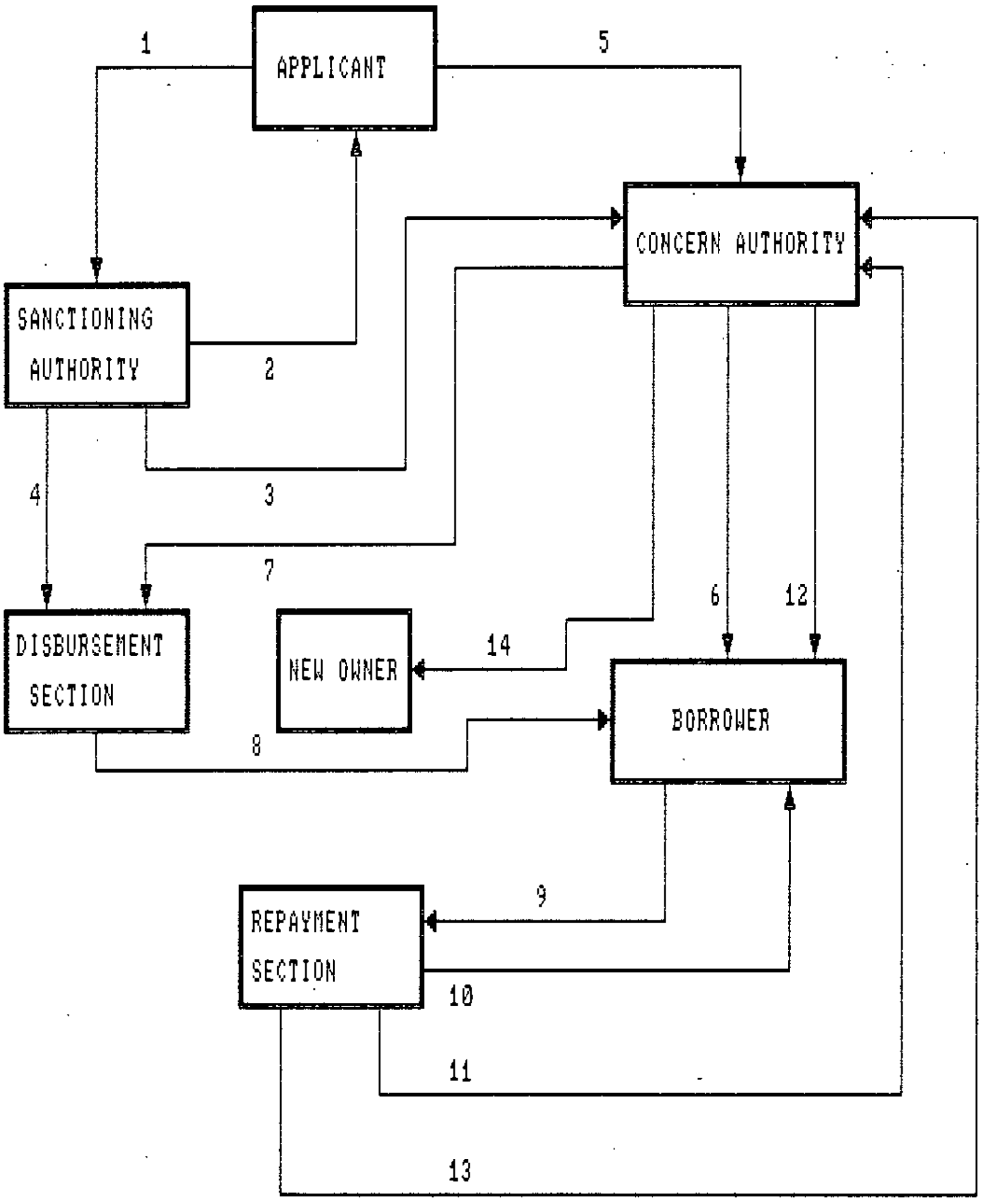


Figure - 3.1
Data Flow Diagram of Loan Ledger Accounting System.

3.4 ENTITY - RELATIONSHIP DIAGRAM

The overall logical structure of a database can be expressed graphically by an E-R diagram. The entity - relationship diagram is based on a perception of a real world which consists of a set of basic objects called *entities* and *relationships* among these objects[10].

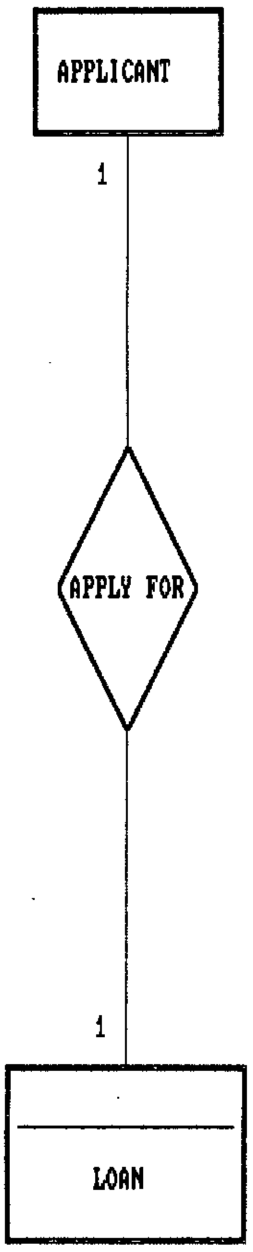
3.4.1 Drawing of ERDs

By analyzing the DFDs, a list of entities and relationship is determined. Figure - 3.2 to 3.7 show the localized ERDs as a result of the identification of entity & relationship types and functionality of relationships.

The logical ERDs are grouped together to form the normalized ERDs. Figure - 3.8 is such a grouped ERD obtained from the localized ERDs of figure - 3.2 to figure - 3.7.

3.4.2 Identification of Attributes

Attributes are the properties of an Entity. These attributes when chosen by itself can serve as a prime indicator and establish relationship with Entities. Thus the attributes can also identify each entity and relationship. The key were shown in figure - 3.2 to figure - 3.7 in the form of underlined attributes.



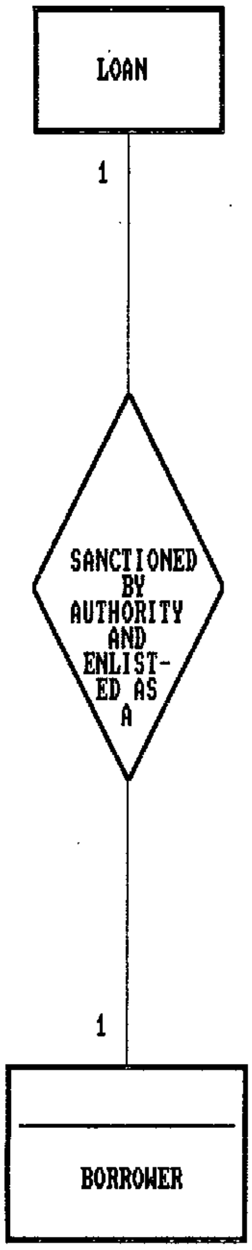
application no, name, address, region, code, category

Functionality of Relationship : 1 : 1

Obligatory side	: Loan
No. of table	: 2

code, category, interest_rate, loan_period

Figure - 3.2
Localized ERDs for the entities of people and loan.



code, category, interest_rate,
loan_period

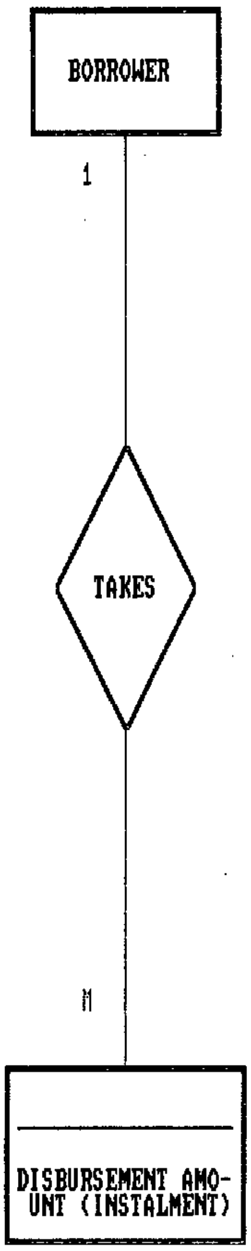
Functionality of Relationship :
1 : 1

Obligatory side :
Borrower

No. of table : 2

code, acc no, category, sanction_
amount, repay_start_date, first_
portion_name, 2nd_portion_name,
address_line1, address_line2, address_
line3, ledger_no,region

Figure - 3.3
Localized ERDs for the entities of loan and borrower.



code, acc no, category, memo_no, pay_date, principal_balance, principal_instalment, principal_instalment_default, penal_interest_on_principal_default, interest_instalment, interest_instalment_default, lump_interest_instalment, lump_int_default, normal_interest_balance

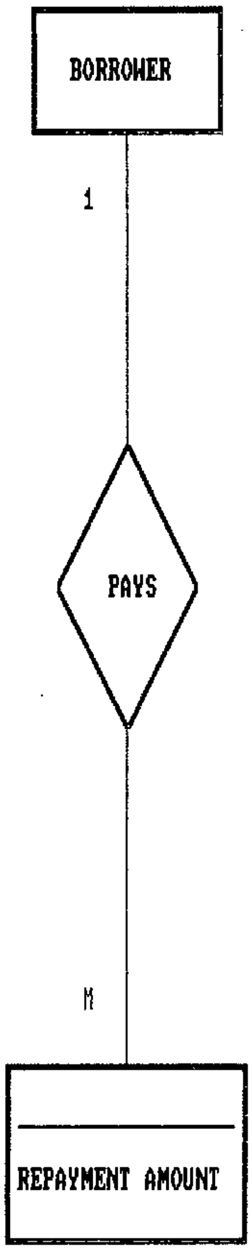
Functionality of Relationship : 1 : M

Obligatory side : Disbursement Amount

No. of table : 2

code, acc no, category, region, issue_date, cheque_no, amount, instal_no, delivery_date,scroll_date, purpose_code

Figure - 3.4
Localized ERDs for the entities of borrower and disbursement amount instalment.



code, acc no, category, memo_no, pay_date, principal_balance, principal_instalment, principal_instalment_default, penal_interest_on_principal_default, interest_instalment, interest_instalment_default, lump_interest_instalment, lump_int_default, normal_interest_balance

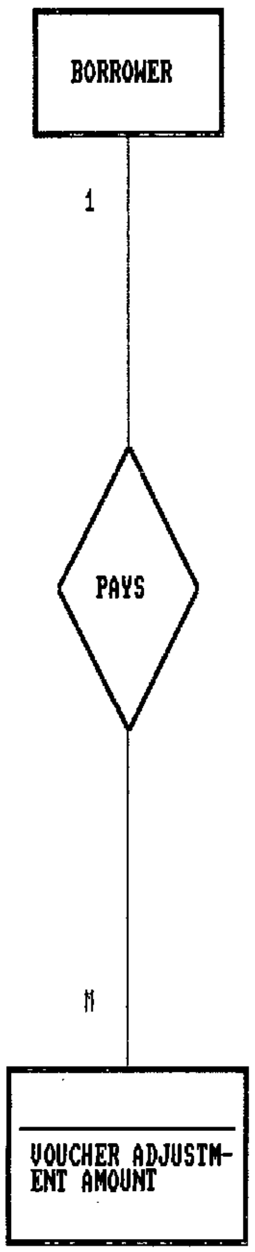
Functionality of Relationship :
1 : M

Obligatory side :
Repayment Amount

No. of table : 2

code, acc no, category, memo_no, bank, pay_date, amount, purpose_code, scroll_date, operator_no

Figure - 3.5
Localized ERDs for the entities of borrower and repayment amount.



code, acc no, category, memo_no, pay_date, principal_balance, principal_instalment, principal_instalment_default, penal_interest_on_principal_default, interest_instalment, interest_instalment_default, lump_interest_instalment, lump_int_default, normal_interest_balance

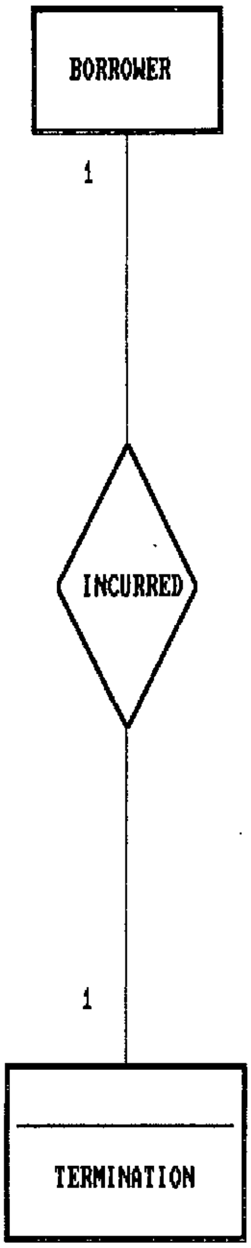
Functionality of Relationship :
1 : M

Obligatory side :
Voucher adjustment amount

No. of table : 2

code, acc no, category, region, purpose_code, voucher_pay_date, amount

Figure - 3.6
Localized ERDs for the entities of borrower and voucher adjustment amount.



code, acc no, category, memo_no, pay_date, principal_balance, principal_instalment, principal_instalment_default, penal_interest_on_principal_default, interest_instalment, interest_instalment_default, lump_interest_instalment, lump_int_default, normal_interest_balance

Functionality of Relationship : 1 : 1

Obligatory side :
Termination

No. of table : 2

code, acc no, category, repay_start_date, sanction_amount

78872

Figure - 3.7
Localized ERDs for the entities of borrower and termination.

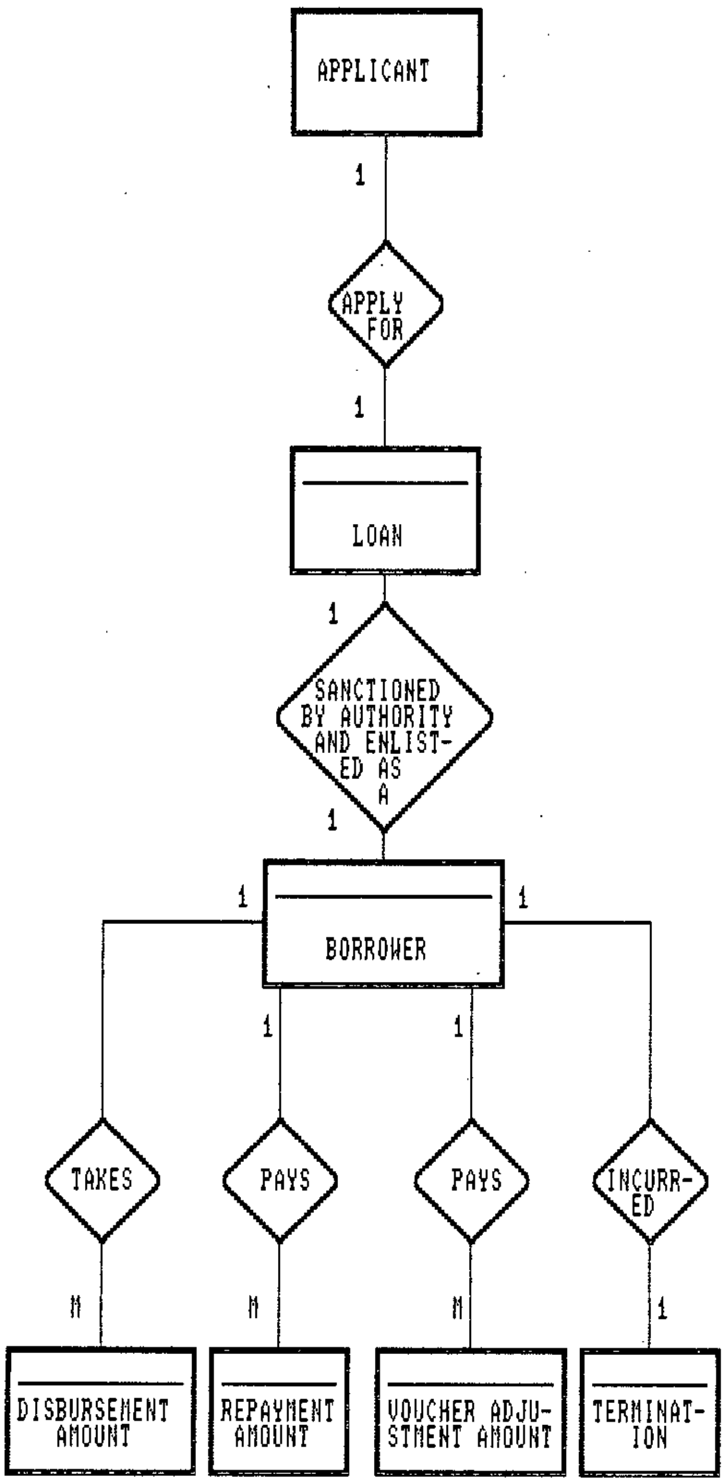


Figure - 3.8
Combined ERDs for Loan Ledger Accounting System.

CHAPTER-4

PHYSICAL APPROACH TO DESIGN OF LOAN LEDGER ACCOUNTING SYSTEM

4.1 DATA STRUCTURE AND ORGANIZATION

From the analysis of DFDs, the data stores were identified. These data have several attributes which are obtained may be kept in a database as per the allocated maximum length described in the Table - 2 to Table - 9. The tables also show the type of data to be kept in that space.

TABLE - 2. Data Structure of new borrower table.

ATTRIBUTE	TYPE	WIDTH	DESCRIPTION
code	character	1	Loan type code
region	character	3	Loan region code
acc_no	integer	4	Account no. of borrower
category	character	2	Category of loan
sanc_amount	decimal	6	Sanctioned amount
repay_date	date	4	Starting date of repay
interest_rate	decimal	3	Rate of interest
name1	character	30	First portion of name
name2	character	30	Second portion of name
address1	character	30	First portion of address
address2	character	30	Second portion of address
address3	character	25	Third portion of address
loan_period	smallint	2	Period of Loan
ledger_no	smallint	2	Ledger No. of borrower
Total (Maximum length)		172	

TABLE - 3. Data Structure of old borrower table.

ATTRIBUTE	TYPE	WIDTH	DESCRIPTION
category	character	2	Category of loan
acc_no	integer	4	Account no. of borrower
code	character	1	Loan type code
memo_no	character	6	Memo no.
p_date	date	4	Repayment date
tot_bal	decimal	6	Total balance
pri_ins	decimal	6	Principal Instalment
in_ins	decimal	6	Interest Instalment
pr_ins_def	decimal	6	Principal Instalment default
id_amt	decimal	6	Instalment default of penal interest
in_ins_def	decimal	6	Interest instalment default
lp_in_amt	decimal	5	Lump interest amount
lp_in_def	decimal	4	Lump interest default
n_in	decimal	6	Normal interest amount
Total (Maximum length)		68	

TABLE - 4. Data Structure of disburse table.

ATTRIBUTE	TYPE	WIDTH	DESCRIPTION
category	character	2	Category of loan
region	character	3	Loan region code
acc_no	integer	4	Account no. of borrower
issue_date	date	4	Cheque issue date
amount	decimal	6	Disbursement instalment amount
cheque_no	character	6	Cheque number
install_no	character	2	Installment number
delivery_date	date	4	Cheque delivery date
sl_date	date	4	Scroll date
purpose	character	1	Purpose of payment
code	character	1	Loan type code
Total (Maximum length)		37	

TABLE - 5. Data Structure of repay table.

ATTRIBUTE	TYPE	WIDTH	DESCRIPTION
memo_no	character	6	Memo number
code	character	1	Loan type code
bank	smallint	2	Bank number
p_date	date	4	Repayment date
amount	decimal	6	Amount of repayment
purpose	character	1	Purpose of repayment
acc_no	integer	4	Account no. of borrower
category	character	2	Category of loan
sl_date	date	4	Scroll date
opno	character	1	Operator number
Total (Maximum length)		31	

TABLE - 6. Data Structure of voucher table.

ATTRIBUTE	TYPE	WIDTH	DESCRIPTION
code	character	1	Loan type code
region	character	3	Region number
acc_no	integer	4	Account no. of borrower
category	character	2	Category of loan
purpose	character	3	purpose of voucher
v_date	date	4	Date of voucher adjustment
amount	decimal	6	Amount of voucher
Total (Maximum length)		23	

TABLE - 7. Data Structure of ledger out table.

ATTRIBUTE	TYPE	WIDTH	DESCRIPTION
code	character	1	Loan type code
category	character	2	Category of loan
acc_no	integer	4	Account no. of borrower
prin_c	decimal	6	Principal credit
prin_d	decimal	6	Principal debit
inst_c	decimal	6	Interest credit
inst_d	decimal	6	Interest debit

Table -7(Contd.)

ATTRIBUTE	TYPE	WIDTH	DESCRIPTION
intcg	decimal	6	Charged interest
n_pay	decimal	6	Normal Interest payment
lum_c	decimal	6	Lump credit
idip_c	decimal	6	IDIP credit
lum_d	decimal	6	Lump debit
idip_d	decimal	6	IDIP debit
prin_pay	decimal	6	Total principal payment
in_to_prin	decimal	6	Deducted amount from from principal instead of interest
Total (Maximum length)		79	

TABLE - 8. Data Structure of Penal rate table.

ATTRIBUTE	TYPE	WIDTH	DESCRIPTION
in_rate	decimal	3	Interest rate
pr_def	decimal	4	Principal default
p_rate	decimal	3	Penal rate
Total (Maximum length)		10	

TABLE - 9. Data Structure of Incom table.

ATTRIBUTE	TYPE	WIDTH	DESCRIPTION
code	character	1	Loan type code
acc_no	integer	4	Account no. of borrower
category	character	2	Category of loan
linstl	character	2	Last instalment no.
instl_date	date	4	Date of instalment
Total (Maximum length)		13	

4.2 HARDWARE CONFIGURATION

The Loan Ledger Accounting System is designed specially for mini-computers, IBM Micro-computers and clones (IBM PC or PC AT Compatible) that run under the UNIX, XENIX, and MS-DOS. The Total Number of bytes required for database is 433. Now,the Loan Ledger Accounting System is implemented for 25,000 borrowers. The system ideally needs a multi-user system for huge data entry. The system requires the following devices :

- CPU with at least 2 MB RAM for mini-computer or CPU (80286 or 80386 or equivalent) with at least 1MB RAM for micro-computer

- 40 MB harddisk

- 4 - data entry terminals

- 132 or more character high speed line printer

4.3 LANGUAGE IMPLEMENTATION

The system is implemented in INFORMIX - 4GL (embedded SQL) under UNIX - V operating system environment but suitable for micro-computer base system (e.g. Oracle, Informix - 4GL) under UNIX or XENIX or PC based Operating System, was not available at that time.

Informix - 4GL has three major components : an interactive SQL capability, an application development tool, and a report writer[11]. The interactive portion of the package to enter SQL query, store it, retrieve a previously stored query, and execute a query.

Results are displayed on the screen, and then scroll forward through them. Options to change databases, create tables, execute queries, and so forth are displayed at the top of the screen. The application developer with a fully functional development tool that can access databases using SQL. Developers can retrieve, update, and insert sets of rows with SQL. You can also use SQL to provide sophisticated, yet concise, editing logic. Informix-4GL contains a full complement of statistical functions, string manipulation commands, and array handling capabilities. It also contains basic assignment and looping constructs. Informix - 4GL is portable to a wide variety of platforms, including many UNIX machines and DEC'S VMS operating system.

4.4 COLLECTION OF DATA

Data are collected from borrowers through disbursement memo, repayment memo, voucher memo, and new borrower list. During data entry, a lot of possible on-line data validation are provided. At that level, data may cause error. To correct the error, validation listing will be generated and supplied to the concern authority. After proper validation, data are updated by the embedded update program. Some of the data collection forms and data entry forms used are attached as Appendix - A .

4.5 INPUT/OUTPUT DESIGN

There are some dialogue designs menus, question & answer, and form filling are used in designing the screen layout. The menus enable the user to select the function to be performed by the system. The Menu chart is shown in Table - 10. The program of the system is composed of four modules or sub-systems shown in figure - 4.1. Each module in turn consists of several sub-modules. This software is designed in a top - down technique. The different modules are briefly explained as follows :

Change the Database

This option is chosen to deal in database modification related to Loan Ledger Accounting System, such as Data Entry, Edit, Delete etc. The explicitly described submenus will help to choose the required option.This module invokes a menu which has five options shown in figure - 4.2 to select tables in which change will be occured.

After selection of any table, a menu will be displayed similar to any of the figure shown in figure 4.3 to figure 4.6 for the corresponding table. The following functions are comprised in the menus shown in figure - 5.3 to figure - 5.6.

<u>Data Entry</u>

By selecting this function for new borrower table, the informations of newly enlisted borrowers to the corporation is to be entered through a screen form shown in figure 4.8. When loan is sanctioned for a borrower, this amount is disbursed through some instalments. This information is entered through a screen form shown in figure 4.9. The borrower is liable to reimburse the borrowed amount. The repayment information is to be entered through a screen form shown in figure 4.10.Some amount is to be paid for voucher adjustment required due to some mistakes or for some discrepencies which may be occured from the borrower side or from the authority of the organization. This amount and its related information is to be entered through a screen form shown in figure 4.11.

After selection of any of these functions shown in figure 4.3 to figure 4.6, a menu will be displayed for choosing the month for which data is to be entered.

Table - 10. Menu Chart of Loan Ledger Accounting System

LOAN LEDGER ACCOUNTING SYSTEM
MENU CHART

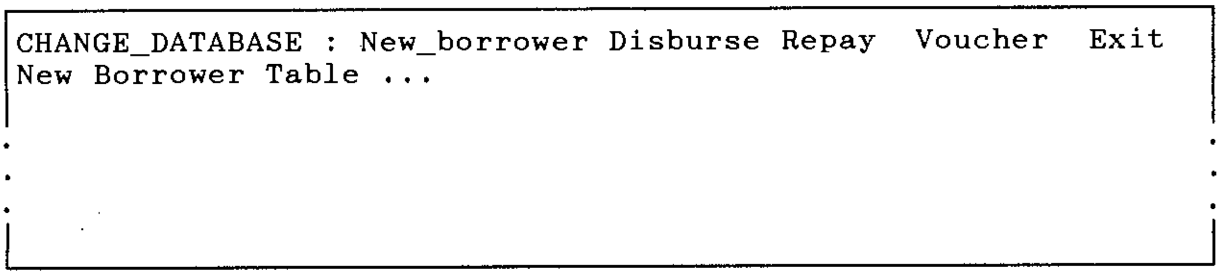
Main Menu	Detailed Functions
1. Change Database	1. New borrower table 1. Data Entry 2. Data Edit 3. Data Delete 4. Exit 2. Disbursement table 1. Data Entry 2. Data Edit 3. Data Delete 4. Exit 3. Repayment table 1. Data Entry 2. Data Edit 3. Data Delete 4. Exit 4. Voucher table 1. Data Entry 2. Data Edit 3. Data Delete 4. Exit 5. Return to Main Menu
2. Validation Listing	1. New borrower table 1. Select starting and end position 2. Select particular month 3. Select particular bank 4. Exit 2. Disbursement table 1. Select starting and end position 2. Select particular month 3. Select particular bank 4. Exit

Table - 10 . (Contd.)

	3. Repayment table 1. Select starting and end position 2. Select particular month 3. Select particular bank 4. Exit 4. Voucher table 1. Select starting and end position 2. Select particular month 3. Select particular bank 4. Exit 5. Return to Main Menu
3. Loan Ledger Statement	1. Selected number of account 1. Print on paper 2. Print on console 3. Save to a file 2. With starting account no. 1. Enter the number of account to be output 1. Print on paper 2. Print on console 3. Save to a file 3. Return to Main Menu
4. Quit the Session	

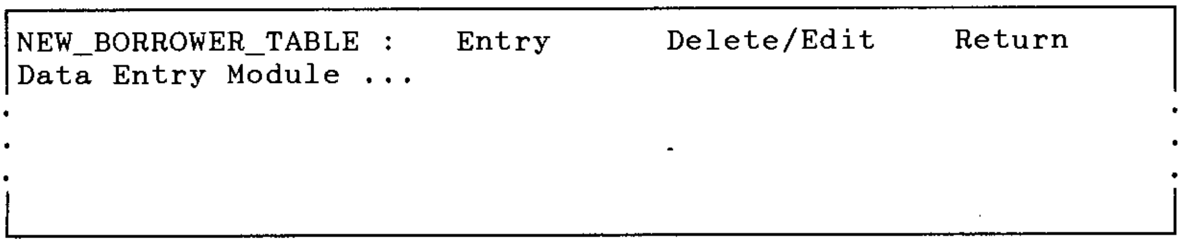
```
MAIN_MENU :Change_database  Validation_list  Loan_ledger  Quit
Changing the database...
.                                                              .
.                                                              .
.                                                              .
```

FIGURE - 4.1
Opening Menu of Loan Ledger Accounting System.



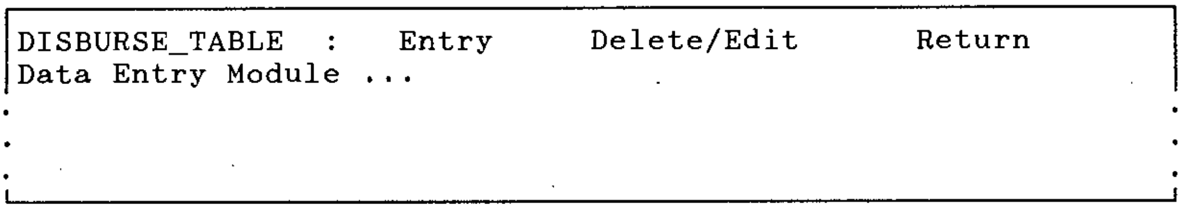
```
CHANGE_DATABASE : New_borrower Disburse Repay  Voucher  Exit
New Borrower Table ...
```

FIGURE - 4.2
Sub-menu for change_database module for selection of the table of Loan Ledger Accounting System Database.



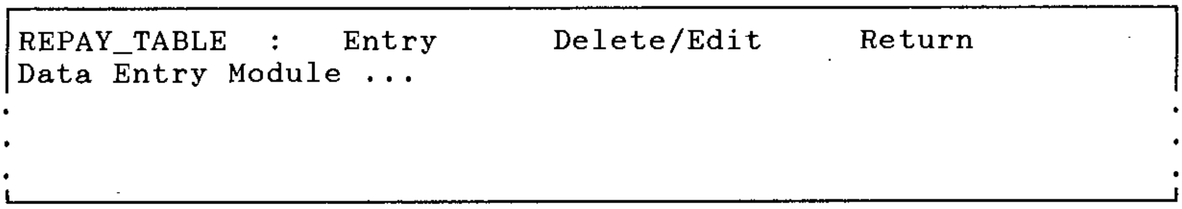
```
NEW_BORROWER_TABLE :   Entry     Delete/Edit    Return
Data Entry Module ...
```

FIGURE - 4.3
Menu for different options of changing the new borrower table of Loan Ledger Accounting System Database.



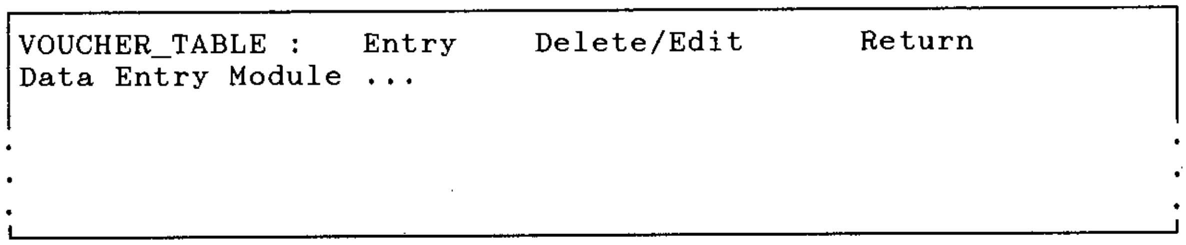
```
DISBURSE_TABLE   :   Entry     Delete/Edit      Return
Data Entry Module ...
```

FIGURE - 4.4
Menu for different options of changing the disbursement table of the Loan Ledger Accounting System Database.



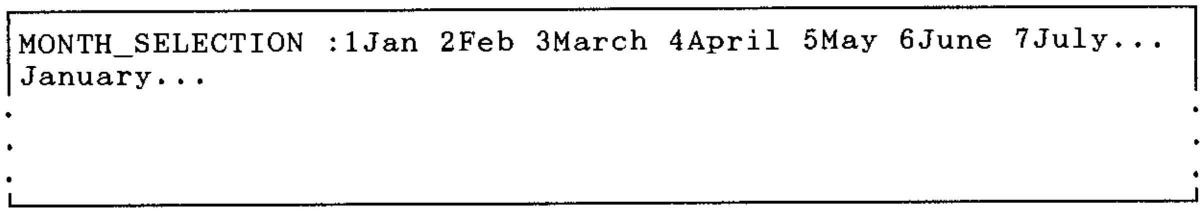
```
REPAY_TABLE  :   Entry     Delete/Edit    Return
Data Entry Module ...
```

FIGURE - 4.5
Menu for different options of changing the repay table of the Loan Ledger Accounting System Database.



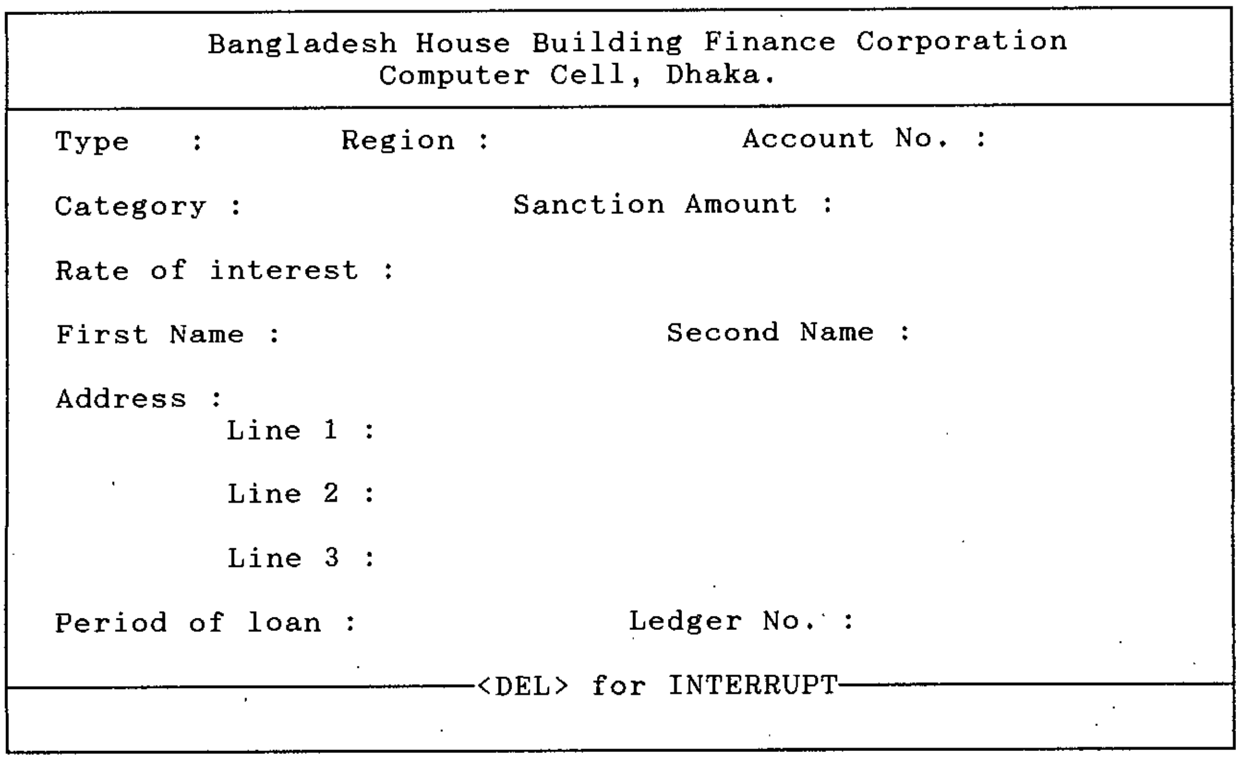
```
VOUCHER_TABLE :   Entry    Delete/Edit      Return
Data Entry Module ...
.
.
.
```

FIGURE - 4.6
Menu for different options of changing the voucher table of the Loan Ledger Accounting System Database.



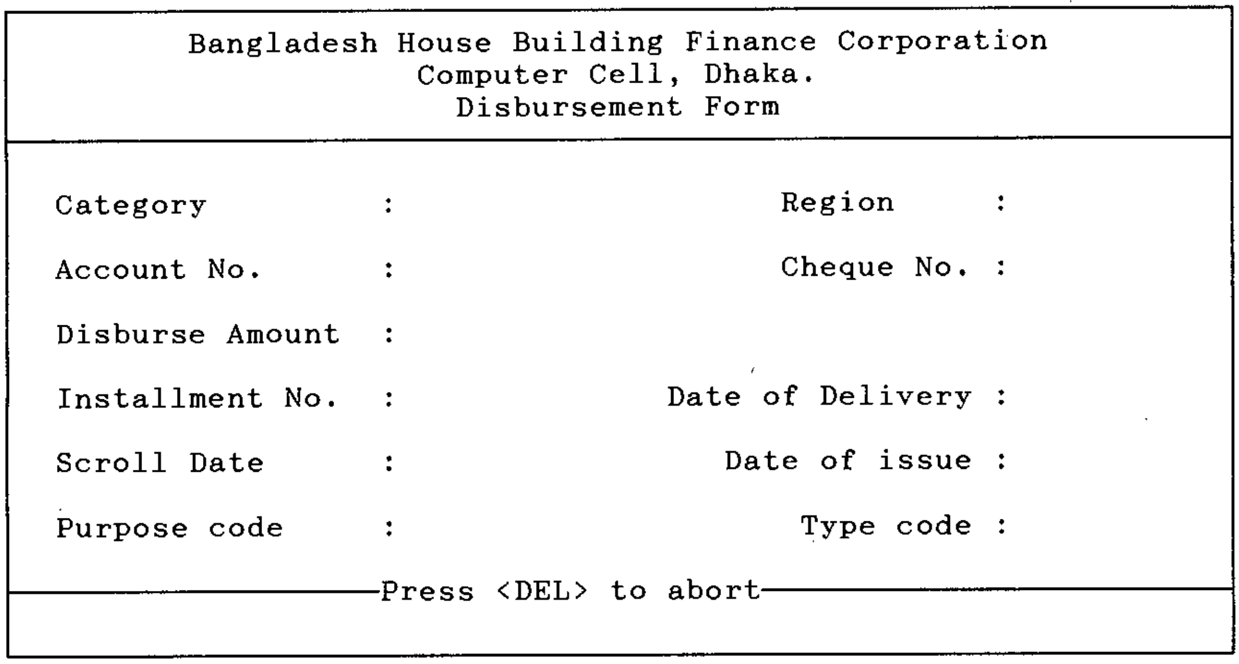
```
MONTH_SELECTION :1Jan 2Feb 3March 4April 5May 6June 7July...
January...
.
.
.
```

FIGURE - 4.7
Menu for month selection.



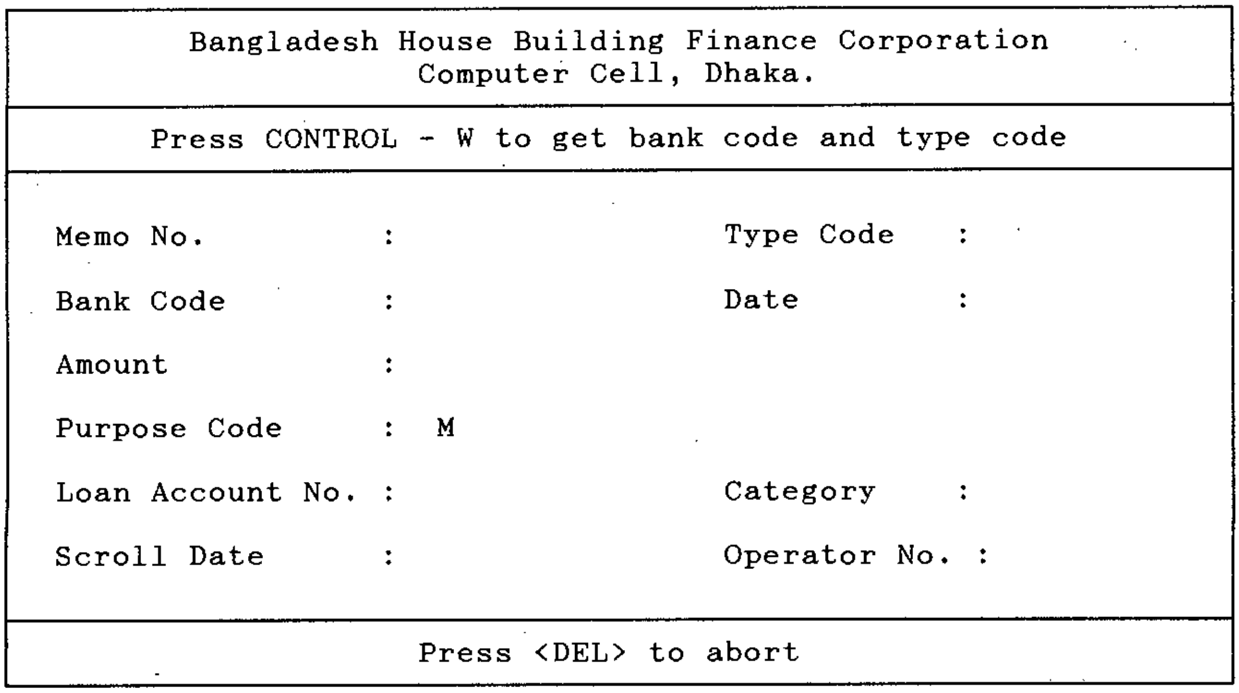
```
          Bangladesh House Building Finance Corporation
                     Computer Cell, Dhaka.

 Type    :        Region :           Account No. :

 Category :                Sanction Amount :

 Rate of interest :

 First Name :                    Second Name :

 Address :
          Line 1 :

          Line 2 :

          Line 3 :

 Period of loan :               Ledger No. :
                  <DEL> for INTERRUPT
```

FIGURE - 4.8
The data entry screen form for new borrower table.



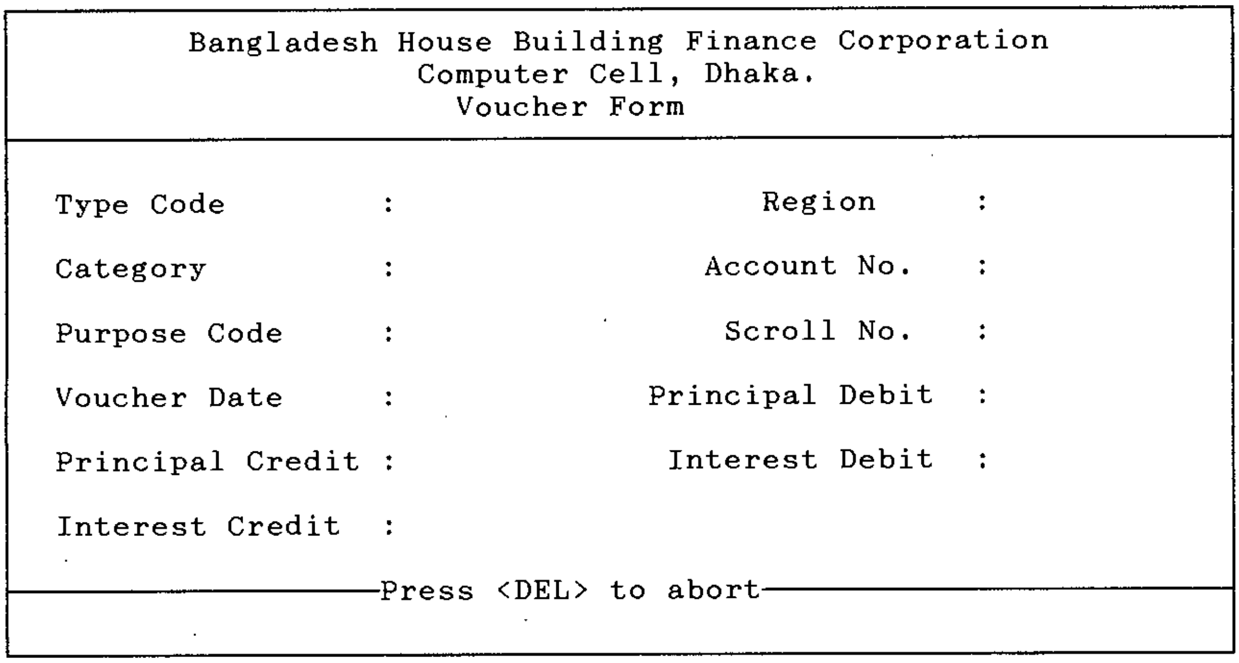
```
          Bangladesh House Building Finance Corporation
                       Computer Cell, Dhaka.
                         Disbursement Form

  Category           :                  Region       :

  Account No.        :                  Cheque No. :

  Disburse Amount    :

  Installment No.    :            Date of Delivery :

  Scroll Date        :               Date of issue :

  Purpose code       :                   Type code :

  -------------------Press <DEL> to abort--------------------------
```

FIGURE - 4.9
The data entry screen form for disbursement table.



```
          Bangladesh House Building Finance Corporation
                       Computer Cell, Dhaka.

       Press CONTROL - W to get bank code and type code

  Memo No.           :               Type Code    :

  Bank Code          :               Date         :

  Amount             :

  Purpose Code       :  M

  Loan Account No. :                 Category     :

  Scroll Date        :               Operator No. :

                     Press <DEL> to abort
```

FIGURE - 4.10
The data entry screen form for repay table.



Bangladesh House Building Finance Corporation
Computer Cell, Dhaka.
Voucher Form

Type Code	:	Region	:
Category	:	Account No.	:
Purpose Code	:	Scroll No.	:
Voucher Date	:	Principal Debit	:
Principal Credit	:	Interest Debit	:
Interest Credit	:		

Press <DEL> to abort

FIGURE - 4.11
The data entry screen form for voucher table.

Data Edit

By selection of this function, the data can be edited in any of the tables. Some mistakes may be occured during data entry, so correction or modification in this case can be done through the forms shown in figure from fig.4.12 to fig.4.15 for respective tables. All the existing information will be displayed on these screen forms and necessary changes may be done.

Data Delete

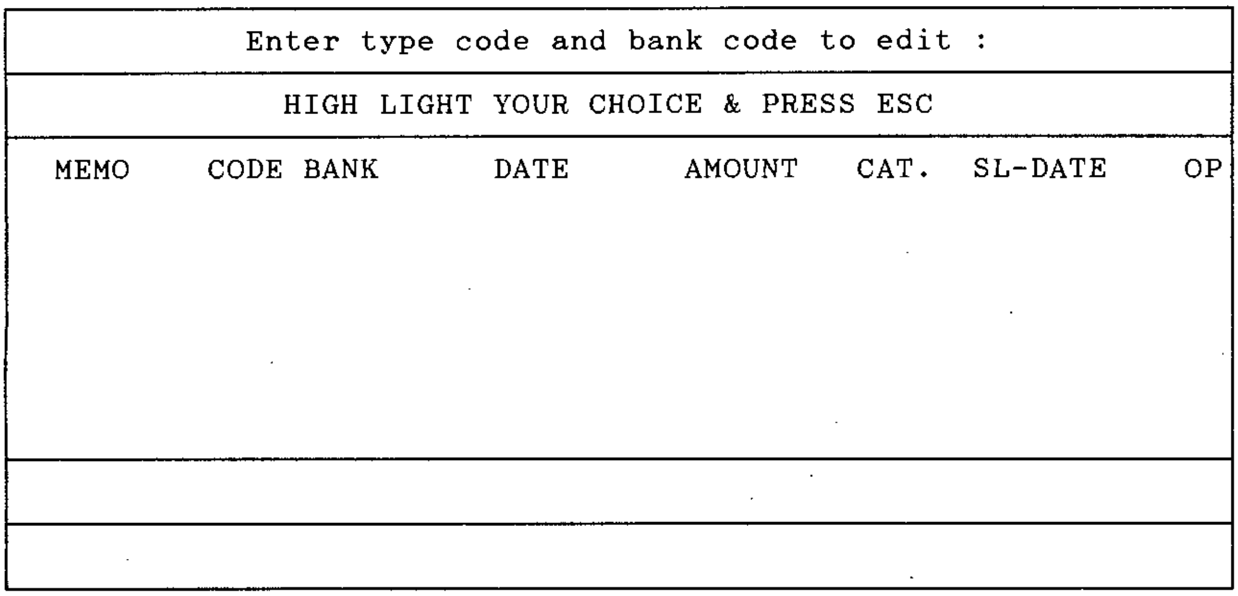
Information may be duplicated due to mistake or unnecessary data may be entered. These data are required to delete from the database. After selection of this function and month, a screen form similar to figure shown in figure 4.12 to figure 4.15. Existing information will be displayed on the screen form and necessary deletion may be done.

```
Type   Region    Account No.    Category    Rate      First Name
            Second Name                     Address(Line 1)
     Address(Line 2)                    Address(Line 3)
```

FIGURE - 4.12
The data edit/delete screen form for new borrower table.

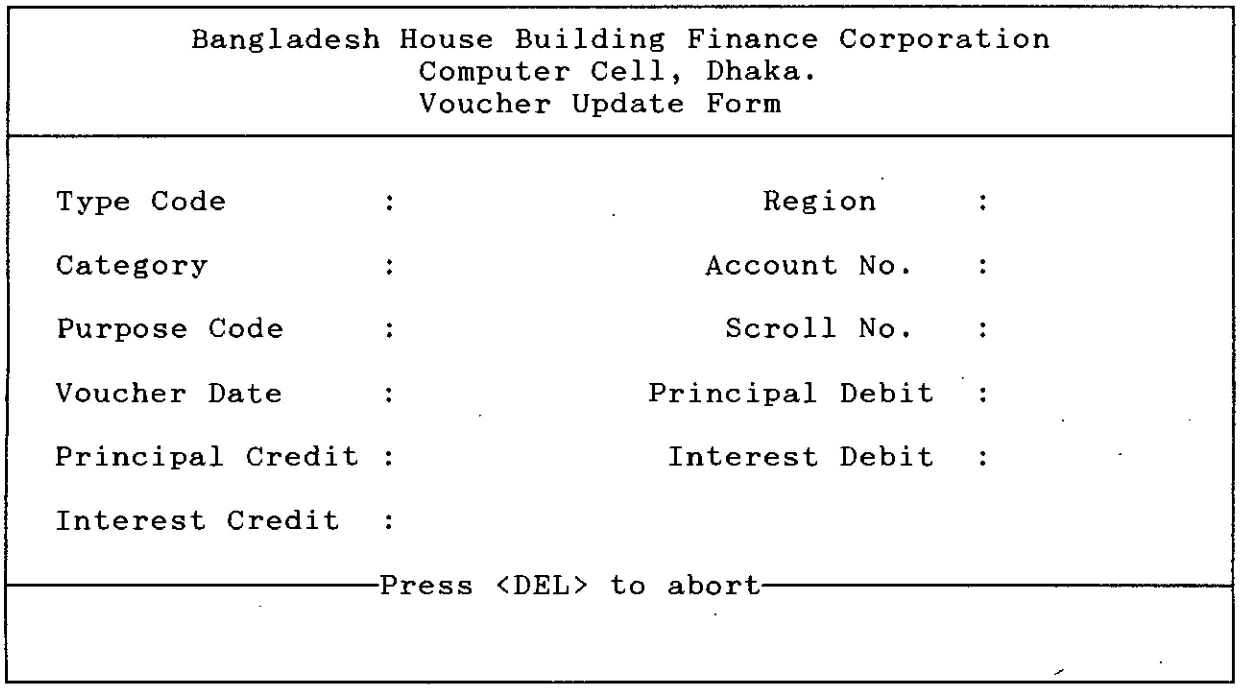
```
          Bangladesh House Building Finance Corporation
                      Disbursement Update

  Category          :                    Region      :

  Account No.       :                    Cheque No.  :

  Disburse Amount   :

  Installment No.   :              Date of Delivery  :

  Scroll Date       :                 Date of issue  :

  Purpose code      :                     Type code  :

Total =             | F2 = Next | F3  = Previous | Del = Exit
                       Esc = Go
```

FIGURE - 4.13
The data edit/delete screen form for disburse table.



```
            Enter type code and bank code to edit :

                 HIGH LIGHT YOUR CHOICE & PRESS ESC

 MEMO     CODE BANK      DATE       AMOUNT   CAT.  SL-DATE    OP
```

FIGURE - 4.14
The data edit/delete screen form for repay table.



```
          Bangladesh House Building Finance Corporation
                      Computer Cell, Dhaka.
                      Voucher Update Form

 Type Code        :                   Region        :

 Category         :                Account No.      :

 Purpose Code     :                 Scroll No.      :

 Voucher Date     :             Principal Debit     :

 Principal Credit :              Interest Debit     :

 Interest Credit  :

───────────────────Press <DEL> to abort──────────────────────────
```

FIGURE - 4.15
The data edit/delete screen form for voucher table.

__Validation Listing__

After completion of data entry, choose this option to validate data for Ledger printing. The validation process is completely done manually. The incorrect data are corrected through the edit options of the Changing Database option.

This module invokes a menu which has four options shown in figure - 4.2. This menu is for selection of table for which validation listing is required. After selection of table, a menu will be displayed which is shown in figure - 4.16. This menu is for selection of some condition for which validation listing is required.

```
SELECT_PRINTING_CONDITION : Start_&_end Particular_month Particular_bank Exit
Select Starting and End Position ...
```

FIGURE - 4.16
Menu to Select Conditions for Validation Listing
of Loan Ledger Accounting System.

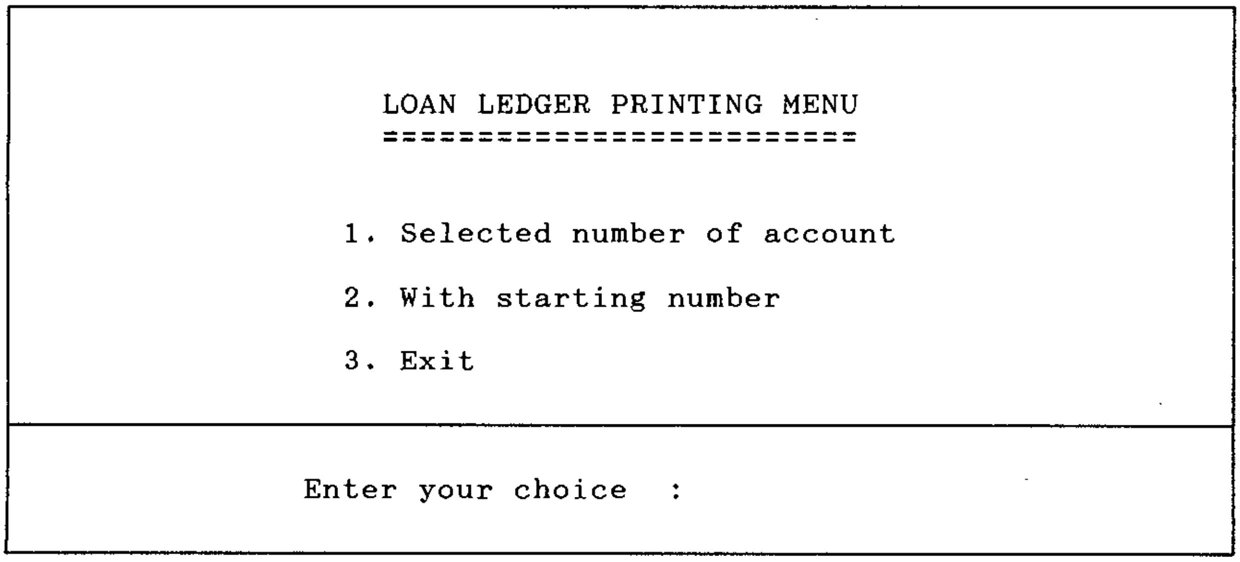
__Loan Ledger Statement__

This option has the facilities for printing Loan Ledger Statement for a selected account or a number of accounts. The submenus under this option opens up a wide variety of choice in which the ledger statement reportsis to be generated.

This module invokes a menu which has three options shown in figure - 4.17. If first option is selected, a print media menu shown in figure - 4.18 is displayed. Each option invokes two queries shown in figure - 4.19 and figure - 4.20 consecutively. After completion of this jobs, a query shown in figure - 4.21 is displayed.

If second option is selected from Ledger Printing Menu shown in fig.4.17, after selection of any options in figure 4.18, a query shown in fig.4.19 is displayed. After selection the starting account no., a query in figure 4.22 is displayed.

If third option is selected from print media menu shown in fig. 4.18, a query shown in fig. 4.23 is displayed for entering file name.



```
                LOAN LEDGER PRINTING MENU
                ========================

              1. Selected number of account

              2. With starting number

              3. Exit

          Enter your choice   :
```

Figure - 4.17
The input format design for Ledger printing menu
of Loan Ledger Accounting System.

```
                    PRINT MEDIA MENU
                    ================

                 1. Print on paper

                 2. Print on console

                 3. Save to a file
--------------------------------------------------------
              Enter your choice  :
```

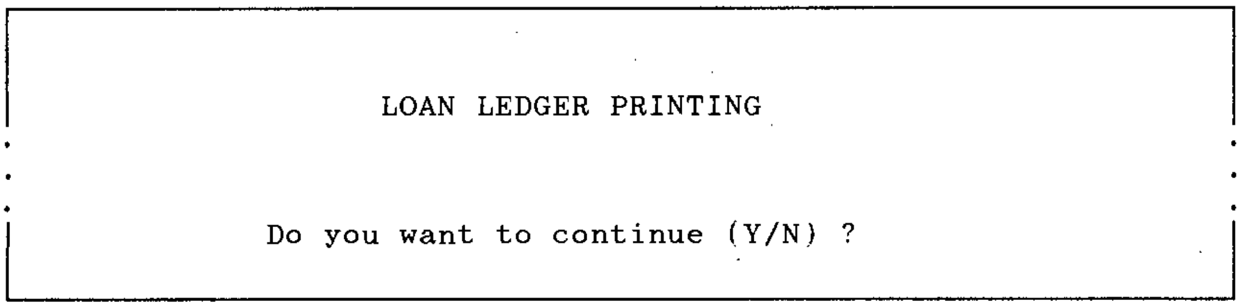
Figure - 4.18
Printing media menu of Loan Ledger.

```
                    LOAN LEDGER PRINTING
.
.
.
               Enter an account number :
```

Figure - 4.19
The input format design of account number query.

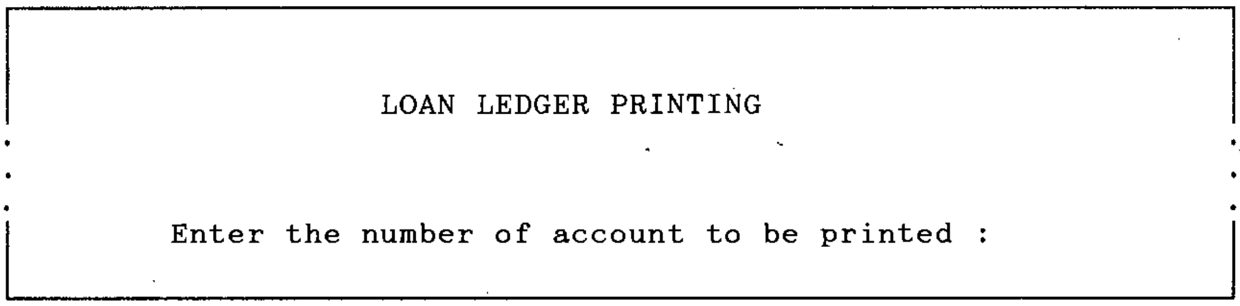
```
                    LOAN LEDGER PRINTING
.
.
.
           Enter the starting year of ledger  :
```

Figure - 4.20
The input format design of starting year query.



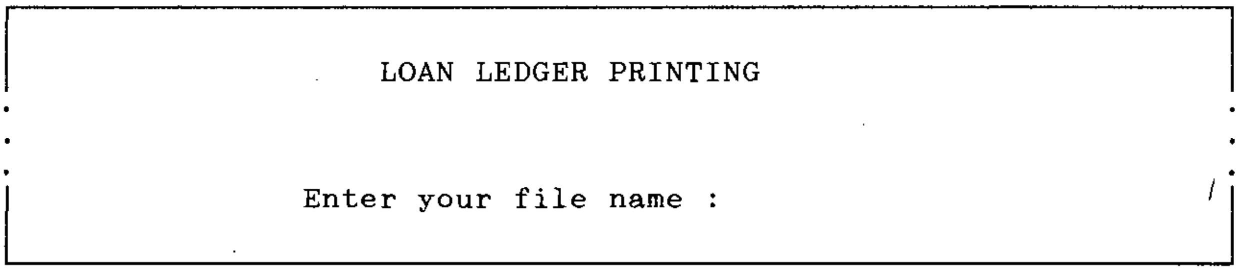
```
                    LOAN LEDGER PRINTING



            Do you want to continue (Y/N) ?
```

Figure - 4.21
The input format design for print continuation query.



```
                    LOAN LEDGER PRINTING



       Enter the number of account to be printed :
```

Figure - 4.22
The input query format design for number of account
to be printed is required.



```
                    LOAN LEDGER PRINTING



               Enter your file name :
```

Figure - 4.23
The input query format design for file name onto which
ledger print out to be stored.

4.6 PROGRAM LISTING

Program listing has a great role to play during maintenance of the system in due course of time or whenever required. Without the listing of program codes the future development of the system is extremely difficult.

The list of the program is given in Appendix - B.

CHAPTER–5

CONCLUSIONS

5.1 RESULT, DISCUSSION, AND CONCLUSION

A practical step-by-step methodology for relational database design can be derived using a variety of extensions to the ER conceptual model. This relational database design approach uses both the ER model and the relational model in successive stages. It benefits from the simplicity and ease of use of the entity-relationship model and the structure (and associated formalism) of the relational model. In order to achieve this approach, it is necessary to build a framework for transforming the variety of ER constructs into relations that can be easily normalized.

In the Loan Ledger Problem, reporting is done in On-line process. To make the processing faster, a number of temporary tables are created during processing and will be deleted after execution. A no. of auxiliary tables are created permanently for keeping information for Management Information System and for some report generation. For data entry into voucher table, to avoid insertion anomalies define a single attribut for different items by one recognisiable character which causes saving of memory.

The data stored in the database needs to be protected from unauthorized access,malicious destruction or alteration,and accidental introduction of inconsistency. To protect the database from malicious abuse and against accidental loss of data consistency, every operator and user is assigned to a particular user number to access to the database. Since a lot of queries are involved in this database, the strategies of *Query Optimization* are used to make faster processing.

In developing the program, the task is defined some set of units which then put together in a defined way, and realised the overall object. The criteria of *Active Decomposition and Modularity* are applied to decompose the whole program to some possible functions and then followed some stepwise *Refinement* which is an early top-down design strategy.

The design process is also of interest since it offers a guide to the design of the databases and their modifications induced from external events. It introduces in a simple way the notion of event that is commonly used in the literature but seldom clearly defined. Analysis of the event ensures the completeness and integrity of the database under various applications of the enterprise.The methodology has been illustrated with a database design problem, showing each design step in detail.

As a result, the facts of the real world which are of interest to the enterprise are considered from both static and dynamic points of view. The perception of data of the user is reflected in a natural way in the design of the Loan Ledger System and also in the design of the programs.

The implementation plan is carefully considered so that the system can be run and modified in any machine. Thus, the language needs to be highly portable and easily modified and maintained by other programmers.

The system is highly interactive and users friendly in nature so that the design of input and output should give ease of use to the operator.

5.2 RECOMMENDATIONS FOR FUTURE WORKS

Additional functions can be added, i.e. Bank reconciliation, Balance of Account, Loan Ledger Summary, Yearly Insurance Statement Summary, Statement of Expired Account, Statement of incomplete cases, Interest wise disbursement and income statement etc. as the present Loan Ledger System has included only limited operations of the Organization. Thus, it is desirable to add other operations so that the users can benefit more this information system.

The overall performance of the Loan Ledger Accounting System can be evaluated in the future in order to determine the efficiency of the system in terms of storage and memory allocations and response time of each function. A review and evaluation can help the person in charge of maintenance to improve the Loan Ledger Accounting System.

Knowledge-based database system such as LDL, NAIL etc. is the recent development of Relational database system. A knowledge - based system is a programming system that has the following characteristics[12] :

1. a declarative language, that is, logic in one of its many possible forms,serving as both a host language and as a query language, and
2. supports the principal capabilities of a database system, that is, efficient access to massive amounts of data, sharing of data,concurrent access to data, and resiliency in the face of failures.

Application areas of knowledge base systems deal with massive amounts of data and need a query facility more powerful that that of typical query languages such as SQL. So, to make the Loan Ledger database system through DBMS efficient and to make faster on-line processing, Knowledge based Database Management System is recommended for future work.

APPENDIX-A

DATA COLLECTION FORMS

REGION | | | |

TYPE | |

NEW ACCOUNT OPENING STATEMENT

		C-1		*C-2*		*C-3*			
Account No.	CAT	CD Ty	Sanctioned Amount (In lakas)	Name Address I (2 fields)	CD Ty	Address II Address III Address IV (3 fields)	CD Ty	Pd of repay	Rate of interest
9 (6)	9(2)	9	9 (6)	x (25)	9	x (22)	9	9 (2)	9(4)

DISBURSEMENT STATEMENT

REGION					
TYPE					
SCROLL DATE					

CAT	Account No.	Date	Cheque No.	Amount	Install No.	Delivery date	Initial

Region : black

Type : General=1
Multi=2

CATEGORY

Original= blank
Addl= 1
	2nd addl =2
	3rd addl =3
	Old Multi =6
		Spl = 4
		Diff = 5
		Old Multi Diff = 7

Install No. 99 =Final

PAYMENT STATEMENT

BANK CODE	
TYPE	
SCROLL DATE	

Cat.	Account No.	Deposit/ Voucher Date	Memo No.	AMOUNT		Purpose Code	Initial
				Cr.	Dr.		

REGION : BLANK

TYPE : GENERAL = 1
MULTI = 2

CATEGORY

ORIGINAL — BLANK
1st Addl.—01, Difft.—05, Multi—00
2nd Addl.—02 Old Multi - 06, Diff.—07
3rd Addl.—03, Spl. ———04, 16%—08

Contd.—P/

VOUCHER STATEMENT

REGION

TYPE

Month ____________________

CAT	Account No.	Purpose code	Voucher Date	Amount to be deposited					
				Principal side		Interest side			
				Debit	Credit	Debit			Credit
						~~Normal~~	~~Lump~~	~~IDIP~~	Normal
Total									

CATEGORY

Addl.	=01	Old Multi	=06
2nd addl.	=02	Old Multi Diff.	=07
3rd addl.	=03		
Spl. ,,	=04		
Diff.	=05	16%	=08

PURPOSE CODE

M =Memo	C =Cheque
I =Insurance	R =Refund
L =Law charge	Li =Lump interest
A =Advertisment	C.B.=Closing Balance
B.F.=Brought forward	V =Adjustment

BANGLADESH HOUSE BUILDING FINANCE CORPORATION
===

LOAN LEDGER NO = 013,FOR THE YEAR 1989-1990

ACCOUNT NO = 6008	REGION = 000 TYPE = 2 CATEGORY = 00 PENAL CODE = 04		
MD.SAHADAT HOSSAIN	AMOUNT OF LOAN SANCTIONED TK. 704000.00	MONTHLY INSTALMENT	LUMP INTEREST TK. 64358.33
			L. INST DEFAULT .000
KALLANPUR,RD.,NO.-5,HO USE-8,DHAKA.	PERIOD OF REPAYMENT - 30 YEARS	PRINCIPAL TK. 2025.96	L.INSTLMENT TK. .00
		INTEREST TK. 3088.56	EXPIRY DATE : 01/09/2020
	INTEREST RATE - 10.50 PA	TOTAL TK. 5114.52	
		REPAYMENT STARTED- 01/09/1990	

PERTICULARS						*	PRINCIPAL SIDE			*	INTEREST SIDE			*	TOTAL DEFAULT	
DATE	PUR CODE	BK	MEMO NO	DL/RP DATE	INS NO	DR/CR	TOTAL BALANCE	INST DEFAULT	PENAL RATE	IDIP	NORMAL INTEREST	TOTAL BALANCE	INST DEFLT	PRIN	INTEREST	TOTAL
	BF		273006	15/06/89		213000.00	213000.00	0.000	0.00	0.00	0.00	3165.82	0.000	0.00	0.00	0.00
	C		273330	09/07/89	04	71000.00	284000.00	0.000	0.00	0.00	0.00	3165.82	0.000	0.00	0.00	0.00
JUL	IN						284000.00	0.000	0.00	0.00	2369.26	5535.08	0.000	0.00	0.00	0.00
	C		273589	08/08/89	05	70000.00	354000.00	0.000	0.00	0.00	0.00	5535.08	0.000	0.00	0.00	0.00
	C		273824	28/08/89	06	70000.00	424000.00	0.000	0.00	0.00	0.00	5535.08	0.000	0.00	0.00	0.00
AUG	IN						424000.00	0.000	0.00	0.00	3096.49	8631.57	0.000	0.00	0.00	0.00
	C		273991	13/09/89	07	70000.00	494000.00	0.000	0.00	0.00	0.00	8631.57	0.000	0.00	0.00	0.00
SEP	IN						494000.00	0.000	0.00	0.00	4021.64	12653.21	0.000	0.00	0.00	0.00
	C		274171	04/10/89	08	70000.00	564000.00	0.000	0.00	0.00	0.00	12653.21	0.000	0.00	0.00	0.00
OCT	IN						564000.00	0.000	0.00	0.00	4969.23	17622.44	0.000	0.00	0.00	0.00
	C		274360	01/11/89	09	70000.00	634000.00	0.000	0.00	0.00	0.00	17622.44	0.000	0.00	0.00	0.00
NOV	IN						634000.00	0.000	0.00	0.00	5471.51	23093.95	0.000	0.00	0.00	0.00
DEC	IN						634000.00	0.000	0.00	0.00	5653.89	28747.84	0.000	0.00	0.00	0.00
	M	26	121950	14/01/90		-634.00	633366.00	0.000	0.00	-0.00	-0.00	28747.84	0.000	0.00	0.00	0.00
JAN	IN						633366.00	0.000	0.00	0.00	5650.61	34398.45	0.000	0.00	0.00	0.00
	M	902	173155	15/02/90		-210.80	633155.20	0.000	0.00	-0.00	-0.00	34398.45	0.000	0.00	0.00	0.00
	C		274921	20/02/90	99	70000.00	703155.20	0.000	0.00	0.00	0.00	34398.45	0.000	0.00	0.00	0.00
FEB	IN						703155.20	0.000	0.00	0.00	5282.02	39680.47	0.000	0.00	0.00	0.00
MAR	IN						703155.20	0.000	0.00	0.00	6270.60	45951.07	0.000	0.00	0.00	0.00
APR	IN						703155.20	0.000	0.00	0.00	6068.33	52019.40	0.000	0.00	0.00	0.00
MAY	IN						703155.20	0.000	0.00	0.00	6270.60	58290.00	0.000	0.00	0.00	0.00
JUN	IN						703155.20	0.000	0.00	0.00	6068.33	64358.33	0.000	0.00	0.00	0.00

CL.BAL (PRINCIPAL 703155.20 N.INT 0.00 IDIP .00 LUMP 64358.33 TOTAL 767513.53)

PRINCIPAL DEBIT : 491000.00 PRINCIPAL CREDIT : 844.80 INTEREST DEBIT : 61192.51 INTEREST CREDIT : 0.00

INT.CHARGED : 61192.51 PAYMENTS (PRINCIPAL - 844.80 N.INT. - 0.00 IDIP - 0.00 LUMP - 0.00 TOTAL :- 844.80)

* * UNLESS DISCREPANCY IS NOTIFIED WITHIN A MONTH, THE STATEMENT WILL BE TREATED AS CORRECT * *

APPENDIX-B

PROGRAM LISTING

```
#-------------------------------------------------------------------
# PROGRAM   : MAIN.4GL
# AUTHOR    : MD. NESAR UDDIN BHUIYAN
# PURPOSE   : Display the Main-menu and Calling the related sub-
#             modules.
#-------------------------------------------------------------------
MAIN
 MENU "MAIN_MENU"
  COMMAND "Change_database" "To Changing the database..."
  RUN "change"
  COMMAND "Validation_list" "To Validation Listing of data...."
  RUN "valid"
  COMMAND "Loan_ledger" "To Loan ledger statement of borrower...."
  RUN "loan"
  COMMAND "Quit" "To Quit the session.... "
  CLEAR SCREEN
  EXIT MENU
 END MENU
END MAIN
#-------------------------------------------------------------------
# PROGRAM   : CHANGE.4GL
# PURPOSE   : Display menu of change database module and calling
#             related sub-modules.
#-------------------------------------------------------------------
MAIN
  MENU "CHANGE_DATABASE"
   COMMAND "New_borrower" "New borrower Table ..."
   RUN "client"
   COMMAND "Disburse" "Disburse Table ..."
   RUN "disburse"
   COMMAND "Repay" "Repay Table ..."
   RUN "repay"
   COMMAND "VOUCHER" "Voucher Table ..."
   RUN "voucher"
   COMMAND "EXIT" "Return to main menu.... "
   CLEAR SCREEN
   EXIT MENU
  END MENU
END MAIN
#-------------------------------------------------------------------
# PROGRAM   : BORROWER.4GL
# PURPOSE   : Different functions for new borrower table
#-------------------------------------------------------------------
MAIN
 MENU "NEW_BORROWER_TABLE"
  COMMAND "ENTRY_DATA" "Entry New borrower data..."
  RUN "cadd"
  COMMAND "_EDIT/DELETE_DATA" "Edit/Delete New borrower data..."
  RUN "cedit"
  COMMAND "RETURN" "Return to the previous menu..."
  CLEAR SCREEN
  EXIT MENU
 END MENU
END MAIN

#-----------------------------------------------------------------
# PROGRAM   : DISBURSE.4GL
# PURPOSE   : Different functions for disbursement
#-----------------------------------------------------------------
MAIN
 MENU "DISBURSE_TABLE"
  COMMAND "ENTRY_DATA" "Entry New borrower data..."
  RUN "cadd"
  COMMAND "_EDIT/DELETE_DATA" "Edit/Delete New borrower data..."
  RUN "cedit"
  COMMAND "RETURN" "Return to the previous menu..."
  CLEAR SCREEN
  EXIT MENU
 END MENU
END MAIN
#-----------------------------------------------------------------
# PROGRAM   : REPAY.4GL
# PURPOSE   : Different functions for repay table
#-----------------------------------------------------------------
MAIN
 MENU "REPAY_TABLE"
  COMMAND "ENTRY_DATA" "Entry Repayment data..."
  RUN "padd"
  COMMAND "_EDIT/DELETE_DATA" "Edit/Delete Repayment data..."
  RUN "pedit"
  COMMAND "RETURN" "Return to the previous menu..."
  CLEAR SCREEN
  EXIT MENU
 END MENU
END MAIN

#-----------------------------------------------------------------
# PROGRAM   : VOUCHER.4GL
# PURPOSE   : Different functions for voucher table
#-----------------------------------------------------------------
MAIN
 MENU "VOUCHER_TABLE"
  COMMAND "ENTRY_DATA" "Entry New voucher data..."
  RUN "vadd"
  COMMAND "_EDIT/DELETE_DATA" "Edit/Delete New borrower data..."
  RUN "vedit"
  COMMAND "RETURN" "Return to the previous menu..."
  CLEAR SCREEN
  EXIT MENU
 END MENU
END MAIN
```

```
# ---------------------------------------------------------------
# PROGRAM   : CADD.4gl
# PURPOSE   : Entry of new borrower information to database
#----------------------------------------------------------------
DATABASE hbfc
MAIN
    DEFINE p_file record like client.*
    DEFINE accpt CHAR(1),bcode CHAR(3),
     btype  char(1),
     bcate char(2),
     brdate date,
     birate decimal(5,2)
  OPTIONS
         PROMPT LINE 21,
         MESSAGE LINE 22
  DEFER INTERRUPT
  open window w1 at 2,2 with 22 rows,78 columns attribute (border)
   OPEN FORM vscr1 FROM "cscr"
   DISPLAY FORM vscr1
   DISPLAY "<DEL> for INTERROPT" at 20,30
   while true
   INITIALIZE p_file.* to NULL
   let p_file.region = bcode
   let p_file.code = btype
   let p_file.category = bcate
   let p_file.repay_date = brdate
   let p_file.interest_rate = birate
   label aaa:
   input by name p_file.* without defaults
   LET bcode = p_file.region
   let btype = p_file.code
   let bcate = p_file.category
   let brdate = p_file.repay_date
   let birate = p_file.interest_rate
   if int_flag != 0 then
        let int_flag = 0
        clear screen
        close form vscr1
        close window w1
        exit while
  end if
   prompt "ACCEPT (Y/y for add,N/n for edit,Q/q for quit) ?"
           FOR CHAR accpt
   if accpt =  "y" or accpt = "Y"  then
   INSERT INTO client VALUES  (p_file.*)
   end if
   if accpt = "N" or accpt = "n" then
   goto aaa
   end if
   continue while
lose window w1
```

```
end while
end main

# ------------------------------------------------------------------
# PROGRAM   : CEDIT.4gl
# PURPOSE   : Edit/deletion of new borrower information
#-------------------------------------------------------------------
DATABASE hbfc
GLOBALS
DEFINE s1_client ARRAY[200] OF
RECORD code LIKE client.code,
       region  LIKE client.region,
       acc_no LIKE client.acc_no,
       category LIKE client.category,
       sanc_amount LIKE client.sanc_amount,
       interest_rate like client.interest_rate,
       name1 LIKE client.name1,
       name2 LIKE client.name2,
       address1 LIKE client.address1,
       address2 LIKE client.address2,
       address3 LIKE client.address3,
       loan_period LIKE client.loan_period
END RECORD
DEFINE s1_clientnum ARRAY[200] OF INTEGER
    DEFINE accpt0   SMALLINT,
           answer,ask   CHAR(1),
           mo CHAR(2),
           no_rec INTEGER,
           i,j,p_i,rec,ssum SMALLINT,
           total DECIMAL(10,2),
           acc SMALLINT,
           sc_curr SMALLINT,
           type,k,kp,ny,while_c CHAR(1),
           select_2,select_1,qr1 CHAR(245),
           quary_1 CHAR(20)
END GLOBALS
MAIN
OPTIONS
        PROMPT line 20,
       NEXT KEY CONTROL-N,
       PREVIOUS KEY CONTROL-P
   DEFER INTERRUPT
     LET while_c = "0"
     LET no_rec = 0
CALL upsam()
LET p_i = 0
END MAIN

FUNCTION upsam()
CLEAR SCREEN
   OPEN WINDOW cw AT 2,2
   WITH 22 ROWS,78 COLUMNS
   ATTRIBUTE (BORDER,COMMENT LINE FIRST,PROMPT LINE LAST)
   OPEN FORM cl1_scr FROM "cscrr"
   DISPLAY FORM cl1_scr
LABEL bbb:
LET k="0"
DISPLAY "Enter type code to edit :" at 1,12 ATTRIBUTE(REVERSE)
   DISPLAY "Press <BREAK> to abort " AT 1,22
LET j = 1
CONSTRUCT BY NAME quary_1 ON code
  LET select_2 = "SELECT code,region,acc_no,category,sanc_amount,
      interest_rate,name1,name2,address1,address2,address3,
      loan_period FROM client WHERE ",quary_1  clipped
  PREPARE sq2 FROM select_2
  DISPLAY "Please wait ...... " AT 1,22 ATTRIBUTE (BLINK,REVERSE)
DECLARE c1_cur SCROLL CURSOR FOR sq2
OPEN c1_cur
display "                          " at 1,22 ATTRIBUTE(REVERSE)
CALL get_no()
   DISPLAY "Record:",ssum AT 1,64
WHILE TRUE
CALL intall()
LET i = 1
IF k = "1" THEN
LET kp = "1"
EXIT WHILE
END IF
LABEL xxx:
IF k = "0" THEN
FETCH c1_cur INTO s1_client[i].code, s1_client[i].region,
   s1_client[i].acc_no, s1_client[i].category,
   s1_client[i].sanc_amount, s1_client[i].interest_rate,
   s1_client[i].name1,s1_client[i].name2, s1_client[i].address1,
   s1_client[i].address2, s1_client[i].address3,
   s1_client[i].loan_period
IF status = notfound THEN
LET p_i = i -1
   LET k ="1"
   GOTO 111
END IF
LET  s1_clientnum[i]=i
LET i= i+1
IF i<201 THEN
GOTO xxx
END IF
LABEL 111:
CALL set_count(i-1)
END IF
IF p_i = 0 THEN
LET no_rec = no_rec + 200
END IF
  IF int_flag != 0 THEN
      LET int_flag = 0
```

```
        GOTO ppp
     END IF
     DISPLAY "Press CONTROLL-N to scroll up:CONTROL-P to scroll down"
                 AT 1,1
     DISPLAY "Press  CONTROLL-B to delete : ESC to update..." AT 2,1
  WHILE TRUE
  LET type = s1_client[1].code
  LET total = get_input()
  LABEL xyz:
  IF total > 0 THEN
     DISPLAY "TOTAL AMOUNT for code ",s1_client[1].code,":-",total
           AT 21,12
     DISPLAY "Press CONTROL-W for next 200 ¦ CONTROL-M for previous
       200 ¦ CONTROL-Z to quit "  at 22,1 ATTRIBUTE (BLINK,REVERSE)
  IF p_i <> 0 THEN
     DISPLAY " There is no 200 more records. Don't press CONTRL-W "
               at 20,9 ATTRIBUTE (BLINK)
  END IF
  ELSE
     DISPLAY "NO RECORD FOUND..........." AT 18,12 ATTRIBUTE(BLINK)
     GOTO ppp
  END IF
  DISPLAY ARRAY s1_client to s_client.*
  ON KEY (ESC)
  LET i = arr_curr()
  LET j = scr_line()
  LET  sc_curr = s1_clientnum[i]
     DISPLAY "       YOU CAN EDIT NOW ..........." AT 1,1
     CALL change_row()
  LET total = get_input()
     DISPLAY "  Press  CONTROLL-N to scroll up: CONTROL-P to scroll
          down" AT 1,1
     DISPLAY "Press  CONTROLL-B to delete : ESC to update..." AT 2,1
     DISPLAY "      TOTAL AMOUNT for code ",s1_client[1].code,":-",
          total AT 21,12
  DISPLAY "Press CONTROL - W for next 200 ¦ CONTROL - M for previous
        200 ¦ CONTROL - Z to quit " at 22,1 ATTRIBUTE(BLINK,REVERSE)
  ON KEY (CONTROL-B)
  LET i = arr_curr()
  LET j = scr_line()
  LET  sc_curr = s1_clientnum[i]
    CALL dele()
  LET total = get_input()
  DISPLAY s1_client[sc_curr].* TO s_client[j].*
     DISPLAY "TOTAL AMOUNT for code ",s1_client[1].code,":-",total
     AT 21,12
  ON KEY(CONTROL-W)
    LET while_c = "1"
    EXIT DISPLAY
  ON KEY(CONTROL-Z)
    LET while_c = "2"
    EXIT DISPLAY
```

```
ON KEY(CONTROL-M)
   LET kp = "0"
IF no_rec >= 400 THEN
  FETCH RELATIVE -400 c1_cur
  LET no_rec = no_rec - 400
ELSE
  FETCH ABSOLUTE 1 c1_cur
END IF
  LET while_c = "1"
  EXIT DISPLAY
END DISPLAY
IF while_c = "1" THEN
     LET while_c = "0"
     EXIT WHILE
END IF
IF while_c = "2" THEN
     EXIT WHILE
END IF
END WHILE
IF while_c = "2" THEN
   LET while_c = "0"
     EXIT WHILE
END IF
END WHILE
LABEL ppp:
   DISPLAY "                                                        " AT 1,1
DISPLAY "Press CONTROL-W for next 200 ¦ CONTROL-M for previous 200¦
             CONTROL-Z to quit "  at 22,1 ATTRIBUTE (BLINK,REVERSE)
CALL intall()
DISPLAY  s1_client[1].* to s_client[1].*
DISPLAY  s1_client[2].* to s_client[2].*
DISPLAY  s1_client[3].* to s_client[3].*
IF kp = "1" THEN
   CLOSE c1_cur
   LET kp = "0"
END IF
   DISPLAY "Record:","     " AT 4,60
PROMPT "Do you want to do it again :" FOR ask
IF ask = "y" OR ask = "Y" THEN
LET p_i = 0
GOTO bbb
END IF
   CLOSE FORM cl1_scr
   CLEAR SCREEN
   CLOSE WINDOW cw
END FUNCTION

FUNCTION dele()
DEFINE cacc_no INTEGER,
       ccate CHAR(2),
       ccode CHAR(1),
       cname1 CHAR(35)
```

```
LET cacc_no = s1_client[i].acc_no
LET ccate = s1_client[i].category
LET ccode = s1_client[i].code
LET cname1  = s1_client[i].name1
PROMPT " Do you want to delete this (Y/N) ? " FOR answer
IF answer="Y" OR answer="y"  THEN
LET select_1 = "DELETE FROM client WHERE code = ? and acc_no = ?
    and category = ? and name1 = ?"
PREPARE sdelete FROM select_1
EXECUTE sdelete  USING ccode,cacc_no,ccate,cname1
LET s1_client[sc_curr].acc_no = NULL
LET s1_client[sc_curr].sanc_amount = NULL
LET s1_client[sc_curr].code = NULL
LET s1_client[sc_curr].category = NULL
LET s1_client[sc_curr].region = NULL
LET s1_client[sc_curr].interest_rate= NULL
LET s1_client[sc_curr].name1 = NULL
LET s1_client[sc_curr].name2 = NULL
LET s1_client[sc_curr].address1 = NULL
LET s1_client[sc_curr].address2 = NULL
LET s1_client[sc_curr].address3 = NULL
LET s1_client[sc_curr].loan_period = NULL
LET ssum = ssum - 1
   DISPLAY "Record:",ssum AT 1,64
END IF

END FUNCTION

FUNCTION change_row()
  DEFINE select_3 CHAR(256)
DEFINE cacc_no INTEGER,
       ccate CHAR(2),
       ccode CHAR(1),
       cname1 CHAR(35)
LET cacc_no = s1_client[i].acc_no
LET ccate = s1_client[i].category
LET ccode = s1_client[i].code
LET cname1  = s1_client[i].name1
 INPUT s1_client[sc_curr].code,s1_client[sc_curr].region,
 s1_client[sc_curr].acc_no,s1_client[sc_curr].category,
 s1_client[sc_curr].sanc_amount,s1_client[sc_curr].interest_rate,
 s1_client[sc_curr].name1,s1_client[sc_curr].name2,
 s1_client[sc_curr].address1,s1_client[sc_curr].address2,
 s1_client[sc_curr].address3,s1_client[sc_curr].loan_period
    without defaults from s_client[j].*
 IF int_flag != 0 THEN
    LET int_flag = 0
    GOTO cc
 END IF
 LET select_3 = " UPDATE client SET address2 = ?,code = ?,region = ?,
 acc_no = ?,sanc_amount = ?,interest_rate = ?,name1 = ?,category = ?,
 name2 = ?,address1 = ? ,address3 = ?,loan_period = ? WHERE code = ?
 and acc_no = ? and category = ? and name1 = ? " CLIPPED
PREPARE supp FROM select_3
EXECUTE supp USING s1_client[sc_curr].address2,
 s1_client[sc_curr].code,s1_client[sc_curr].region,
 s1_client[sc_curr].acc_no,s1_client[sc_curr].sanc_amount,
 s1_client[sc_curr].interest_rate,s1_client[sc_curr].name1,
 s1_client[sc_curr].category,s1_client[sc_curr].name2,
 s1_client[sc_curr].address1,s1_client[sc_curr].address3,
 s1_client[sc_curr].loan_period,ccode,cacc_no,ccate,cname1
MESSAGE "................... ROW UPDATED ......................"
MESSAGE ""
LABEL cc:
END FUNCTION

FUNCTION get_input()
DEFINE ttot CHAR(145),sst decimal(12,2)
LET ttot = "SELECT SUM(sanc_amount) FROM  client WHERE ",quary_1
        clipped
PREPARE snam FROM ttot
DECLARE icur CURSOR FOR snam
OPEN icur
FETCH icur INTO sst
CLOSE icur
RETURN sst
END FUNCTION

FUNCTION get_no()
DEFINE se_1 CHAR(145)
LET se_1 = "SELECT COUNT(*) FROM client WHERE ",quary_1 CLIPPED
PREPARE sew_1 FROM se_1
DECLARE icu CURSOR FOR sew_1
OPEN icu
FETCH icu INTO ssum
CLOSE icu
END FUNCTION

FUNCTION intall()
DEFINE t SMALLINT
FOR t = 1 TO 200
LET s1_client[t].acc_no = 0
LET s1_client[t].sanc_amount = 0.0
LET s1_client[t].code = NULL
LET s1_client[t].category=NULL
LET s1_client[t].region= NULL
LET s1_client[t].interest_rate= 0.0
LET s1_client[t].name1 = NULL
LET s1_client[t].name2 = NULL
LET s1_client[t].address1 = NULL
LET s1_client[t].address2 = NULL
LET s1_client[t].address3 = NULL
LET s1_client[t].loan_period = 0
END FOR
END FUNCTION
```

```
#-----------------------------------------------------------------
# PROGRAM : DISBENT.4GL
# Purpose : Disbursement data entry program.
#-----------------------------------------------------------------
DATABASE hbfc
GLOBALS
        DEFINE f_disb  RECORD
                category         like   disburse.category,
                region           like   disburse.region,
                acc_no           like   disburse.acc_no,
                cheque_no        like   disburse.cheque_no,
                amount           like   disburse.amount,
                install_no       like   disburse.install_no,
                delivery_date    like   disburse.delivery_date,
                sl_date          like   disburse.sl_date,
                issue_date       like   disburse.issue_date,
                purpose          like   disburse.purpose,
                code             like   disburse.code
                END RECORD
        DEFINE    f_var1                 CHAR(1),
                  dcategory              char(2),
                  dregion                char(3),
                  dacc                   char(6),
                  dtype_code             char(1),
                  dsldate                date,
                  ddeliver               date
END GLOBALS
MAIN
        OPTIONS
                PROMPT LINE 23
                DEFER INTERRUPT
                OPEN FORM fs_1 FROM "dscr"
                DISPLAY FORM fs_1
                DISPLAY " Press <DEL> to quit " AT 21,30
    WHILE TRUE
        INITIALIZE f_disb.* TO NULL
        LET f_disb.category = dcategory
        LET f_disb.region = dregion
        LET f_disb.code = dtype_code
        LET f_disb.purpose =  "C"
        let f_disb.sl_date = dsldate
        let f_disb.issue_date = dsldate
        let f_disb.delivery_date = ddeliver
        # -------------------- re-edit -----------------------
        label aaa:
       INPUT f_disb.*  WITHOUT DEFAULTS FROM dis_inp.*
             AFTER FIELD sl_date
             DISPLAY f_disb.sl_date to issue_date
             NEXT FIELD code
       END INPUT
       LET dcategory = f_disb.category
       LET dregion = f_disb.region
```

```
        LET dtype_code = f_disb.code
        let dsldate = f_disb.sl_date
        let dacc = f_disb.acc_no
        let ddeliver  = f_disb.delivery_date
        IF int_flag != 0 THEN
               LET int_flag = 0
               CLOSE FORM fs_1
               CLEAR SCREEN
               EXIT WHILE
        END IF
        LABEL bbb:
        PROMPT "Accept (y/1 for add, n/N  for re-edit, q for QUIT) :"
                  FOR CHAR f_var1
           IF int_flag != 0 THEN
                LET int_flag = 0
                CLOSE FORM fs_1
                CLEAR SCREEN
                EXIT WHILE
           END IF
           IF f_var1 = "y" OR f_var1 = "Y" or f_var1 = "1" THEN
                 INSERT INTO disburse
                    VALUES ( f_disb.category, f_disb.region,
                             f_disb.acc_no,f_disb.issue_date,
                             f_disb.amount,f_disb.cheque_no,
                             f_disb.install_no, f_disb.delivery_date,
                             f_disb.sl_date,f_disb.purpose,
                             f_disb.code )
                 DISPLAY "..... added" AT 23,32
                 SLEEP 1
                 DISPLAY " " AT 23,32
                 CLEAR FORM
           ELSE
                 IF f_var1 = "n" or f_var1 = "N" then
                         GOTO aaa
                 ELSE
                         GOTO bbb
                 END IF
           END IF
           display "Last entered a/c : ", dacc at 24,3
         END WHILE
END MAIN

# ---------------------------------------------------------------
# PROGRAM  : DEDIT.4gl
# PURPOSE  : Edit/deletion of disbursement information
#----------------------------------------------------------------
DATABASE hbfc
GLOBALS
    DEFINE p_disb RECORD LIKE disburse.*
    DEFINE accpt0   CHAR(6),
           answer   CHAR(1)
END GLOBALS
```

```
MAIN
   DEFER INTERRUPT
   OPEN FORM d_scr FROM "dscr"
   DISPLAY FORM d_scr
   DISPLAY "Press <BREAK> to abort " AT 21,22
WHILE TRUE
   INITIALIZE p_disb.* TO NULL
   IF int_flag != 0 THEN
      LET int_flag = 0
      CLOSE FORM d_scr
      CLEAR SCREEN
      EXIT WHILE
   END IF
   CALL dis_inp()
   IF int_flag != 0 THEN
      LET int_flag = 0
      CLOSE FORM d_scr
      CLEAR SCREEN
      EXIT WHILE
   END IF
SELECT *  INTO p_disb.*  FROM disburse
  WHERE  acc_no = accpt0 AND
         cheque_no = accpt1
IF status=NOTFOUND THEN
        DISPLAY " DATA IS NOT AVAILABLE ..." AT 20,21
        SLEEP 3
        DISPLAY "" AT 20,21
        CONTINUE WHILE
END IF
DISPLAY BY NAME p_repay.*
PROMPT "Do you want to add or change any information(y/n) ? "
        FOR CHAR answer
   IF int_flag != 0 THEN
      LET int_flag = 0
      CLOSE FORM d_scr
      CLEAR SCREEN
      EXIT WHILE
   END IF
IF answer="Y" OR answer="y" THEN
   CALL change_row()
END IF
   IF int_flag != 0 THEN
      LET int_flag = 0
      CLOSE FORM d_scr
      CLEAR SCREEN
      EXIT WHILE
   END IF
IF answer="N" OR answer="n"  THEN
   CLOSE FORM pay_scr
   CLEAR SCREEN
      CLOSE WINDOW cw
   EXIT WHILE
```

```
END IF
END WHILE
END MAIN

FUNCTION change_row()
   INPUT BY NAME p_repay.* WITHOUT DEFAULTS
   IF int_flag != 0 THEN
      LET int_flag = 0
      CLOSE FORM pay_scr
      CLEAR SCREEN
      CLOSE WINDOW cw
      goto cc
   END IF
   UPDATE repaynov
   SET repaynov.* = p_repay.*
   WHERE memo_no = accpt0
MESSAGE "............................ ROW UPDATED ........."
SLEEP 6
MESSAGE ""
label cc:
END FUNCTION

# ------------------------------------------------------------------
# PROGRAM   : PADD.4GL
# PURPOSE   : Entry of repayment information to database
#-------------------------------------------------------------------
DATABASE hbfc
GLOBALS
    DEFINE accpt,codever  CHAR(1),bankver smallint,categoryver CHAR(2),
        ver  DATE,perver CHAR(1),pdate DATE ,bee CHAR(1),popno char(1),
        oo,no_a char(1),mo char(2),tabl,tablame char(8),scr char(7),
        select_2 char(145),ssum smallint
    DEFINE p_repay RECORD LIKE repaymay.*
END GLOBALS
MAIN
OPTIONS
PROMPT LINE 22
    DEFER INTERRUPT
LET no_a = "0"
PROMPT "Enter your operational code : " FOR oo
CLEAR SCREEN
MENU "DATA-ENTRY"
     COMMAND "FIRST_TIME" "First time data entry....."
     COMMAND "UPDATE_TIME" "Update time data entry......"
     let upmark = "*"
END MENU
MENU "DATA_ENTRY"
     COMMAND "A" "JANUARY"
      LET tablame = "repayjan"
          LET p_repay.sl_date = "31/01/90"
CALL fop()
CALL pt()
```

```
     COMMAND "B" "FEBRUARY"
      LET tablame = "repayfeb"
          LET p_repay.sl_date = "28/02/90"
CALL fop()
CALL pt()
     COMMAND "C" "MARCH"
      LET tablame = "repaymar"
          LET p_repay.sl_date = "31/03/90"
CALL fop()
CALL pt()
     COMMAND "D" "APRILL"
      LET tablame = "repayapr"
          LET p_repay.sl_date = "30/04/90"
CALL fop()
CALL pt()
     COMMAND "E" "MAY"
      LET tablame = "repaymay"
          LET p_repay.sl_date = "31/05/90"
CALL fop()
CALL pt()
     COMMAND "F" "JUNE"
      LET tablame = "repayjun"
          LET p_repay.sl_date = "30/06/90"
CALL fop()
CALL pt()
     COMMAND "G" "JULY"
      LET tablame = "repay"
          LET p_repay.sl_date = "31/07/89"
CALL fop()
CALL pt()
     COMMAND "H" "AUGUST"
      LET tablame = "repayaug"
          LET p_repay.sl_date = "31/08/89"
CALL fop()
CALL pt()
     COMMAND "I" "SEPTEMBER"
      LET tablame = "repaysep"
      LET p_repay.sl_date = "30/09/89"
CALL fop()
CALL pt()
     COMMAND "J" "OCTOBER"
      LET tablame = "repayoc"
          LET p_repay.sl_date = "31/10/89"
CALL fop()
CALL pt()
     COMMAND "K" "NOVEMBER"
      LET tablame = "repaynov"
          LET p_repay.sl_date = "30/11/89"
CALL fop()
CALL pt()
     COMMAND "L" "DECEMBER"
      LET tablame = "repaydec"
```

```
           LET p_repay.sl_date = "31/12/89"
CALL fop()
CALL pt()
     COMMAND "_EXIT"
      EXIT MENU
END MENU

CLEAR SCREEN
END MAIN

FUNCTION fop()
CLEAR SCREEN
     OPEN WINDOW cw AT 2,4
       WITH FORM "payscrt"
       ATTRIBUTE (BORDER,COMMENT LINE FIRST,PROMPT LINE LAST )
    OPEN FORM pay_scr FROM "payscrt"
    DISPLAY FORM pay_scr
    DISPLAY "   Press <DEL> to abort" AT 20,21
               LET bee = ASCII 7
END FUNCTION

FUNCTION pt()
DEFINE select_1 CHAR(240)
       LET p_repay.purpose = "M"
       LET categoryver = null
       LET pdate = null
       LET select_1 = "INSERT INTO ",tablame,"(code,bank,category,
        acc_no,memo_no,p_date,sl_date,amount,opno,purpose,upcode)
        VALUES (?,?,?,?,?,?,?,?,?,?,?)"
        PREPARE sco FROM select_1
        DECLARE in_cur CURSOR FOR sco
        OPEN in_cur
        DISPLAY " Press CONTROL - W to get bank code and type code"
                AT 2,2
 WHILE TRUE
       INITIALIZE p_repay.memo_no TO NULL
       INITIALIZE p_repay.acc_no TO NULL
       INITIALIZE p_repay.amount TO NULL
       LET p_repay.category=categoryver
       LET p_repay.p_date = pdate
       LET p_repay.opno  = oo
       DISPLAY "Press CONTROL - W to get bank code and type code"
                AT 2,2
       LABEL aaa:
          DISPLAY upmark TO cclient.upcode
          INPUT BY NAME p_repay.* WITHOUT DEFAULTS
             ON KEY (CONTROL-W)
               call get_bank()
               DISPLAY p_repay.code TO cclient.code
               DISPLAY p_repay.bank TO cclient.bank
               NEXT FIELD memo_no
            AFTER FIELD amount
```

```
                IF p_repay.amount IS NULL THEN
                   DISPLAY "Entry must......" AT 22,32
                   DISPLAY bee  AT 22,32
                   DISPLAY "                " AT 22,32
                   NEXT FIELD amount
                END IF
                   IF int_flag != 0 THEN
                       LET int_flag = 0
                       LET no_a = "2"
                       EXIT INPUT
                   END IF
                   END INPUT
                   LET pdate = p_repay.p_date
                   LET categoryver=p_repay.category

    WHILE TRUE
        IF no_a = "2" THEN
        EXIT WHILE
        END IF
        CALL ans()
        IF no_a = "3" OR no_a = "1" OR no_a = "4" OR no_a = "5" THEN
        EXIT WHILE
        END IF
    END WHILE
              IF int_flag != 0 THEN
                    LET int_flag = 0
                    CLOSE FORM pay_scr
                    CLEAR SCREEN
                    CLOSE WINDOW cw
                    EXIT WHILE
              END IF
    IF no_a = "1" THEN
    LET no_a = "0"
    GOTO aaa
    END IF
    IF no_a = "3" OR no_a = "2" THEN
    EXIT WHILE
    END IF
    END WHILE
    CLOSE in_cur
    END FUNCTION

    FUNCTION ans()
      label cat:
      PROMPT " Press 1 or Y to SAVE, N to edit & Q to abort data :"
               FOR CHAR accpt
              IF int_flag != 0 THEN
                    LET int_flag = 0
                    CLOSE FORM pay_scr
                    CLEAR SCREEN
                    CLOSE WINDOW cw
               let no_a = "3"
```

```
                goto aa
          END IF
    IF accpt = "1" or accpt = "Y"or accpt = ascii 100  THEN
     PUT in_cur FROM p_repay.code,p_repay.bank,p_repay.category,
     p_repay.acc_no,p_repay.memo_no,p_repay.p_date,p_repay.sl_date,
     p_repay.amount,p_repay.opno,p_repay.purpose,p_repay.upcode
     FLUSH in_cur
     DISPLAY " One record added "   AT 2,20
     SLEEP 1
     DISPLAY "" AT 2,20
     CLEAR FORM
     let no_a = "5"
     goto aa
    END IF
          IF accpt = "n" or accpt = "N" THEN
               let no_a = "1"
               goto aa
           END IF
          IF accpt = "Q" OR accpt ="q" THEN
           let no_a = "4"
               goto aa
          else
            goto cat
          end if
LABEL aa:
END FUNCTION

FUNCTION get_bank()
PROMPT "Enter bank :" FOR bankver
PROMPT "Enter code :" FOR codever
LET  p_repay.bank = bankver
LET  p_repay.code = codever
END FUNCTION

#-----------------------------------------------------------------
# PROGRAM : LOAN.4GL
# PURPOSE : Program for loan ledger statement
#-----------------------------------------------------------------
DATABASE hbfc
GLOBALS
DEFINE c
RECORD
       c1 LIKE client.code,
       c2 LIKE client.region,
       c3 LIKE client.acc_no,
       c4 LIKE client.category,
       c5 LIKE client.sanc_amount,
       c6 LIKE client.repay_date,
       c7 LIKE client.interest_rate,
       c8 LIKE client.name1,
       c9 LIKE client.name2,
       c10 LIKE client.address1,
       c11 LIKE client.address2,
       c12 LIKE client.address3,
       c13 LIKE client.loan_period,
       c14 LIKE client.ledger_no
END RECORD
DEFINE c2 RECORD
        clc LIKE client2.code,
        clg LIKE  client2.category,
        cla LIKE  client2.acc_no,
        cl1 LIKE client2.p_date,
        cl2 LIKE client2.memo_no,
        cl3 LIKE client2.tot_bal,
        cl4 LIKE client2.pri_ins,
        cl5 LIKE client2.in_ins,
        cl6 LIKE client2.pr_ins_def,
        cl8 LIKE client2.in_ins_def,
        cl9 LIKE client2.lp_in_amt,
        cl0 LIKE client2.lp_in_def,
        cln LIKE client2.n_in,
        cll LIKE client2.lp_ins,
        cli like client2.id_amt,
        clt like client2.lst_trn,
        clp like client2.purpose,
        clb like client2.bank,
        clci like client2.dins
END RECORD
DEFINE t DECIMAL(9,2), rd DECIMAL(2,0)
DEFINE u RECORD
        xx char(1),
        xy char(6),
        xz char(2),
        xx1 char(3),
        xx2 smallint,
        xx3 char(6),
        xx4 date,
        xx5 char(2),
        xx6 decimal(11,4)
END RECORD
DEFINE
       ns char(1),
       mc SMALLINT,
       v_instd decimal(9,2),
       p DECIMAL(6,3),
       int1 DECIMAL(11,4),
       cm CHAR(3),
       rep DECIMAL(9,2),
       pfbptot DECIMAL(11,4),
       botot DECIMAL(9,2),
       dbotot DECIMAL(9,2),
       ct SMALLINT,
       mon1 SMALLINT,
       id_ins decimal(8,3),
       mon SMALLINT,
       rinterest DECIMAL(11,4),
       xtt DATE,
       idp decimal(6,3),
       v_instc decimal(9,2),
       v_princ decimal(9,2)
DEFINE z array[100] of record
       x1  CHAR(3),
       x2  CHAR(3),
       x3  CHAR(3),
       x4  CHAR(6),
       x5  DATE,
       x6  CHAR(2),
       x7  DECIMAL(9,2),
       x8  DECIMAL(9,2),
       x9  DECIMAL(6,3),
       x10 DECIMAL(4,2),
       x11 DECIMAL(9,2),
       x12 DECIMAL(9,2),
       x13 DECIMAL(9,2),
       x14 DECIMAL(6,3),
       x15 DECIMAL(9,2),
       x16 DECIMAL(9,2),
       x17 DECIMAL(9,2)
end record
define
        tdate DATE,
        l SMALLINT,
        nval decimal(9,2),
        vval decimal(9,2),
        d smallint,
        ll smallint,
        xcdate date,
        pa char(2),
        noacc smallint,
        no_acc smallint,
        ddd smallint,
        sdd smallint,
        lttt2 smallint,
        amt_of_lum_d decimal(9,2),
        fsave char(1),
        loption char(1),
        lno  char(6),
        int3 char(1),
        d_is char(1),
        fname char(9),
        princ decimal(9,2),
        prind decimal(9,2),
        instc  decimal(9,2),
        instd  decimal(9,2),
        intcg decimal(9,2),
        n_pay decimal(9,2),
        id_pay decimal(9,2),
        lp_pay decimal(9,2),
        lum_d decimal(9,2),
        idip_d decimal(9,2),
        linstl char(2),
        instl_date date,
        amt_pr decimal(9,2),
        amt_ins decimal(9,2),
        ask char(1),
        icha decimal(9,2),
        in_to_prin decimal(9,2),
        prin_pay decimal(9,2),
        cacc smallint
define xt1 smallint,xt2 smallint
END GLOBALS
MAIN
options
prompt line 20
create temp  table xttt
    (code char(1),
     acc_no integer,
     cate    char(2),
     pur  char(3),
     bank  smallint,
     num char(6),
     dat date,
     ins char(2),
     amt decimal(9,2))
label pppp:
call win()
if loption = "1" then
call cpchoice()
while true
call snumber()
call master1()
prompt " Do you want to do it again?  " for ask
if ask <> "y" or ask <> "Y" then
exit while
end if
end while
goto pppp
end if
if loption = "2" then
call snumber()
prompt  "Enter the number of account to be printed: " for noacc
call cpchoice()
call master1()
#display "Ledger have been printed from",lno,"to",(lno+noacc)
goto pppp
end if
clear screen
end main
```

```
function master1()
   clear screen
   prompt "        Enter the starting year of ladger:" for ddd
   let sdd = ddd
if fsave = "1" then
    START REPORT samar to printer
end if
if fsave = "2" then
    START REPORT samar
end if
if fsave = "3" then
   start report samar to fname
end if
    LET mon = 7
    LET mon1 = 1
    LET l = 1
    INITIALIZE tdate TO NULL
    INITIALIZE xtt TO NULL
if loption = "1" then
 DECLARE q1_cur SCROLL CURSOR FOR
  SELECT  client.code,client.region,client.acc_no,client.category,
     client.sanc_amount,client.repay_date,client.interest_rate,
     client.name1,client.name2,client.address1,client.address2,
     client.address3,client.loan_period,client.ledger_no
     FROM client where client.acc_no = lno
     ORDER BY client.code,client.region,client.acc_no,
     client.category
end if
if loption = "2" then
 DECLARE q2_cur SCROLL CURSOR FOR
  SELECT client.code,client.region,client.acc_no,client.category,
         client.sanc_amount,client.repay_date,client.interest_rate,
         client.name1,client.name2,client.address1,client.address2,
         client.address3,client.loan_period,client.ledger_no
         FROM client
           where client.acc_no > (lno-1)
           ORDER BY client.code,client.region,client.acc_no,
           client.category
end if
    INITIALIZE c.* TO NULL
if loption = "1" then
OPEN q1_cur
    FETCH FIRST q1_cur INTO c.*
    IF status = NOTFOUND THEN
    INITIALIZE c.* to NULL
    close q1_cur
    goto ppp
    END IF
end if
if loption = "2" then
OPEN q2_cur
    FETCH FIRST q2_cur INTO c.*
```

```
        IF status = NOTFOUND THEN
        INITIALIZE c.* to NULL
        close q2_cur
        goto ppp
        END IF
    end if
       let no_acc = 0
    WHILE TRUE
    let prin_pay = 0.0
    let idp = 0.0
    let in_to_prin = 0.0
    let v_instc = 0.0
    let v_princ = 0.0
    let v_instd = 0.0
    let xt1 = 0
    let xt2 = 0
    let lttt2 = 0
    INITIALIZE xtt TO NULL
    let linstl = null
    let instl_date = null
    let amt_pr = 0.0
    let amt_ins = 0.0
    let princ = 0.0
    let prind = 0.0
    let instc = 0.0
    let instd = 0.0
    let icha = 0.0
    let n_pay = 0.0
    let id_pay= 0.0
    let lp_pay= 0.0
    let rep = 0.0
    let rinterest = 0.0
    let idip_d = 0.0
    let lum_d = 0.0
    let xcdate = c.c6
         IF c.c4 = "00" AND c.c6 is null THEN
          CALL ical()
         END IF
       DECLARE x_cur CURSOR FOR
       SELECT client2.code, client2.category, client2.acc_no,
              client2.p_date, client2.memo_no,client2.tot_bal,
              client2.pri_ins, client2.in_ins, client2.pr_ins_def,
              client2.in_ins_def, client2.lp_in_amt,client2.lp_in_def,
              client2.n_in,client2.lp_ins,client2.id_amt FROM client2
              WHERE (client2.code = c.c1 and
                     client2.category = c.c4 and
                     client2.acc_no = c.c3)
       OPEN x_cur
       CALL c2in()
       FETCH x_cur INTO c2.*
       IF status = NOTFOUND THEN
       CALL c2in()
```

```
   close x_cur
   END IF
   if c2.cl4 = 0 or  c2.cl5 = 0 then
   call inscal()
   end if
   close x_cur
   display "Account no:",c.c3," Category no:",c.c4,"Type :",c.cl
                call pcal()
                CALL xin()
                LET z[l].xl = "   "
                LET z[l].x2 = "BF"
                LET z[l].x3 = "   "
                LET z[l].x4 = c2.cl2
                LET z[l].x5 = c2.cl1
                LET z[l].x6 = "  "
                LET z[l].x7 = c2.cl3
                LET z[l].x8 = c2.cl3
                LET z[l].x9 = c2.cl6
                LET z[l].x10 = p
                LET z[l].x11 = c2.cli
                let z[l].x12 = c2.cln
                LET z[l].x13 = c2.cln
                LET z[l].x14 = c2.cl8
                LET z[l].x15 = c2.cl4*c2.cl6
                LET z[l].x16 = c2.cl8*c2.cl5+c2.cli+(c2.cl1*c2.cl0)
                LET z[l].x17 = z[l].x15+z[l].x16
                let l = l + 1
     DECLARE r_cur CURSOR FOR
         SELECT repay.code, repay.acc_no ,repay.category,
                repay.purpose,repay.bank,repay.memo_no,
                repay.p_date, repay.amount FROM repay
          WHERE repay.code = c.cl and
                repay.acc_no = c.c3 and
                repay.category = c.c4 order by repay.p_date
DECLARE d_cur CURSOR FOR SELECT disburse.code,disburse.acc_no,
        disburse.category,disburse.purpose,disburse.cheque_no,
        disburse.delivery_date,disburse.install_no, disburse.amount
        FROM disburse WHERE disburse.code = c.cl  and
                            disburse.acc_no = c.c3 and
                            disburse.category = c.c4
         order by disburse.delivery_date
 DECLARE v_cur CURSOR FOR select voucher.code code, voucher.acc_no
        acc_no,voucher.category cate,voucher.purpose pur,
        voucher.v_date dat,voucher.amount amt FROM voucher
          WHERE voucher.code = c.cl and
             voucher.acc_no = c.c3  and
             voucher.category = c.c4 and
             voucher.region = c.c2 order by voucher.v_date
  foreach r_cur into u.xx,u.xy,u.xz,u.xx1,u.xx2,u.xx3,u.xx4,u.xx6
   let u.xx5 = null
   insert into xttt(code,acc_no,cate,pur,bank,num,dat,ins,amt)
    values (u.xx,u.xy,u.xz,u.xx1,u.xx2,u.xx3,u.xx4,u.xx5,u.xx6)
  end foreach
  foreach d_cur into u.xx,u.xy,u.xz,u.xx1,u.xx3,u.xx4,u.xx5,u.xx6
   let u.xx2 = 0
  insert into xttt(code,acc_no,cate,pur,bank,num,dat,ins,amt)
    values (u.xx,u.xy,u.xz,u.xx1,u.xx2,u.xx3,u.xx4,u.xx5,u.xx6)
  end foreach
    foreach v_cur into u.xx,u.xy,u.xz,u.xx1,u.xx4,u.xx6
    let u.xx3 = null
    let u.xx2 = 0
    let u.xx5 = null
    insert into xttt(code,acc_no,cate,pur,bank,num,dat,ins,amt)
     values (u.xx,u.xy,u.xz,u.xx1,u.xx2,u.xx3,u.xx4,u.xx5,u.xx6)
    end foreach
IF c.c4 = "00" and tdate IS NOT NULL and c.c6 is null THEN
     insert into xttt(code,acc_no,cate,pur,bank,num,dat,ins,amt)
          values(c.c1,c.c3,c.c4,"C",null,null,tdate,"99",0.0)
END IF
call pan()
    WHILE mon1 < 13
      LET int3 = "1"
      let d_is = "0"
      call dcount()
      let ns = "0"
     LET pfbptot = 0.0
declare u_cur scroll cursor for select code,acc_no,cate,pur,
           bank,num,dat,ins,amt
           from xttt where month(dat)=mon
           order by dat
        IF mon = 6   THEN
            let prind = prind + c.c5*(.90)/1000
              CALL xin()
              LET c2.c13 = c2.c13 + c.c5*(.90)/1000
if (year(c.c6)+c.c13)<(ddd+1900) or (year(c.c6)+c.c13=(ddd+1900)
              and month(c.c6)<mon+1) then
              let c2.c16 = c2.c13/c2.c14
end if
              LET z[1].x1 = "    "
              LET z[1].x2 = "I"
              LET z[1].x3 = "    "
              LET z[1].x4 = "       "
              LET z[1].x5 = "01/06/89"
              LET z[1].x6 = "   "
              LET z[1].x7 = .90*c.c5/1000
              LET z[1].x8 = c2.c13
              LET z[1].x9 = c2.c16
              LET z[1].x10 = p
              LET z[1].x11 = 0.0
              LET z[1].x12 = 00.00
              LET z[1].x13 = c2.c1n
              LET z[1].x14 = c2.c18
              LET z[1].x15 = c2.c14*c2.c16
              LET z[1].x16 = c2.c18*c2.c15+c2.c1i+(c2.c11*c2.c10)
          LET z[l].x17 = z[l].x15+z[l].x16
          let l = l + 1
          END IF
  CALL uin()
foreach u_cur into u.*
if status = notfound then
 goto j
end if
case
    when u.xx1="C"
         call xin()
         let c2.clp = u.xx1
         let c2.clb = u.xx2
         let c2.cl2 = u.xx3
         let c2.cl1 = u.xx4
         let c2.clci = u.xx5
         let c2.clt = u.xx6
         call scont()
         let linstl = u.xx5
         let instl_date = u.xx4
         let amt_pr = c2.cl3
         let amt_ins = c2.cln
         call assval(u.xx6,0.0)
    when u.xx1="M"
         CALL xin()
         let c2.clp = u.xx1
         let c2.clb = u.xx2
         let c2.cl2 = u.xx3
         let c2.cl1 = u.xx4
         let c2.clt = u.xx6
         let c2.clci = u.xx5
         CALL rcont()
         let prin_pay = prin_pay+rep
         let id_ins = -idp
         call assval((-rep),(-rinterest))
         let idp = 0.0
         let rinterest = 0.0
         let rep = 0.0
    otherwise
         CALL xin()
         let id_ins = 0.0
         CALL ucont()
         call assval(vval,nval)
end case
call uin()
let id_ins = 0.0
end foreach
label j:
        call xin()
        call incont()
        LET z[l].x1 = cm
        LET z[l].x2 = "IN"
```

```
                LET z[l].x3 = "  "
                LET z[l].x4 =  "       "
                LET z[l].x5 = "       "
                LET z[l].x6 = "  "
                LET z[l].x7 = "         "
                LET z[l].x8 = c2.cl3
                LET z[l].x9 = c2.cl6
                LET z[l].x10 = p
                LET z[l].x11 = id_ins
                LET z[l].x12 = pfbptot
                LET z[l].x13 = c2.cln
                LET z[l].x14 = c2.cl8
                LET z[l].x15 = c2.cl4*c2.cl6
                LET z[l].x16 = c2.cl8*c2.cl5+c2.cli+(c2.cll*c2.cl0)
                LET z[l].x17 = z[l].x15+z[l].x16
                let l = l + 1
    let instd = instd + pfbptot
    let instd = instd + id_ins
          CALL xin()
          CLOSE u_cur
          LET mon1 = mon1 +1
          LET   mon = mon +1
         IF mon > 12 THEN
          LET mon = mon - 12
          LET ddd = ddd + 1
         END IF
    END WHILE
          delete from xttt where code = c.c1 and
                                 acc_no = c.c3 and
                                 cate = c.c4
    let instc = instc + id_pay
    let intcg = instd
    let intcg = intcg -v_instd
    let icha = 0.0
    let n_pay = instc-id_pay-lp_pay+v_instc
    INSERT INTO client3(code,category,acc_no,p_date,memo_no,tot_bal,
           pri_ins,in_ins,pr_ins_def,in_ins_def,lp_in_amt,lp_in_def,
           n_in,lp_ins,id_amt,lst_trn,purpose,bank,dins) VALUES
           (c2.clc,c2.clg,c2.cla,c2.cl1,c2.cl2,c2.cl3,c2.cl4,c2.cl5,
            c2.cl6,c2.cl8,c2.cl9,c2.cl0,c2.cln,c2.cll,c2.cli,c2.clt,
            c2.clp,c2.clb,c2.clci)
       if xcdate is null then
       UPDATE client set(client.repay_date)=(c.c6)
                WHERE client.code = c.c1 and
                    client.region = c.c2 and
                    client.acc_no = c.c3 and
                    client.category = c.c4
       end if
      let amt_of_lum_d = lttt2*c2.cll
      insert into ladg_out(code,category,acc_no,prin_c,prin_d,inst_c,
          inst_d,intcg,n_pay,idip_c,lum_c,lum_d,idip_d,prin_pay,
          in_to_prin) values (c2.clc,c2.clg,c2.cla,princ,prind,instc,
      instd,intcg,n_pay,id_pay,lp_pay,amt_of_lum_d,idip_d,prin_pay,
      in_to_prin)
if linstl is not null and c.c6>mdy(7,1,ddd+1900) then
 insert into incom(code,acc_no,category,linstl,instl_date) values
       (c2.clc,c2.cla,c2.clg,linstl,instl_date)
end if
     LET mon = 7
     LET mon1 = 1
     LET ddd = sdd
    for l1 = 1 to l-1
     output to report samar()
    end for
    let intcg = 0.0
    let l = 1
    INITIALIZE  c.* TO NULL
    let p = 0.0
    if loption = "1" then
    FETCH NEXT q1_cur INTO c.*
    IF status = NOTFOUND THEN
    INITIALIZE  c.* TO NULL
    CLOSE q1_cur
    EXIT WHILE
    END IF
    end if
    if loption = "2" then
    FETCH NEXT q2_cur INTO c.*
    IF status = NOTFOUND THEN
    INITIALIZE  c.* TO NULL
    CLOSE q2_cur
    EXIT WHILE
    END IF
    end if
    INITIALIZE tdate TO NULL
    let no_acc = no_acc + 1
    if no_acc = noacc then
    goto ppp
    end if
    END WHILE
label ppp:
FINISH REPORT samar
end function

REPORT samar()
define cyear1 char(4), cyear2 char(4)
OUTPUT
     LEFT MARGIN 3
     TOP MARGIN 0
     BOTTOM MARGIN 0
     RIGHT MARGIN 163
     PAGE LENGTH 66
FORMAT
     PAGE HEADER
```

```
      print " "
      PRINT COLUMN 140, "PAGE NO -",PAGENO USING "<<<"
ON EVERY ROW
IF  l1 = 1 THEN
let cyear1 = 1900+sdd
let cyear2 = 1901+sdd
PRINT COLUMN 65, "BANGLADESH HOUSE BUILDING FINANCE CORPORATION"
PRINT COLUMN 65, "============================================="
SKIP 1 LINE
PRINT COLUMN 65, "LOAN LEDGER NO = ",COLUMN 80,c.c14 using "&&&,",
      COLUMN 85, "FOR THE YEAR ",cyear1 clipped,"-" clipped,cyear2
      SKIP 1 LINE
PRINT "ACCOUNT NO =",COLUMN 14, c.c3 using "######",COLUMN 30,
 "REGION = ",COLUMN 37, c.c2 USING "&&&",COLUMN 43, "TYPE = ",
 COLUMN 50, c.c1 using "&",COLUMN 53,"CATEGORY = ",COLUMN 65,
 c.c4 USING "&&",COLUMN 68,"PENAL CODE = ",COLUMN 82, pa
PRINT c.c8,COLUMN 29, "AMOUNT OF LOAN",COLUMN 75, "MONTHLY
 INSTALMENT",COLUMN 110, "LUMP INTEREST    TK.",COLUMN 130, c2.c19
            USING "#####.##"
PRINT c.c9,COLUMN 29, "SANCTIONED TK.",COLUMN 45, c.c5 USING
  "#######.##",COLUMN 75, "========================",COLUMN 110,
  "L. INST DEFAULT   ",COLUMN 130, c2.c10 using "#######.##"
PRINT c.c10,COLUMN 29, "PERIOD OF          ",COLUMN 75, "PRINCIPAL
 TK.",COLUMN 92, c2.c14 USING "#######.##",COLUMN 110, "L.INSTLMENT
  TK.",COLUMN 130, c2.c11
  LET rd = year(c.c6)+c.c13
PRINT  c.c11,COLUMN 29, "REPAYMENT - ",COLUMN 45, c.c13 USING "##",
  COLUMN 48, "YEARS",COLUMN 75, "INTEREST  TK.",COLUMN 92, c2.c15
  USING "#######.##",COLUMN 110, "EXPIRY DATE : ",COLUMN 130,
  mdy(month(c.c6),day(c.c6),rd) using "dd/mm/yyyy"
  LET t = c2.c14+c2.c15
PRINT c.c12,COLUMN 29,"INTEREST RATE - ",COLUMN 45,c.c7 USING "##.##",
  COLUMN 52, "PA",COLUMN 75,"TOTAL TK.",COLUMN 92,t USING "#######.##"
PRINT COLUMN 75,"REPAYMENT STARTED-",COLUMN 92,c.c6 using "dd/mm/yyyy"
  SKIP 1 LINE
PRINT "-------------------------------------------------------
-------------------------------------------------------------
------------------------"
PRINT "                    PERTICULARS",COLUMN 35,
 "*              PRINCIPAL SIDE       ",COLUMN 68,
 "*          INTEREST SIDE        ",
 COLUMN 96, "             *           TOTAL DEFAULT"
PRINT "-------------------------------------------------------
-------------------------------------------------------------
------------------------"
PRINT "DATE PUR      MEMO    DL/RP    INS     DR/CR     TOTAL
 INST      PENAL    IDIP    NORMAL      TOTAL      INST       PRIN
 INTEREST    TOTAL"
PRINT "     CODE BK    NO     DATE      NO                 BALANCE
 DEFAULT   RATE                INTEREST    BALANCE    DEFLT"
PRINT "---------------------------------------------------------
-------------------------------------------------------------
----------------------"
```

```
END IF
if l1 = 35 then
PRINT "-----------------------------------------------------------
-----------------------------------------------------------
------------------"
skip 2 lines
print "                    Continue to the next page.........."
 SKIP TO TOP OF PAGE
 SKIP 3 LINE
PRINT "-----------------------------------------------------------
-----------------------------------------------------------
-----------------------"
PRINT "                      PERTICULARS",COLUMN 35,
  "*                PRINCIPAL SIDE      ",COLUMN 68,
  "*            INTEREST SIDE        ",
  COLUMN 96, "              *           TOTAL DEFAULT"
PRINT "-----------------------------------------------------------
-----------------------------------------------------------
-----------------------"
PRINT "DATE PUR      MEMO    DL/RP    INS     DR/CR     TOTAL
 INST      PENAL    IDIP    NORMAL      TOTAL      INST        PRIN
 INTEREST    TOTAL"
PRINT "      CODE BK    NO      DATE        NO              BALANCE
 DEFAULT   RATE               INTEREST    BALANCE     DEFLT"
PRINT "-------------------------------------------------------------
-------------------------------------------------------------
-----------------------"
END IF
PRINT z[l1].x1,COLUMN 6,z[l1].x2,COLUMN 11,z[l1].x3,COLUMN 15,
 z[l1].x4,COLUMN 23,z[l1].x5,COLUMN 33,z[l1].x6,COLUMN 37,
 z[l1].x7,COLUMN 47,z[l1].x8,COLUMN 61,z[l1].x9,COLUMN 71,
 z[l1].x10,COLUMN 74,z[l1].x11,COLUMN 82,z[l1].x12,COLUMN  95,
 z[l1].x13,COLUMN 109,z[l1].x14,COLUMN 116,z[l1].x15,COLUMN 126,
 z[l1].x16,COLUMN 136,z[l1].x17
IF  l1 = (l-1) THEN
   PRINT "-----------------------------------------------------
-----------------------------------------------------------
-----------------"
   PRINT "                   CL.BAL ( PRINCIPAL         ",
        COLUMN 31,c2.cl3,
        COLUMN 42,"    N.INT       ",
        COLUMN 60,c2.cln-c2.cli-c2.cl9,
        COLUMN 70,"       IDIP     ",
        COLUMN 90 ,c2.cli using "#####.##",
        COLUMN 100,"     LUMP        ",
        COLUMN 110,c2.cl9,
        COLUMN 120,"    TOTAL",
        COLUMN 130,(c2.cl3+c2.cln),")".
   PRINT "-----------------------------------------------------
-----------------------------------------------------------
----------------"
PRINT "PRINCIPAL DEBIT :",prind,"     PRINCIPAL CREDIT :",princ,"
      INTEREST DEBIT  :", instd,"      INTEREST CREDIT : ",instc
```

```
PRINT "   "
PRINT "INT.CHARGED :",intcg," PAYMENTS (PRINCIPAL -",princ+v_princ,"
 N.INT. - ",n_pay,"IDIP  - ",id_pay," LUMP - ",lp_pay," TOTAL :- ",
   (princ+instc+v_princ+v_instc),")"
PRINT "    "
PRINT "            * *   UNLESS DISCREPANCY IS NOTIFIED WITHIN
 A MONTH, THE STATEMENT WILL BE TREATED AS CORRECT   * *    "
  SKIP TO TOP OF PAGE
END IF
END REPORT

FUNCTION scont()
call pcal()
LET dbotot = c2.cl3
let c2.cl3 = c2.cl3 + u.xx6
let prind = prind + u.xx6
CALL nint()
IF u.xx5 = "99" THEN
CALL rdate()
END IF
END FUNCTION

FUNCTION ucont()
call pcal()
LET dbotot = c2.cl3
CALL vou()
CALL nint()
END FUNCTION

FUNCTION rcont()
call pcal()
LET botot = c2.cl3
CALL  repl()
let ns= "1"
CALL nint()
END FUNCTION

FUNCTION incont()
CALL mont()
IF pfbptot <> 0.0 THEN
     LET c2.cln = c2.cln + pfbptot+id_ins
END IF
IF pfbptot = 0.0 THEN
     LET pfbptot = c2.cl3*d*c.c7/36500
     LET c2.cln = c2.cln + pfbptot+id_ins
END IF
IF  xt1=0 and xt2 = 0 THEN
     CALL lptim()
END IF
IF xt1 = mon and xt2 = (1900+ddd) THEN
    CALL lpcal()
    goto pp
END IF
IF c2.cll <> 0.0 then
if year(c.c6) = (1900+ddd) and int3 = "1" then
if month(c.c6) > mon THEN
goto rrr
end if
CALL int2()
end if
end if
if year(c.c6) < (1900+ddd) and int3 = "1" then
call int2()
end if
label rrr:
if c.c6 is not null and c2.cl1= 0.0 and year(c.c6)=(1900+ddd) then
LET c2.cl9 = c2.cln
END IF
     LET botot = 0
     LET dbotot = 0
IF (year(c.c6)+c.c13)<(ddd+1900) or (year(c.c6)+c.c13=(ddd+1900) and
   month(c.c6)<mon+1) THEN
LET c2.cl6 = c2.cl3/c2.cl4
LET c2.cl8 = c2.cln/c2.cl5
END IF
label pp:
    call idip()
let c2.cli = c2.cli + id_ins
END FUNCTION

FUNCTION assval(a,b)
DEFINE a DECIMAL(9,2),
       b DECIMAL(9,2)
       LET z[1].x1 = "   "
       LET z[1].x2 = u.xx1
       if u.xx2 = 0 then
        LET z[1].x3 = null
       else
        LET z[1].x3 = u.xx2
        end if
        LET z[1].x4 = u.xx3
        LET z[1].x5 = u.xx4
        LET z[1].x6 = u.xx5
        LET z[1].x7 =  a
        LET z[1].x8 = c2.cl3
        LET z[1].x9 = c2.cl6
        LET z[1].x10 = p
        LET z[1].x11 = id_ins
        LET z[1].x12 = b
        LET z[1].x13 = c2.cln
        LET z[1].x14 = c2.cl8
        LET z[1].x15 = c2.cl4*c2.cl6
        LET z[1].x16 = c2.cl8*c2.cl5+c2.cli+(c2.cl1*c2.cl0)
        LET z[1].x17 = z[1].x15+z[1].x16
        LET l = l + 1
 END FUNCTION
```

```
FUNCTION ical()
SELECT disburse.delivery_date INTO tdate FROM disburse
       where disburse.code = c.cl and
       disburse.acc_no = c.c3 and
       disburse.category = "08" and
       disburse.install_no = "99"
END FUNCTION

FUNCTION vou()
DEFINE vl SMALLINT,
       vr CHAR(1)
if u.xxl = "pi" or u.xxl = "PI" then
let c2.cli = c2.cli + u.xx6
let id_ins = u.xx6
if u.xx6>0.0 then
 let idip_d = idip_d + u.xx6
 let v_instd = v_instd + u.xx6
else
 let id_pay = id_pay - u.xx6
 let v_instc = v_instc + u.xx6
end if
let c2.cln = c2.cln + u.xx6
let nval = 0.0
let vval = 0.0
goto yy
end if
IF u.xxl = "li" or u.xxl = "LI" THEN
 LET c2.cl9 = c2.cl9 + u.xx6
 let c2.cln = c2.cln + u.xx6
 if u.xx6<0.0 then
  let instc = instc -u.xx6
  let v_instc = v_instc + u.xx6
 else
  let instd = instd + u.xx6
  let v_instd = v_instd + u.xx6
 end if
 let c2.cl0 = c2.cl0 + (u.xx6/c2.cll)
 let nval = u.xx6
 let vval = 0.0
 goto yy
end if
LET vl = LENGTH(u.xxl)
LET vr = u.xxl[vl,vl]
LET u.xxl = u.xxl[1,(vl-1)]
IF vr = "p" or vr = "P" THEN
label ipp:
let nval = 0.0
let vval = u.xx6
LET c2.cl3 = c2.cl3+u.xx6
if u.xx6<0.0 then
  let princ = princ - u.xx6
  let v_princ = v_princ + u.xx6
```

```
else
  let prind = prind + u.xx6
end if
if u.xx1 = "mv" or u.xx1 = "MV" then
  let c2.cl6 = c2.cl6 + (u.xx6/c2.cl4)
end if
goto yy
END IF
IF vr = "i" or vr = "I" THEN
if u.xx6 < 0.0 then
 if c2.cl8 = 0.0 then
  let in_to_prin = in_to_prin + u.xx6
  goto ipp
 end if
end if
let nval = u.xx6
let vval = 0.0
LET c2.cln=c2.cln +u.xx6
if u.xx6<0.0 then
 let instc = instc - u.xx6
 let v_instc = v_instc + u.xx6
else
 let instd = instd + u.xx6
 let v_instd = v_instd + u.xx6
end if
if u.xx1="mv" or u.xx1="MV" or u.xx1="cv" or u.xx1="CV" then
  let c2.cl8 = c2.cl8 + (u.xx6/c2.cl5)
end if
END IF
label yy:
END FUNCTION

FUNCTION mont()
CASE
  WHEN mon = 1 LET cm = "JAN"
  WHEN mon = 2 LET cm = "FEB"
  WHEN mon = 3 LET cm = "MAR"
  WHEN mon = 4 LET cm = "APR"
  WHEN mon = 5 LET cm = "MAY"
  WHEN mon = 6 LET cm = "JUN"
  WHEN mon = 7 LET cm = "JUL"
  WHEN mon = 8 LET cm = "AUG"
  WHEN mon = 9 LET cm = "SEP"
  WHEN mon = 10 LET cm = "OCT"
  WHEN mon = 11 LET cm = "NOV"
  OTHERWISE
     LET cm = "DEC"
END CASE

END FUNCTION
```

```
FUNCTION nint()
IF pfbptot = 0.0 THEN
if ns="1" then
IF day(u.xx4) > 7 THEN
LET pfbptot = (botot*(day(u.xx4)-1)+(c2.cl3*(d - day(u.xx4)+1)))
              *c.c7/36500
END IF
if day(u.xx4) < 8 then
LET pfbptot = c2.cl3*d*c.c7/36500
END IF
goto aa
END IF
LET pfbptot = (dbotot*(day(u.xx4)-1)+(c2.cl3*(d - day(u.xx4)+1)))
               *c.c7/36500
goto aa
END IF
IF pfbptot <> 0.0 THEN
if ns="1" then
LET intl = (c2.cl3-botot)*(d-day(u.xx4)+1)*c.c7/36500
LET pfbptot = pfbptot + intl
if intl < 0.0 then
let icha = icha + intl
end if
goto aa
END IF
LET intl = (c2.cl3-dbotot)*(d-day(u.xx4)+1)*c.c7/36500
LET pfbptot = pfbptot + intl
if intl < 0.0 then
let icha = icha + intl
end if
END IF
label aa:
let intl = 0.0
let ns = "0"
let botot = 0
let dbotot = 0
END FUNCTION

FUNCTION rep1()
let rinterest = 0.0
let rep = 0.0
IF c.c6 IS NULL or c.c6 > u.xx4 THEN
LET c2.cl3 = c2.cl3 - u.xx6
let rep = u.xx6
let princ = princ + rep
GOTO a
END IF
IF c2.cl9>0.0 and c2.cll <> 0.0 THEN
 if c2.cl9 <c2.cll then
  LET u.xx6=u.xx6-c2.cl9
  let rinterest = rinterest + c2.cl9
  let lp_pay = lp_pay + c2.cl9
  LET c2.cl9 = 0.0
  let c2.cl0 = 0.0
  goto ss
 end if
 if d_is  = "0" then
  if (c2.cl0+1) > c2.cl9/c2.cll then
   let lttt2= lttt2 + (c2.cl9/c2.cll - c2.cl0)
   let c2.cl0 = c2.cl9/c2.cll
  else
   let c2.cl0 = c2.cl0+1
   let lttt2= lttt2 + 1
  end if
 end if
if (c2.cl0*c2.cll) > u.xx6 then
 LET c2.cl0 = c2.cl0- (u.xx6/c2.cll)
 LET c2.cl9 = c2.cl9-u.xx6
 let rep = 0.0
 let rinterest = rinterest + u.xx6
 let lp_pay = lp_pay + u.xx6
 LET u.xx6 = 0.0
ELSE
 LET u.xx6=u.xx6-(c2.cl0 * c2.cll)
 LET c2.cl9 = c2.cl9 - (c2.cl0 * c2.cll)
 let rinterest = rinterest + (c2.cl0 * c2.cll)
 let lp_pay = lp_pay + (c2.cl0 * c2.cll)
 LET c2.cl0 = 0
 let rep = 0.0
end if
END IF
label ss:
if d_is  = "0" then
let c2.cl8 = c2.cl8 + 1
let c2.cl6 = c2.cl6 + 1
end if
IF u.xx6 = 0.0 THEN
GOTO a
ELSE
IF u.xx6<c2.cli THEN
LET c2.cli= c2.cli-u.xx6
let id_pay = id_pay + u.xx6
let idp = u.xx6
LET u.xx6 = 0
ELSE
LET u.xx6 = u.xx6 - c2.cli
let id_pay = id_pay + c2.cli
let idp = c2.cli
LET c2.cli = 0
END IF
END IF

IF u.xx6=0 THEN
GOTO a
```

```
ELSE
IF (c2.cl5*c2.cl8)>u.xx6 THEN
LET rinterest = rinterest + u.xx6
LET c2.cl8 = c2.cl8 - (u.xx6/c2.cl5)
let c2.cln = c2.cln -rinterest-idp
LET u.xx6 = 0.0
goto a
ELSE
 LET rinterest =  rinterest + (c2.cl5*c2.cl8)
 let c2.cln = c2.cln - rinterest
 LET u.xx6 = u.xx6 - (c2.cl5*c2.cl8)
 LET c2.cl8 = 0.0
END IF
END IF
IF u.xx6=0.0 THEN
 GOTO a
ELSE
 let c2.cl3 = c2.cl3 - u.xx6
 LET rep = u.xx6
 let princ = princ + rep
IF (c2.cl4*c2.cl6)>u.xx6 THEN
 LET c2.cl6 = c2.cl6 - (u.xx6/c2.cl4)
 LET u.xx6 = 0.0
ELSE
 LET c2.cl6 = c2.cl6-(u.xx6/c2.cl4)
END IF
END IF
LABEL a:
let instc = instc + rinterest
let d_is = "1"
let int3 = "0"
END FUNCTION

function pcal()
DEFINE ct SMALLINT
if c2.cl6 < 0 or c2.cl6 = 0 then
let ct = 0
let p = 0
goto abu
end if
if c.c7 =16.0 then
if c2.cl6>0 then
let ct = 1
end if
goto nesar
end if
if c.c7= 13.00 then
IF c2.cl6>0 and c2.cl6<1.5 THEN
LET ct = 1
END IF
IF c2.cl6<2.5 and c2.cl6>1.5 THEN
LET ct = 2
```

```
END IF
IF c2.c16<3.5 and c2.c16>2.5 THEN
LET ct = 3
END IF
IF  c2.c16>3.5 THEN
LET ct = 4
END IF
goto nesar
end if
IF c2.c16>0 and c2.c16<1.5 THEN
 LET ct = 1
END IF
IF c2.c16<2.5 and c2.c16>1.5 THEN
 LET ct = 2
END IF
IF c2.c16<3.5 and c2.c16>2.5 THEN
 LET ct = 3
END IF
IF c2.c16<4.5 and c2.c16>3.5 THEN
 LET ct = 4
END IF
IF c2.c16<5.5 and c2.c16>4.5 THEN
 LET ct = 5
END IF
IF c2.c16>5.5 THEN
 LET ct = 6
END IF
label nesar:
let p = 0.0
if ct <> 0 then
SELECT p_rate INTO p FROM panel WHERE in_rate = c.c7 and pr_def=ct
end if
label abu:
end function

FUNCTION idip()
call pcal()
LET id_ins = (c2.c16*c2.c14*p*d)/36500
let c2.cln = c2.cln + id_ins
let idip_d = idip_d + id_ins
END FUNCTION

FUNCTION rdate()
DEFINE m SMALLINT
if (month(u.xx4) + 7) > 12 then
LET c.c6 = mdy((month(u.xx4)+7-12),1,(year(u.xx4)+1))
else
LET c.c6 = mdy((month(u.xx4)+7),1,year(u.xx4))
end if
END FUNCTION
```

```
FUNCTION dcount()
CASE
WHEN mon = 1  LET d =31
WHEN mon = 2  LET d =28
WHEN mon = 3  LET d =31
WHEN mon = 4  LET d =30
WHEN mon = 5  LET d =31
WHEN mon = 6  LET d =30
WHEN mon = 7  LET d =31
WHEN mon = 8  LET d =31
WHEN mon = 9  LET d =30
WHEN mon = 10  LET d =31
WHEN mon = 11  LET d =30
OTHERWISE LET d =31
END CASE
END FUNCTION

FUNCTION int2()
IF c.c6 IS NOT NULL then
if c2.cl9 <> 0.0 THEN
if (c2.cl0+1) > c2.cl9/c2.cll then
let lttt2= lttt2 + (c2.cl9/c2.cll - c2.cl0)
let c2.cl0 = c2.cl9/c2.cll
else
let c2.cl0 = c2.cl0+1
let lttt2= lttt2 + 1
end if
let lum_d = lum_d + (c2.cl0*c2.cll)
end if
LET c2.cl8 = c2.cl8+1
LET c2.cl6 = c2.cl6 +1
end if
END FUNCTION

FUNCTION uin()
LET u.xx6 = 0.0
INITIALIZE u.xx1 TO NULL
let u.xx2 = 0
INITIALIZE u.xx3 TO NULL
INITIALIZE u.xx4 TO NULL
INITIALIZE u.xx5 TO NULL
END FUNCTION

FUNCTION c2in()
LET c2.cl3 = 0.0
LET c2.cl4 = 0.0
LET c2.cl5 = 0.0
LET c2.cl6 = 0.0
LET id_ins = 0.0
LET c2.cl8 = 0.0
LET c2.cl9 = 0.0
LET c2.cl0 = 0.0
LET c2.cln = 0.0
LET c2.cll = 0.0
let c2.cli = 0.0
INITIALIZE c2.cl1 TO NULL
INITIALIZE c2.clc TO NULL
INITIALIZE c2.clg TO NULL
INITIALIZE c2.cla TO NULL
INITIALIZE c2.cl2 TO NULL
END FUNCTION

function lptim()
if month(c.c6) < 3 then
let xt1 = (month(c.c6) + 12 - 2)
let xt2 = year(c.c6)
let xt2 = xt2 -1
else
let xt1 = month(c.c6) - 2
let xt2 = year(c.c6)
end if
end function

function lpcal()
let c2.cl9 = c2.cln
let c2.cll = c2.cl9/12
end function

FUNCTION xin()

                INITIALIZE z[1].x1 TO NULL
                INITIALIZE z[1].x2 TO NULL
                INITIALIZE z[1].x3 TO NULL
                INITIALIZE z[1].x4 TO NULL
                INITIALIZE z[1].x5 TO NULL
                INITIALIZE z[1].x6 TO NULL
                LET z[1].x7 =  0.0
                LET z[1].x8 = 0.0
                LET z[1].x9 = 0.0
                LET z[1].x10 = 0.0
                LET z[1].x11 = 0.0
                LET z[1].x12 = 0.0
                LET z[1].x13 = 0.0
                LET z[1].x14 = 0.0
                LET z[1].x15 = 0.0
                LET z[1].x16 = 0.0
                LET z[1].x17 = 0.0
END FUNCTION

function  inscal()
let c2.cl4 = c.c5*((1/(c.c13*12))+.0001)
let c2.cl5 = c.c5*c.c7*((c.c13*12)+1)/(c.c13*28800)
end function
```

```
function pan()
   case
   when c.c7 = 5.0 let pa= "01"
   when c.c7 = 6.0 let pa= "00"
   when c.c7 = 6.25 let pa= "00"
   when c.c7 = 7.0 let pa= "00"
   when c.c7 = 7.25 let pa= "00"
   when c.c7 = 7.5 let pa= "02"
   when c.c7 = 8.0 let pa= "00"
   when c.c7 = 9.5 let pa= "03"
   when c.c7 = 10.5 let pa= "04"
   when c.c7 = 11.0 let pa= "00"
   when c.c7 = 13.0 let pa= "05"
   otherwise
     let pa= "06"
   end case
 end function

function win()
display "==================================================" at 5,2
display "     LEDGER PRINTING " AT 6,15
display "              MENU                " at 7,15
display "1. Selected number of account " at 9,15
display "2. With starting number           " at 11,15
display "3. Exit                           " at 13,15
display "------------------------------------------------"   at 16,2
prompt  "  Enter your choice    : " for loption
end function

function snumber()
define rl smallint,
       cl smallint,
       cw char(2),
       xn char(3)
clear screen
display "==================================================" at 5,2
display "     LEDGER PRINTING " AT 7,15
display "--------------------------------------------------"   at 22,2
prompt  "    Enter an account number: " for lno
end function

function cpchoice()
display "================================================" at 5,2
display "     LEDGER PRINTING " AT 6,15
display "              MENU                " at 7,15
display "1. Print on paper                 " at 9,15
display "2. Print on consol                " at 11,15
display "3. Save to a file                 " at 13,15
display "----------------------------------------------"   at 16,2
prompt  "  Enter your choice    : " for fsave
if fsave = "3" then
prompt "   Enter your file name : " for fname
end if
end function
```

REFERENCES

[1] CHEN,P, "The entity - relationship model - Toward a unified view of Data", ACM Trans. Database System 1, 1976.

[2] CHEN,P, "A Preliminary Framework for Entity - Relationship Models", ER Institute, 1981.

[3] JOSEPH,A. VASTA, "UNDERSTANDING DATABASE MANAGEMENT SYSTEMS, Wadsworth Publishing Company, Belmont California, 1989.

[4] DATE,C, "An Introduction to Database Systems",vol.1, 4th ed. Addison - Wesley, 1985.

[5] TOBY,J, P.FRY, D.YANG, "A Logical Design Methodology for Relational Databases Using Extended Entity-Relationship Model", Computing Surveys, Vol.18, Michigan, 1986.

[6] CODD, "A Relational Model for large shared data banks.", commun. ACM 13, 6 (June) 377 - 387, 1970.

[7] KORTH,SILBERSCHATZ, "Database System Concept", McGRAW-HILL INTERNATIONAL EDITIONS, Computer Science Series,Stanford University, 1988.

[8] Rules and Regulations, BHBFC, 1973.

[9] ROGER,S.PRESSMAN, "SOFTWARE ENGINEERING", 2nd ed.,McGRAW-HILL INTERNATIONAL EDITIONS, Computer Science Series, 1987.

[10] CHEN,P, " English Sentance Structure and Entity-Relationship Diagrams". Infor sciences. 29,127-149.,Elsevier Science Publishing Co., Inc., 52 Vonderbilt Ave., New York, NY 10017,1983.

[11] RICHARD FINKELSTEIN AND FABIAN PASCAL, "SQL DATABASE MANAGEMENT SYSTEMS", January 1988, BYTE, 111-118.

[12] JEFFREY D. ULLMAN,1988, "PRINCIPLES OF DATABASE AND KNOWLEDGE BASE SYSTEMS", Vol.I, Computer Science Press.