NEAR EAST UNIVERSITY

FUCLTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

DATABASE SYSTEM FOR AN ESTATE AGENCY USING ORACLE

GRADUATION PROJECT

COM-400

Student: **Bashar Abd Alnabi**

Supervisor: **Asst. Prof. Dr Firudin Muradov**

Nicosia-2005

Table of Contents

ACKNOWLEDGEMENTS

Foremost I would like to pay my special thanks to my parents, who helped me on every phase of my life. They boosted me up about my studies as well as my life. I am very much thankful and grateful to my father whose sacrifice to make this day come true and my mother whose prayers and love for me has encouraged me so make this day come true. I will never ever forget my uncles and my grand father, for his encouraged and love. It is only because of them that today I am capable of completing my degree.

Secondly I would like to present my special appreciation to my supervisor Assist. Prof. Dr. Firudin Muradov , without him it is impossible for me to complete the project. His trust in my work and me and his priceless awareness for the project has made me do my work with full interest.

Further I am thankful to Near East University academic staff and my friends who helped me and encouraged me completion of my project.

"Thanks"

Abstract

Database system is a computerized record keeping system. The database can be regarded as a kind of electronic filling cabinet.

Oracle is a smart software program that can join the database and facilitate the requirement job for any user to use the designed program. The oracle programming language has many characteristics, like the security, safety; understand ability, Reliability, Flexibility.

In the present project, we first analyzed the requirements for the activities of the state agency sell company, to develop software. After that we created required tables oracle 8i and then developer 6i to automate some of the activities.

The project fulfils the request of a database system for Estate Agency. In this system user can add, view, delete, and change the database as of his /her requirements.

INTRODUCTION

A Database management system is collection of programs that enables users to create and maintain a database.

A RDBMS is computerized records keeping system that stores maintains and provides access to the information. A Database system consists of four major components that are Data, Hardware, Software, and Users. DBMS are used by any reasonably self contained commercial, scientific, technical or other organization for a single individual to a large company and a DBMS is used for many reasons. The objective of this project was to design software for an Estate Agency.

This project consists of four chapters and conclusion.

The Chapter One: presents an introduction to the database management system, data models and advantages of DBMS.

Chapter Tow: describes useful features of relational database management system (RDBMS), such as security and authorization.

Chapter Three: is a description for the entities and relationships, schema and ER diagram for state agency.

Chapter Four: presents an implementation of state agency software and forms designed with programming languages which used

Chapter 1

INTRODUCTION TO DBMS

1.1 Database

In a typical file-processing environment, each user area, such as payroll, personnel, and the speakers' bureau, has it own collection of files and programs that access files. Since there is usually overlap of data between user areas, there is redundancy in the system. The address of a faculty member can occur in many places, i.e. while this is certainly wasteful trying to produce reports or respond to queries that span user areas can be extremely difficult. These problems lead to the idea of a pool of data, or database, rather than separate collections of individual files.

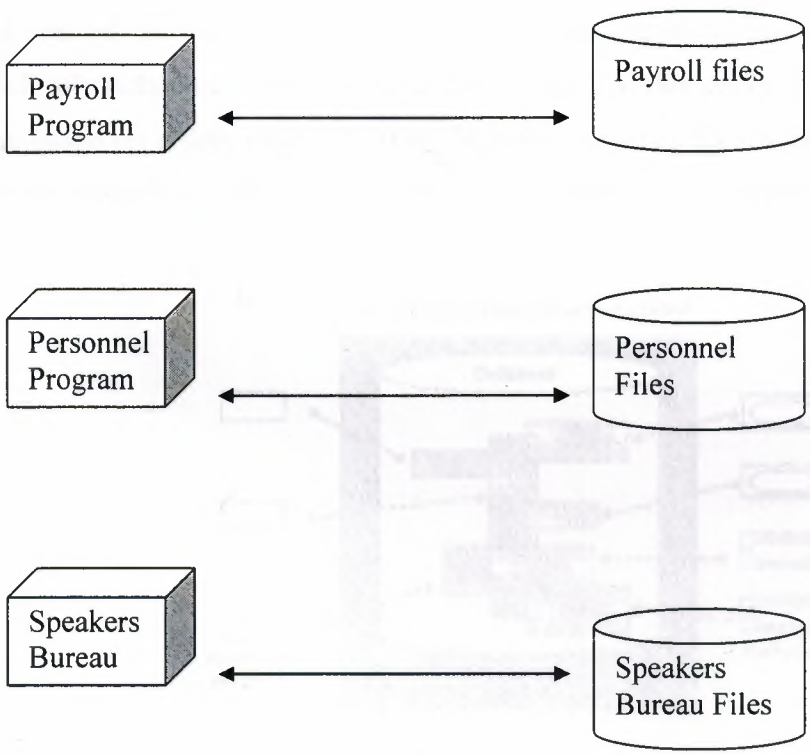


Figure 1.1 Databases

1.2 **What Makes Up A DBMS?**

A DBMS is a computerized record-keeping system that stores, maintains and provides access to information. A database system involves four major components, which are as follows.

1. DATA
2. HARDWARE
3. SOFTWARE
4. USERS

DBMS are used by any reasonably self-contained commercial, scientific, technical or other organization from a single individual to a large company and a DBMS may be used for many reasons. Data itself consists of individual entities, in addition to which there will be relationships between entity types linking them together. Given an enterprise with a nebulously defined collection of data, the mapping of this collection onto the real DBMS is done based on a data model. Various architectures exits for databases and various models have been purposed including the relational, network, and hierarchic models.

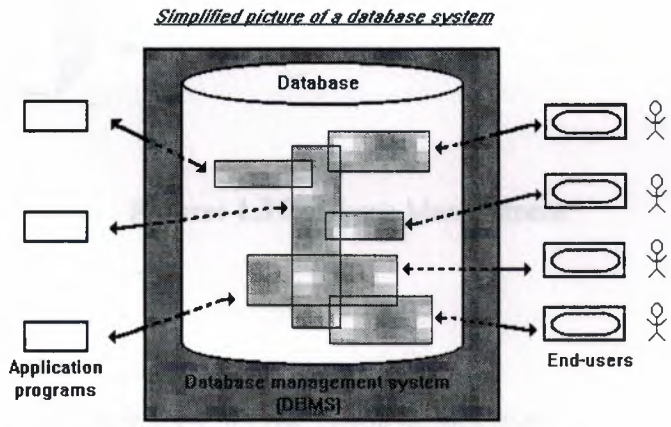


Figure: 1.2 [web_pagecs111notesDBMS.htm]

1.3 **Database Management System**

Fortunately, software package called database management system can do the job of manipulating actual database for us. A database management system (or DBMS), at its simplest, is a software product through which users interact with a database. The actual manipulating of the underlying database structures is handled by the DBMS.

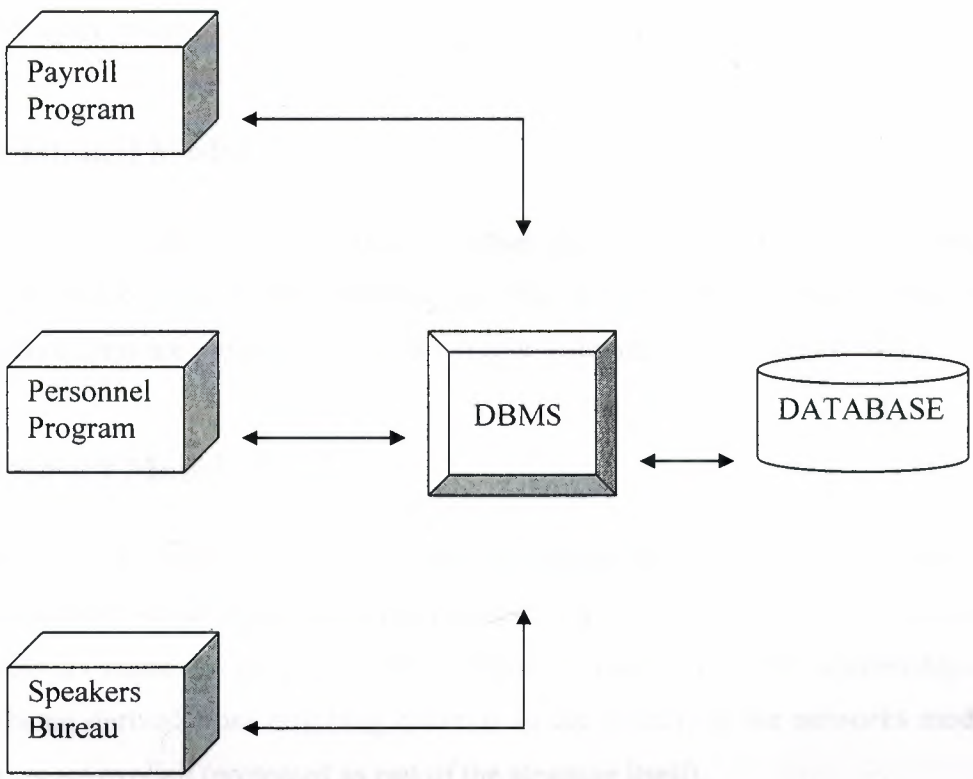


Figure: 1.3 Database Management

1.4 Data Models

The model of data that they follow characterizes database management systems. A Data model has two components-structure and operations. The structure refers to the way the system structures data or, at least, the way the users of the DBMS feel that the data is structured. The operations re the facilities given to the users of the DBMS to manipulate

data within the database. What is crucial is the way things feel to the user, it does not matter how the designers of the DBMS choose o implement these facilities behind the scenes.

There are three models, or categories, for the vast majority of DBMS:

- Relational model
- Network model
- Hierarchical model.

1.4.1 Relational Model

The user as begin just a collection of tables perceives a relational model database. Formally, these tables are called **relations,** and this is where the relational model gets its name. Relationships are implanted through common columns in two or more tables.

1.4.2 Network Model

The user as a collection of record types an relationships between these record types perceive a network model database such a structure is a network, and it is form this that the model takes its name. In contrast to the relational model, in which relationships were implicit (being derived from matching columns in the tables), in the networks model the relationships are explicit (presented as part of the structure itself).

1.4.3 Hierarchical Model

A user as a collection of hierarchies (or trees) perceives a hierarchies model database. A hierarchy is really a network with am added restriction; no box can have more than one arrow entering the box. (It doesn't matter how many arrows leave a box). A hierarchy is thus a more restrictive structure than a network.

1.5 Advantages of DBMS

The main advantages of using a DBMS is that the formalism of the model of data underlying the DBMS is imposed upon the data set to yield a logical and structured organization of the data. Given a fuzzy, real-world data set, when a model's formalism is imposed in that data set the result is easier to manage, define an manipulate. Different models of data lead to different organizations. In general the relational model is the most popular because that model is the most abstract and easiest to apply to data while still begin powerful.

Therefore, using a DBMS we have the following advantages.

- Clear picture of logical organization of data set.
- Centralization for multi-users.
- Data independence.

1.6 The 3 Levels Architecture

The three levels architecture is architecture for a DBMS to provide a framework for describing database concepts and structures. Not all DBMS fit neatly into this architecture, but most do. The model has been proposed by ANSI/SPARC and has three levels. Mappings exist between the three levels and it is the responsibility of the DBA to ensure these mappings are correct.

- External level (individual users view)
- Conceptual level (community user view)
- Internal level (storage)

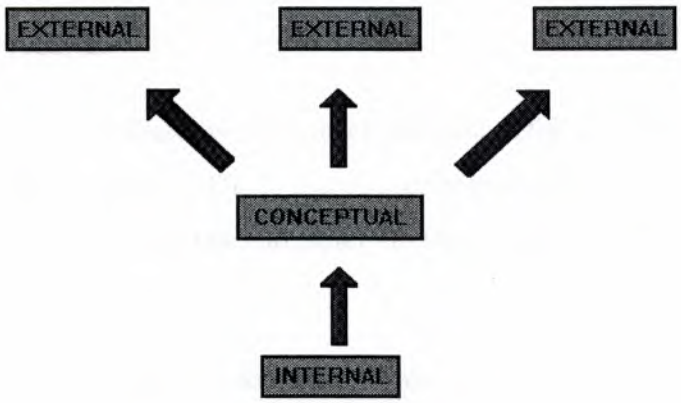


Figure: 1.4. Three Level Architecture

1.6.1 External Level

The external level of the three level architecture is the individual user level. At this level each user has a language at their disposal of which they will use a "data sub language" i.e. a subset of the total language that is concerned specifically with database operations and objects. For the application programmer, the language will be a conventional language e.g. COBOL with embedded SQL, or a specific one e.g. Database. For the end user, it will normally be a query language like SQL or a special purpose language. In principle, any given data sub language consists of a DDL (to declare data objects) and a DML (data manipulation language) to manipulate these objects

An individual user's view is an external view, which is thus the content of the database as seen by that particular user. There will thus be multiple occurrences of multiple types of external records. The external view is defined by an external schema, which in turn is defined by the DDL part of the user's data sub language

1.6.2 Conceptual Level

The conceptual level of the three levels architecture is essentially a representation of the entire information content of the database in a form abstracted from physical storage. It may also be quit different or similar to external views held by a particular user. It is data as

it really is. Rather than as users are forced to see it- it is multiple occurrences of multiple types of conceptual records.

The conceptual schema is defined by the conceptual data definition language (DDL). There is no reference in the conceptual DDL to stored record concepts, sequences, indexing, hash addressing, pointers etc. the references are solely to the definition of information content, in order to preserve data independence.

Conceptual schemas will also include security and integrity constraints as well as data definitions. Normally the conventional schema is little more than a union of all individual external schemas, plus some security/integrity checks.

1.6.3 Internal Level

The internal level of the three levels architecture is a low level representation of the entire database; it consists of multiple types of internal record. It does not deal with block/pages or device-dependant concepts like cylinders and tracks. The internal system defines types of stored records and indexes, how fields are represented, various storage structures used, whether they use pointer chains or hashing, what sequence they are in, and so on. The internal schema is written using yet another data definition language, the internal DDL.

Programs accessing this level directly (i.e. utility programs) are dangerous since they have by-passed the security and integrity checks which the DBMS program normally takes responsibility for.

1.7 Properties of DBMS Data

DBMS are available on any machine, from small micros to large mainframes, and can be single or multi-user obviously, there will be special problem in multi-user environments in order to make other users invisible, but these problems are internal to DBMS.

Data may be shared over many databases, giving a distributed DBMS, though quite often it is centralized and stored in just one database on one machine. In general, the data in the database, at least in a large system, will be both integrated and shared.

1.8 Who Uses A DBMS

There are three broad classes of users who use a DBMS

- Application programmers
- End users
- Database administrator

1.9 Hardware for A DBMS

Conventional DBMS hardware consists of secondary storage devices, usually hard disks, on which the database physically resides, together with the associated I/O devices, device controllers, I/O channels and so forth. Databases run on a range of machines, from microcomputers to large mainframes.

Other hardware issues for a DBMS includes database machines, which is hardware designed specifically to support a database system.

1.10 Database Security

The DBA can set up the DBMS such that only certain users or certain application programs are allowed perform certain operations to the dataset e.g. only admissions are allowed create records for students, only library are allowed to create records for books etc. Different checks can be established for each type of access to each type of information in the database. Different users should have different access rights to different objects.

SQL provides two methods for implementing security restrictions. These are:

- Views - can be provided to hide sensitive data.
- GRANT/REVOKE - grant or remove access privileges to specific users for specific tables.
- There is, however, a major drawback to SQL security

1.11 How Data is Stored

A data model is defined as a set of guidelines for representing the logical organization of data in the database; a pattern according to which data and relationships can be organized; an underlying mathematical formulation for building logical data organizations.

A data model consists of:

- A named logical unit (record type, data item)
- Relationships among logical units

A data item is the smallest logical unit of data, an instance of which is known as a data item value.

A record type is a collection of data items, and a record is hence defined as an instance of a record type.

Note: A data model does not specify the data, data implementations or physical organization only the way it can be logically organized.

1.12 Definition of Entity

An entity is any distinguishable real world object that is to be represented in the database; each entity will have attributes or properties e.g.

The entity lecture has the properties place and time. A set of similar entities is known as an entity type.

1.13 Relationship

it is defined as an association among entities or the entities in a database are likely to interact with other entities . The inter connection between the entity sets are called relationship.

Chapter 2

RELATIONAL DATABASE MANAGEMENT SYSTEM

2.1 Introduction to Relational Database Management System

In recent years, database management systems (DBMS) have established themselves as the primary means of data storage for information system ranging from large commercial transaction processing applications to PC-based desktop applications. At the heart of most of today's information systems is a relational database management system (RDBMS).

RDBMS have been the workhorse from data management operations fro over a decade and continue to evolve and mature, providing sophisticated storage, retrieved, and distribution functions to enterprise-wide data processing and information management system. Compared to the file systems, relational database management system provides organization data into meaningful information systems. The evolution of high-powered database engines has fostered the development of advanced "enabling" technologies including client/server, data warehousing, and online analytical processing all of which comprise the core of today's state-of-the-art information management systems.

Examine the components of the term relational database management system. First, a database is an integrated collection of related data. Given a specific data item, the structure of a database facilitates the access to data related to it, such as a student and all of his registered courses or an employee and his dependents. Next, a relational database is a type of database based in the relational model; non-relational database commonly use a hierarchical, network, or object-oriented model as their basis. Finally, a relational database management system is the software that manages a relational database. These systems come in several varieties, ranging form single-user desktop systems to full featured, global, enterprise-wide systems.

2.2 The Relational Database Models

Most of the database management systems used by commercial applications today are based on one of three basic models:

1. Hierarchical Model; Network Model OR
2. Relational Model

2.2.1 Hierarchical Model

The first commercially available database management systems were of the CODEASYL type, and many of them are still in use with mainframe-based, COBOL applications. Both network and hierarchical database are quite complex in that they rely on the use of permanent internal pointers to relate records to each other. i.e. in an accounts payable application, a vendor record might contain a physical pointer in its record structure that points to purchase order records. Each purchase order record in turn contains pointers to purchase order line item records.

The process of inserting, updating and deleting records using these types of database required synchronization of the pointers, a task that must be performed by the application. As you might imagine, this pointer maintenance required a significant amount of application code (usually written in COBOL) that at times could be quite cumbersome.

2.2.2 Relational Model

Relational database rely on the actual attribute values as opposed to internal pointers to link records. Instead of using an internal pointer from the vendor record to purchase order records, you would link the purchase order record to the vendor record using a common attributer form each record, such as the vendor identification number.

Although the concepts of academic theory underlying the relational model are somewhat complex, you should be familiar with are some basic concepts and terminology.

Essentially, there are three basic components of the relational model:

- Relation Data Structure
- Constraints that Govern the Organization of the Structure
- Operations that are Perform on the Data Structure.

2.3 RDBMS Components

Two important pieces of RDBMS architecture are the Kernel, which is the software, and the data dictionary, which consists of the system-level data structures used by the kernel to manage the database.

2.4 Relational Data Base Management Issues

- Integrity
- Security
- Recovery
- Concurrency

2.4.1 Security

The advantage of having shared access to data is in fact a disadvantage also

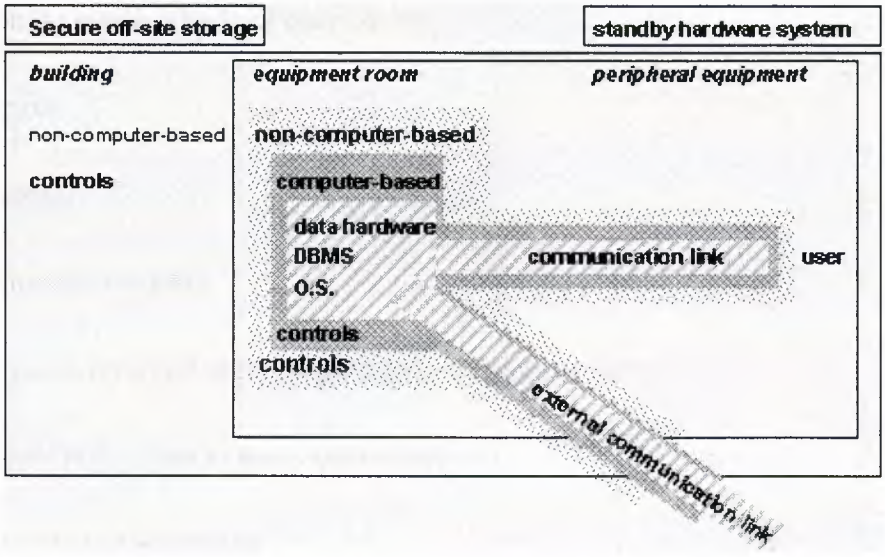


Figure: 2.1 Security [Aptech]

- Consequences: loss of competitiveness, legal action from individual
- Restrictions
    - Unauthorized users seeing data
    - Corruption due to deliberate incorrect updated
    - Corruption due to accidental incorrect updated
- Reading ability allocated to those who have a right to know
- Writing capabilities restricted for the casual user - who may accidentally corrupt data due to lack of understanding
- Authorization is restricted to the chosen few to avoid deliberate corruption

2.5 Countermeasures (computer based)

2.5.1 Authorization

- Determine user is who they claim to be
- Privileges
- Passwords
- Low storage overhead
- Many passwords and users forget them - write them down!!
- User time high - type in many passwords
- Held in file and encrypted.

2.6 Countermeasures (cont.)

- Initial password entry to system
- User name checked against control list
- The access control list has very limited access, superuser
- If many users and applications and data then list can be large

2.7 READ, WRITE, and MODIFY access controls

- Restrictions at many levels
- Database Level: 'Adds a new DB'
- Record Level: 'delete a new record'
- Data Level: 'delete an attribute'
- Remember there are overheads with security mechanisms

2.8 Countermeasures (cont.)

- Views
- Subschema
- Dynamic result of one or more relational operations operating on base relations to produce another relations
- Virtual relation - doesn't exist but is produce at runtime
- Back-up
- Periodic copy of database and log file (programs) onto offline storage
- Stored in secure location

2.9 Countermeasures (cont.)

- Keeping log file of all changes made to database to enable recovery in the event of failure
- Check pointing
- Synchronization point where all buffers in the DBMS is force-written to secondary storage
- Integrity (see later)
    - Encryption
    - Data encoding by special algorithm that render data unreadable without Decryption key
    - Degradation in performance
    - Good for communication

2.10 Countermeasures (cont.) Associated procedures:

- Specify procedures for authorization and backup/recovery
- Audit: auditor observe manual and computer procedures
- Installation/upgrade procedures
- Contingency plan
- Escrow agreement.

2.11 Non-Computer Counter Measures:

- Establishment of security policy and contingency plan
- Personnel controls
- Secure positing of equipment, data and software
- Escrow agreements (3rd party holds source code)
- Maintenance agreements
- Physical access controls
- Building controls
- Emergency arrangements.

2.12 Privacy in Oracle:

- User gets a password and user name
- Privileges:

  Connect: users can read and update tables (can't create)

Resource: create tables, grant privileges and control auditing

DBA: any table in complete DB

- User owns tables they create

  They grant other users privileges:

  Select: retrieval

  Insert: new rows

  Update: existing rows

  Delete: rows

  Alter: column def.

  Index: on tables

- Owner can offer GRANT to other users as well

  This can be revoked

- Users can get audits of:
  - List of successful/unsuccessful attempts to access tables
  - Selective audit e.g. update only
  - Control level of detail reported
- DBA has this and logon, logoff oracle, grants/revolts privilege
- Audit is stored in the Data Dictionary.

2.13 Integrity:

- Introduction
- Basic concepts
- Integrity constraints
- Relation constraints
- Domain constraints
- Referential integrity
- Explicit constraints
- Static and Dynamic Constraint

Chapter 3

DESIGN OF THE TABLES AND RELATIONSHIPS FOR ESTATE AGENCY

3.1 Introduction

This chapter presents the database and the relations between these entities. Also this chapter described each entity with data types for each attribute and the relation are identified by ER diagram.

3.2 Identification of Entities and Relationships

The Entity is: things or object in the real world that is distinguishable from object to other objects.

The Relation: is association among several entities.

The identification of entities and relationships that will help to understand the relations in this application design as following.

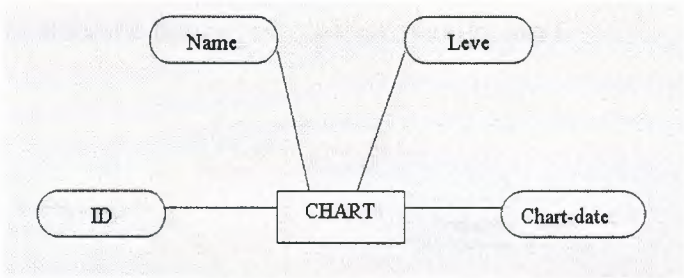
The entities and relations in this application are as a following:

- Chart.
- Gl_master.
- Gl_details.
- Pay_master.
- Pay_details.
- Res_master.
- Res_details.

3.2.1 Chart

Chart entity contains the following attributes:

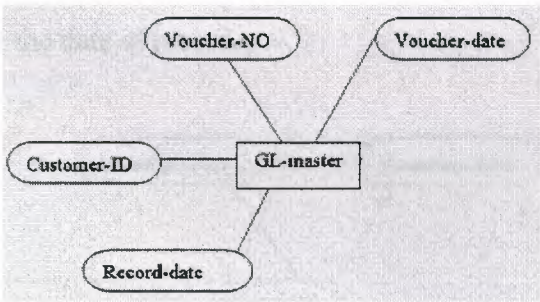
- ID to store the IDs of (banks, debtors, properties) which we have (primary key).
- Name to store the names of (banks, debtors, properties) which we have.
- Leve (type) to store the types of (banks, debtors, properties) which we have.
- Chart date to store the date when we insert the record.



3.2.2 Gl _master

Gl _master entity contains the following attributes:

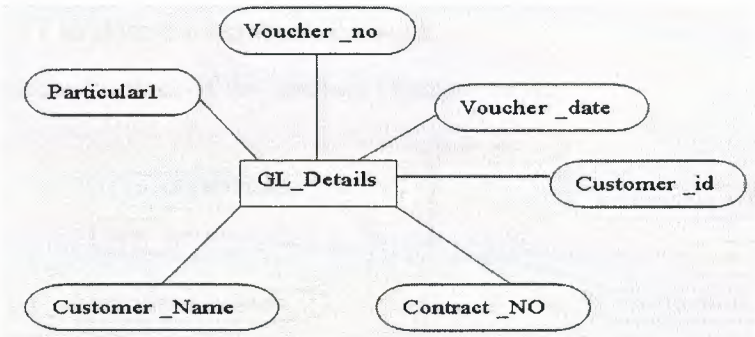
- Voucher _no to store the voucher number (primary key).
- Voucher _date to store the date of voucher and it is always taking the system date.
- Customer _id to store the id of customer.
- Record _date to store the date of record.



3.2.3 Gl _details

Gl _details entity contains the following attributes:

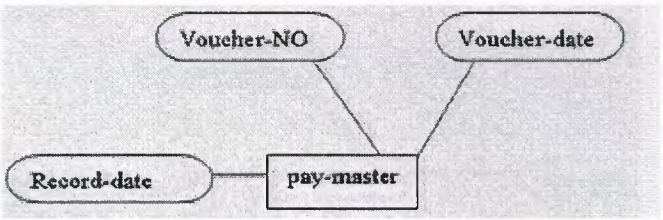
- Voucher _no to store the voucher number (primary key) (foreign key).
- Voucher _date to store the date of voucher and it is always taking the system date.
- Customer _id to store the id of customer.
- Customer _name to store the name of the customer.
- Particular1 to store any note you want about the receipts of the customer.
- Contract _no to store the number of contract (foreign key).



3.2.4 Pay _master

Pay _master entity contains the following attributes:

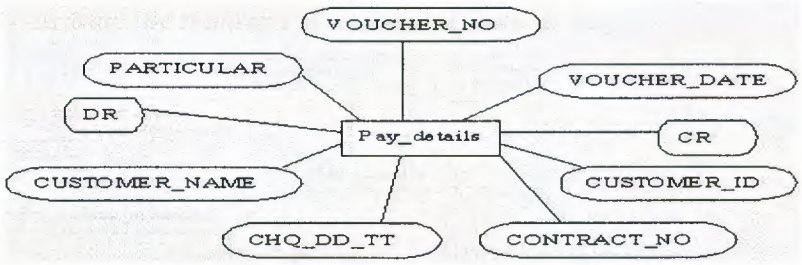
- Voucher _no to store the voucher number (primary key).
- Voucher _date to store the date of voucher and it is always taking the system date.
- Record _date to store the date of record.



3.2.5 Pay _details

Pay _details entity contains the following attributes:

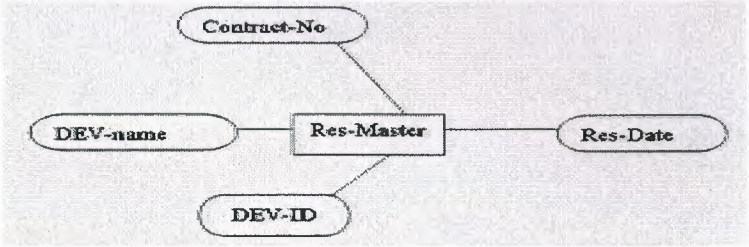
- Voucher _no to store the voucher number (primary key) (foreign key).
- Voucher _date to store the date of voucher.
- Customer _id to store the id of customer (foreign key).
- Customer _name to store the name of the customer.
- Particular to store any note you want about the receipts of the customer.
- DR (Debit).
- CR (Credit).
- CHQ_DD_TT to store the number of cheque.
- Contract _no the number of the contract (foreign key).



3.2.6 Res _master

Res _master entity contains the following attributes:

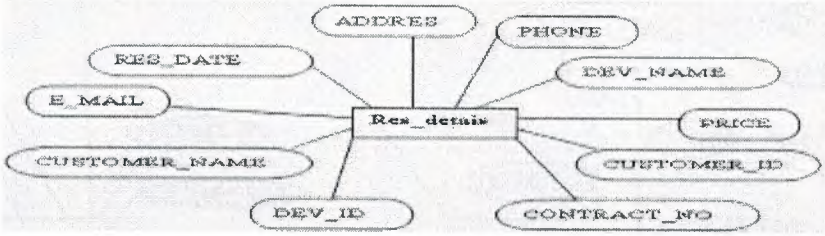
- Contract _no the number of the contract (primary key).
- Res _date the reservation date.
- DEV _name property name.
- DEV _ID property Id.



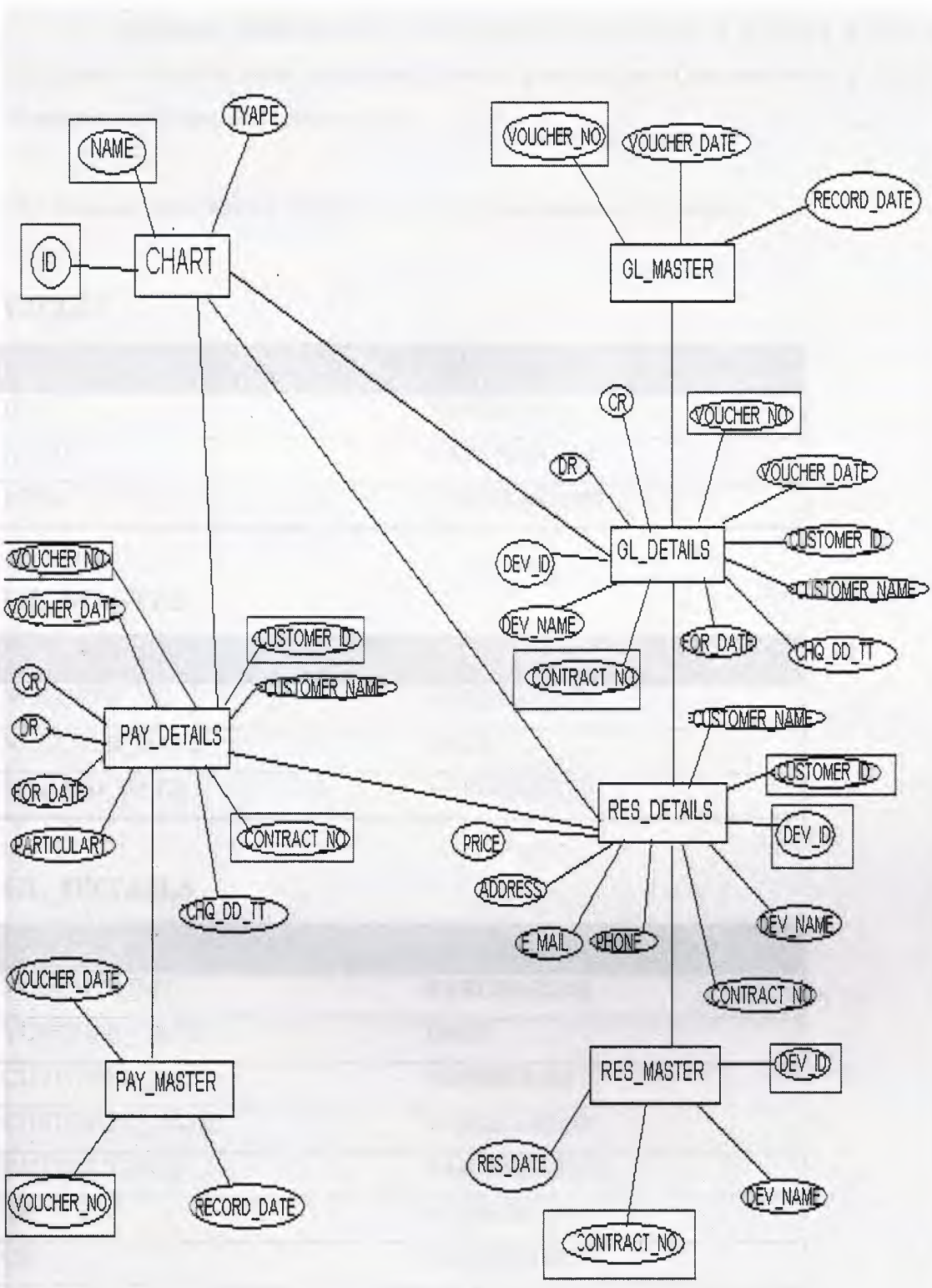
3.2.7 Res _details

Res _details entity contains the following attributes:

- Contract _no the number of the contract (primary key) (foreign key).
- Res _date the reservation date.
- DEV _name property name.
- DEV_ID property id.
- Price to store the price (the amount of money).
- Phone to store the customer phone number.
- E _mail to store the customer e _mail.
- Address to store the customer address.
- Customer _name to store the names of the customers.
- Customer _No to store the numbers of customers (foreign key).



3.3 Identify ER Diagram



3.4 My Schema

It is graphical representation of the database system, and it provides a high-level conceptual data model them its supports the user's perception of the data and it is composed of entities, attributes, and relationships.

The database described by divided the schema into tables as following:

CHART

NAME	TYPE
ID	NUMBER(10)
NAME	VARCHAR2(60)
LEVE	VARCHAR2(30)

GL_MASTER

NAME	TYPE
VOUCHER_NO	VARCHAR2(8)
VOUCHER_DATE	DATE
RECORD_DATE	VARCHAR2(30)

GL_DETAILS

NAME	TYPE
VOUCHER_NO	VARCHAR2(8)
VOUCHER_DATE	DATE
CUSTOMER_ID	NUMBER(10)
CUSTOMER_NAME	VARCHAR2(60)
PARTICULAR1	VARCHAR2(150)
DR	NUMBER(15,2)
CR	NUMBER(15,2)
CHQ_DD_TT	VARCHAR2(20)
FOR_DATE	DATE

DEV_ID	NUMBER(10)
DEV_NAME	VARCHAR2(35)
CONTRACT_NO	NUMBER

RES_MASTER

NAME	TYPE
CONTRACT_NO	NUMBER(10)
DEV_ID	NUMBER(10)
DEV_NAME	VARCHAR2(35)
RES_DATE	DATE

RES_DETAILS

NAME	TYPE
CONTRACT_NO	NUMBER
DEV_ID	NUMBER(10)
DEV_NAME	VARCHAR2(30)
CUSTOMER_ID	NUMBER(10)
CUSTOMER_NAME	VARCHAR2(60)
RES_DATE	DATE
PRICE	NUMBER(10)
ADDRESS	VARCHAR2(150)
E_MAIL	VARCHAR2(50)
PHONE	NUMBER(10)

PAY_MASTER

NAME	TYPE
VOUCHER_NO	VARCHAR2(8)
VOUCHER_DATE	DATE
RECORD_DATE	DATE

PAY_DETAILS

NAME	TYPE
VOUCHER_NO	VARCHAR2(8)
VOUCHER_DATE	DATE
CUSTOMER_ID	NUMBER(10)
CUSTOMER_NAME	VARCHAR2(60)
PARTICULAR1	VARCHAR2(150)
DR	NUMBER(15,2)
CR	NUMBER(15,2)
CHQ_DD_TT	VARCHAR2(20)
FOR_DATE	DATE
DEV_ID	NUMBER(10)
DEV_NAME	VARCHAR2(35)
CONTRACT_NO	NUMBER

Chapter 4

IMPLEMENTATION OF ESTATE AGENCY SOFTWARE AND FORMS DESIGNED

4.1 Introduction

The section illustrates the rules of programming languages SQL, PL/SQL, Oracle8i and Oracle developer 6i.

Also the implementations of this system phases are presents in the project.

4.2 Programming Languages

All languages which applied in this computerize system, and it is roles in this system presents as following.

4.2.1 SQL (Structured Query Language)

SQL language is formal language provide notation for representing queries, it is more user friendly and it is combination of relational algebra and relational calculus concepts.

This language used to contain systems database tables that can by generate script file which contain SQL statements to create tables in database. As shown in source code in appendix.

4.2.2 PL/SQL (Procedural Language / Structured Query Language)

A procedural programming language is a programming language that users detailed, sequential instructions to process data. A PL/SQL program combines SQL commands (such as SELECT AND UPDATE) with procedural commands for tasks ,such as manipulating variables values ,evaluating IF/THEN logic structures ,and creating loop structures that repeat multiple times until an exist condition is reached . The PL/SQL can contain SQL commands.

4.3 Oracle Developer 6i

This platform used to manipulate with developer forms which used to implement the desktop application forms.

4.4 Reusable Component

The components are used for all interfaces (forms) .That called when needed by a system. As shown in appendix.

4.4.1 Menu Bar

It's used to move between forms.

Menu's Items:

1) CHART.
2) Bank payments.
3) Bank receipts.
4) Reservation form.
5) Window.

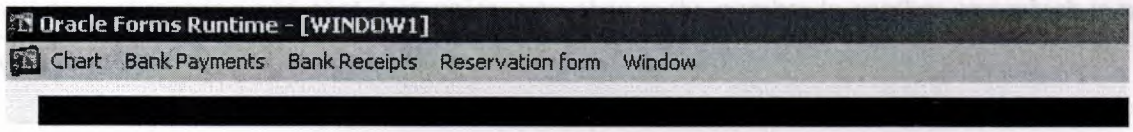


Figure 4.1 Menu Bar

This menu item is to move between the CHAR, Bank payment, Bank receipts, Reservation form and window forms that move to insert and update forms

4.4.2 Function Reusable

These functions are used for all interfaces (forms). That called when needed by a system and any another application by oracle can be use these function.

These function descried as a following, the coding of these functions shown in appendix.

→WHEN-NEW-FORM-INSTANCE

It contains the function built-in which will maximize the window through the run process also contain that trigger which will see if the user was granted to use a specific form or not , if he don't have the privileges to use a specific form then system will prevent user from getting into this form and give him a message .

→ON-CLEAR-DETAILS

This trigger will clear all data in master_details in form, so when you get to form you won't see data that in database until you execute query. (Fires when a coordination-causing event occurs in a block that is a master block in a Master/Detail relation. A coordination-causing event is any event that makes a different record the current record in the master block)

4.4.3 Form Functions

In Oracle (Triggers)

The function of some triggers used in this system. More details in appendix.

→KEY-NEXT-ITEM

This trigger works when you insert data and then moves into next item to insert another data so when there was error as having repetition in number which is primary key then system will prevent moving into next item unless you change the number to another one which is no repeated .

→POST-CHANGE

Post-change means that the trigger will activate after the action is done.

If we tried to change the primary keys numbers then system will refused this change and give message, also this trigger will directly call the name of the data that you insert its number ID and show it in the field.

→ON-CHECK-DELETE-MASTER

Form Builder creates this trigger automatically when you define a master/detail relation and set the Delete Record Behavior property to Non-Isolated.

It fires when there is an attempt to delete a record in the master block of a master/detail relation.

.

→ON-POPULATE-DETAILS

Form Builder creates this trigger automatically when a Master/Detail relation is defined. It fires when Form Builder would normally need to populate the detail block in a Master/Detail relation.

→WHEN-VALIDATE-ITEM

Fires during the Validate the Item process. Specifically, it fires as the last part of item validation for items with the New or Changed validation status.

→WHEN-BUTTON-PRESSED

Fires when an operator selects a button, by clicking with a mouse, or using the keyboard.

4.5 Forms Designed

To meeting the system requirement (nonfunctional requirements) that concern on simplicity and clarity of software application, project application optimal interfaces design used that help a user to understand the application with low training courses and help guide. The architecture of interfaces designed is show below.

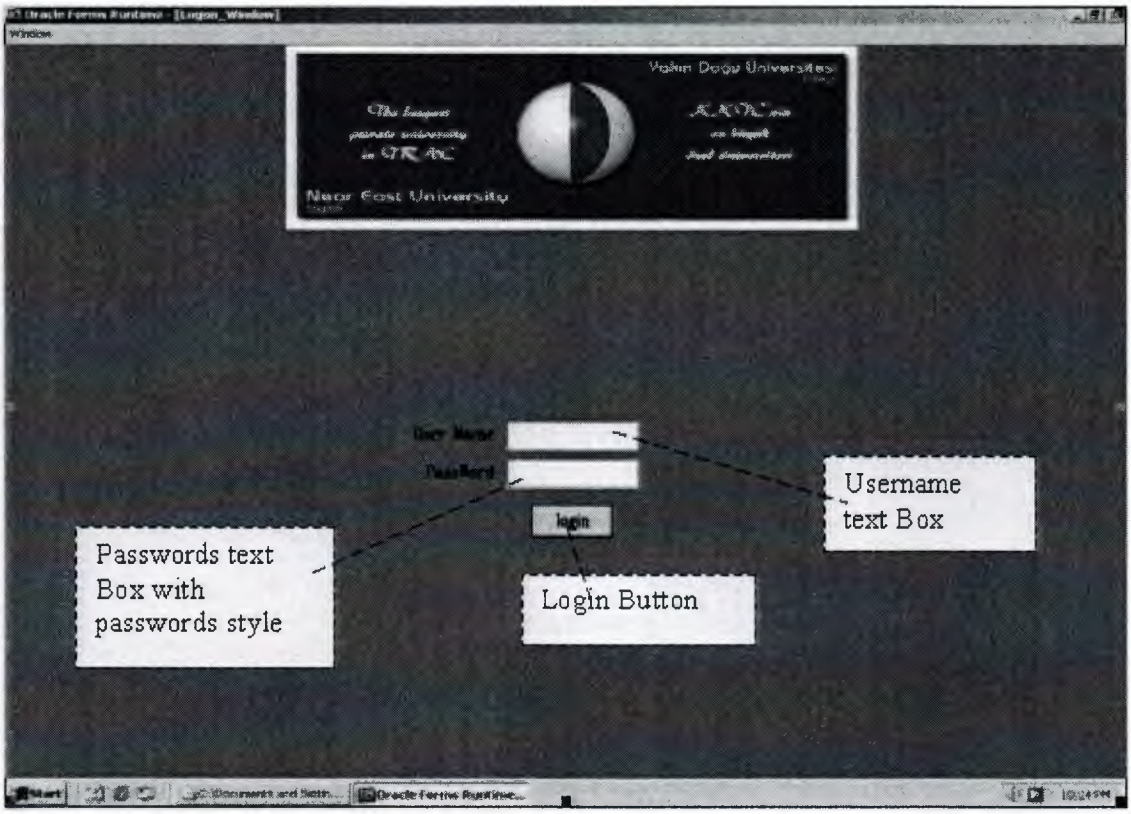


Figure 4.2 login form

There is two fields in this form when user want to uses the forms he has to know the password and the user name, the login button when the user enter the password, and the user name and press login button there is program inside the button to check if the username and password is right after that he can use the forms.

Figure 4.3 Menu Form

This form contains the menu items is to move between the CHAR , Bank payment , Bank receipts, Reservation form, window forms to insert and update forms and also there is buttons to move between the forms to make the forms flexible and easy to use.

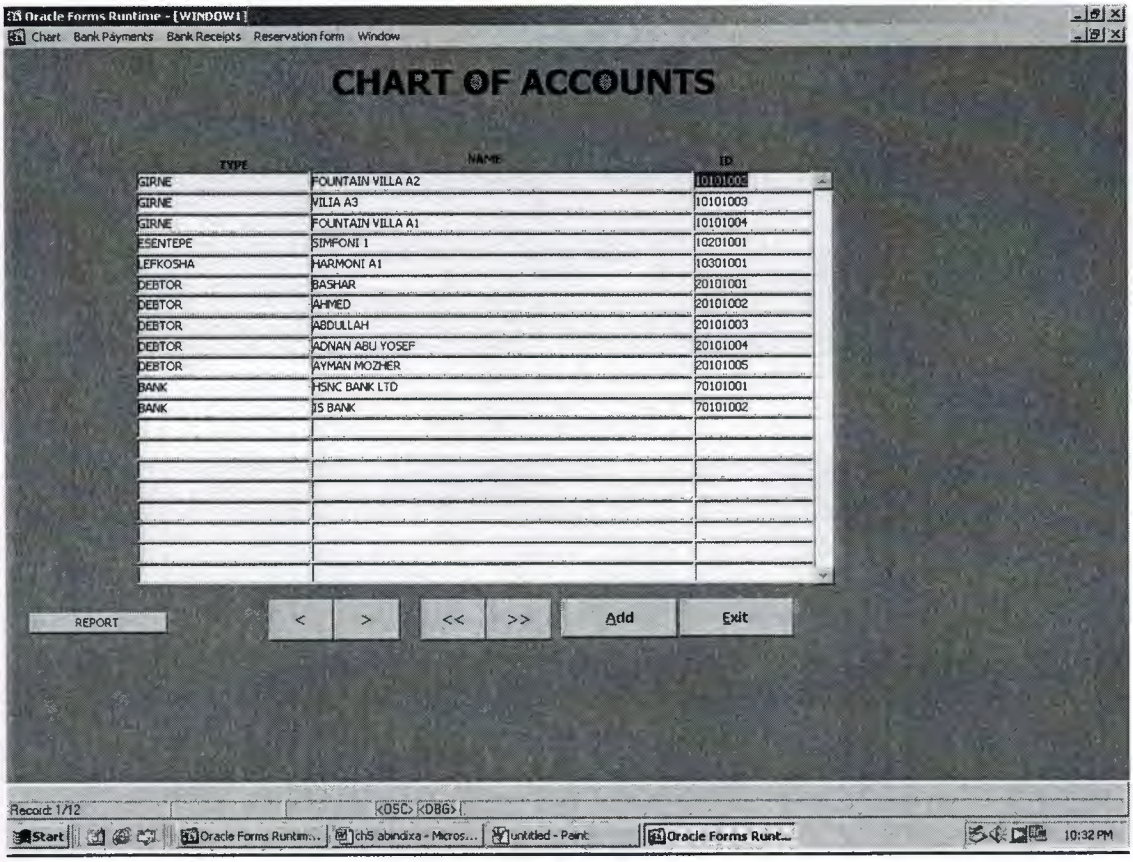


Figure 4.4 Charts of Accounts

This form for adding the id ,name, type for all the customers ,banks and properties to store the records in chart table to be use in other forms when the record inserting like reservation form ,payments and receipts to make the forms flexible ,easy to insert and more secure for the user .and the user can get the reports about any accounts he has to count the number of each account e.g. to know how many customer (debtor) they have .

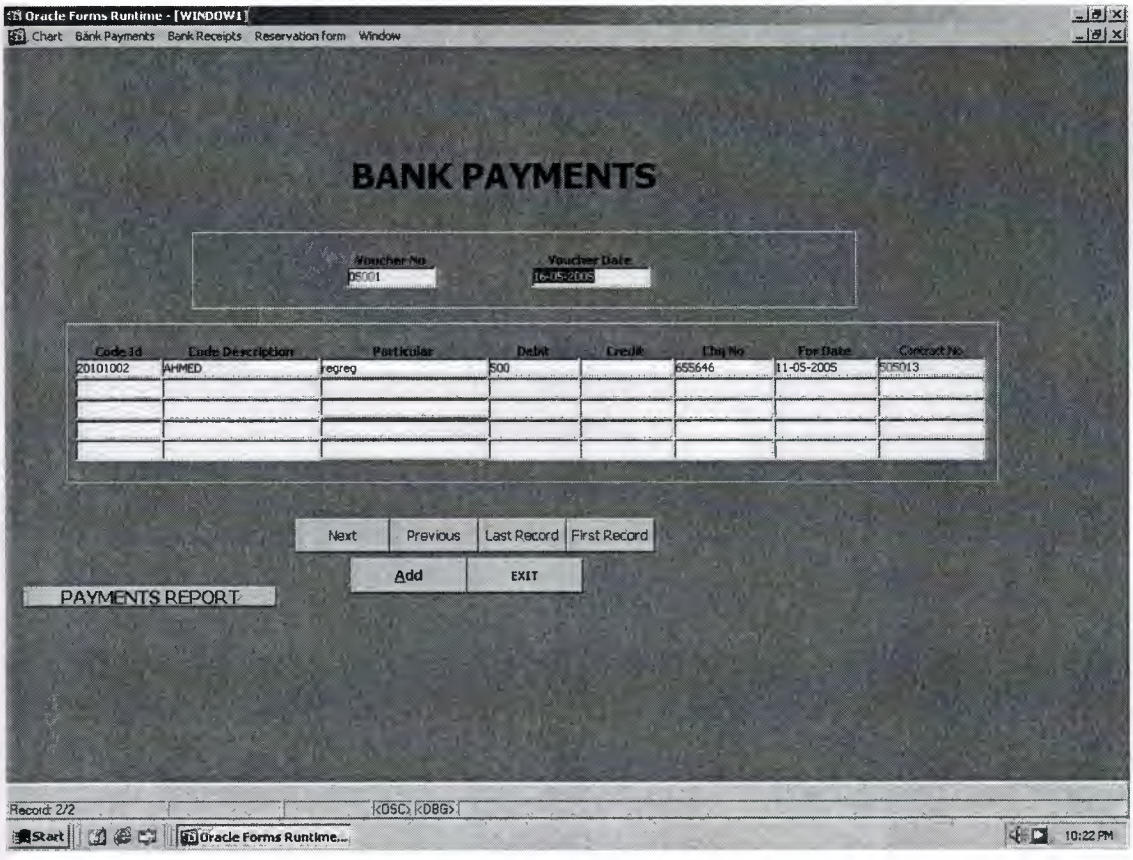


Figure 4.5 Bank Payments

This form will be used for all kind of bank payments. Here in this form voucher number is a consecutive number for the record. where first two digit of voucher display the year second two for year and the rest three for voucher number in that respective month.

As we are working on the double entry accounting system so here we used debit and credit terms for flexibility to produce different kind of reports on the reports level. for forming future budgets. Like cash flow, debtors report etc.

At the same time in the details table we keeping the reference for the property so that use could not enter the record and development name is coming for reservation form from which it's evident that this is relational database system.

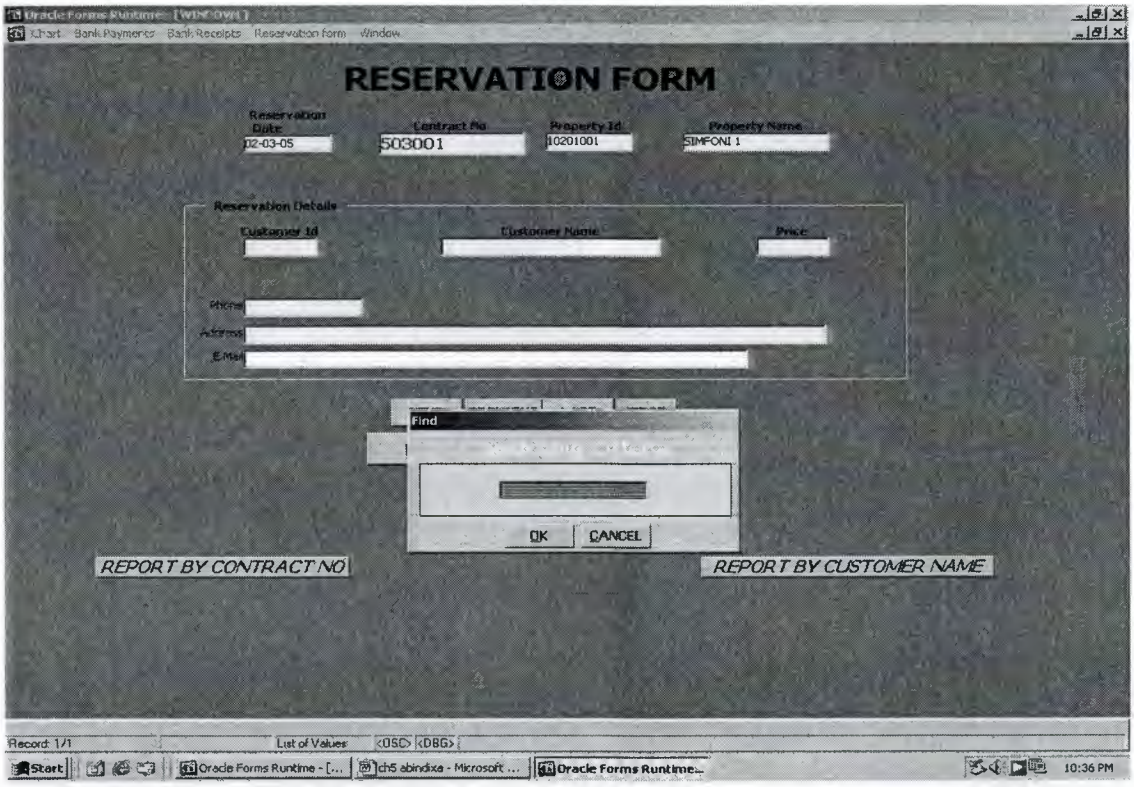


Figure 4.5 Reservation Form

This form for inserting the information about the customer when he comes to bay new development of the estate agency and to make the contract,

By this form user can see the data saved for each customer by searching about the data by using the find button.

Also the user can get report about each customer by the report button.

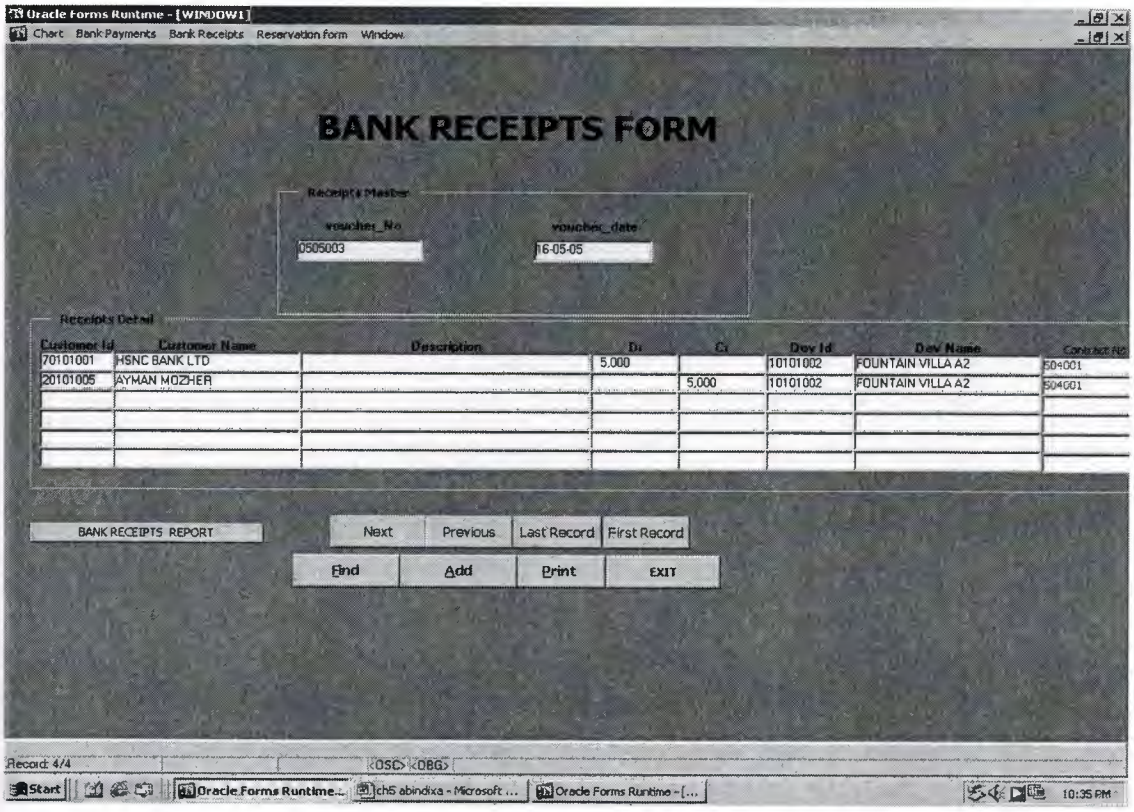


Figure 4.6 Bank Receipts Form

This form to insert the records and data about the bank receipts by inserting the information about the voucher no, voucher date and the information about debtor and creditor and the amount of money.

RESARVATION REPORT

Friday, May 20 2005

Customer **AHMED**

Contract No	Dev Name	Price	Address	E Mail	Phone
505001	FOUNTAIN VILLA A1	25000	MAGOSA	lion@hotmail.com	43532636
505012	FOUNTAIN VILLA A2	30000	jakarta	info@neu.edu	53453453
505002	FOUNTAIN VILLA A2	254412	girne	info@neu.edu	45658787
505008	FOUNTAIN VILLA A1	32000	jordan	jordan@yahoo.comm	555555522
505009	SIMFONI 1	40000	turkey	turkey@hotmail.com	5555555
505013	FOUNTAIN VILLA A1	5500	lefkosha	bashar@yahoo.com	545456465

Figure 4.7 Reports for the Reservation by Customer Name

By this software user can get reports about each form.

For example:

The report above showing us the information's about the customers and the reports are ready for all the forms to use it the user jus has to press the reports buttons.

Future Work

The advise to those who will take this project as reference is to carry with the following topics:

The Database system developed in the project could be expanded to include "Automation of the system" that is all calculations and operations will be automatically done by the system. Reservation deposit and late fees could also be automatically calculated. Since the system will be developed to cover large companies, the commissions of the agents will be calculated by the system.

On the security issue of the system, when prices are automatically updated they will be taken from the price table and there will be a password to protect this table so that no other user aside from the administration department can change the prices on these tables. Commission will also be automatically calculated based on the total amount, the commission will be calculated only when certain percentage of the total amount is paid, and once the commission will be paid it will be recorded, if the agent will claim that they have not received their commission even though they have been paid, the system will keep track of that. When reservations are made for the customers and they request that the furniture's also be purchased by the company, the system will generate a list of utilities each customer has ordered and will automatically relate them with the reservation table.

Since the database is expanding, the number of staff will also expand, so there will be a pay roll system. To achieve all this we will have to create tables and forms with more triggers.

APPENDIX

Triggers:

1. PASS_WORD_FORM :

WHEN-NEW-FORM-INSTANCE

```
SET_WINDOW_PROPERTY(FORMS_MDI_WINDOW,WINDOW_STATE,MAXIMIZE
);
SET_WINDOW_PROPERTY('W_LOGON',WINDOW_STATE,MAXIMIZE);
```

WHEN-WINDOW-CLOSED

```
EXIT_FORM;
```

WHEN-WINDOW-DEACTIVATED

```
EXIT_FORM;
```

ON-LOGON

```
LOGON('bashar','z');
```

1.1 CONTROL BLOCK:

- **LOGON BUTTON :**

WHEN-BUTTON-PRESSED

```
:GLOBAL.CO:=0;
IF  :U_NAME='BASHAR'
     AND :PASS='Z' THEN
     OPEN_FORM('C:\DEV6I\BIN\MENU.FMX');
ELSE
     : U_NAME:=NULL; :PASS:=NULL;
        :GLOBAL.CO:=:GLOBAL.CO+1;
        IF :GLOBAL.CO=3 THEN
```

```
        EXIT_FORM;
ELSE
        MESSAGE ('NOT VAILD USER_NAME OR PASSWORD');
        END IF;
END IF;
```

2. CHART FORM :

WHEN-NEW-FORM-INSTANCE

```
BEGIN
SET_WINDOW_PROPERTY
(FORMS_MDI_WINDOW,WINDOW_STATE,MAXIMZE);
SET_WINDOW_PROPERTY ( 'WINDOW1' , WINDOW_STATE , MAXIMIZE);
                GO_BLOCK ( 'CHART' ) ;
                LAST_RECORD;
                EXECUTE_QUERY;
                        END;
```

2.1 CHART BLOCK:

WHEN-NEW-FORM-INSTANCE

```
EXECUTE_QUERY;
```

KEY-NEXT-ITEM

```
        BEGIN
                IF      (:CHART.LEVE='DEBTOR') THEN
                        ELECT NVL(MAX(ID),0)+1
                        INTO :CHART.ID
                        FROM CHART
                        WHERE LEVE  LIKE'%DEBTOR%';
                 ELSIF
                         (:CHART.LEVE='BANK') THEN
                          SELECT NVL(MAX(ID),0)+1
```

```
            INTO :CHART.ID
            FROM CHART
            WHERE LEVE  LIKE'%BANK%';
ELSIF
           (:CHART.LEVE='GIRNE') THEN
           SELECT NVL(MAX(ID),0)+1
           INTO :CHART.ID
           FROM CHART
           WHERE LEVE  LIKE'%GIRNE%';
ELSIF
           (:CHART.LEVE='LEFKOSHA') THEN
           SELECT NVL(MAX(ID),0)+1
           INTO :CHART.ID
           FROM CHART
           WHERE LEVE  LIKE'%LEFKOSHA%';
ELSIF
           (:CHART.LEVE='LEFKA') THEN
           SELECT NVL(MAX(ID),0)+1
           INTO :CHART.ID
           FROM CHART
           WHERE LEVE  LIKE'%LEFKA%';
 ELSIF
          (:CHART.LEVE='MAGOSA') THEN
          SELECT NVL(MAX(ID),0)+1
          INTO :CHART.ID
          FROM CHART
          WHERE LEVE  LIKE'%MAGOSA%';
  ELSIF
          (:CHART.LEVE='GOZALYORD') THEN
          SELECT NVL(MAX(ID),0)+1
          INTO :CHART.ID
          FROM CHART
```

```
                    WHERE LEVE  LIKE'%GOZALYORD%';
            END IF;
GO_ITEM('CHART.NAME');
        END;
```

2. 2 BUTTON BLOCK (BTN_BLOCK):

- **NEXT BUTTON:**

WHEN-BUTTON-PRESSED

```
BEGIN
        GO_BLOCK('CHART');
        NEXT_RECORD;
                    END;
```

- **PREVIOUS PUTTON**

WHEN-BUTTON-PRESSED

```
BEGIN
        GO_BLOCK('CHART');
        PREVIOUS_RECORD;
        END;
```

- **LAST BUTTON**

WHEN-BUTTON-PRESSED

```
BEGIN
        GO_BLOCK('CHART');
        LAST_RECORD;
        GO_ITEM('BTN_FIRST');
        END;
```

- **FIRST BUTTON**

WHEN-BUTTON-PRESSED

```
BEGIN
```

```
GO_BLOCK('CHART');
FIRST_RECORD;
GO_ITEM('BTN_LAST');
END;
```

- **EXIT BUTTON**

WHEN-BUTTON-PRESSED

```
BEGIN
    ERASE('GLOBAL.CASH_AC');
        ERASE('GLOBAL.MAS_ADD');
        ERASE('GLOBAL.MAS_MODIFY');
        ERASE('GLOBAL.DET_ADD');
        ERASE('GLOBAL.DET_MODIFY');
        ERASE('GLOBAL.DET_DEL');
        ERASE('GLOBAL.MAS_DEL');
        ERASE('GLOBAL.ADD_MASTER');
        ERASE('GLOBAL.VIEWMODE');
    EXIT_FORM(NO_VALIDATE);
END;
```

- **ADD BUTTON**

WHEN-BUTTON-PRESSED

```
BEGIN
    IF    GET_ITEM_PROPERTY('BTN_ADD',LABEL) = '&ADD' THEN
        SET_BLOCK_PROPERTY('CHART',INSERT_ALLOWED,PROPERTY_
        TRUE);
        SET_BLOCK_PROPERTY('CHART',UPDATE_ALLOWED,PROPERTY_
        TRUE);
        IF :SYSTEM.RECORD_STATUS != 'INSERT' THEN
         GO_ITEM('CHART.LEVE');
         LAST_RECORD;
```

```
            CREATE_RECORD;
            END IF;

        SET_ITEM_PROPERTY('BTN_ADD'     ,LABEL,'SA&VE');
        SET_ITEM_PROPERTY('BTN_PREVIOUS',   ENABLED,PROPERTY_TRUE);
        SET_ITEM_PROPERTY('BTN_NEXT',       ENABLED,PROPERTY_TRUE);
        SET_ITEM_PROPERTY('BTN_FIRST',      ENABLED,PROPERTY_TRUE);
        SET_ITEM_PROPERTY('BTN_LAST',       ENABLED,PROPERTY_TRUE);

ELSE
            IF (SHOW_ALERT('SAVE') = ALERT_BUTTON1) THEN
            GO_BLOCK('CHART');
            COMMIT_FORM;
            IF :SYSTEM.FORM_STATUS = 'QUERY' THEN
            SET_ITEM_PROPERTY('BTN_ADD'     ,LABEL      ,'&ADD');
SET_BLOCK_PROPERTY('CHART',INSERT_ALLOWED,PROPERTY_FALSE);
SET_BLOCK_PROPERTY('CHART',UPDATE_ALLOWED,PROPERTY_FALSE);
          GO_BLOCK('CHART');
          GO_ITEM('BTN_BLOCK.BTN_ADD');
          MESSAGE('CODE HAS SUCCESSFULLY SAVED.....');
ELSE
            MESSAGE('CODE SUCCESSFULLY COULD NOT BE SAVED!');
            MESSAGE('CODE SUCCESSFULLY COULD NOT BE SAVED!!');
            END IF;

       END IF;
      END IF;
END;
```

3. PAY_MASTER_DETAILS FORM:

ON-CLEAR-DETAILS

```
BEGIN
 CLEAR_ALL_MASTER_DETAILS;
END;
```

WHEN-NEW-FORM-INSTANCE

```
BEGIN
SET_WINDOW_PROPERTY(FORMS_MDI_WINDOW,WINDOW_STATE,MAXIMIZE
);
SET_WINDOW_PROPERTY('WINDOW1',WINDOW_STATE,MAXIMIZE);
EXECUTE_QUERY;
LAST_RECORD;
GO_BLOCK('PAY_MASTER');
GO_ITEM('PAY_MASTER.VOUCHER_DATE');
END;
```

3.1 PAY_MASTER BLOCK:

ON-POPULATE-DETAILS

```
DECLARE
 RECSTAT    VARCHAR2(20) := :SYSTEM.RECORD_STATUS;
 STARTITM   VARCHAR2(61) := :SYSTEM.CURSOR_ITEM;
 REL_ID     RELATION;
BEGIN
 IF ( RECSTAT = 'NEW' OR RECSTAT = 'INSERT' ) THEN
  RETURN;
 END IF;
IF ( (:PAY_MASTER.VOUCHER_NO IS NOT NULL) OR
(:PAY_MASTER.VOUCHER_DATE IS NOT NULL) OR
(:PAY_MASTER.DOCUMENT_TYPE_ID IS NOT NULL) ) THEN
  REL_ID := FIND_RELATION('PAY_MASTER.PAY_MASTER_PAY_DETAILS');
  QUERY_MASTER_DETAILS(REL_ID, 'PAY_DETAILS');
 END IF;
IF ( :SYSTEM.CURSOR_ITEM <> STARTITM ) THEN
```

```
   GO_ITEM(STARTITM);
   CHECK_PACKAGE_FAILURE;
 END IF;
END;
```

ON-CHECK-DELETE-MASTER

```
DECLARE
 DUMMY_DEFINE CHAR(1);
CURSOR PAY_DETAILS_CUR IS
   SELECT 1 FROM PAY_DETAILS P
   WHERE P.VOUCHER_NO = :PAY_MASTER.VOUCHER_NO AND
P.VOUCHER_DATE = :PAY_MASTER.VOUCHER_DATE AND
P.DOCUMENT_TYPE_ID = :PAY_MASTER.DOCUMENT_TYPE_ID;
BEGIN
OPEN PAY_DETAILS_CUR;
 FETCH PAY_DETAILS_CUR INTO DUMMY_DEFINE;
 IF ( PAY_DETAILS_CUR%FOUND ) THEN
MESSAGE('CANNOT DELETE MASTER RECORD WHEN MATCHING DETAIL
RECORDS EXIST.');
CLOSE PAY_DETAILS_CUR;
RAISE FORM_TRIGGER_FAILURE;
 END IF;
 CLOSE PAY_DETAILS_CUR;
END;
```

- **VOUCHER_DATE ITEM :**

WHEN-VALIDATE-ITEM

```
        IF SUBSTR(:PAY_MASTER.VOUCHER_DATE,4,6) = 'JAN-03' THEN
        SELECT      NVL(MAX(TO_NUMBER(VOUCHER_NO)),0)+1
        INTO        : PAY_MASTER.VOUCHER_NO
        FROM        PAY_MASTER
        WHERE             SUBSTR (DOCUMENT_TYPE_ID,1,2) = 'BP'
        AND               SUBSTR (VOUCHER_DATE,4,6) = 'JAN-03'
```

```
        AND                 TO_CHAR (VOUCHER_DATE,'MM/RRRR') =
        TO_CHAR(:PAY_MASTER.VOUCHER_DATE,'MM/RRRR');
              ELSE
        SELECT
        TO_CHAR(:PAY_MASTER.VOUCHER_DATE,'MM')||TO_CHAR(NVL(MAX
        (TO_NUMBER(SUBSTR(VOUCHER_NO,5,4))),0)+1,'FM009')
        NTO         :PAY_MASTER.VOUCHER_NO
        FROM        PAY_MASTER
        WHERE               SUBSTR(DOCUMENT_TYPE_ID,1,2) = 'BP'
        AND                 SUBSTR(VOUCHER_DATE,4,6) <> 'JAN-03'
        AND                 TO_CHAR(VOUCHER_DATE,'MM/RRRR') =
        TO_CHAR(:PAY_MASTER.VOUCHER_DATE,'MM/RRRR');
        END IF;
```

KEY-NEXT-ITEM

```
BEGIN
      GO_BLOCK('PAY_DETAILS');
GO_ITEM('PAY_DETAILS.CUSTOMER_ID');
END;
```

- **RECORD_DATE ITEM:**

WHEN-VALIDATE-ITEM

```
SELECT SYSDATE INTO :RECORD_DATE FROM DUAL;
```

3.2 PAY_DETAILS BLOCK:

- **PARTICULAR1 ITEM :**

KEY-NEXT-ITEM

```
DECLARE
      T_NARRATION PAY_DETAILS.PARTICULAR1%TYPE;
BEGIN
      IF MOD(TO_NUMBER(:SYSTEM.TRIGGER_RECORD),2) = 0 THEN
            T_NARRATION := :PAY_DETAILS.PARTICULAR1;
            PREVIOUS_RECORD;
            :PAY_DETAILS.PARTICULAR1 := T_NARRATION;
```

```
            NEXT_RECORD;
            ELSE
            IF (:PAY_DETAILS.DR IS  NULL) THEN
                  T_NARRATION := :PAY_DETAILS.PARTICULAR1;
                  NEXT_RECORD;
                  IF (:PAY_DETAILS.DR IS NOT NULL) THEN
                        :PAY_DETAILS.PARTICULAR1 := T_NARRATION;
                        PREVIOUS_RECORD;
                  END IF;
                  DELETE_RECORD;
            END IF;
            END IF;
 END;
            :PARTICULAR1 := INITCAP(:PARTICULAR1);
 NEXT_ITEM;
```

KEY-ENTER ITEM:

```
      :PARTICULAR1 := INITCAP(:PARTICULAR1);
NEXT_ITEM;
```

- **DEV_ID ITEM :**

KEY-NEXT-ITEM

```
BEGIN
      NEXT_RECORD;
END;
```

3.3 BTN_BLOCK BLOCK:

- **EXIT BUTTON:**

WHEN-BUTTON-PRESSED

```
EXIT_FORM;
```

- ***BTN_NEXT BUTTON:***

WHEN-BUTTON-PRESSED

```
BEGIN
        GO_BLOCK('PAY_MASTER');
        NEXT_RECORD;
                END;
```

- **BTN_PREVIOUS BUTTON:**

WHEN-BUTTON-PRESSED

```
BEGIN
        GO_BLOCK('PAY_MASTER');
        PREVIOUS_RECORD;
        END;
```

- **BTN_LAST BUTTON:**

WHEN-BUTTON-PRESSED

```
BEGIN
        GO_BLOCK('PAY_MASTER');
        LAST_RECORD;
        GO_ITEM('BTN_FIRST');
        END;
```

- **BTN_FIRST BUTTON :**

WHEN-BUTTON-PRESSED

```
BEGIN
        GO_BLOCK('PAY_MASTER');
        FIRST_RECORD;
        GO_ITEM('BTN_LAST');
        END;
```

- **BTN_ADD BUTTON:**

WHEN-BUTTON-PRESSED

```
BEGIN
    IF GET_ITEM_PROPERTY('BTN_ADD',LABEL) = '&ADD' THEN
            SET_BLOCK_PROPERTY('PAY_MASTER',INSERT_ALLOWED,
            PROPERTY_TRUE);
            SET_BLOCK_PROPERTY('PAY_MASTER',UPDATE_ALLOWED,
            PROPERTY_TRUE);
            SET_BLOCK_PROPERTY('PAY_DETAILS',INSERT_ALLOWED,
            PROPERTY_TRUE);
            SET_BLOCK_PROPERTY('PAY_DETAILS',UPDATE_ALLOWED,
            PROPERTY_TRUE);

             GO_ITEM('PAY_MASTER.VOUCHER_DATE');

      IF :SYSTEM.RECORD_STATUS != 'INSERT' THEN
          CREATE_RECORD;
          SELECT SYSDATE INTO :PAY_MASTER.VOUCHER_DATE FROM
          DUAL;
      END IF;
  SET_ITEM_PROPERTY('BTN_ADD'              ,LABEL,'SA&VE');
  SET_ITEM_PROPERTY('BTN_PREVIOUS'  ,ENABLED,PROPERTY_TRUE);
  SET_ITEM_PROPERTY('BTN_NEXT'        ,ENABLED,PROPERTY_TRUE);
  SET_ITEM_PROPERTY('BTN_FIRST'       ,ENABLED,PROPERTY_TRUE);
  SET_ITEM_PROPERTY('BTN_LAST'        ,ENABLED,PROPERTY_TRUE);
ELSE
  COMMIT_FORM;
IF :SYSTEM.FORM_STATUS = 'QUERY' THEN
SET_ITEM_PROPERTY('BTN_ADD' ,  LABEL   ,'&ADD');
 SET_BLOCK_PROPERTY('PAY_MASTER',INSERT_ALLOWED,
```

```
        PROPERTY_FALSE);
SET_BLOCK_PROPERTY('PAY_MASTER',UPDATE_ALLOWED,
        PROPERTY_FALSE);
SET_BLOCK_PROPERTY('PAY_DETAILS',INSERT_ALLOWED,
        PROPERTY_FALSE);
SET_BLOCK_PROPERTY('PAY_DETAILS',UPDATE_ALLOWED,
        PROPERTY_FALSE);
GO_BLOCK('BTN_BLOCK');
GO_ITEM('BTN_BLOCK.BTN_ADD');
MESSAGE('BANK RECEIPT VOUCHER SUCCESSFULLY SAVED....');
ELSE
        MESSAGE('BANK RECEIPT VOUCHER COULD NOT BE SAVED!');
        MESSAGE('BANK RECEIPT VOUCHER COULD NOT BE SAVED!');
        END IF;
 END IF;
END;
```

3.4 FIND_BLK BLOCK:

- **W_BR_ID_FROM ITEM:**

KEY-NEXT-ITEM

```
BEGIN
        IF (:FIND_BLK.W_BR_ID_FROM IS NULL) THEN
                SELECT MIN(VOUCHER_NO)
                INTO   :W_BR_ID_FROM
                FROM   GL_MASTER
                WHERE  VOUCHER_DATE >= :W_BR_DATE_FROM
                AND DOCUMENT_TYPE_ID LIKE 'BR%';
        ELSE
                IF LENGTH(:W_BR_ID_FROM) <= 4 THEN :W_BR_ID_FROM :=
           TO_CHAR(:W_BR_DATE_FROM,'RRMM')||lpad(:W_BR_ID_FROM,4,'0');
                END IF;
        END IF;
        :FIND_BLK.W_BR_ID_TO :=       :FIND_BLK.W_BR_ID_FROM;
```

```
        GO_ITEM('FIND_BLK.W_BR_ID_TO');
END;
```

- **W_BR_ID_TO ITEM:**

KEY-NEXT-ITEM

```
BEGIN
        IF (:FIND_BLK.W_BR_ID_TO IS NULL) THEN
                SELECT MAX(VOUCHER_NO)
                INTO   :W_BR_ID_TO
                FROM   GL_MASTER
                WHERE  VOUCHER_DATE <= :W_BR_DATE_TO
                AND DOCUMENT_TYPE_ID LIKE 'BR%';
        ELSE
           IF LENGTH(:W_BR_ID_TO) <= 4 THEN :W_BR_ID_TO :=
           TO_CHAR(:W_BR_DATE_FROM,'RRMM')||LPAD(:W_BR_ID_TO,4,'0');
                END IF;
        END IF;
        NEXT_ITEM;
END;
```

- **W_BR_DATE_FROM ITEM:**

KEY-NEXT-ITEM

```
DECLARE
        V_ALERT_BTN NUMBER;
BEGIN
        IF (:FIND_BLK.W_BR_DATE_FROM IS NULL) THEN
                V_ALERT_BTN := SHOW_ALERT('NULL_ERROR');
                RAISE FORM_TRIGGER_FAILURE;
        ELSE
                IF :W_BR_DATE_TO < :W_BR_DATE_FROM THEN
                        :W_BR_DATE_TO := :W_BR_DATE_FROM;
                END IF;
```

```
        END IF;
        :FIND_BLK.W_BR_DATE_TO     :=     :FIND_BLK.W_BR_DATE_FROM;
        GO_ITEM('FIND_BLK.W_BR_DATE_TO');
END;
```

POST-CHANGE

```
BEGIN
        IF (:FIND_BLK.W_BR_DATE_FROM IS NOT NULL) THEN
           SELECT MIN(VOUCHER_NO)          INTO  :W_BR_ID_FROM
              FROM  GL_MASTER
WHERE  VOUCHER_DATE >= :W_BR_DATE_FROM
              AND DOCUMENT_TYPE_ID LIKE 'BR%';
              END IF;
END;
```

- **W_BR_DATE_TO ITEM:**

KEY-NEXT-ITEM

```
BEGIN
        IF (:FIND_BLK.W_BR_DATE_TO IS NULL) THEN
              :W_BR_DATE_TO := :W_BR_DATE_FROM;
        ELSE
              IF :W_BR_DATE_TO < :W_BR_DATE_FROM THEN
                    :W_BR_DATE_TO := :W_BR_DATE_FROM;
              END IF;
        END IF;

        GO_ITEM('FIND_BLK.BTN_OK');
END;
```

POST-CHANGE

```
BEGIN
        IF (:FIND_BLK.W_BR_DATE_TO IS NOT NULL) THEN
              SELECT MAX(VOUCHER_NO)  INTO  :W_BR_ID_TO
```

```
            FROM   GL_MASTER
            WHERE  VOUCHER_DATE BETWEEN :W_BR_DATE_FROM AND
                :W_BR_DATE_TO  AND DOCUMENT_TYPE_ID LIKE 'BR%';
            END IF;
END;
```

- **BTN_OK BUTTON:**

WHEN-BUTTON-PRESSED

```
DECLARE
      WHERECLAUSE VARCHAR2(300);
BEGIN
    WHERECLAUSE := 'WHERE (GL_MASTER.VOUCHER_NO BETWEEN '
                    || :FIND_BLK.W_BR_ID_FROM
                    || ' AND '
                    || :FIND_BLK.W_BR_ID_TO
                    || ')'
                    || ' AND '
                    || '(GL_MASTER.VOUCHER_DATE BETWEEN '''
                    || :FIND_BLK.W_BR_DATE_FROM
                    || ''' AND '''
                    || :FIND_BLK.W_BR_DATE_TO
                    || ''') AND '
                    || '(GL_MASTER.DOCUMENT_TYPE_ID LIKE "BR%")' ;

   CLEAR_FORM(NO_VALIDATE);
    GO_BLOCK('GL_MASTER');
   SET_BLOCK_PROPERTY('GL_MASTER',DEFAULT_WHERE,WHERECLAUSE
    );
    EXECUTE_QUERY;
    LAST_RECORD;
     GO_ITEM('BTN_FIND');
END;
```

- **ITEM33 BUTTON:**

WHEN-BUTTON-PRESSED

```
GO_ITEM('GL_MASTER.VOUCHER_NO');
SET_ITEM_PROPERTY('BTN_BLOCK.BTN_CLEAR',ENABLED,PROPERTY_FALSE)
;
```

4. RECEIPTS_GL_MASTER_DETAILS FORM :

ON-CLEAR-DETAILS

```
BEGIN
 Clear_All_Master_Details;
END;
```

WHEN-NEW-FORM-INSTANCE

```
BEGIN
SET_WINDOW_PROPERTY(FORMS_MDI_WINDOW,WINDOW_STATE,MAXIMIZE
);
SET_WINDOW_PROPERTY('WINDOW1',WINDOW_STATE,MAXIMIZE);
EXECUTE_QUERY;
LAST_RECORD;
GO_BLOCK('GL_MASTER');
GO_ITEM('GL_MASTER.VOUCHER_DATE');
END;
```

4.1 GL_MASTER BLOCK :

ON-POPULATE-DETAILS

```
DECLARE
        RECSTAT    VARCHAR2(20) := :SYSTEM.RECORD_STATUS;
        STARTITM   VARCHAR2(61) := :SYSTEM.CURSOR_ITEM;
        REL_ID     RELATION;
BEGIN
        IF ( RECSTAT = 'NEW' OR RECSTAT = 'INSERT' ) THEN
            RETURN;
 END IF;
IF ( (:GL_MASTER.VOUCHER_NO IS NOT NULL) OR
   (:GL_MASTER.VOUCHER_DATE IS NOT NULL) OR
   (:GL_MASTER.DOCUMENT_TYPE_ID IS NOT NULL) ) THEN
   REL_ID := FIND_RELATION('GL_MASTER.GL_MASTER_GL_DETAILS');
   QUERY_MASTER_DETAILS(REL_ID, 'GL_DETAILS');
 END IF;
IF ( :SYSTEM.CURSOR_ITEM <> STARTITM ) THEN
   GO_ITEM(STARTITM);
   CHECK_PACKAGE_FAILURE;
 END IF;
END;
```

ON-CHECK-DELETE-MASTER

```
DECLARE
        DUMMY_DEFINE CHAR(1);
CURSOR GL_DETAILS_CUR IS
        SELECT 1 FROM GL_DETAILS G
   WHERE G.VOUCHER_NO = :GL_MASTER.VOUCHER_NO AND
   G.VOUCHER_DATE = :GL_MASTER.VOUCHER_DATE AND
   G.DOCUMENT_TYPE_ID = :GL_MASTER.DOCUMENT_TYPE_ID;

BEGIN
```

```
    OPEN GL_DETAILS_CUR;
     FETCH GL_DETAILS_CUR INTO DUMMY_DEFINE;
     IF ( GL_DETAILS_CUR%FOUND ) THEN
  MESSAGE('CANNOT DELETE MASTER RECORD WHEN MATCHING DETAIL
   RECORDS EXIST.');
   CLOSE GL_DETAILS_CUR;
  RAISE FORM_TRIGGER_FAILURE;
 END IF;
 CLOSE GL_DETAILS_CUR;
END;
```

- **VOUCHER_DATE ITEM :**

WHEN-VALIDATE-ITEM

```
IF SUBSTR(:GL_MASTER.VOUCHER_DATE,4,6) = 'JAN-03' THEN
   SELECT       NVL(MAX(TO_NUMBER(VOUCHER_NO)),0)+1
   INTO                              :GL_MASTER.VOUCHER_NO
   FROM                                        GL_MASTER
   WHERE            SUBSTR(DOCUMENT_TYPE_ID,1,2) = 'BR'
   AND               SUBSTR(VOUCHER_DATE,4,6) = 'JAN-03'
   AND             TO_CHAR(VOUCHER_DATE,'MM/RRRR') =
                 TO_CHAR(:GL_MASTER.VOUCHER_DATE,'MM/RRRR');
   ELSE
    SELECT
TO_CHAR(:GL_MASTER.VOUCHER_DATE,'RR')||TO_CHAR(:GL_MASTER.VOUCH
ER_DATE,'MM')||TO_CHAR(NVL(MAX(TO_NUMBER(SUBSTR(VOUCHER_NO,5,4))
),0)+1,'FM009')
   INTO                                    :GL_MASTER.VOUCHER_NO
   FROM                                        GL_MASTER
   WHERE            SUBSTR(DOCUMENT_TYPE_ID,1,2) = 'BR'
   AND              SUBSTR(VOUCHER_DATE,4,6) <> 'JAN-03'
   AND             TO_CHAR(VOUCHER_DATE,'MM/RRRR') =
```

```
                    TO_CHAR(:GL_MASTER.VOUCHER_DATE,'MM/RRRR');
END IF;
```

KEY-NEXT-ITEM

```
BEGIN
        GO_BLOCK('GL_DETAILS');
        GO_ITEM('GL_DETAILS.CUSTOMER_ID');
END;
```

- **PUSH_BUTTON31 ITEM:**

WHEN-BUTTON-PRESSED

```
DECLARE
        LIST_ID PARAMLIST;
        LIST_NAME       VARCHAR2(10) := 'TEMP';

BEGIN
         LIST_ID := GET_PARAMETER_LIST(LIST_NAME);
      IF NOT ID_NULL(LIST_ID) THEN
      DESTROY_PARAMETER_LIST(LIST_ID);
  END IF;
        LIST_ID := CREATE_PARAMETER_LIST(LIST_NAME);

      IF ID_NULL(LIST_ID) THEN
   MESSAGE('ERROR CREATING PARAMETER LIST FOR REPORT');
   RAISE FORM_TRIGGER_FAILURE;
ELSE
      IF :GL_MASTER.VOUCHER_DATE = '12-JAN-2005' THEN
         ADD_PARAMETER(LIST_ID, 'VCH_NO', TEXT_PARAMETER,
        :GL_MASTER.VOUCHER_NO);
        ADD_PARAMETER(LIST_ID, 'VCH_NO1', TEXT_PARAMETER,
```

```
        :GL_MASTER.VOUCHER_NO);
        ADD_PARAMETER(LIST_ID, 'VCH_DATE', TEXT_PARAMETER,
        :GL_MASTER.VOUCHER_DATE);
        ADD_PARAMETER(LIST_ID, 'VCH_DATE1', TEXT_PARAMETER,
        :GL_MASTER.VOUCHER_DATE);
        ADD_PARAMETER(LIST_ID, 'VCH_TYPE', TEXT_PARAMETER,
        :GL_MASTER.DOCUMENT_TYPE_ID);
        ADD_PARAMETER(LIST_ID, 'PARAMFORM', TEXT_PARAMETER,
           'NO');
       RUN_PRODUCT(REPORTS, 'C:\DEV6I\BIN\GL_LEDGER.REP',
       ASYNCHRONOUS, RUNTIME, FILESYSTEM, LIST_ID, NULL);
    END IF;
  END IF;
END;
```

- **BTN_FIND BUTTON :**

WHEN-BUTTON-PRESSED

```
BEGIN
SET_ITEM_PROPERTY('btn_clear',enabled,PROPERTY_TRUE);
go_item('find_blk.W_BR_ID_FROM');
SELECT                    MIN(VOUCHER_DATE),
                          MAX(VOUCHER_DATE),
LPAD(TO_CHAR(MIN(TO_NUMBER(VOUCHER_NO))),8,'0'),
LPAD(TO_CHAR(MAX(TO_NUMBER(VOUCHER_NO))),8,'0')
INTO                      :FIND_BLK.W_BR_DATE_FROM,
:FIND_BLK.W_BR_DATE_TO, :FIND_BLK.W_BR_ID_FROM,
:FIND_BLK.W_BR_ID_TO FROM  GL_MASTER
WHERE  DOCUMENT_TYPE_ID LIKE 'BR%';
SET_ITEM_PROPERTY('W_BR_DATE_FROM',ITEM_IS_VALID,PROPERTY_TRUE);
 SET_ITEM_PROPERTY('W_BR_DATE_TO',ITEM_IS_VALID,PROPERTY_TRUE);

END;
```

4. 2 GL_DETAILS BLOCK:

- **CUSTOMER_ID ITEM:**

KEY-NEXT-ITEM

```
IF :GL_DETAILS.CUSTOMER_ID IS NULL THEN
                        GO_ITEM('BTN_BLOCK.SAVE');

END IF;
```

- **PARTICULAR1 ITEM:**

KEY-NEXT-ITEM

```
DECLARE
        T_NARRATION GL_DETAILS.PARTICULAR1%TYPE;
BEGIN
        IF MOD(TO_NUMBER(:SYSTEM.TRIGGER_RECORD),2) = 0 THEN
        T_NARRATION := :GL_DETAILS.PARTICULAR1;
  PREVIOUS_RECORD;
        :GL_DETAILS.PARTICULAR1 := T_NARRATION;
  NEXT_RECORD;
ELSE
IF (:GL_DETAILS.DR IS  NULL) THEN
  T_NARRATION := :GL_DETAILS.PARTICULAR1;
   NEXT_RECORD;
IF (:GL_DETAILS.DR IS NOT NULL) THEN
    :GL_DETAILS.PARTICULAR1 := T_NARRATION;
   PREVIOUS_RECORD;
END IF;
   DELETE_RECORD;
 END IF;
END IF;

END;
```

```
:PARTICULAR1 := INITCAP(:PARTICULAR1);
NEXT_ITEM;
```

KEY-ENTER

```
    :PARTICULAR1 := INITCAP(:PARTICULAR1);
NEXT_ITEM;
```

- **DEV_ID ITEM :**

KEY-NEXT-ITEM

```
BEGIN
                                NEXT_RECORD;
END;
```

4. 3 BTN_BLOCK BLOCK:

- **EXIT BUTTON :**

WHEN-BUTTON-PRESSED

```
EXIT_FORM;
```

- **BTN_NEXT BUTTON :**

WHEN-BUTTON-PRESSED

```
BEGIN
      GO_BLOCK('GL_MASTER');
 NEXT_RECORD;
END;
```

- **BTN_PREVIOUS BUTTON:**

WHEN-BUTTON-PRESSED

```
BEGIN
      GO_BLOCK('GL_MASTER');
      LAST_RECORD;
     GO_ITEM('BTN_FIRST');
```

```
END;
```

- **BTN_LAST BUTTON :**

WHEN-BUTTON-PRESSED

```
BEGIN
        GO_BLOCK('GL_MASTER');
     LAST_RECORD;
         GO_ITEM('BTN_FIRST');
END;
```

- **BTN_FIRST BUTTON :**

WHEN-BUTTON-PRESSED

```
BEGIN
        GO_BLOCK('GL_MASTER');
    FIRST_RECORD;
       GO_ITEM('BTN_LAST');
END;
```

- **BTN_ADD BUTTON :**

WHEN-BUTTON-PRESSED

```
BEGIN
          IF GET_ITEM_PROPERTY('BTN_ADD',LABEL) = '&ADD' THEN

SET_BLOCK_PROPERTY('GL_MASTER',INSERT_ALLOWED,PROPERTY_TRUE);

SET_BLOCK_PROPERTY('GL_MASTER',UPDATE_ALLOWED,PROPERTY_TRUE);

SET_BLOCK_PROPERTY('GL_DETAILS',INSERT_ALLOWED,PROPERTY_TRUE);

SET_BLOCK_PROPERTY('GL_DETAILS',UPDATE_ALLOWED,PROPERTY_TRUE);

 GO_ITEM('GL_MASTER.VOUCHER_DATE');
```

```
 IF :SYSTEM.RECORD_STATUS != 'INSERT' THEN

CREATE_RECORD;
      SELECT SYSDATE INTO :GL_MASTER.VOUCHER_DATE FROM DUAL;
      END IF;
 SET_ITEM_PROPERTY('BTN_ADD'                    ,LABEL,'SA&VE');
 SET_ITEM_PROPERTY('BTN_PREVIOUS'    ,ENABLED,PROPERTY_TRUE);
 SET_ITEM_PROPERTY('BTN_NEXT'        ,ENABLED,PROPERTY_TRUE);
 SET_ITEM_PROPERTY('BTN_FIRST'       ,ENABLED,PROPERTY_TRUE);
  SET_ITEM_PROPERTY('BTN_LAST'       ,ENABLED,PROPERTY_TRUE);
ELSE
COMMIT_FORM;
        IF :SYSTEM.FORM_STATUS = 'QUERY' THEN
SET_ITEM_PROPERTY('BTN_ADD'          ,LABEL          ,'&ADD');
SET_BLOCK_PROPERTY('GL_MASTER',INSERT_ALLOWED,PROPERTY_FALSE);

SET_BLOCK_PROPERTY('GL_MASTER',UPDATE_ALLOWED,PROPERTY_FALSE);

SET_BLOCK_PROPERTY('GL_DETAILS',INSERT_ALLOWED,PROPERTY_FALSE);

SET_BLOCK_PROPERTY('GL_DETAILS',UPDATE_ALLOWED,PROPERTY_FALSE);

GO_BLOCK('BTN_BLOCK');
GO_ITEM('BTN_BLOCK.BTN_ADD');
      MESSAGE('BANK RECEIPT VOUCHER SUCCESSFULLY SAVED....');
ELSE
       MESSAGE('BANK RECEIPT VOUCHER COULD NOT BE SAVED!');
       MESSAGE('BANK RECEIPT VOUCHER COULD NOT BE SAVED!');
      END IF;
 END IF;
END;
```

4. 4 FIND_BLK BLOCK:

- **W_BR_ID_FROM ITEM :**

KEY-NEXT-ITEM

```
BEGIN
IF (:FIND_BLK.W_BR_ID_FROM IS NULL) THEN
SELECT MIN(VOUCHER_NO)
INTO  :W_BR_ID_FROM     FROM  GL_MASTER
WHERE  VOUCHER_DATE >= :W_BR_DATE_FROM
AND DOCUMENT_TYPE_ID LIKE 'BR%';
ELSE
IF LENGTH(:W_BR_ID_FROM) <= 4 THEN
:W_BR_ID_FROM :=
TO_CHAR(:W_BR_DATE_FROM,'RRMM')||lpad(:W_BR_ID_FROM,4,'0');
END IF;
END IF;
:FIND_BLK.W_BR_ID_TO:=:FIND_BLK.W_BR_ID_FROM;
GO_ITEM('FIND_BLK.W_BR_ID_TO');
END;
```

- **W_BR_ID_TO ITEM:**

KEY-NEXT-ITEM

```
BEGIN
IF (:FIND_BLK.W_BR_ID_TO IS NULL) THEN
SELECT MAX(VOUCHER_NO)
INTO  :W_BR_ID_TO
FROM  GL_MASTER
WHERE  VOUCHER_DATE <= :W_BR_DATE_TO
AND DOCUMENT_TYPE_ID LIKE 'BR%';
ELSE
IF LENGTH(:W_BR_ID_TO) <= 4 THEN :W_BR_ID_TO :=
TO_CHAR(:W_BR_DATE_FROM,'RRMM')||lpad(:W_BR_ID_TO,4,'0');
END IF;
```

```
END IF;
NEXT_ITEM;
END;
```

5. RES_MASTER_DETAILS FORM.

ON-CLEAR-DETAILS

```
BEGIN
 CLEAR_ALL_MASTER_DETAILS;
END;
```

WHEN-NEW-FORM-INSTANCE

```
BEGIN
SET_WINDOW_PROPERTY(FORMS_MDI_WINDOW,WINDOW_STATE,MAXIMIZE
);
SET_WINDOW_PROPERTY('WINDOW1',WINDOW_STATE,MAXIMIZE);

EXECUTE_QUERY;
GO_BLOCK('FIND_BLK');
LAST_RECORD;
END;
```

5.1 RES_MASTER BLOCK:

ON-POPULATE-DETAILS

```
DECLARE
 RECSTAT    VARCHAR2(20) := :SYSTEM.RECORD_STATUS;
 STARTITM   VARCHAR2(61) := :SYSTEM.CURSOR_ITEM;
 REL_ID     RELATION;

BEGIN
 IF ( RECSTAT = 'NEW' OR RECSTAT = 'INSERT' ) THEN
  RETURN;
 END IF;
```

```
IF ( (:RES_MASTER.CONTRACT_NO IS NOT NULL) OR (:RES_MASTER.DEV_ID IS
NOT NULL) OR (:RES_MASTER.DEV_NAME IS NOT NULL) OR
(:RES_MASTER.RES_DATE IS NOT NULL) ) THEN
REL_ID := FIND_RELATION('RES_MASTER.RES_MASTER_RES_DETAILS');
QUERY_MASTER_DETAILS(REL_ID, 'RES_DETAILS');
END IF;
IF ( :SYSTEM.CURSOR_ITEM <> STARTITM ) THEN
GO_ITEM(STARTITM);
CHECK_PACKAGE_FAILURE;
END IF;
END;
```

ON-CHECK-DELETE-MASTER

```
DECLARE
 DUMMY_DEFINE CHAR(1);

CURSOR RES_DETAILS_CUR IS
   SELECT 1 FROM RES_DETAILS R
   WHERE R.CONTRACT_NO = :RES_MASTER.CONTRACT_NO AND R.DEV_ID
    = :RES_MASTER.DEV_ID AND R.DEV_NAME = :RES_MASTER.DEV_NAME
   AND R.RES_DATE = :RES_MASTER.RES_DATE;

BEGIN
      OPEN RES_DETAILS_CUR;
       FETCH RES_DETAILS_CUR INTO DUMMY_DEFINE;
IF ( RES_DETAILS_CUR%FOUND ) THEN
    MESSAGE('CANNOT DELETE MASTER RECORD WHEN MATCHING
    DETAIL RECORDS EXIST.');
    CLOSE RES_DETAILS_CUR;
    RAISE FORM_TRIGGER_FAILURE;
 END IF;
 CLOSE RES_DETAILS_CUR;
```

```
END;
```

- **RES_DATE ITEM:**

WHEN-VALIDATE-ITEM

```
IF SUBSTR(:RES_MASTER.RES_DATE,4,6) = 'JAN-01' THEN
SELECT     NVL(MAX(TO_NUMBER(CONTRACT_NO)),0)+1
INTO                              :RES_MASTER.CONTRACT_NO
FROM                                        RES_MASTER
WHERE                          SUBSTR(RES_DATE,4,6) = 'JAN-01'
AND                             TO_CHAR(RES_DATE,'MM/RRRR') =
                  TO_CHAR(:RES_master.RES_date,'MM/RRRR');
ELSE
SELECT
TO_CHAR(:RES_MASTER.RES_DATE,'RR')||TO_CHAR(:RES_MASTER.RES_DATE,'
MM')||TO_CHAR(NVL(MAX(TO_NUMBER(substr(CONTRACT_NO,5,4))),0)+1,'FM00
9')
INTO                                          :RES_MASTER.CONTRACT_NO
FROM                                                RES_MASTER
WHERE                       SUBSTR(RES_DATE,4,6) <> 'JAN-01'
AND                             TO_CHAR(RES_DATE,'MM/RRRR') =
                TO_CHAR(:RES_MASTER.RES_DATE,'MM/RRRR');
END IF;
```

- **DEV_ID ITEM:**

KEY-NEXT-ITEM

```
:RES_DETAILS.DEV_ID :=:RES_MASTER.DEV_ID;
:RES_DETAILS.DEV_NAME :=:RES_MASTER.DEV_NAME;

SELECT SYSDATE INTO :RES_MASTER.ENTRY FROM DUAL;
GO_ITEM('RES_DETAILS.CUSTOMER_ID');
```

- **BTN_ADD**

WHEN-BUTTON-PRESSED

```
BEGIN
    IF GET_ITEM_PROPERTY('BTN_ADD',LABEL) = '&ADD' THEN
SET_BLOCK_PROPERTY('RES_MASTER',INSERT_ALLOWED,PROPERTY_TRUE);

SET_BLOCK_PROPERTY('RES_MASTER',UPDATE_ALLOWED,PROPERTY_TRUE);

SET_BLOCK_PROPERTY('RES_DETAILS',INSERT_ALLOWED,PROPERTY_TRUE);
SET_BLOCK_PROPERTY('RES_DETAILS',UPDATE_ALLOWED,PROPERTY_TRUE);

GO_ITEM('RES_MASTER.VOUCHER_DATE');
 IF :SYSTEM.RECORD_STATUS != 'INSERT' THEN
CREATE_RECORD;
SELECT SYSDATE INTO :RES_MASTER.RES_DATE FROM DUAL;
END IF;
IF :SYSTEM.RECORD_STATUS != 'INSERT' THEN
CREATE_RECORD;
SELECT SYSDATE INTO :RES_MASTER.ENTRY FROM DUAL;
END IF;
SET_ITEM_PROPERTY('BTN_ADD'                    ,LABEL,'SA&VE');
SET_ITEM_PROPERTY('BTN_PREVIOUS'       ,ENABLED,PROPERTY_TRUE);
SET_ITEM_PROPERTY('BTN_NEXT'             ,ENABLED,PROPERTY_TRUE);
SET_ITEM_PROPERTY('BTN_FIRST'            ,ENABLED,PROPERTY_TRUE);
SET_ITEM_PROPERTY('BTN_LAST'             ,ENABLED,PROPERTY_TRUE);

ELSE
COMMIT_FORM;
IF :SYSTEM.FORM_STATUS = 'QUERY' THEN
SET_ITEM_PROPERTY('BTN_ADD'          ,LABEL          ,'&ADD');
SET_BLOCK_PROPERTY('RES_MASTER',INSERT_ALLOWED,PROPERTY_FALSE);
```

```
SET_BLOCK_PROPERTY('RES_MASTER',UPDATE_ALLOWED,PROPERTY_FALSE
);
SET_BLOCK_PROPERTY('RES_DETAILS',INSERT_ALLOWED,PROPERTY_FALSE);
SET_BLOCK_PROPERTY('RES_DETAILS',UPDATE_ALLOWED,PROPERTY_FALSE
);
GO_BLOCK('BTN_BLOCK');
GO_ITEM('BTN_BLOCK.BTN_ADD');
MESSAGE('BANK RECEIPT VOUCHER SUCCESSFULLY SAVED....');
ELSE
MESSAGE('BANK RECEIPT VOUCHER COULD NOT BE SAVED!');
MESSAGE('BANK RECEIPT VOUCHER COULD NOT BE SAVED!');
END IF;
 END IF;
END;
```

- **EXIT BUTTON :**

WHEN-BUTTON-PRESSED

```
EXIT_FORM;
```

- **BTN_NEXT BUTTON :**

WHEN-BUTTON-PRESSED

```
BEGIN
NEXT_RECORD;
END;
```

- **BTN_PREVIOUS BUTTON :**

WHEN-BUTTON-PRESSED

```
BEGIN
PREVIOUS_RECORD;
```

```
END;
```

- **BTN_LAST BUTTON :**

WHEN-BUTTON-PRESSED

```
BEGIN
GO_BLOCK('CHART');
LAST_RECORD;
GO_ITEM('BTN_FIRST');
END;
```

- **BTN_FIRST BUTTON :**

WHEN-BUTTON-PRESSED

```
BEGIN
GO_BLOCK('CHART');
FIRST_RECORD;
GO_ITEM('BTN_LAST');
END;
```

- **BTN_FIND BUTTON :**

WHEN-BUTTON-PRESSED

```
BEGIN
GO_ITEM('ID');
SELECT                    MAX(CONTRACT_NO)
INTO                              :ID
FROM  RES_MASTER;
END;
```

- **ENTRY ITEM :**

WHEN-NEW-FORM-INSTANCE

```
SELECT SYSDATE INTO :RES_MASTER.ENTRY FROM DUAL;
```

5.2 RES_DETAILS BLOCK :

WHEN-NEW-FORM-INSTANCE

```
SET_ITEM_PROPERTY('RES_DETAILS.CUSTOMER_ID'
,ENABLED,PROPERTY_FALSE);
SET_ITEM_PROPERTY('RES_DETAILS.CUSTOMER_NAME'
,ENABLED,PROPERTY_FALSE);
SET_ITEM_PROPERTY('RES_DETAILS.PRICE'
,ENABLED,PROPERTY_FALSE);

SET_ITEM_PROPERTY('RES_DETAILS.COMMISSION_AGENT'
,ENABLED,PROPERTY_FALSE);
```

- **PRICE ITEM :**

KEY-NEXT-ITEM

```
GO_ITEM('RES_DETAILS.PHONE');
```

- **ADDRESS ITEM :**

KEY-NEXT-ITEM

```
GO_ITEM('RES_DETAILS.E_MAIL');
```

- **E_MAIL ITEM :**

KEY-NEXT-ITEM

```
GO_ITEM('RES_MASTER.BTN_ADD');
```

- **PHONE ITEM :**

KEY-NEXT-ITEM

```
GO_ITEM('RES_DETAILS.ADDRESS');
```

5.2 FIND_BLK BLOCK :

- **BTN_OK BUTTON :**

WHEN-BUTTON-PRESSED

```
DECLARE
WHERECLAUSE VARCHAR2(300);
```

```
BEGIN
WHERECLAUSE := 'WHERE (RES_MASTER.CONTRACT_NO = '
                                        || :ID
                                        || ')' ;
CLEAR_FORM(NO_VALIDATE);
GO_BLOCK('RES_MASTER');
SET_BLOCK_PROPERTY('RES_MASTER',DEFAULT_WHERE,WHERECLAUSE );
EXECUTE_QUERY;
LAST_RECORD;
GO_ITEM('BTN_FIND');
END;
```

- **CANCEL BUTTON :**

WHEN-BUTTON-PRESSED

```
BEGIN
                GO_ITEM('RES_MASTER.RES_DATE');
END;
```

5.4 CONTRACT BLOCK:

- **FROM_CONTRACT ITEM :**

KEY-NEXT-ITEM

```
BEGIN
IF (:CONTRACT.FROM_CONTRACT IS NULL) THEN
SELECT MIN(CONTRACT_NO)
INTO   :CONTRACT.FROM_CONTRACT
FROM   RES_MASTER;
END IF;

:CONTRACT.TO_CONTRACT:=:CONTRACT.FROM_CONTRACT;
GO_ITEM('CONTRACT.CONTRACT_FROM');
```

```
END;
```

- **TO_CONTRACT ITEM :**

KEY-NEXT-ITEM

```
BEGIN
        IF (:CONTRACT.FROM_CONTRACT IS NULL) THEN
        SELECT MAX(CONTRACT_NO)
        INTO   :TO_CONTRACT
        FROM RES_MASTER;
END IF;
END;
```

- **BTN_OK BUTTON :**

WHEN-BUTTON-PRESSED

```
DECLARE
            WHERECLAUSE VARCHAR2(300);
BEGIN
WHERECLAUSE := 'WHERE (ERS_MASTER.CONTRACT_NO BETWEEN '

                    || :CONTRACT.FROM_CONTRACT
                    || ' AND '
                    || :CONTRACT.TO_CONTRACT
                    || ')' ;
CLEAR_FORM(NO_VALIDATE);
 GO_BLOCK('GL_MASTER');
SET_BLOCK_PROPERTY('GL_MASTER',DEFAULT_WHERE,WHERECLAUSE );
EXECUTE_QUERY;
LAST_RECORD;
GO_ITEM('CONTRACT');
END;
```

- **CANCEL BUTTON :**

WHEN-BUTTON-PRESSED

```
Go_item('GL_MASTER.VOUCHER_NO');
SET_ITEM_PROPERTY('BTN_BLOCK.BTN_CLEAR',ENABLED,PROPERTY_FALSE)
;
```

Source Code:

```
CREATE TABLE CHART
  (ID                    NUMBER(10,0) NOT NULL,
  NAME                    VARCHAR2(60) NOT NULL,
  LEVE                    VARCHAR2(30),
  CHART_DATE              DATE,
 CONSTRAINT CHART_IDNAME_PK PRIMARY KEY (ID, NAME))
/
CREATE TABLE GL_MASTER
  (VOUCHER_NO             VARCHAR2(8) NOT NULL,
  VOUCHER_DATE            DATE NOT NULL,
  DOCUMENT_TYPE_ID        VARCHAR2(6) NOT NULL,
  RECORD_DATE             VARCHAR2(30),
CONSTRAINT GL_MASTERIDNAMENO_PK PRIMARY KEY (VOUCHER_NO,
VOUCHER_DATE,
 DOCUMENT_TYPE_ID))
/
CREATE TABLE RES_MASTER
  (CONTRACT_NO            NUMBER(10,0) NOT NULL,
  DEV_ID                  NUMBER(10,0) NOT NULL,
  DEV_NAME                VARCHAR2(35) NOT NULL,
  RES_DATE                DATE NOT NULL,
  ENTRY                   DATE,
CONSTRAINT RES_MASTERDEVIDNAME_PK PRIMARY KEY (CONTRACT_NO,
DEV_ID, DEV_NAME,RES_DATE))
```

```
/
CREATE TABLE GL_DETAILS
  (VOUCHER_NO              VARCHAR2(8),
  VOUCHER_DATE             DATE,
  DOCUMENT_TYPE_ID         VARCHAR2(6),
  CUSTOMER_ID              NUMBER(10,0),
  CUSTOMER_NAME            VARCHAR2(60),
  PARTICULAR1              VARCHAR2(150),
  DR                  NUMBER(15,2),
  CR                  NUMBER(15,2),
  CHQ_DD_TT                VARCHAR2(20),
  FOR_DATE                 DATE,
  DEV_ID                   NUMBER(10,0),
  DEV_NAME                 VARCHAR2(35),
  CONTRACT_NO              NUMBER,
CONSTRAINT GL_DETAILS_ VOUCHER_NO_PK  PRIMARY KEY (VOUCHER_NO
))
/
CREATE TABLE PAY_DETAILS
  (VOUCHER_NO              VARCHAR2(8),
  VOUCHER_DATE             DATE,
  DOCUMENT_TYPE_ID         VARCHAR2(6),
  CUSTOMER_ID              NUMBER(10,0),
  CUSTOMER_NAME            VARCHAR2(60),
  PARTICULAR1              VARCHAR2(150),
  DR                  NUMBER(15,2),
  CR                  NUMBER(15,2),
  CHQ_DD_TT                VARCHAR2(20),
  FOR_DATE                 DATE,
  DEV_ID                   NUMBER(10,0),
  DEV_NAME                 VARCHAR2(35),
  CONTRACT_NO              NUMBER,
```

```
CONSTRAINT PAY_DETAILS_VOUCHER_NO_PK PRIMARY KEY
(VOUCHER_NO))
/
CREATE TABLE PAY_MASTER
  (VOUCHER_NO              VARCHAR2(8),
  VOUCHER_DATE             DATE,
  DOCUMENT_TYPE_ID         VARCHAR2(6),
  RECORD_DATE              DATE)
CONSTRAINT PAY_MASTER_VOUCHERNO_PK PRIMARY KEY (VOUCHER_NO))
/
CREATE TABLE RES_DETAILS
  (CONTRACT_NO             NUMBER,
  DEV_ID                   NUMBER(10,0),
  DEV_NAME                 VARCHAR2(35),
  CUSTOMER_ID              NUMBER(10,0),
  CUSTOMER_NAME            VARCHAR2(60),
  RES_DATE                 DATE,
  PRICE                    NUMBER(10,0) NOT NULL,
CONSTRAINT RES_DETAILS_CONTRACT_NO_PK PRIMARY
KEY(CONTRACT_NO))
/
ALTER TABLE CHART
ADD CONSTRAINT CHART_ID_UK UNIQUE (ID))
/
ALTER TABLE CHART
ADD CONSTRAINT CHART_NAME_UK UNIQUE (NAME))
/
ALTER TABLE RES_MASTER
ADD CONSTRAINT RES_MASTER_DEV_ID_IK UNIQUE (DEV_ID, DEV_NAME))
/
ALTER TABLE GL_DETAILS
```

```
ADD CONSTRAINT GL_DETAILS_FK_MASTER FOREIGN KEY (VOUCHER_NO,
VOUCHER_DATE, DOCUMENT_TYPE_ID)
REFERENCES GL_MASTER
(VOUCHER_NO,VOUCHER_DATE,DOCUMENT_TYPE_ID))
/
ALTER TABLE RES_DETAILS
ADD CONSTRAINT RES_DETAILS_CUSTOMER_IDNAME_FK FOREIGN KEY
(CUSTOMER_ID, CUSTOMER_NAME)REFERENCES CHART (ID,NAME))
/
ALTER TABLE RES_DETAILS
ADD CONSTRAINT RES_DETAILS_DEVIDNAME_FK FOREIGN KEY
(CONTRACT_NO, DEV_ID, DEV_NAME, RES_DATE)REFERENCES RES_MASTER
(CONTRACT_NO,DEV_ID,DEV_NAME,RES_DATE)
/
ALTER TABLE GL_DETAILS
ADD CONSTRAINT GL_DETAILS_CONTRACTNO_FK
 FOREIGN KEY(CONTRACT_NO)
REFERENCES RES_DETAILS(CONTRACT_NO)
/
ALTER TABLE PAY_DETAILS
ADD CONSTRAINT PAY_DETAILS_CONTRACTNO_FK  FOREIGN
KEY(CONTRACT_NO) REFERENCES RES_DETAILS(CONTRACT_NO)
/
ALTER TABLE GL_DETAILS_ID_NAME_FK FOREIGN
KEY(CUSTOMER_ID,CUSTOMER_NAME) REFERENCES CHART(ID,NAME)
/
```

Conclusion

In this project a computer program for an estate egency office using oracle8i and developer6i was designed.

The oracle8i was used to create the database. The programs forms were made via the developer6i.

The field database has evolved from ordinary files to a complete database management system. There are over thousand RDBMS software's now available in the world.

Some has advantages and other lack behind in many respects. Amongst these is Oracle which has tremendous amount of capability in handling database Applications, from simple desktop to mainframes.

The Oracle RDBMS is an enormous environment with unlimited potential. The advantages of using Oracle as a RDBMS is that we can get voice mail, wireless access, email server, enterprise file sharing, can be used on Linux, Unix and Windows operating systems, safe, cost lesser than most other RDBMS, and is less prone to viruses.

In the project presents theoretical information about data models design, and the properties of DBMS it also describes software designed to manipulate the activities of the estate agency sell company.

References

[1]. Jolene Morrison, Mike Morrison, "Enhanced Guide to Oracle8i"

[2]. Data C.J, " An Introduction To Database System " .1997

[3]. Nilesh Shah, " Database Systems Using Oracle" .

[4]. Oracle University," Introduction to SQL and PL/SQL "

[5]. ASP free from the World Wide Web *"http//www.aspfree.com"*

[6]. Arab oracle from the World Wide Web *"http://www.araboracle.com"*

[7]. Oracle Corporation from the World Wide Web "http://*www.oracle.com*"

[8]. Oracle University "*http://education.oracle.com*"