DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

COURSE NAME : Database Management System Lab

COURSE CODE : 17CSCC22

SEM/YEAR : III/II

FACULTY IN-CHARGE : Mr.S.Muthuselvan

Ex.No.01 DDL COMMANDS

Aim:-

To use simple DDL commands.

Introduction:-

SQL statements are divided into two major categories DDL(Data Definition Language) and DML(Data Manipulation Language). DDL allows you perform the basic operation such as creating tables or views, altering table, dropping tables or views. DML allows you to insert, update, delete, and selecting rows from the table in the database.

Creating Table:-

Description

The create table command is used to create a table in SQL. The table consists of various fields that may be number or characters.

Syntax:-

Create table <table-name> (colunm1 type, column type …)

Example:-

SQL>create table sample (rollno number(10), class varchar2(5));

Creating View:-

Description

The create view command is used to create a view in SQL. The view consists of columns that are specified in the select statement.

Syntax:-

Create view <view-name> as (select statement);

Example:-

SQL>create table sample1 as select * from sample;

Desc command:-

Description

The command is used to view the columns in the table or view, and their corresponding data types.

Syntax:-

desc <table-name> / <view-name>;

Example:-

SQL> desc sample;

Alter table command:-

Description

This command is used either to add a column or modify the data type of an existing column.

Syntax:-

Alter table <table-name> add (column type);
Alter table <table-name> modify (column type);

Example:-

SQL> alter tablesample add(year number(5));
SQL> alter table sample modify(class char);

Drop command:-

Description

This command is used to deletes the records and table structure or view structure from the database.

Syntax:-

Drop table <table-name>;
Drop view <view-name>;

Example:-

```
SQL> drop table sample;
SQL> drop view sample1;
```

Result:-

Thus the above various DDL (Data Definition Language) commands are studied and executed successfully.

Ex.No.02 DML, TCL, AND DCL COMMANDS

Aim:-

To use simple DML,TCL, and DCL commands.

Introduction:-

Data Manipulation Language (DML) allows the users to query and manipulate the data in existing schema of object. It allows following data to insert, delete, update and recovery data in schema object.

Transaction Control Language (TCL) manages the changes made by the DML commands. Commands are commit, rollback, and savepoint.

Data Control Language (DCL) allows the user to perform the operation like granting and revoking the privileges using grant and revoke command.

DML COMMANDS

Insert command:-

Description:-

values can be inserted into the table using 'Insert' command.

Syntax:-

Insert into <table-name> values (value1, value2…);

Example:-

SQL> insert into college values(404072,'Mahesh',22,'cse',60000);

Update command:-

Description:-

This allows the user to update the particular column value using the WHERE clause condition.

Syntax:-

Update <table-name> set <col1=value…..> where <col=value>;

Example:-

SQL> update college set fees=50000 where stu_rno=404072;

Delete command:-

Description:-

This command allows you to delete a particular column value using WHERE clause condition.

Syntax:-

Delete from <table-name> where <condition>;

Example:-

SQL> delete from college where stu_rno=404072;

Select command:-

Description:-

This command is used to select the records from the table or view.

Syntax:-

Select <col1…..>/* from <table-name>;

Example:-

SQL> select * from college; // select all records.

SQL> select name from college; // select the particular column.

The various select statements:-

I) Select using WHERE clause

Syntax:-

Select <colname1……>/* from <table-name> where <condition>;

Example:-

SQL> select stu_rno, name from college where age=20;

II) Select using Pattern Matching:-

Syntax:-

Select <colname1……>/* from <table-name> where <condition> like <pattern>;

Example:-

SQL> select name from college where name like 'm%';

III) Select ORDER BY clause:-

Syntax:-

Select <colname1……>/* from <table-name> ORDER BY <colname>;

Example:-

SQL> select stu_rno, name from college order by stu_rno;

IV) Select using logic operators:-

Syntax:-

Select <colname1……>/* from <table-name> where (colname <condition>values);

Example:-

SQL> select * from college where fees!=40000;

V) Select using set operators:-

Syntax:-

Select <colname1……>/* from <table-name> <minus/ union/ union all/ intersect> (select statement);

Example:-

SQL> select stu_rno from college union select stu_rno from hostel;
SQL> select stu_rno from college union all select stu_rno from hostel;
SQL> select stu_rno from hostel minus select stu_rno from college;
SQL> select stu_rno from hostel intersect select stu_rno from college;

VI) Select using joins and sub queries:-

Syntax:-

Select <table2.colname,table1.colname……> from <table1,table2> where (table2.colname <condition> table1.colname);

Select * from <table1,table2>where table2.colname=any(select statement);

Example:-

SQL> select college.stu_rno, hostel.name from college, hostel where college.stu_rno=hostel.stu_rno;

TCL COMMANDS

Commit command:-

Description:-

The commit statement explicitly makes permanent any changes that to the database during the current transaction. A commit also makes those changes visible to other users. So the commit statement ends the current transaction.

Syntax:-

Commit;

Example:-

SQL> commit;

Savepoint command:-

Description:-

Savepoint names and marks the current point in the processing of a transaction. Using savepoint with rollback, we can undo a part of a transaction instead of the whole transaction. The maximum number of savepoint per transaction is 5.

Syntax:-

SAVEPOINT savepoint-name;

Example:-

SQL> savepoint s1;

Rollback command:-

Description:-

The rollback statement does exactly opposite to the commit. It ends the current transaction and discards any changes made during that transaction after the commit or savepoint.

Syntax:-

Rollback [to [SAVEPOINT] savepoint-name];

Example:-

SQL> rollback to s1;

SAVEPOINT is optional and is used to rollback a partial transaction at the specified savepoint.

DCL COMMANDS

Grant command:-

Description:-

By using the grant command, you can grant any system the privileges or role to another role. The 'WITH ADMIN OPTION' clause permits the grantee to after the privilege or role on other users or roles. The grantor can also revoke a role from a users as well. We can also specify ALL to grant all the privileges.

Syntax:-

Grant privileges ON <object-name> TO <user-name>;

Example:-

SQL> grant select, insert, delete on college to mohan;

Revoke command:-

Description:-

Privileges granted can be taken away by the REVOKE command. This command is almost similar to that of the grant command in its format.

Syntax:-

Revoke privileges ON <object-name> FROM <user-name>;

Example:-

SQL> revoke insert, delete on college from mohan;

Result:-

Thus the above DML, TCL, and DCL commands are studied and executed successfully.

Ex.No.03 PL/SQL FOR CURSOR

Aim:-

Introduction:-

Cursor is a memory area created by oracle which is used to store the table data temporarily while we manipulate them in Oracle.

Description:-

i. Create a table for student and insert the appropriate values in the table database.
ii. Then create a cursor with cursor name, and this cursor name refers all the fields of the student table.
iii. Then declare the required variables.
iv. Now start the definition of the cursor using begin statement and open the cursor.
v. Check whether the cursor opened or not. If it is opened display the message “Cursor Opened…..”
vi. Define the loop for encountered the student table and fetch the required output value into the table.
vii. Then count the fetched values in the student table and print the output in the display monitor.
viii. Finally close the cursor.

Before Creating PL/SQL:-

Oracle Table:-

```
SQL> create table student(regno number(7),dbms number(3),ethics number(3),dpsd number(3),total
number(3),average number(3));
SQL> insert into student(regno,dbms,ethics,dpsd) values (&regno,&dbms,&ethics,&dpsd);
SQL> update student set total=0;
SQL> update student set average=0;
```

Default Table Contents:-

```
SQL> select * from student;
```

REGNO	DBMS	ETHICS	DPSD	TOTAL	AVERAGE
404072	90	90	90	0	0
404071	95	95	95	0	0
404073	85	85	85	0	0
404075	88	88	88	0	0
404074	92	92	92	0	0

PL/SQL:-

```
SQL> set serveroutput on
SQL> declare
  2  cursor stu is select regno,dbms,ethics,dpsd from student;
  3  rno  student.regno%type;
  4  m1  student.dbms%type;
  5  m2  student.ethics%type;
  6  m3 student.dpsd%type;
  7  tot student.total%type;
  8  av student.average%type;
  9  begin
 10  open stu;
 11  if stu%ISOPEN then
 12  dbms_output.put_line('Cursor Opened…');
 13       loop
```

```
 14          fetch stu into rno,m1,m2,m3;
 15          exit when stu%NOTFOUND;
 16          tot:=(m1+m2+m3);
 17          av:=tot/3;
 18          update student set average=av where regno=rno;
 19          update student set total=tot where regno=rno;
 20       end loop;
 21  dbms_output.put_line('Total Records :' ||stu%ROWCOUNT);
 22  close stu;
 23  dbms_output.put_line('Cursor Closed…');
 24   end if;
 25  end;
 26  /

Cursor Opened…
Total Records : 5
Cursor Closed…
SQL>
```

Output:-

```
SQL> select * from student;
REGNO      DBMS       ETHICS     DPSD       TOTALAVERAGE
---------- ---------- ---------- ---------- ---------- ----------
    404072         90         90         90        270         90
    404071         95         95         95        285         95
    404073         85         85         85        255         85
    404075         88         88         88        264         88
    404074         92         92         92        276         92
SQL>
```

Result:-

Thus the above cursor in PL/SQL program was performed and verified successfully.

Ex.No.04 PL/SQL FOR TRIGGERS

Aim:-

Introduction:-

A database triggers is a stored procedure that is implicitly executed when an insert, update or delete statement is issued against the associated table. Database triggers can be used for the following purposes.

i. To generate data automatically
ii. To prevent invalid transaction
iii. To enforce complex security authorizations
iv. To enforce referential integrity
v. To maintain synchronous table replicates
vi. To gather statistics on table access

Description:-

i. Create two tables named voters_master and voters_passed with appropriate data types to illustrate the functioning of the triggers.
ii. Insert the required values in the voters_master table.
iii. Write a proper PL/SQL block to delete the details of the voters whenever a row is inserted in the table voters_passed.
iv. Select the table and see now, the particular row has been deleted.

Before Creating PL/SQL:-

Oracle Table:-

```
SQL> create table voters_master(voter_id number(5),name varchar2(30), ward_no number(4),dob
date,address varchar2(30), primary key(voter_id, ward_no));
SQL> create table voters_passed(voter_id number(5),ward_no number(4), primary
key(voter_id,ward_no));
```

Default Table Contents:-

```
SQL> select * from voters_master;
 VOTER_ID NAME           WARD_NO DOB       ADDRESS
--------- -------------- ------- --------- ----------
        1 saradha p            1 12-MAR-67 nagarcoil-4
        2 janarthan m          5 01-FEB-78 kattur
        3 vasu l               5 31-DEC-79 vadasery
SQL> select * from voters_passed;
no rows selected.
```

Trigger Function:-

```
SQL> create or replace trigger vote_trig after insert on voters_passed for each  row
  2  declare
  3  v_id number(5);
  4  w_id number(4);
  5  begin
  6  v_id:=&voter_id;
  7  w_id:=&ward_no;
  8  delete from voters_master where voter_id=v_id and ward_no=w_id;
  9  end;
 10 /
Trigger created.
```

Output:-

Insert value into voters_passed:-

```
SQL> insert into voters_passed values(1,1);
```

After Executing Trigger:-

```
SQL> select * from voters_master;
 VOTER_ID    NAME           WARD_NO    DOB            ADDRESS
-----------  -------------  ---------  -------------  -------------
        2    janarthan m            5  01-FEB-78      kattur
        3    vasu l                 5  31-DEC-79      vadasery
SQL> select * from voters_passed;
VOTER_ID     WARD_NO
-----------  ---------
        1            1
```

Result:-

Thus the above triggers in PL/SQL program was performed and verified successfully.

Ex.No.05(A) PL/SQL FOR PROCEDURE

Aim:-

Introduction:-

A procedure is a subprogram the performs a specific action and accepts more than one argument and returns more than one value.

Description:-

i. Create a table named order_master and insert the appropriate values in the database table.
ii. Now create the procedure named by proc with the required arguments
iii. Select the item code, ordered quantity and delivered quantity from the table and store them in the declared variables.
iv. Check if delivered quantity is less than the ordered quantity.
v. If delivered quantity < ordered quantity, return item code to the called procedure.
vi. In the calling procedure declare the two variables and get the input for one, then call the procedure with the arguments.
vii. Finally print the output statement in the display monitor.

Before Creating PL/SQL:-

Oracle Table:-

```
SQL> create table order_master(qty_ord number(5),qty_del number(5),itemcode number(3),ordno
number(2));
```

Default Table Contents:-

```
SQL> select * from order_master;
QTY_ORD     QTY_DEL     ITEMCODE ORDNO
---------- ---------- ---------- ----------
       100         75        101          1
        70         70        102          2
```

Create Procedure:-

```
SQL> create procedure proc(or_no in number,b in out number) is
  2  qtyord number;
  3  qtydel number;
  4  code number;
  5  begin
  6  select qty_ord,qty_del,itemcode into qtyord,qtydel,code from order_master where ordno=or_no;
  7  if qtydel<qtyord then
  8  b:=code;
  9  end if;
 10  end;
 11  /
Procedure created.
```

Calling Procedure:-

```
SQL> declare
  2  a number;
  3  b number;
  4  begin
  5  a:=&enter_ordno;
  6  proc(a,b);
  7  if b>0 then
  8  dbms_output.put_line('The item code '||to_char(b)||' has to be delivered');
  9  else
 10  dbms_output.put_line('The item has been delivered');
```

```
 11  end if;
 12  end;
13 /
```

Output:-

```
SQL> set serveroutput on
SQL> /
Enter value for enter_ordno: 2
old 5: a:=&enter_ordno;
new  5: a:=2;

The item has been delivered.
PL/SQL procedure successfully completed.

SQL> /
Enter value for enter_ordno: 1
old 5: a:=&enter_ordno;
new  5: a:=1;
The item code 101 has to be delivered.
PL/SQL procedure successfully completed.
```

Result:-

Thus the above procedure in PL/SQL program was performed and verified successfully.

Ex.No.05(B) PL/SQL FOR FUNCTIONS

Aim:-

Introduction:-

A function is a subprogram that accepts more than one argument and returns only one value. In function, RETURN keyword is used to the give a value to the PL/SQL program. Function can be called as many times as we need.

Description:-

i. Create a table named order_master and insert the appropriate values in the database table.
ii. Now create the function named by items with the required arguments
iii. Select the ordered quantity and delivered quantity from the table and store them in the declared variables.
iv. Check if delivered quantity is less than the ordered quantity.
v. If delivered quantity < ordered quantity, return 0, else return 1 to the called procedure.
vi. In the calling function declare the two variables and get the input for one, then call the function with the arguments.
vii. Finally print the output statement in the display monitor.

Before Creating PL/SQL:-

Oracle Table:-

```
SQL> create table order_master(qty_ord number(5),qty_del number(5),itemcode number(3),ordno
number(2));
```

Default Table Contents:-

```
SQL> select * from order_master;
QTY_ORD     QTY_DEL     ITEMCODE ORDNO
---------- ---------- ---------- ----------
       100         75        101          1
        70         70        102          2
```

Create Function:-

```
SQL> create function items(it number)return number is args number;
  2  qtyord number;
  3  qtydel number;
  4  begin
  5  select qty_ord,qty_del into qtyord,qtydel from order_master where  ordno=it;
  6  if qtydel<qtyord then
  7  args:=0;
  8  return args;
  9  else
 10  args:=1;
 11  return args;
 12  end if;
 13  end;
 14  /
Function created.
```

Calling Function:-

```
SQL> declare
  2  a number;
  3  b number;
  4  begin
  5  a:=&enter_ordno;
```

```
 6  b:=items(a);
 7  if b=0 then
 8  dbms_output.put_line('The item has to be delivered.');
 9  else
10  dbms_output.put_line('The item has been delivered.');
11  end if;
12  end;
13 /
```

Output:-

```
SQL> set serveroutput on
SQL> /
Enter value for enter_ordno: 1
old 5: a:=&enter_ordno;
new 5: a:=1;
The item has to be delivered.
PL/SQL procedure successfully completed.

SQL> /
Enter value for enter_ordno: 2
old 5: a:=&enter_ordno;
new  5: a:=2;
The item has been delivered.
PL/SQL procedure successfully completed.
```

Result:-

Thus the above function in PL/SQL program was performed and verified successfully.

Ex.No.06 EMBEDDED SQL

Aim:-

To write a program in java to retrieve the data from the database.

Hardware Requirements:-

1. Pentium III 600 MHz.
2. 256 Mb RAM
3. 14 inch color monitor.
4. 101 keys keyboard
5. 20 GB hard disk
6. 3 button mouse

Software Requirements:-

- JDK 1.3.
- MS-ACCESS.

Concept:-

Definition:-

The SQL structures permitted in the host language (i.e. The language in which the SQL queries can be embedded) are called as EMBEDDED SQL.

Features:-

1. Not all Queries can be expressed in SQL. These Queries can be embedded in languages like C, JAVA, or COBOL that cannot be resolved in SQL.
2. SQL does not support actions like printing a report, interacting with a user, or sending the results of a query to a graphical user interface. By Embedding the SQL in the host language these actions can be performed.

Algorithm:-

1. Start.
2. Create a Student table in MS-ACCESS.
3. Create a Data Source Name using MS-ACCESS Driver.
4. Establish a connection from Java MS-ACCESS Database using JDBC:ODBC Driver
5. Embed the SQL statements like Insert and Select statements in the JAVA program.
6. Compile and Run the JAVA program.
7. Stop.

Program:-

```
import java.io.*;
import java.sql.*;
public class jdeg
{
public static void main(String args[])throws IOException
{
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
String rno,name,dept,mark;
System.out.println("enter the value(rno,name,dept,mark)to be inserted");
rno=br.readLine();
name=br.readLine();
mark=br.readLine();
dept=br.readLine();
try
```

```
{
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connection con=DriverManager.getConnection("jdbc:odbc:student");
Statement st=con.createStatement();
st.executeUpdate("insert into student values('"+rno+"','"+name+"','"+dept+"','"+mark+"')");
ResultSet rs=st.executeQuery("select * from student");
while(rs.next())
{
System.out.println(rs.getString("rno"));
System.out.println(rs.getString("name"));
System.out.println(rs.getString("dept"));
System.out.println(rs.getString("mark"));
}
}
catch(Exception e)
{
System.out.println(e);
}
}
}
```

Output:-

```
D:\jdk1.3\bin>javac jdeg.java
D:\jdk1.3\bin>java jdeg
enter the values(rno,name,dept,marks) to insert into the table
3
aravind
cse
90
RNO     NAME    DEPT    MARKS
1       Achelal                 cse       90
2       Ahamath       cse      100
3       aravind       cse       90
```

IN MS-ACCESS:

	RollNo	Name	Dept	Mark
	1	Achelal	cse	90
	2	Ahamath	cse	100
▶	3	aravind	cse	90

Result:-

Thus a program is written to retrieve the data from the database.

Ex.No.07 NORMALIZATION

Aim:-

To design a database using E-R diagram and normalization.

Hardware Requirements:-

1. Pentium III 600 MHz.
2. 256 Mb RAM
3. 14 inch color monitor.
4. 101 keys keyboard
5. 20 GB hard disk
6. 3 button mouse

Software Requirements:-

- Oracle 8i server.

CONCEPT AND DEFINITION

Normalization:-

Normalization is the analysis of functional dependencies between attributes/data itemsa of user views. It reduces a complex user view to a set of small and stable subgroups of the fields amd relations. This process helps to design a logical data model known as conceptual data model.

There are different normal forms

1. First normal form (1NF)
2. Second Normal Form (2NF)
3. Third Normal Form (3NF)

First Normal Form(1NF)

1NF states that the domain of an attribute must include only atomic (simple,indivisible) values and that value of any attribute in a tuple must be a single value from the domain of that attribute.Hence 1NF disallows multi-valued attributes,composite attributes.It disallows “relations within relations”.

Second Normal Form(2NF)

A relation is said to be in 2NF if it is already in 1NF and it has no partial dependency.2NF is based on the concept of full functional dependency.

A functional dependency(FD) $X \rightarrow Y$ is full functional dependency if $(X-(A)) \rightarrow Y$ does not hold dependency any more if $A \varepsilon X$.

A functional dependency $X \rightarrow Y$ is partial dependency if A can be removed which does not affect the dependency i.e. $(X-(A)) \rightarrow Y$ holds.

A relation is in 2NF if it is in 1NF and every non-primary key attribute is fully and functionally dependent on primary key.

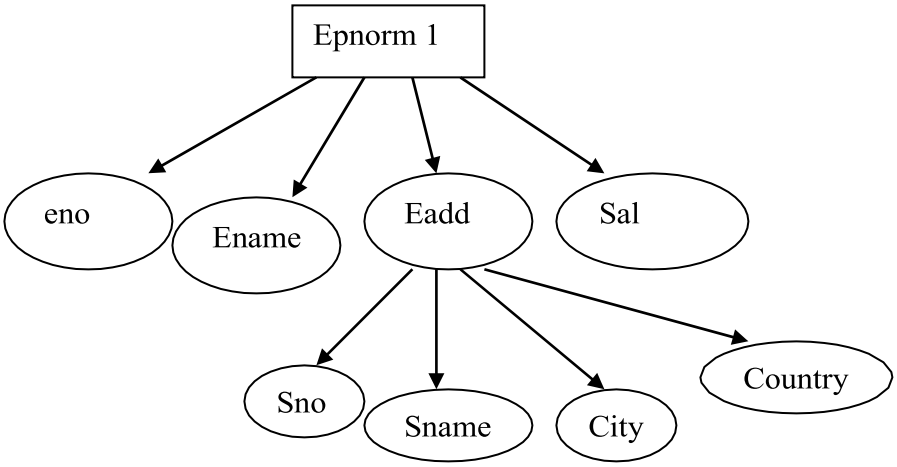
A relation in the 1NF will be in the 2NF if one of the following conditions is satisfied:
The primary key consist of only one attribute.
No non-key attribute exist in relation i.e. all the attributes in the relation are components of the primary key.
Every non-key attribute is FD on full set of primary key attributes.

Third Normal Form(3NF)

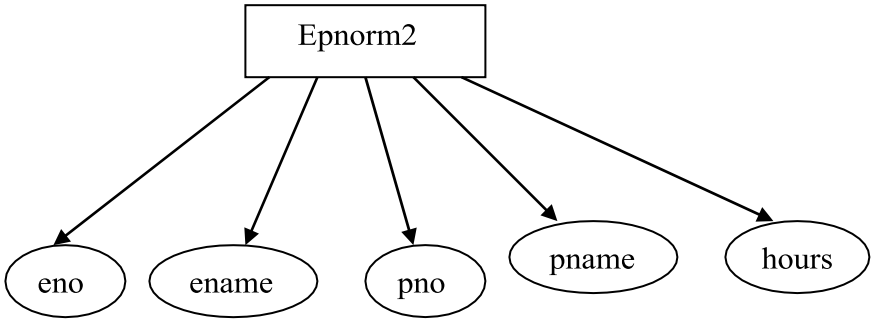
A relation is said to be in 3NF if it is already in 2NF and it has no transitive dependency.

A FD X→Y in a relation schema R is a transitive dependency if there is a set of attributes Z that is neither a candidate key nor a subset of any key of the relation and both X→Z and Z→Y hold.

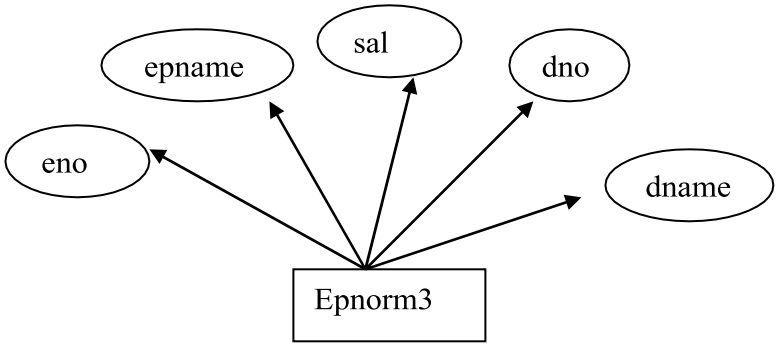
E-R DIAGRAMS
First Normal Form



Second Normal Form



Third Normal Form



ALGORITHM
FIRST NORMAL FORM

1. Create a type address for the composite address attribute.
   create or replace type address as object

(sno number(3),sname varchar2(30),city varchar2(20),country varchar2(20));

2. Create a employee table with the following fields eno,ename,eadd,sal and having the eno as the primary key.

create table emp(eno number(3) primary key,ename varchar2(20),eadd

address,sal number(7,2));

SQL> desc employees

Name	Null?	Type
ENO	NOT NULL	NUMBER(3)
ENMAE		VARCHAR2(20)
EADD		ADDR
SAL		NUMBER(7,2)

3. Insert values in the emp table

insert into emp values(&eno,'&ename',address

(&sno,'&sname','&city','&country'),&sal);

SQL> insert into employees
values(&eno,'&enmae',addr(&sno,'&sname','&city','&state'),&sal);
Enter value for eno: 001
Enter value for enmae: anbu
Enter value for sno: 12
Enter value for sname: Ist street
Enter value for city: chennai
Enter value for state: tamilnadu
Enter value for sal: 10000
old 1: insert into employees
values(&eno,'&enmae',addr(&sno,'&sname','&city','&state'),&sal)
new 1: insert into employees values(001,'anbu',addr(12,'Ist
street','chennai','tamilnadu'),10000)

1 row created.

5. Emp table is not in the first normal form since it has a composite attribute. So it has been normalized to first normal form.

Before Normalization

Eno	Ename	Eadd	Sal

Normalization To First Normal Form

1. creating the en11 table with eno,ename and esal from emp;
   create table en11 as select eno,ename,sal from emp;
2. creating the table en12 with eno and eadd from emp
   create table en12 as select eno,eadd from emp;
3. altering the table en11 with primary key on eno
   alter table en11 add constraint k1 primary key(eno);
4. altering the table en12 with foreign key on eno with reference from en11
   alter table en12 add constraint c1 foreign key(eno) references en11(eno)

After Normalization

En11

Eno	Ename	Sal

en12

Eno	Eadd

SECOND NORMAL FORM

1. Creating the emp project table
   SQL> create table epnorm2(eno number(3) primary key,pno number(3) unique,pname varchar2(20),hours number(3),ename varchar2(20))
2. checking the table
   SQL> desc epnorm2

Name	Null?	Type
ENO	NOT NULL	NUMBER(3)
PNO		NUMBER(3)
PNAME		VARCHAR2(20)
HOURS		NUMBER(3)
ENAME		VARCHAR2(20)

3. inserting the values in the table;
   insert into epnorm2 values(&eno,&pno,'&pname',&hours,'&ename')

example of insertion

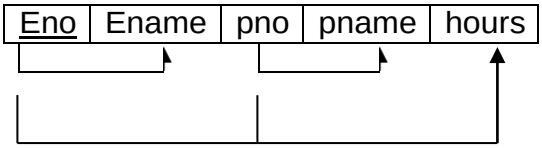
```
SQL> insert into epnorm2 values(&eno,&pno,'&pname',&hours,'&ename')

Enter value for eno: 1
Enter value for pno: 101
Enter value for pname: Sharma
Enter value for hours: 75
Enter value for ename: Aravind
old   1: insert into epnorm2 values(&eno,&pno,'&pname',&hours,'&ename')
new 1: insert into epnorm2 values(1,101,'Sharma',75,'Aravind')

1 row created.
```

4. To normalize the above table to second normal form.

Before Normalization



Normalization To Second Normal Form

a) create the table en21 with eno,ename from the table epnorm2
   SQL> create table en21 as select eno,ename from epnorm2;
   Table created.
b) Create the table en22 with pno,pname from table epnorm2
   SQL> create table en22 as select pno,pname from epnorm2;
   Table created.

c) Alter table en21 with a primary key constraint on eno.
SQL> alter table en21 add constraint en21 primary key(eno);
Table altered.

d) Alter table en22 with a primary key constraint on pno.
SQL> alter table en22 add constraint en22 primary key(pno);
Table altered.

e) Create table en23 with eno,pno and hours from the table epnorm2.
SQL> create table en23 as select eno,pno,hours from epnorm2;
Table created.

f) Alter table en23 with a foreign key on eno with references on eno from en21
SQL> alter table en23 add constraint en231 foreign key(eno) references en21(eno);
Table altered.

g) Alter table en23 with foreign key on pno with references on pno from en22
SQL> alter table en23 add constraint en232 foreign key(pno) references en22(pno);
Table altered.

After Normalization

En21

Eno	**E NAME**

En22

PNO	Pname

En23

ENO	Pno	Hours

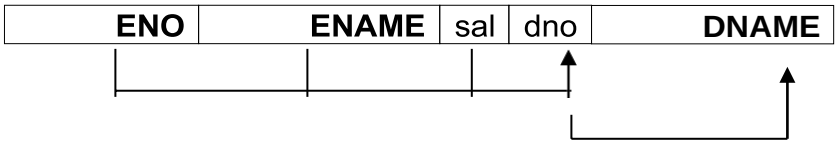
THIRD NORMAL FORM

1. create the table emp_dept with eno,ename,sal,dno and dname as attributes.
   SQL> create table emp_dept(eno number(3) primary key,ename varchar2(20),sal number(7,2),dno number(3),dname varchar2(20));
   Table created.
2. insert the values in the table.
   SQL> insert into emp_dept values(&eno,'&ename',&sal,&dno,'&dname');
   Example record
   SQL> insert into emp_dept values(&eno,'&ename',&sal,&dno,'&dname')
   Enter value for eno: 1
   Enter value for ename: Mr. Brown
   Enter value for sal: 10000
   Enter value for dno: 1
   Enter value for dname: cse
   old 1: insert into emp_dept values(&eno,'&ename',&sal,&dno,'&dname')
   new 1: insert into emp_dept values(1,'Mr. Brown',10000,1,'cse')

   1 row created.
3. The relation does not satisfy the 3rd normal form since dno is not a primary key. So normalization is done for the third normal form.

Before Normalization

Empdept



Normalization To Third Normal Form

a) Create table en31 with eno,ename,sal,dno from the table emp_dept.
SQL> create table en31 as select eno,ename,sal,dno from emp_dept
Table created.

b) Create table en32 with dno,dname from the table emp_dept.
SQL> create table en32 as select dno,dname from emp_dept;
Table created.

c) Alter the table en31 with the constraint primary key on eno.
SQL> alter table en31 add constraint en31 primary key(eno);
Table altered.

d) Alter table en32 with the constraint primary key on dno;
SQL> alter table en32 add constraint en32 primary key(dno);
Table altered.

e) Alter table en31 with the constraint foreign key on dno with reference from dno in en32
SQL> alter table en31 add constraint en311 foreign key(dno) references en32(dno)
Table altered.

After Normalization

En31

Eno	ENAME	sal	dno

En32

Dno	Dname

INPUT / OUTPUT

FIRST NORMAL FORM

```
SQL> select * from emp;
ENO   ENAME      EADD(SNO, SNAME, CITY, COUNTRY)            SAL
  1     Ahamath    ADDRESS(45, 'cross street', 'Chennai', 'India')      50000
  2    Bhaylool    ADDRESS(46, 'Cross Street', 'Chennai', 'India')     90000
SQL> select * from en11;
     ENO ENAME                SAL
       1    Ahamath              50000
       2    Bhaylool             90000
SQL> select * from en12;
    ENO      EADD(SNO, SNAME, CITY, COUNTRY)
       1      ADDRESS(45, 'cross street', 'Chennai', 'India')
       2      ADDRESS(46, 'Cross Street', 'Chennai', 'India')
```

SECOND NORMAL FORM

```
SQL> select * from epnorm2;
      ENO ENAME                PNO      PNAME                HOURS
        1   Achelal                5          AXE                    8
        2   Sharma                 3        SETWET                   8
SQL> select * from en21;
      ENO ENAME
        1      Achelal
        2     Sharma
SQL> select * from en22;
      PNO    PNAME
        5        AXE
        3     SETWET
SQL> select * from en23;
      ENO       PNO      HOURS
        1         5        8
        2         3        8
```

THIRD NORMAL FORM

```
SQL> select * from epnorm3;
    ENO   ENAME              SAL       DNO     DNAME
      1     Ahamath           50000        2       HR
      2     Bhaylool          90000        5       R&D
SQL> select * from en31
      ENO ENAME              SAL       DNO
        1     Ahamath           50000        2
        2     Bhaylool          90000        5
SQL> select * from ed2;
      DNO    DNAME
        2        HR
        4       R&D
```

RESULT:-

Thus the database was designed using E-R diagrams and Normalizations.

Ex.No.08 PAYROLL SYSTEM

Aim:-

To write a program for payroll system for employees using the Visual Basic 6.0 for front-end and the oracle database for back-end.

Introduction:-

Front-end:-

Visual Basic 6.0 is a windows programming language that has developed at Microsoft Corporation. Visual Basic is a powerful programming language to develop sophisticated windows programs very quickly. Visual Basic is one of the RAD (Rapid Application Development) tools as it enables the programmer to develop applications very easily and very quickly. Visual Basic applications are very popular as front-end to many client/server database systems like SQL/Server, Oracle etc.

Back-end:-

Oracle 8 is an object-oriented relational database management system (ORDBMS). It only offers capabilities of both relational and object-oriented database systems. In Visual Basic a data source identifies a path to data that may include a network library, server, database and other attributes – in this case, the data source is the path to an oracle database.

Procedure:-

Initial steps:-

i. First, we have to design the Visual Basic form like shown below diagram.
ii. Then create the given tables in the oracle database. The table also given below
iii. Insert the required values in the oracle database table if necessary.

ODBC Connection:-

i. Select the “Administrator Tools” in the Control Panel.
ii. In Data Source (ODBC), add the data source name using “Microsoft ODBC for Oracle”.
iii. Then click the “Finish” button for closing the window.

ADODC Connection in Visual Basic:-

i. Add the “Microsoft ADO Data Control 6.0 (OLEDB)” using components toolbox.
ii. Then select the property of ADODC Control.
iii. In the general tab, select the “Use Connection String” option and click the “Build” button.
iv. In the “Data Link Properties” select the “Microsoft OLEDB provider for ODBC Drivers” and then select the “Next” button.
v. Then select the data source name and give the username and password, and then check the “Test Connection” button.
vi. Click the “Ok” button for data link properties box.
vii. Then select the Authentication tab, and give the username and password.
viii. In Record Source tab, select the Command Type and table name.
ix. Finally, click the “Ok” button for ADODC control properties box.
x. Then do the above steps for all the included ADODC controls.
xi. Now write the necessary coding in the Visual Basic coding window for the payroll system requires.

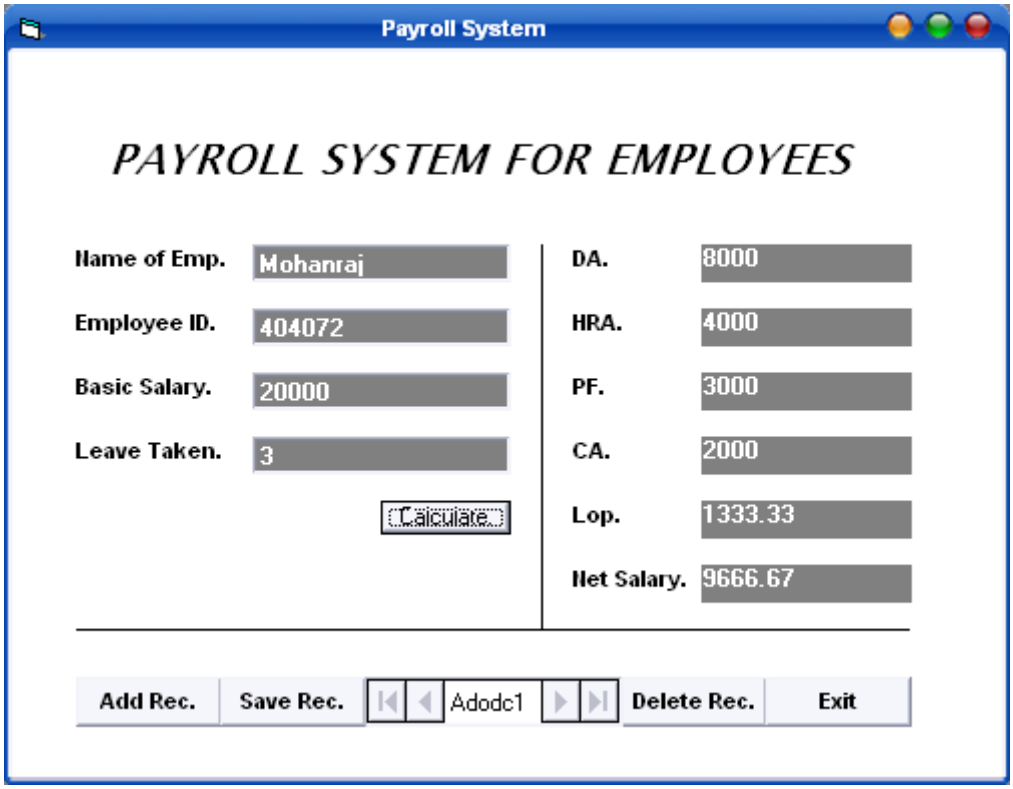
Oracle Table:-

Sql> create table payroll(Ename varchar(15),Eno number(7),Bsalary number(5,2),LT number(2),DA number(5,2),HRA number(5,2),PF number(5,2),CA number(5),LOP number(5,2), NetSal number(6,2));

Program Coding:-

```
Private Sub Command1_Click()
Text5.Text = 0.4 * Val(Text3.Text)
Text6.Text = 0.2 * Val(Text3.Text)
Text7.Text = 0.15 * Val(Text3.Text)
Text8.Text = 0.1 * Val(Text3.Text)
Text9.Text = Format(((Val(Text4.Text) - 1) / 30) * Val(Text3.Text), "#0000.00")
Text10.Text = Val(Text5.Text) + Val(Text6.Text) + Val(Text8.Text) - Val(Text7.Text) -
Val(Text9.Text)
End Sub
Private Sub Command2_Click()
Adodc1.Recordset.AddNew
End Sub
Private Sub Command3_Click()
Adodc1.Recordset.Update
End Sub
Private Sub Command4_Click()
Adodc1.Recordset.Delete
End Sub
Private Sub Command5_Click()
Unload Me
End Sub
```

Output Form:-



Oracle Database Output:-

Sql> Select * from payroll;

Ename	Eno	Bsalary	LT	DA	HRA	PF	CA	LOP	NetSal
Ahamed	404073	25000	3	10000	5000	3750	2500	1666.67	12083.33
Mohanraj	404072	20000	3	8000	4000	3000	2000	1333.33	9666.67

Result:-

Thus the above payroll system for employees has been performed and executed successfully.

Ex.No.09 BANKING SYSTEM

Aim:-

To write a program for banking system for customers using the Visual Basic 6.0 for front-end and the oracle database for back-end.

Introduction:-

Front-end:-

Visual Basic 6.0 is a windows programming language that has developed at Microsoft Corporation. Visual Basic is a powerful programming language to develop sophisticated windows programs very quickly. Visual Basic is one of the RAD (Rapid Application Development) tools as it enables the programmer to develop applications very easily and very quickly. Visual Basic applications are very popular as front-end to many client/server database systems like SQL/Server, Oracle etc.

Back-end:-

Oracle 8 is an object-oriented relational database management system (ORDBMS). It only offers capabilities of both relational and object-oriented database systems. In Visual Basic a data source identifies a path to data that may include a network library, server, database and other attributes – in this case, the data source is the path to an oracle databas e.

Procedure:-

Initial steps:-

i. First, we have to design the Visual Basic form like shown below diagram.
ii. Then create the given tables in the oracle database. The table also given below
iii. Insert the required values in the oracle database table if necessary.

ODBC Connection:-

i. Select the “Administrator Tools” in the Control Panel.
ii. In Data Source (ODBC), add the data source name using “Microsoft ODBC for Oracle”.
iii. Then click the “Finish” button for closing the window.

ADODC Connection in Visual Basic:-

i. Add the “Microsoft ADO Data Control 6.0 (OLEDB)” using components toolbox.
ii. Then select the property of ADODC Control.
iii. In the general tab, select the “Use Connection String” option and click the “Build” button.
iv. In the “Data Link Properties” select the “Microsoft OLEDB provider for ODBC Drivers” and then select the “Next” button.
v. Then select the data source name and give the username and password, and then check the “Test Connection” button.
vi. Click the “Ok” button for data link properties box.
vii. Then select the Authentication tab, and give the username and password.
viii. In Record Source tab, select the Command Type and table name.
ix. Finally, click the “Ok” button for ADODC control properties box.
x. Then do the above steps for all the included ADODC controls.
xi. Now write the necessary coding in the Visual Basic coding window for the payroll system requires.

Oracle Table:-

SQL> create table bank(cname varchar(15),cno number(6),qualify varchar(15),balance number(7,2),add1 varchar(10),add2 varchar(10),loan number(7,2));

Program Coding:-

Variable Initialization:-

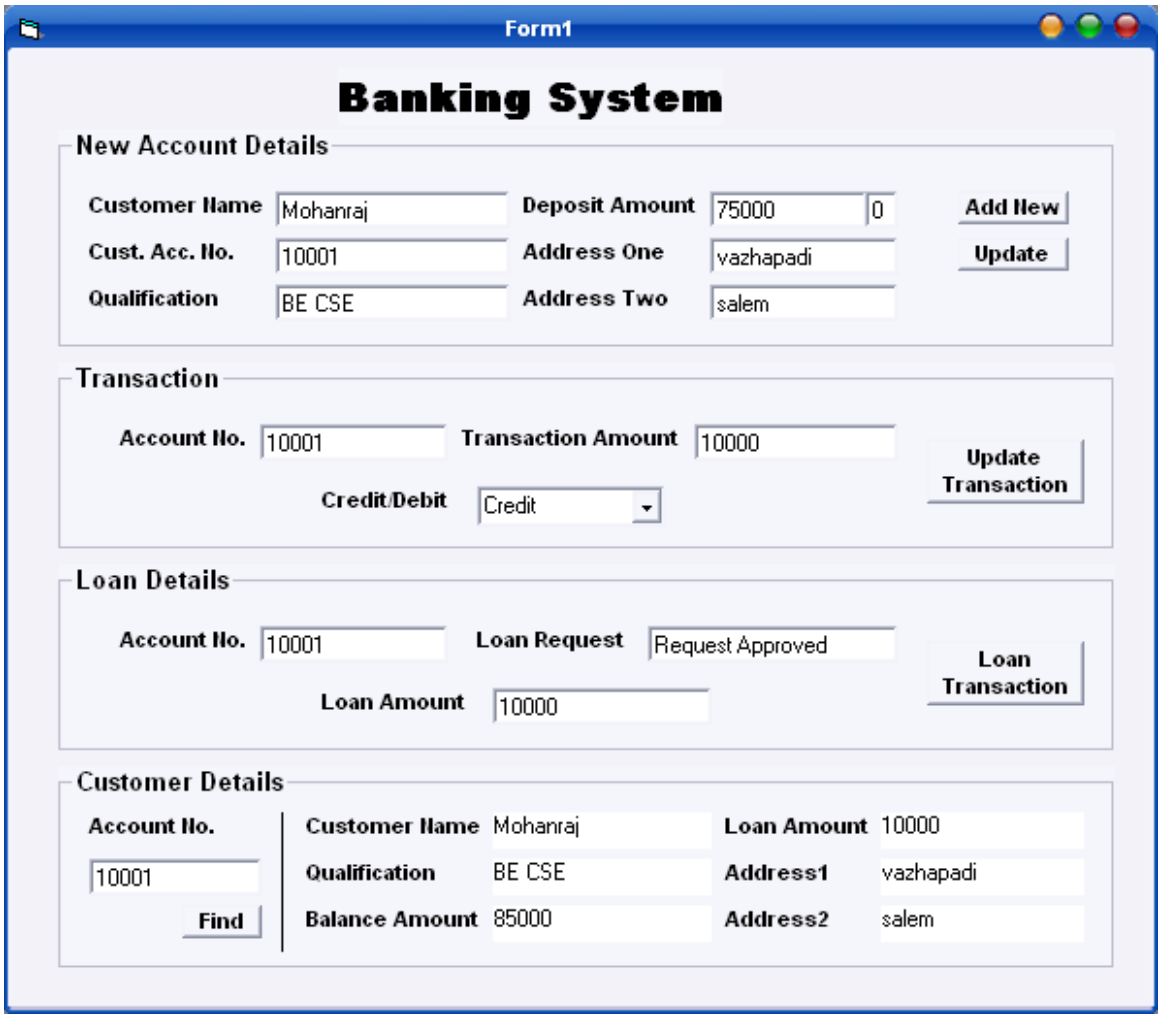
```
Dim con As ADODB.Connection
Dim rs As ADODB.Recordset
```
Connecting Database While Form Load:-
```
Private Sub Form_Load()
Set con = New ADODB.Connection
Set rs = New ADODB.Recordset
con.Provider = "MSDASQL.1;Password=tiger;Persist Security Info=True;User ID=scott;Data
Source=mybank"
con.CursorLocation = adUseClient
con.Mode = adModeReadWrite
con.Open
rs.Open "select * from bank", con, adOpenDynamic, adLockOptimistic
Text1.Text = ""          Text2.Text = ""          Text3.Text = ""          Text4(0).Text =
""      Text4(1).Text = ""      Text5.Text = ""          Text6.Text = ""
End Sub
```
New Account Details:-
```
Private Sub Command1_Click()
Adodc1.Recordset.AddNew
End Sub
Private Sub Command2_Click()
Adodc1.Recordset.Update
End Sub
Private Sub Text4_GotFocus(Index As Integer)
Text4(1).Text = 0
End Sub
```
Transaction Details:-
```
Private Sub Command4_Click()
rs.Close
rs.Open "select * from bank where cno=" & Val(Text7.Text), con, adOpenDynamic
   If rs.EOF = False Or rs.BOF = False Then
      MsgBox "Record Updated...", vbInformation
      If Combo1.Text = "Credit" Then
         rs.Close
         rs.Open "update bank set balance = balance + '" & Val(Text8.Text) & "' where cno=" &
Val(Text7.Text), con, adOpenDynamic
      Else
         rs.Close
         rs.Open "update bank set balance = balance - '" & Val(Text8.Text) & "' where cno=" &
Val(Text7.Text), con, adOpenDynamic
      End If
   Else
      MsgBox "Record Not Found...", vbInformation
   End If
End Sub
```
Loan Details:-
```
Private Sub Command5_Click()
rs.Close
rs.Open "select * from bank where cno=" & Val(Text9.Text), con, adOpenDynamic
   If rs.EOF = False Or rs.BOF = False Then
      MsgBox "Loan Transaction Successed...", vbInformation
         rs.Close
         rs.Open "update bank set loan = loan + '" & Val(Text11.Text) & "' where cno=" & Val(Text9.Text),
con, adOpenDynamic
   Else
      MsgBox "Record Not Found...", vbInformation
```

```
    End If
End Sub
Private Sub Text10_GotFocus()
On Error Resume  Next
rs.Close
rs.Open "select * from bank where cno=" & Val(Text9.Text), con, adOpenDynamic
If rs.EOF = False Or rs.BOF = False Then
    If rs.Fields("loan") = 0 Then
        Text10.Text = "Request Approved"
    Else
        Text10.Text = "Request Not Approved"
    End If
Else
    MsgBox "Record Not Found...", vbInformation
End If
End Sub
```

Customer Details:-

```
Private Sub Command6_Click()
On Error Resume Next
rs.MoveFirst
rs.Find "cno=" & Val(Text12.Text), , adSearchForward
If rs.EOF = False Or rs.BOF = False Then
    Text13.Text = rs("cname")
    Text14.Text = rs("qualify")
    Text15.Text = rs("balance")
    Text16.Text = rs("loan")
    Text17.Text = rs("add1")
    Text18.Text = rs("add2")
Else
    MsgBox "Record not found..."
End If
End Sub
```

Output:-



Oracle Database Output:-

SQL> select * from bank;

CNAME	CNO	QUALIFY	BALANCE	ADD1	ADD2	LOAN
Mohanraj	10001	BE CSE	85000	vazhapadi	salem	10000
Venkatesh	10002	BE CSE	50000	padur	chennai	0
Ahamed	10003	BE ECE	40000	saidapet	chennai	
0						
Durai	10004	BE EIE	30000	thambaram	chennai	0

Result:-

Thus the above library management system for students has been performed and executed successfully.

Ex.No.10 LIBRARY MANAGEMENT SYSTEM

Aim:-

To write a program for library management system for students using the Visual Basic 6.0 for front-end and the oracle database for back-end.

Introduction:-

Front-end:-

Visual Basic 6.0 is a windows programming language that has developed at Microsoft Corporation. Visual Basic is a powerful programming language to develop sophisticated windows programs very quickly. Visual Basic is one of the RAD (Rapid Application Development) tools as it enables the programmer to develop applications very easily and very quickly. Visual Basic applications are very popular as front-end to many client/server database systems like SQL/Server, Oracle etc.

Back-end:-

Oracle 8 is an object-oriented relational database management system (ORDBMS). It only offers capabilities of both relational and object-oriented database systems. In Visual Basic a data source identifies a path to data that may include a network library, server, database and other attributes – in this case, the data source is the path to an oracle database.

Procedure:-

Initial steps:-

i. First, we have to design the Visual Basic form like shown below diagram.
ii. Then create the given tables in the oracle database. The table also given below
iii. Insert the required values in the oracle database table if necessary.

ODBC Connection:-

i. Select the “Administrator Tools” in the Control Panel.
ii. In Data Source (ODBC), add the data source name using “Microsoft ODBC for Oracle”.
iii. Then click the “Finish” button for closing the window.

ADODC Connection in Visual Basic:-

i. Add the “Microsoft ADO Data Control 6.0 (OLEDB)” using components toolbox.
ii. Then select the property of ADODC Control.
iii. In the general tab, select the “Use Connection String” option and click the “Build” button.
iv. In the “Data Link Properties” select the “Microsoft OLEDB provider for ODBC Drivers” and then select the “Next” button.
v. Then select the data source name and give the username and password, and then check the “Test Connection” button.
vi. Click the “Ok” button for data link properties box.
vii. Then select the Authentication tab, and give the username and password.
viii. In Record Source tab, select the Command Type and table name.
ix. Finally, click the “Ok” button for ADODC control properties box.
x. Then do the above steps for all the included ADODC controls.
xi. Now write the necessary coding in the Visual Basic coding window for the payroll system requires

Oracle Tables:-

Sql> create table studlib(sname varchar(15) NOT NULL, scard number(7) UNIQUE, nobooks number(2), sfine number(5,2));

Insert into studlib(sname,scard) values (‘&sname’,&scard);

Sql> create table booklib(bname varchar(15) NOT NULL, bauthor varchar(15), bcode number(5), bstatus number(2));
Insert into booklib values ('&bname','&bauthor',&bcode,&bstatus);
Sql> create table dayt(scard number(7), bcode number(7), bname varchar(15), bardate date, retdate date, fdue number(5,2));
Sql> update studlib set nobooks=0;
Sql> update studlib set fdue=0;

Default Table Contents:-

Sql> select * from studlib;

Sname	scard	nobook	sfine
Mohanraj S	404072	0	0
Venkatesh K	404071	0	0
Mahesh S	404074	0	0
Ahamad Ali	404073	0	0

Sql> select * from booklib;

Bname	bauthor	bcode	bstatus
C program	balagurusamy e	1001	3
C++ program	balagurusamy e	1002	2
Java program	silberscatz	1003	3
Vc++ program	silberscatz	1004	2
Oracle	silberscatz	1005	3

Sql> select * from dayt;
no selected items.

Program Coding:-

Variable Initializing:-

```
Dim con As ADODB.Connection
Dim rs1 As ADODB.Recordset
Dim rs2 As ADODB.Recordset
Dim rs3 As ADODB.Recordset
```

Book Searching:-

```
Private Sub Command1_Click()
On Error Resume Next
   rs2.Close
   rs2.Open "select * from booklib where lower(bname)='" & Trim(LCase(Text1.Text)) & "' and lower(bauthor)='" & Trim(LCase(Text2.Text)) & "'", con, adOpenDynamic
   If rs2.EOF = False Or rs2.BOF = False Then
      MsgBox "Record found"
      If rs2.Fields("bstatus") = 0 Then
         MsgBox "There is No Books available"
         Else
         Text3.Text = rs2.Fields("bstatus")
         Text4.Text = rs2.Fields("bcode")
      End If
      Else
      MsgBox "Record Not found"
   End If
End Sub
```

Book Borrowing:-

```
Private Sub Command2_Click()
On Error Resume Next
rs1.Close
rs2.Close
```

```
If Text8.Text = "Approved" Then
   rs1.Open "update studlib set nobooks=nobooks+1 where scard=" & Val(Text7.Text), con,
adOpenDynamic
   rs2.Open "update booklib set bstatus=bstatus-1 where bcode=" & Val(Text5.Text), con,
adOpenDynamic
   rs2.Close
   rs2.Open "select * from booklib where bcode=" & Val(Text5.Text), con, adOpenDynamic
   Text9.Text = rs2.Fields("bstatus")
   rs3.Close
   rs3.Open "SELECT * FROM DAYT"
   rs3.AddNew
      rs3.Fields("scard") = Text7.Text
      rs3.Fields("bcode") = Text5.Text
      rs3.Fields("bname") = Text1.Text
      rs3.Fields("bardate") = Date
      rs3.Fields("retdate") = DateAdd("D", 3, Date)
      rs3.Fields("fdue") = 0
   rs3.Update
Else
   MsgBox "Not approved detected, sorry... unable to give the book"
End If
Command2.Enabled = False
End Sub
Private Sub Text5_GotFocus()
   Text5.Text = Text4.Text
End Sub
Private Sub Text6_GotFocus()
   Text6.Text = Text3.Text
End Sub
Private Sub Text8_GotFocus()
On Error Resume Next
Command2.Enabled = True
rs1.Close
rs2.Close
rs3.Close
rs1.Open "select * from studlib where scard=" & Val(Text7.Text),  con, adOpenDynamic
rs2.Open "Select * from booklib where bcode=" & Val(Text5.Text), con, adOpenDynamic
rs3.Open "SELECT * FROM DAYT WHERE BCODE=" & Val(Text5.Text) & " AND SCARD="  &
Val(Text7.Text) & " AND BARDATE='" & Format(Date, "DD-MMM-YY") & "'", con,  adOpenDynamic
If rs3.EOF = True Or rs3.BOF = True Then
   If rs2.Fields("bstatus") <= 0 Then
   MsgBox "There is No Books available"
   Text8.Text = "Not Approved"
   Else
      If rs1.EOF = True Or rs1.BOF = True Then
      MsgBox "Student Not found"
      Text8.Text = "Not Approved"
      Else
         If rs1.Fields("nobooks") <= 3 Then
         Text8.Text = "Approved"
         Else
         Text8.Text = "Not Approved"
         End If
      End If
   End If
```

```
    Else
    MsgBox "The SAME BOOK IS NOT ALLOWED to taking..."
    Text8.Text = "Not Approved"
End If
End Sub
```

Book Renewal:-

```
Private Sub Command3_Click()
rs1.Close
rs3.Close
rs1.Open "update STUDLIB set SFINE=SFINE+" & Val(Text11.Text) & " where scard=" &
Val(Text22.Text), con, adOpenDynamic
rs3.Open "update dayt set BARDATE='" & Format(Date, "DD-MMM-YY") & "' where bcode=" &
Val(Text10.Text) & " and scard=" & Val(Text22.Text), con,  adOpenDynamic
rs3.Open "update dayt set RETDATE='" & Format(DateAdd("D", 3, Date), "DD-MMM-YY") & "' where
bcode=" & Val(Text10.Text) & " and scard=" & Val(Text22.Text), con, adOpenDynamic
rs3.Open "update dayt set FDUE=" & Val(Text11.Text) & " where bcode=" & Val(Text10.Text) & " and
scard=" & Val(Text22.Text), con, adOpenDynamic
End Sub
Private Sub Text11_GotFocus()
Dim chk As String
    rs3.Close
    rs3.Open "select * from dayt where bcode=" & Val(Text10.Text) & " and scard=" & Val(Text22.Text),
con, adOpenDynamic
    If rs3.EOF = False Or rs3.BOF = False Then
    MsgBox "Record found"
    chk = DateDiff("D", rs3.Fields("bardate"), Date)
    'MsgBox "" & chk
    'chk = Format(chk, "#.00")
    If chk >= 4 Then
        chk = chk - 3
        Text11.Text = Val((chk) * 1.25)
    End If
    Text12.Text  =  rs3.Fields("bardate")
    Text13.Text  =  rs3.Fields("retdate")
    End If
End Sub
```

Book Return:-

```
Private Sub Command4_Click()
On Error Resume Next
Dim chk As String
rs1.Close
rs2.Close
rs3.Close
    rs3.Open "select * from dayt where bcode=" & Val(Text14.Text) & " and scard=" & Val(Text23.Text),
con, adOpenDynamic
    If rs3.EOF = False Or rs3.BOF = False Then
    chk = DateDiff("D", rs3.Fields("bardate"), Date)
    If chk >= 4 Then
        chk = chk - 3
        Text15.Text = Val((chk) * 1.25)
    End If
    End If
    rs3.Open "select * from dayt where bcode=" & Text14.Text & " and scard=" & Text23.Text, con,
adOpenDynamic
    If rs3.EOF = False Or rs3.BOF = False Then
```

```
        MsgBox "Record Found"
        Text16.Text  = rs3.Fields("bname")
        rs2.Open "Select * from booklib where bcode=" & Text14.Text, con, adOpenDynamic
        Text17.Text = rs2.Fields("bstatus")
        rs1.Open "update STUDLIB set SFINE=SFINE+" & Val(Text15.Text) & " where scard=" &
Val(Text23.Text), con, adOpenDynamic
        rs3.Delete
        rs3.Close
        rs1.Close
        rs2.Close
        rs1.Open "update studlib set nobooks=nobooks-1 where scard=" & Text23.Text, con,
adOpenDynamic
        rs2.Open "update booklib set bstatus=bstatus+1 where bcode=" & Text14.Text, con,
adOpenDynamic
    Else
        MsgBox "Record Not found"
    End If
End Sub
```

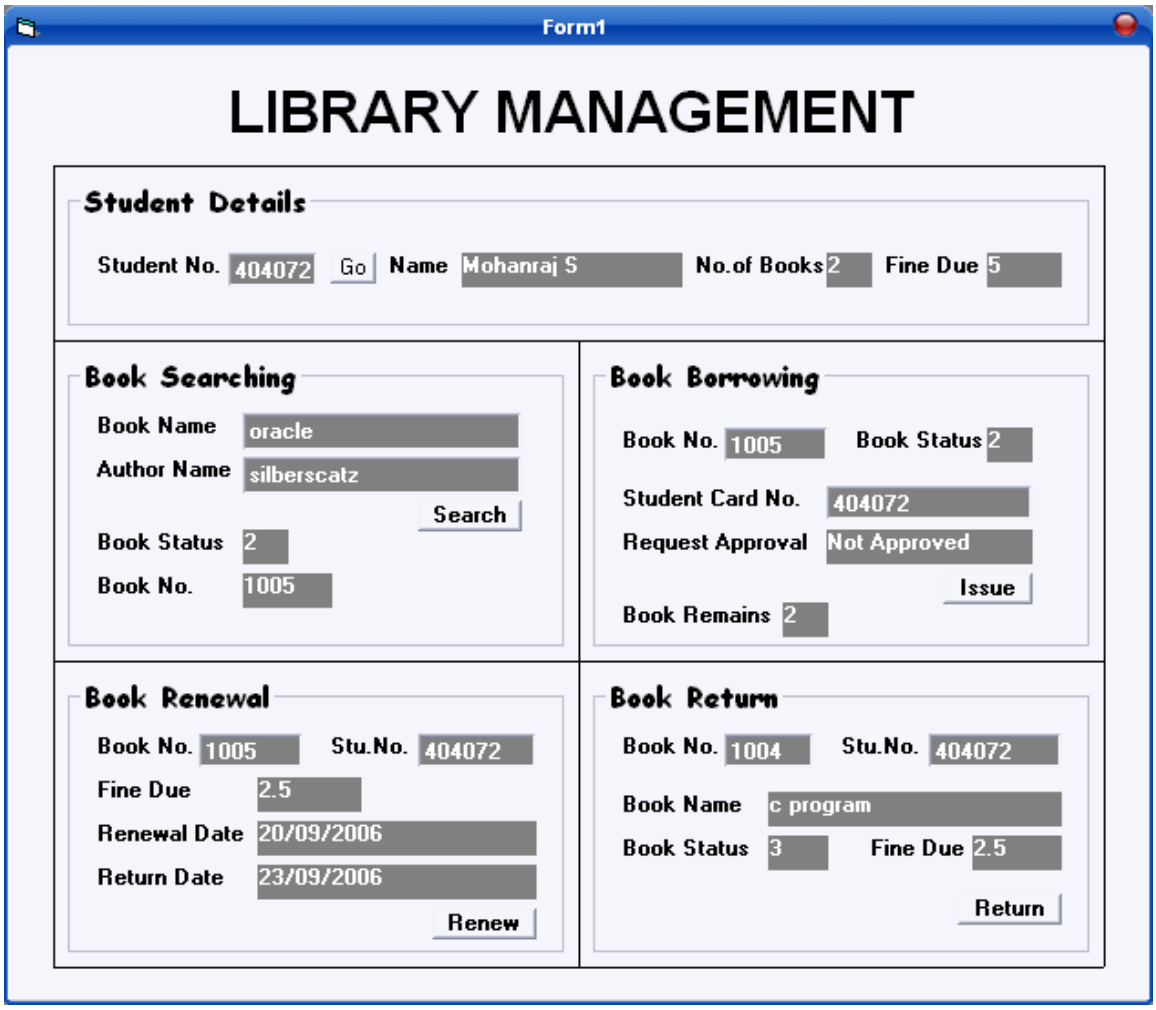
Searching Student Details:-

```
Private Sub Command6_Click()
On Error Resume Next
rs1.MoveFirst
rs1.Find "scard=" & Val(Text18.Text), , adSearchForward
If rs1.EOF = False Or rs1.BOF = False Then
    Text19.Text = rs1(0)
    Text20.Text = rs1(2)
    Text21.Text = rs1(3)
    Else
    MsgBox "Record not found"
End If
End Sub
```

Connecting Database while Form Load:-

```
Private Sub Form_Load()
Set con = New ADODB.Connection
Set rs1 = New ADODB.Recordset
Set rs2 = New ADODB.Recordset
Set rs3 = New ADODB.Recordset
con.Provider = "MSDASQL.1;Password=tiger;Persist Security Info=True;User ID=scott;Data
Source=mydb"
con.CursorLocation = adUseClient
con.Mode = adModeReadWrite
con.Open
rs1.Open "select * from studlib", con, adOpenDynamic, adLockOptimistic
rs2.Open "select * from booklib", con, adOpenDynamic, adLockOptimistic
rs3.Open "select * from dayt", con, adOpenDynamic, adLockOptimistic
End Sub
```

Output Form:-



Result:-

Thus the above library management system for students has been performed and executed successfully.