



**SRM**  
INSTITUTE OF SCIENCE & TECHNOLOGY  
(Deemed to be University u/s 3 of UGC Act, 1956)

**TIRUCHIRAPPALLI**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**  
*TIRUCHIRAPPALLI*  
**SCHOOL OF COMPUTING**

# **18CSC303J – DATABASE MANAGEMENT SYSTEMS**

## **Record Work**

**Register No.** : \_\_\_\_\_

**Name of the Student** : \_\_\_\_\_

**Semester** : \_\_\_\_\_

**Programme** : B.Tech. \_\_\_\_\_

**REGISTER NO:** \_\_\_\_\_

### **BONAFIDE CERTIFICATE**

Certified as the bonafide record of work done by \_\_\_\_\_,

Register No. \_\_\_\_\_ of \_\_\_\_\_ (Semester/Year),

B. Tech \_\_\_\_\_

programme in the practical course **18CSC303J – Database Management Systems** at SRM  
Institute of Science and Technology, Tiruchirappalli during the academic year 2023 – 24.

**Faculty In-charge**

**Head of the Department**

Submitted for the End Semester Examination held on \_\_\_\_\_

**Examiner-1**

**Examiner-2**

# **SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

## **Tiruchirappalli**

### **SCHOOL OF COMPUTING**

#### **VISION AND MISSION OF THE DEPARTMENT**

**Vision** To become a world class department in imparting high-quality knowledge and in providing students a unique learning and research experience in Computer Science and Engineering.

#### **Mission**

1. To impart knowledge in cutting edge Computer Science and Engineering technologies in par with industrial standards.
2. To collaborate with renowned academic institutions to uplift innovative research and development in Computer Science and Engineering and its allied fields to serve the needs of society
3. To demonstrate strong communication skills and possess the ability to design computing systems individually as well as part of a multidisciplinary teams.
4. To instill societal, safety, cultural, environmental, and ethical responsibilities in all professional activities
5. To produce successful Computer Science and Engineering graduates with personal and professional responsibilities and commitment to lifelong learning

#### **Program Educational Objectives (PEO)**

PEO - 1 Graduates will be able to perform in technical/managerial roles ranging from design, development, problem solving to production support in software industries and R&D sectors.

PEO - 2 Graduates will be able to successfully pursue higher education in reputed institutions.

PEO - 3 Graduates will have the ability to adapt, contribute and innovate new technologies and systems in the key domains of Computer Science and Engineering.

PEO - 4 Graduates will be ethically and socially responsible solution providers and entrepreneurs in Computer Science and other engineering disciplines.

PEO - 5 Graduates will possess the additional skills in core computer science discipline with knowledge of Hardware, Software, Programming, Logic & Reasoning.

#### **Mission of the Department to Program Educational Objectives (PEO) Mapping**

	Mission Stmt -1	Mission Stmt -2	Mission Stmt -3	Mission Stmt -4	Mission Stmt -5
PEO - 1	H	H	H	H	H
PEO - 2	L	H	H	H	H
PEO - 3	H	H	M	L	H
PEO - 4	M	H	M	H	H
PEO - 5	H	H	M	M	H

H – High correlation, M – Medium Correlation, L – Low Correlation

#### **Program Outcomes as defined by NBA (PO)**

##### **Engineering Graduates will be able to:**

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

#### PSO – Program Specific Outcomes (PSO)

**PSO - 1** Ability to understand client requirements and suggest solutions

**PSO - 2** Ability to create Software for automation and function

**PSO - 3** Ability to utilize Logic & Reasoning Skills

Mapping Program Educational Objectives (PEO) to Program Learning Outcomes (PLO)

Program Learning Outcomes (PLO)														
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Engineering Knowledge	Problem Analysis	Design & Development	Analysis, Design, Research	Modern Tool Usage	Society & Culture	Environment & Sustainability	Ethics	Individual & Team Work	Communication	Project Mgt & Finance	Life Long Learning	PSO-1	PSO-2	PSO-3
H	H	H	H	H	-	-	-	H	H	H	H	-	-	-
H	H	H	H	H	-	-	-	H	H	H	H	-	-	-
H	H	H	H	H	-	-	-	H	H	H	H	-	-	-
H	H	L	M	L	-	-	-	M	M	M	L	-	-	-
H	L	L	L	L	-	-	-	H	L	L	L			

H – High correlation, M – Medium Correlation, L – Low Correlation

## COURSE DESIGN

<b>Course Learning Outcomes (CLO):</b>	At the end of this course, learners will be able to:
CLO – 1	Acquire the knowledge on DBMS Architecture and Languages
CLO – 2	Apply the fundamentals of data models to model an application's data requirements using conceptual modelling tools like ER diagrams
CLO – 3	Apply the method to convert the ER model to a database schema based on the conceptual relational model
CLO – 4	Apply the knowledge to create, store and retrieve data using Structure Query Language (SQL) and PL/SQL
CLO – 5	Apply the knowledge to improve database design using various normalization criteria and optimize queries
CLO – 6	Appreciate the fundamental concepts of transaction processing-concurrency control techniques and recovery procedures

## CLO-PO Mapping

CLOs	Program Learning Outcomes (PLO)														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	Engineering Knowledge	Problem Analysis	Design & Development	Analysis, Design, Research	Modern Tool Usage	Society & Culture	Environment & Sustainability	Ethics	Individual & Team Work	Communication	Project Mgt. & Finance	Life Long Learning	PSO - 1	PSO - 2	PSO – 3
CLO1	H	H	L	-	-	-	L	-	H	H	M	M	-	-	-
CLO2	H	H	H	H	H	-	M	-	H	H	H-	M	-	-	-
CLO3	H	H	M	H	H	M	M	L	H	H	M	-	-	-	-
CLO4	H	H	H	-	H	-	-	M	H	M	H	-	-	-	-
CLO5	H	M	M	M	M	M	M	-	H	H	-	M	-	-	-

## LIST OF EXPERIMENTS & SCHEDULE

Exp. No.	Title	Date	Signature
1	Data Definition Language (DDL) commands in RDBMS		
2	Data Manipulation Language (DML) and Data Control Language (DCL)		
3	High level language extensions with cursors		
4	High level language extension with Triggers		
5	Procedures and Functions		
6	Embedded SQL		
7	Database design using E-R model and Normalization		
8	Design and implementation of payroll processing system		
9	Design and implementation of Banking system		
10	Design and implementation of Library Information System		
11	Design and implementation of Student Information System		
12	Automatic Backup of Files and Recovery of Files		

**Course Co-ordinator**

**Head of the Department**

# **HARDWARE AND SOFTWARE REQUIREMENTS**

## **HARDWARE REQUIREMENTS:**

INTEL PENTIUM 915 GV

80GB HDD

512MB DDR

## **SOFTWARE REQUIREMENTS:**

ORACLE 8i,9i.

MY SQL, DB2.



**AIM:**

To create a DDL to perform creation of table, alter, modify and drop column.

**DDL COMMANDS**

1. The Create Table Command: - it defines each column of the table uniquely. Each column has minimum of three attributes, a name , data type and size.

**Syntax:**

Create table <table name> (<col1> <datatype>(<size>),<col2> <datatype><size>)); Ex:create table emp(empno number(4) primary key, ename char(10));

2. Modifying the structure of tables.

**a) Add new columns Syntax:**

Alter table <tablename> add(<new col><datatype>(size),<new col>datatype(size)); Ex:alter table emp add(sal number(7,2));

3. Dropping a column from a table.

**Syntax:**

Alter table <tablename> drop column <col>; Ex:alter table emp drop column sal;

4. Modifying existing columns.

**Syntax:**

Alter table <tablename> modify(<col><newdatatype>(<newsize>));

Ex:alter table emp modify(ename varchar2(15));

5. Renaming the tables **Syntax:**

Rename <oldtable> to <new table>; Ex:rename

emp to emp1;

6. truncating the tables.

**Syntax:**

Truncate table <tablename>;

Ex:trunc table emp1;

7. Destroying tables.

**Syntax:**

Drop table <tablename>;

Ex:drop table emp;

**CREATION OF TABLE:**

**SYNTAX:**

create table<tablename>(column1 datatype,column2 datatype...);

**EXAMPLE:**

SQL>create table std(sno number(5),sname varchar(20),age number(5),sdob date,sm1 number(4,2),sm2 number(4,2),sm3 number(4,4)); Table created.

SQL>insert into std values(101,'AAA',16,'03-jul-88',80,90,98); 1  
row created.

SQL>insert into std values(102,'BBB',18,'04-aug-89',88,98,90); 1  
row created.

**OUTPUT:**

Select \* from std;

SNO	SNAME	AGE	SDOB	SM1	SM2	SM3
101	AAA	16	03-jul-88	80	90	98
102	BBB	18	04-aug-89	88	98	90

**ALTER TABLE WITH ADD:**

SQL>create table student(id number(5),name varchar(10),game varchar(20)); Table created.

SQL>insert into student values(1,'mercy','cricket'); 1 row created.

**SYNTAX:**

alter table<tablename>add(col1 datatype,col2 datatype..);

**EXAMPLE:**

SQL>alter table student add(age number(4));

SQL>insert into student values(2,'sharmi','tennis',19);

**OUTPUT:**

ALTER: select \* from student;

ID	NAME	GAME
----	------	------

1	Mercy	Cricket
---	-------	---------

ADD: select \* from student;

ID	NAME	GAME	AGE
----	------	------	-----

1	Mercy	cricket
---	-------	---------

2	Sharmi	Tennis	19
---	--------	--------	----

## **ALTER TABLE WITH MODIFY:**

### **SYNTAX:**

Alter table<tablename>modify(col1 datatype,col2 datatype..);

### **EXAMPLE:**

SQL>alter table student modify(id number(6),game varchar(25));

### **OUTPUT: MODIFY**

desc student;

NAME NULL? TYPE

Id      Number(6)

Name    Varchar(20)

Game    Varchar(25) Age

Number(4) **DROP:**

**SYNTAX:** drop table<tablename>; **EXAMPLE:**

SQL>drop table student; SQL>Table

dropped.

## **TRUNCATE TABLE**

**SYNTAX:** TRUNCATE TABLE <TABLE NAME>;

Example: Truncate table stud;

## **DESC**

Example: desc emp;

Name Null? Type

-----

EmpNo NOT NULL number(5)

EName VarChar(15)

Job NOT NULL Char(10)

DeptNo NOT NULL number(3)

PHONE\_NO number (10)

### **CONSTRAINTS:**

Create table tablename (column\_name1 data\_type constraints, column\_name2 data\_type constraints ...) **Example:**

Create table Emp ( EmpNo number(5), EName VarChar(15), Job Char(10) constraint unique, DeptNo number(3) CONSTRAINT FKKey2 REFERENCES DEPT(DeptNo));

Create table stud (sname varchar2(20) not null, rollno number(10) not null, dob date not null);

### **DOMAIN INTEGRITY**

**Example:** Create table cust(custid number(6) not null, name char(10)); Alter table cust modify (name not null);

### **CHECK CONSTRAINT**

Example: Create table student (regno number (6), mark number (3) constraint b check (mark >=0 and mark <=100)); Alter table student add constraint b2 check (length(regno)<=4));

### **ENTITY INTEGRITY**

a) Unique key constraint

Example: Create table cust(custid number(6) constraint unique, name char(10)); Alter table cust add(constraint c unique(custid)); b) Primary Key Constraint

Example: Create table stud(regno number(6) constraint primary key, name char(20));

### **RESULT:**

Thus, the DDL commands have been executed successfully.

**AIM;**

To study the various DML commands and implement them on the database.

**DML COMMANDS**

DML commands are the most frequently used SQL commands and is used to query and manipulate the existing database objects. Some of the commands are Insert, Select, Update, Delete.

**Insert Command** This is used to add one or more rows to a table. The values are separated by commas and the data types char and date are enclosed in apostrophes. The values must be entered in the same order as they are defined.

**Select Commands** It is used to retrieve information from the table. It is generally referred to as querying the table. We can either display all columns in a table or only specify column from the table.

**Update Command** It is used to alter the column values in a table. A single column may be updated or more than one column could be updated.

**Delete command** After inserting row in a table we can also delete them if required. The delete command consists of a from clause followed by an optional where clause.

**Sample Queries**

Q1: Insert a single record into dept table.

Ans: SQL> insert into dept values (1,'IT','Tholudur'); 1

row created.

Q2: Insert more than a record into emp table using a single insert command.

Ans: SQL> insert into emp values(&empno,&ename,&'&job',&deptno,&sal);

Enter value for empno: 1

Enter value for ename: Mathi

Enter value for job: AP

Enter value for deptno: 1 Enter value for sal: 10000 old 1: insert into  
emp values(&empno,'&ename','&job',&deptno,&sal) new 1: insert  
into emp values(1,'Mathi','AP',1,10000) 1 row created.

SQL> / Enter value for empno: 2

Enter value for ename: Arjun

Enter value for job: ASP

Enter value for deptno: 2 Enter value for sal: 12000 old 1: insert into  
emp values(&empno,'&ename','&job',&deptno,&sal) new 1: insert  
into emp values(2,'Arjun','ASP',2,12000) 1 row created.

SQL> / Enter value for empno: 3

Enter value for ename: Gudan

Enter value for job: ASP

Enter value for deptno: 1 Enter value for sal: 12000 old 1: insert into  
emp values(&empno,'&ename','&job',&deptno,&sal) new 1: insert  
into emp values(3,'Gudan','ASP',1,12000) 1 row created.

Q3: Update the emp table to set the salary of all employees to Rs15000/- who are working as  
ASP

Ans: SQL> select \* from emp;

EMPNO ENAME JOB DEPTNO SAL

-----

1 Mathi AP 1 10000

2 Arjun ASP 2 12000

3 Gudan ASP 1 12000

SQL> update emp set sal=15000 where job='ASP'; 2 rows updated.

SQL> select \* from emp;

EMPNO ENAME JOB DEPTNO SAL

-----

1 Mathi AP 1 10000

2 Arjun ASP 2 15000

3 Gungan ASP 1 15000

Q4: Create a pseudo table employee with the same structure as the table emp and insert rows into the table using select clauses.

Ans: SQL> create table employee as select \* from emp; Table created.

SQL> desc employee;

Name Null? Type

-----  
EMPNO NUMBER(6)

ENAME NOT NULL VARCHAR2(20)

JOB NOT NULL VARCHAR2(13)

DEPTNO NUMBER(3)

SAL NUMBER(7,2)

Q5: select employee name, job from the emp table

Ans: SQL> select ename, job from emp;  
ENAME JOB

-----  
Mathi AP  
Arjun ASP  
Gungan ASP  
Karthik Prof  
Akalya AP  
suresh lect  
6 rows selected.

Q6: Delete only those who are working as lecturer

Ans: SQL> select \* from emp;  
EMPNO ENAME JOB DEPTNO SAL

-----  
1 Mathi AP 1 10000  
2 Arjun ASP 2 15000  
3 Gungan ASP 1 15000  
4 Karthik Prof 2 30000  
5 Akalya AP 1 10000  
6 suresh lect 1 8000  
6 rows selected.



SQL> delete from emp where job='lect';

1 row deleted.

SQL> select \* from emp;

EMPNO ENAME JOB DEPTNO SAL

- -----  
1 Mathi AP 1 10000  
2 Arjun ASP 2 15000  
3 Gungan ASP 1 15000  
4 Karthik Prof 2 30000  
5 Akalya AP 1 10000

Q7: List the records in the emp table orderby salary in ascending order.

Ans: SQL> select \* from emp order by sal;

EMPNO ENAME JOB DEPTNO SAL

-----  
1 Mathi AP 1 10000  
5 Akalya AP 1 10000  
2 Arjun ASP 2 15000  
3 Gungan ASP 1 15000  
4 Karthik Prof 2 30000

Q8: List the records in the emp table orderby salary in descending order.

Ans: SQL> select \* from emp order by sal desc;

EMPNO ENAME JOB DEPTNO SAL

-----  
4 Karthik Prof 2 30000  
2 Arjun ASP 2 15000  
3 Gungan ASP 1 15000  
1 Mathi AP 1 10000  
5 Akalya AP 1 10000

Q9: Display only those employees whose deptno is 30.

Solution: Use SELECT FROM WHERE syntax.

Ans: SQL> select \* from emp where deptno=1;

EMPNO ENAME JOB DEPTNO SAL

-----  
1 Mathi AP 1 10000  
3 Gungan ASP 1 15000  
5 Akalya AP 1 10000

Q10: Display deptno from the table employee avoiding the duplicated values.

Solution:

1. Use SELECT FROM syntax.

2. Select should include distinct clause for the deptno.

Ans: SQL> select distinct deptno from emp;

DEPTNO

-----

1

2

## **IMPLEMENTATION OF DATA AND BUILT IN FUNCTIONS IN SQL**

### **CHARACTER/STRING FUNCTION:**

SQL> select upper('welcome') from dual;

-----

WELCOME

SQL> select upper('hai') from dual;

---

HAI

SQL> select lower('HAI') from dual;

LOW

--- hai

SQL> select initcap('hello world') from dual;

INITCAP('Hello

-----

Hello World

SQL> select ltrim(' hai') from dual;

LTR

--- hai

SQL> select rtrim('hai ')from dual;

RTR

--- hai

SQL> select rtrim(' hai ')from dual;

RTRIM('

-----

hai

SQL> select concat('SRM',' university')from dual;

-----

SRM university

SQL> select length('SRM')from dual;

LENGTH('SRM')

-----

12

SQL> select replace('SRM university', 'SRM','Anna')from dual;

----- Anna

university

SQL> select substr('SRM', 7,6)from dual;

SUBSTR

-----

lingam

```
SQL> select rpad('hai',3,'*')from dual;
```

RPAD('

-----

hai\*\*\*

```
SQL> select lpad('hai',3,'*')from dual;
```

LPAD('

-----

\*\*\*hai

```
SQL> select replace('Dany','y','ie')from dual;
```

REPLACE

-----

Danie

```
SQL> select translate('cold','ld','ol')from dual;
```

TRANSL

----- cool

**Date and**

**Time**

**Function**

```
SQL> select sysdate from dual;
```

SYSDATE

-----

07-APR-10

```
SQL> select round(sysdate)from dual;
```

```
ROUND(SYS
```

```
-----
```

```
07-APR-10
```

```
SQL> select add_months(sysdate,3)from dual;
```

```
ADD_MONTH
```

```
-----
```

```
07-JUL-10
```

```
SQL> select last_day(sysdate)from dual;
```

```
LAST_DAY(
```

```
-----
```

```
30-APR-10
```

```
SQL> select sysdate+20 from dual;
```

```
SYSDATE+2
```

```
-----
```

```
27-APR-10
```

```
SQL> select next_day(sysdate,'tuesday')from dual;
```

```
NEXT_DAY(
```

```
-----
```

```
13-APR-10
```

### **NUMERIC FUNCTION**

```
SQL> select round(15.6789)from dual;
```

```
ROUND(15.6789)
```

```
-----
```

```
16
```

```
SQL> select ceil(23.20)from dual;
```

CEIL(23.20)

-----

24

SQL> select floor(34.56)from dual;

FLOOR(34.56)

-----

34

SQL> select trunc(15.56743)from dual;

TRUNC(15.56743)

-----

15

SQL> select sign(-345)from dual;

SIGN(-345)

-----

-1

SQL> select abs(-70)from dual;

ABS(-70)

-----

70

### **MATH FUNCTION:**

SQL> select abs(45) from dual;

ABS(45)

-----

45

SQL> select power(10,12) from dual;

POWER(10,12)

-----  
1.000E+12

SQL> select mod(11,5) from dual;

MOD(11,5)

-----  
1

SQL> select exp(10) from dual;

EXP(10)

-----  
22026.466

SQL> select sqrt(225) from dual;

SQRT(225)

-----  
15

### NESTED QUERIES AND JOIN QUERIES

Q1: Display all employee names and salary whose salary is greater than minimum salary of the company and job title starts with \_M'.

#### **Solution:**

1. Use select from clause.
2. Use like operator to match job and in select clause to get the result.

Ans: SQL> select ename,sal from emp where sal>(select min(sal) from emp where job like 'A%');

ENAME SAL

-----

Arjun 12000

Gugan 20000

Karthik 15000

Q2: Issue a query to find all the employees who work in the same job as Arjun.

Ans: SQL> select \* from emp;

EMPNO ENAME JOB DEPTNO SAL

-----

1 Mathi AP 1 10000

2 Arjun ASP 2 12000

3 Gugan ASP 2 20000

4 Karthik AP 1 15000

SQL> select ename from emp where job=(select job from emp where ename='Arjun');  
ENAME

-----

Arjun

Gugan

### SET OPERATORS QUERIES:

Q1: Display all the dept numbers available with the dept and emp tables avoiding duplicates.

#### Solution:

1. Use select from clause.

2. Use union select clause to get the result.

Ans: SQL> select deptno from emp union select deptno from dept;

DEPTNO

-----

1

2



12

30

40

Q2: Display all the dept numbers available with the dept and emp tables.

**Solution:**

1. Use select from clause.

2. Use union all in select clause to get the result.

Ans: SQL> select deptno from emp union all select deptno from dept;

DEPTNO

-----

1

2

2

1

12

1

2

30

40

9 rows selected.

Q3: Display all the dept numbers available in emp and not in dept tables and vice versa.

**Solution:**

1. Use select from clause.

2. Use minus in select clause to get the result.

Ans: SQL> select deptno from emp minus select deptno from dept;

DEPTNO

-----

12

SQL> select deptno from dept minus select deptno from emp;

DEPTNO

-----

30

40

## VIEWS

Q1: The organization wants to display only the details of the employees those who are ASP.

### **Solution:**

1. Create a view on emp table named managers
2. Use select from clause to do horizontal portioning

Ans: SQL> create view empview as select \* from emp where job='ASP'; View created.

SQL> select \* from empview;

EMPNO ENAME JOB DEPTNO SAL

-----

2 Arjun ASP 2 12000 3 Gungan ASP 2 20000

Q2: The organization wants to display only the details like empno, empname, deptno, deptname of the employees. (Vertical portioning) **Solution:** 1. Create a view on emp table named general

2. Use select from clause to do vertical partitioning Ans:

SQL> create view empview1 as select ename,sal from emp;  
View created.

Q3: Display all the views generated.

Ans: SQL> select \* from tab;

TNAME TABTYPE CLUSTERID

-----

DEPT TABLE

EMP TABLE

EMPVIEW VIEW

EMPVIEW1 VIEW

Q4: Execute the DML commands on the view created.

Ans: SQL> select \* from empview;

EMPNO ENAME JOB DEPTNO SAL

-----

2 Arjun ASP 2 12000

3 Gungan ASP 2 20000

Q5: Drop a view.

Ans: SQL> drop view empview1; View

dropped.

Q3: Issue a query to display information about employees who earn more than any employee in dept 1. Ans: SQL> select \* from emp where sal>(select max(sal) from emp where empno=1);

EMPNO ENAME JOB DEPTNO SAL

-----

2 Arjun ASP 2 12000

3 Gungan ASP 2 20000

4 Karthik AP 1 15000

JOINS Tables used

SQL> select \* from emp;

EMPNO ENAME JOB DEPTNO SAL

-----

1 Mathi AP 1 10000

2 Arjun ASP 2 12000

3 Gungan ASP 2 20000

4 Karthik AP 1 15000

SQL> select \* from dept;

DEPTNO DNAME LOC

-----  
1 ACCOUNTING NEW YORK  
2 RESEARCH DALLAS  
30 SALES CHICAGO  
40 OPERATIONS BOSTON  
EQUI-JOIN

Q4: Display the employee details, departments that the departments are same in both the emp and dept. Solution: 1. Use select from clause.

2. Use equi join in select clause to get the result.

Ans: SQL> select \* from emp,dept where emp.deptno=dept.deptno;  
EMPNO ENAME JOB DEPTNO SAL DEPTNO DNAME LOC

-----  
1 Mathi AP 1 10000 1 ACCOUNTING NEW YORK  
2 Arjun ASP 2 12000 2 RESEARCH DALLAS  
3 Gugan ASP 2 20000 2 RESEARCH DALLAS  
4 Karthik AP 1 15000 1  
ACCOUNTING NEW YORK  
NON-EQUIJOIN

Q5:

Display the employee details, departments that the departments are not same in both the emp and dept. Solution: 1. Use select from clause. 2. Use non equi join in select clause to get the result.

Ans: SQL> select \* from emp,dept where emp.deptno!=dept.deptno;  
EMPNO ENAME JOB DEPTNO SAL DEPTNO DNAME LOC

-----  
2 Arjun ASP 2 12000 1 ACCOUNTING NEW YORK  
3 Gugan ASP 2 20000 1 ACCOUNTING NEW YORK  
1 Mathi AP 1 10000 2 RESEARCH DALLAS  
EMPNO ENAME JOB DEPTNO SAL DEPTNO DNAME LOC

-----  
4 Karthik AP 1 15000 2 RESEARCH DALLAS  
1 Mathi AP 1 10000 30 SALES CHICAGO  
2 Arjun ASP 2 12000 30 SALES CHICAGO  
EMPNO ENAME JOB DEPTNO SAL DEPTNO DNAME LOC

-----  
3 Gugan ASP 2 20000 30 SALES CHICAGO  
4 Karthik AP 1 15000 30 SALES CHICAGO  
1 Mathi AP 1 10000 40 OPERATIONS BOSTON  
EMPNO ENAME JOB DEPTNO SAL DEPTNO DNAME LOC

-----  
2 Arjun ASP 2 12000 40 OPERATIONS BOSTON  
3 Gugan ASP 2 20000 40 OPERATIONS BOSTON  
4 Karthik AP 1 15000 40 OPERATIONS BOSTON  
12 rows selected.

LEFTOUT-JOIN Tables used SQL> select \* from stud1;

Regno Name Mark2 Mark3 Result

-----  
101 john 89 80 pass

102 Raja 70 80 pass

103 Sharin 70 90 pass

104 sam 90 95 pass

SQL> select \* from stud2;

NAME GRA

-----  
john s raj s sam a sharin a

Q6: Display the Student name and grade by implementing a left outer join.

Ans: SQL> select stud1.name,grade from stud1 left outer join stud2 on

stud1.name=stud2.name;                      Name Gra

-----  
john s raj

s sam a sharin a smith null      RIGHT OUTER-JOIN

Q7: Display the Student name, register no, and result by implementing a right outer join.

Ans: SQL> select stud1.name, regno, result from stud1 right outer join stud2 on stud1.name  
= stud2.name; Name Regno Result

-----  
john 101 pass

raj 102 pass

sam 103 pass

sharin 104 pass

Rollno Name Mark1 Mark2 Total

-----  
1 sindu 90 95 185

2 arul 90 90 180

FULL OUTER-JOIN

Q8: Display the Student name register no by implementing a full outer join.

Ans: SQL> select stud1.name, regno from stud1 full outer join stud2 on (stud1.name=  
stud2.name);                      Name Regno

-----  
john 101

raj 102 sam

103 sharin 104

SELFJOIN

Q9: Write a query to display their employee names

Ans: SQL> select distinct ename from emp x, dept y where x.deptno=y.deptno;

ENAME

-----  
Arjun

Gugan

Karthik

Mathi

Q10: Display the details of those who draw the salary greater than the average salary.

Ans: SQL> select distinct \* from emp x where x.sal >= (select avg(sal) from emp);

EMPNO ENAME JOB DEPTNO SAL

-----  
3 Guggan ASP 2 20000

4 Karthik AP 1 15000

11 kavitha designer 12 17000

### **DCL COMMANDS**

The DCL language is used for controlling the access to the table and hence securing the database. DCL is used to provide certain privileges to a particular user. Privileges are rights to be allocated. The privilege commands are namely, Grant and Revoke. The various privileges that can be granted or revoked are, Select Insert Delete Update References Execute All.

**GRANT COMMAND:** It is used to create users and grant access to the database. It requires database administrator (DBA) privilege, except that a user can change their password. A user can grant access to their database objects to other users.

**REVOKE COMMAND:** Using this command, the DBA can revoke the granted database privileges from the user.

### **TCL COMMAND**

**COMMIT:** command is used to save the Records.

**ROLL BACK:** command is used to undo the Records.

**SAVE POINT** command is used to undo the Records in a particular transaction.

### **Queries:**

Tables Used: Consider the following tables namely “DEPARTMENTS” and “EMPLOYEES”

Their schemas are as follows, Departments ( dept\_no, dept\_name, dept\_location );  
Employees ( emp\_id, emp\_name, emp\_salary );

Q1: Develop a query to grant all privileges of employees table into departments table

Ans: SQL> Grant all on employees to departments; Grant succeeded.

Q2: Develop a query to grant some privileges of employees table into departments table Ans:  
SQL> Grant select, update , insert on departments to departments with grant option; Grant  
succeeded.

Q3: Develop a query to revoke all privileges of employees table from departments table Ans:  
SQL> Revoke all on employees from departments; Revoke succeeded.

Q4: Develop a query to revoke some privileges of employees table from departments table  
Ans: SQL> Revoke select, update , insert on departments from departments; Revoke  
succeeded.

Q5: Write a query to implement the save point

Ans: SQL> SAVEPOINT S1;

Savepoint created.

SQL> select \* from emp;

EMPNO ENAME JOB DEPTNO SAL

-----

1 Mathi AP 1 10000

2 Arjun ASP 2 15000

3 Gungan ASP 1 15000

4 Karthik Prof 2 30000

SQL> INSERT INTO EMP VALUES(5,'Akalya','AP',1,10000); 1 row created.

SQL> select \* from emp;

EMPNO ENAME JOB DEPTNO SAL

-----

1 Mathi AP 1 10000

2 Arjun ASP 2 15000

3 Gungan ASP 1 15000

4 Karthik Prof 2 30000

5 Akalya AP 1 10000

Q6: Write a query to implement the rollback

Ans: SQL> rollback s1; SQL> select \* from emp;

EMPNO ENAME JOB DEPTNO SAL

-----  
1 Mathi AP 1 10000

2 Arjun ASP 2 15000

3 Gungan ASP 1 15000

4 Karthik Prof 2 30000

Q6: Write a query to implement the commit

Ans: SQL> COMMIT; Commit

complete.

## RESULT

Thus, the DML, DCL, TCL commands was performed successfully and executed.



**CURSOR PROGRAM FOR ELECTRICITY BILL CALCULATION:**

SQL> create table bill(name varchar2(10), address varchar2(20), city varchar2(20), unit number(10)); Table created.

SQL> insert into bill values('&name','&address','&city','&unit');

Enter value for name: yuva

Enter value for address: srivi

Enter value for city: srivilliputur Enter value for unit: 100 old

1: insert into bill values('&name','&address','&city','&unit') new

1: insert into bill values('yuva','srivi','srivilliputur','100')

1 row created.

SQL> /

Enter value for name: nithya

Enter value for address: Lakshmi nagar

Enter value for city: sivakasi Enter value for unit: 200 old 1: insert

into bill values('&name','&address','&city','&unit') new 1: insert into

bill values('nithya','Lakshmi nagar','sivakasi','200')

1 row created.

SQL> /

Enter value for name: maya

Enter value for address: housing board

Enter value for city: sivakasi Enter value for unit: 300 old 1: insert  
into bill values('&name','&address','&city','&unit') new 1: insert  
into bill values('maya','housing board','sivakasi','300')

1 row created.

SQL> /

Enter value for name: jeeva

Enter value for address: RRR nagar

Enter value for city: sivaganagai Enter value for unit: 400 old 1:

insert into bill values('&name','&address','&city','&unit') new 1:

insert into bill values('jeeva','RRR nagar','sivaganagai','400')

1 row created.

SQL> select \* from bill;

NAME	ADDRESS	CITY	UNIT
-----	-----	-----	yuva
srivi	srivilliputur	100	nithya
Lakshmi nagar	sivakasi	200	maya
housing board	sivakasi	300	jeeva
RRR nagar	sivaganagai	400	SQL>

declare

2 cursor c is select \* from bill;

3 b bill %ROWTYPE;

4 begin

5 open c;

```

6  dbms_output.put_line('Name  Address  city  Unit  Amount');
7  loop
8  fetch c into b;
9  if(c % notfound) then
10 exit;
11 else
12 if(b.unit<=100) then
13 dbms_output.put_line(b.name||'  '||b.address||'  '||b.city||'  '||b.unit||'  '||b.uni t*1);
14 elsif(b.unit>100 and b.unit<=200) then
15 dbms_output.put_line(b.name||'  '||b.address||'  '||b.city||'  '||b.unit||'  '||b.
unit*2);
16 elsif(b.unit>200 and b.unit<=300) then
17 dbms_output.put_line(b.name||'  '||b.address||'  '||b.city||'  '||b.unit||'  '||b.
unit*3);
18 elsif(b.unit>300 and b.unit<=400) then
19 dbms_output.put_line(b.name||'  '||b.address||'  '||b.city||'  '||b.unit||'  '||b.unit*
4);
20 else
21 dbms_output.put_line(b.name||'  '||b.address||'  '||b.city||'  '||b.unit||'  '||b.unit*
5);
22 end if;
23 end if;
24 end loop;
25 close c;
26 end;
27 /

```

Name	Address	city	Unit	Amount	yuva	srivi
srivilliputur	100	100	nithya	Lakshmi	nagar	
sivakasi	200	400	maya	housing	board	
sivakasi	300	900	jeeva	RRR	nagar	
sivaganagai	400	1600				

PL/SQL procedure successfully completed.

### **PROGRAM FOR STUDENT GRADE CALCULATION**

```
SQL> create table std(name varchar(10), rollno number(3), mark1 number(3), mark2
number(3), mark3 nu mber(3));
```

Table created.

```
SQL> insert into std values('&name','&rollno','&mark1','&mark2','&mark3');
```

Enter value for name: gowri

Enter value for rollno: 101

Enter value for mark1: 78

Enter value for mark2: 89 Enter value for mark3: 99 old 1: insert into std

values('&name','&rollno','&mark1','&mark2','&mark3') new 1: insert into

std values('gowri','101','78','89','99')

1 row created.

```
SQL> /
```

Enter value for name: prem

Enter value for rollno: 102

Enter value for mark1: 88

Enter value for mark2: 99 Enter value for mark3: 90 old 1: insert into std  
values('&name','&rollno','&mark1','&mark2','&mark3') new 1: insert into  
std values('prem','102','88','99','90')

1 row created.

SQL> /

Enter value for name: ravathi

Enter value for rollno: 103

Enter value for mark1: 67

Enter value for mark2: 89 Enter value for mark3: 99 old 1: insert into std  
values('&name','&rollno','&mark1','&mark2','&mark3') new 1: insert into  
std values('ravathi','103','67','89','99')

1 row created.

SQL> /

Enter value for name: arun

Enter value for rollno: 104

Enter value for mark1: 56

Enter value for mark2: 66 Enter value for mark3: 77 old 1: insert into std  
values('&name','&rollno','&mark1','&mark2','&mark3') new 1: insert into  
std values('arun','104','56','66','77')

1 row created.

SQL> set serveroutput on;

```

SQL> declare
2  tot number;
3  average number;
4  cursor c is select * from std;
5  s std %ROWTYPE;
6  begin
7  open c;
8  dbms_output.put_line('Name  Rollno  Mark1  Mark2  Mark3  Total  Average
  Grade');
9  loop
10 fetch c into s;
11 tot:=s.mark1+s.mark2+s.mark3;
12 average:=floor(tot/3);
13 if(c % notfound)then
14 exit;
15 else
16 if(s.mark1<50 or s.mark2<50 or s.mark3<50)then
17 dbms_output.put_line(s.name||'  '||s.rollno||'  '||s.mark1||'  '||s.mark2||'  '||s.mark3||
  '  '||tot||'  '||average||'  '||F');
18 elsif(average>=90 and average<=100)then
19 dbms_output.put_line(s.name||'  '||s.rollno||'  '||s.mark1||'  '||s.mark2||'  '||s.mark3||
  '  '||tot||'  '||average||'  '||S');
20 elsif(average>=80 and average<90)then
21 dbms_output.put_line(s.name||'  '||s.rollno||'  '||s.mark1||'  '||s.mark2||'  '||s.mark3||'
  '  '||tot||'  '||average||'  '||A+');
22 elsif(average>=70 and average<80)then
23 dbms_output.put_line(s.name||'  '||s.rollno||'  '||s.mark1||'  '||s.mark2||'  '||s.mark3||

```

```

' ||tot||' ||average||' '||B');
24 elsif(average>=60 and average<70)then
25 dbms_output.put_line(s.name||' '||s.rollno||' '||s.mark1||' '||s.mark2||' '||s.mark3|| '
    ||tot||' ||average||' '||C');
26 else
27 dbms_output.put_line(s.name||' '||s.rollno||' '||s.mark1||' '||s.mark2||' '||s.mark3|| '
    ||tot||' ||average||' '||D');
28 end if;
29 end if;
30 end loop;
31 close c;
32 end;
33 /

```

Name	Rollno	Mark1	Mark2	Mark3	Total	Average	Grade
gowri	101	78	89	99	266	88	A+
prem	102	88	99	90	277	92	S
ravathi	103	67	89	99	255	85	A+
arun	104	56	66	77	199	66	C

PL/SQL procedure successfully completed.

## RESULT:

Thus, the program to implement cursors was executed and output was verified successfully.

**TRIGGER FOR DISPLAYING GRADE OF THE STUDENT**

SQL> create table stdn(rollno number(3),name varchar(2),m1 number(3),m2 number(3),m3 number(3),tot num

ber(3),avrg number(3),result varchar(10)); Table

created.

SQL> create or replace trigger t1 before insert on stdn

2 for each row

3 begin

4 :new.tot:=:new.m1+:new.m2+:new.m3;

5 :new.avrg:=:new.tot/3;

6 if(:new.m1>=50 and :new.m2>=50 and :new.m3>=50) then

7 :new.result:='pass';

8 else

9 :new.result:='Fail';

10 end if;

11 end;

12 /

Trigger created.

SQL> insert into stdn values(101,'SM',67,89,99,"",""); 1

row created.

SQL> select \* from stdn;

ROLLNO	NA	M1	M2	M3	TOT	AVRG	RESULT
--------	----	----	----	----	-----	------	--------

101	SM	67	89	99	255	85	pass
-----	----	----	----	----	-----	----	------



## **PROGRAM TO INDICATE INVALID CONDITION USING TRIGGER**

```
SQL> create table emp (name varchar(10),empno number(3),age number(3));
```

Table created.

```
SQL>
```

```
1 create or replace trigger t2 before insert on emp
2 for each row
3 when(new.age>100)
4 begin
5 RAISE_APPLICATION_ERROR(-20998,'INVALID ERROR');
6* end; SQL>
/
```

Trigger created.

```
SQL> insert into emp values('nithya',101,24);
```

1 row created.

```
SQL> insert into emp values('nithya',101,103); insert into emp values('nithya',101,103)
```

\*

ERROR at line 1:

ORA-20998: INVALID ERROR

ORA-06512: at "SCOTT.T2", line 2

ORA-04088: error during execution of trigger 'SCOTT.T2'

**RESULT:**

Thus, triggers were implemented successfully.

---

**EXNO:5****PROCEDURES AND FUNCTIONS**

---

PROCEDURE TO INSERT NUMBER

SQL> create table emp1(id number(3),First\_name varchar2(20));

Table created.

SQL> insert into emp1 values(101,'Nithya');

1 row created.

SQL> insert into emp1 values(102,'Maya');

1 row created.

SQL> select \* from emp1;

ID FIRST\_NAME

-----

101 Nithya

102 Maya

SQL> set serveroutput on;

SQL> create or replace

2 procedure insert\_num(p\_num number)is

```
3 begin
4 insert into emp1(id,First_name) values(p_num,user);
5 end insert_num;
6 /
```

Procedure created.

```
SQL> exec insert_num(3);
```

PL/SQL procedure successfully completed.

```
SQL> select * from emp1;
```

ID	FIRST_NAME
----	------------

101	Nithya
102	Maya
103	SCOTT

### **FUNCTION TO FIND FACTORIAL**

```
SQL> create or replace function fact(n number)
```

```
2 return number is
```

```
3 i number(10);
```

```
4 f number:=1;
```

```
5 begin
```

```
6 for i in 1..N loop
```

```
7 f:=f*i;
```

```
8  end loop;  
9  return f;  
10 end;  
11 /
```

Function created.

```
SQL> select fact(2) from dual;
```

```
FACT(2)
```

```
-----
```

```
2
```

### **RESULT:**

Thus, procedures and functions were implemented successfully.

---

**EX.NO:6**

**IMPLEMENTATION OF EMBEDDED SQL**

---

**PL/SQL PROGRAM FOR BONUS CALCULATION**

SQL> set serveroutput on;

SQL> declare

2 salary number;

3 bonus number;

4 begin

5 salary:=&sa;

6 if salary>5000 then

7 bonus:=salary\*0.5;

8 else

9 bonus:=0;

10 end if;

11 dbms\_output.put\_line(bonus);

12 end;

13 /

Enter value for sa: 10000

old 5: salary:=&sa; new

5: salary:=10000;

5000

PL/SQL procedure successfully completed.

### **PROGRAM FOR ARMSTRONG NUMBER**

SQL> set serveroutput on;

SQL> declare

2 a number;

3 b number;

4 i number;

5 begin

6 i:=&num;

7 a:=i;

8 b:=0;

9 while a>0

10 loop

11 b:=b+power(mod(a,10),3);

12 a:=trunc(a/10);

13 end loop;

14 if b=i then

15 dbms\_output.put\_line(i||'IS AN ARMSTRONG NUMBER');

16 else

17 dbms\_output.put\_line(i||'IS NOT AN ARMSTRONG NUMBER');

18 end if;

19 end;

20 /

Enter value for num: 123

old 6: i:=&num; new 6:

i:=123;

123 IS NOT AN ARMSTRONG NUMBER

PL/SQL procedure successfully completed.

SQL> / Enter value for

num: 407 old 6:

i:=&num; new 6:

i:=407;

407IS AN ARMSTRONG NUMBER

PL/SQL procedure successfully completed.

### **PROGRAM FOR MULTIPLICATION TABLE:**

SQL> set serveroutput on;

SQL> declare

2 a number;

3 b number;

4 i number;

5 n number;

6 s number;

7 begin

8 a:=&ulimit;

9 b:=&llimit;

10 n:=&n;

11 for i in a..b loop

12 s:=i\*n;

13 dbms\_output.put\_line(i||'\*'||n||'='||s);

14 end loop;

15 end;

16 /

Enter value for ulimit: 1

old 8: a:=&ulimit; new

8: a:=1; Enter value for

llimit: 10 old 9:

b:=&llimit; new 9:

b:=10; Enter value for n:

5

old 10: n:=&n; new

10: n:=5; 1\*5=5

2\*5=10

3\*5=15

4\*5=20

5\*5=25

6\*5=30

7\*5=35

8\*5=40

9\*5=45

10\*5=50

PL/SQL procedure successfully completed.

### **RESULT:**

Thus, Embedded SQL was executed successfully.



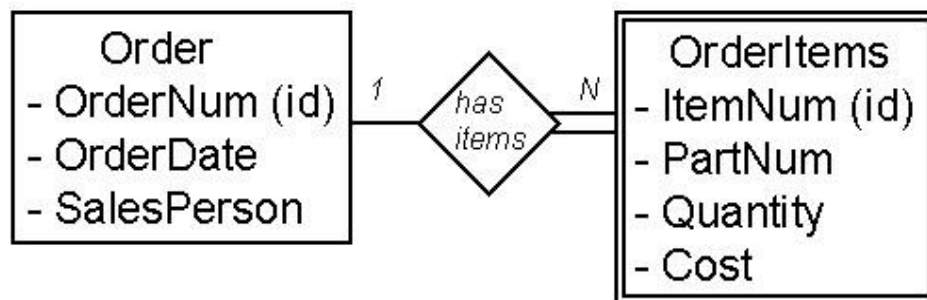
---

## EX NO:7 Database design using E-R model and Normalization

---

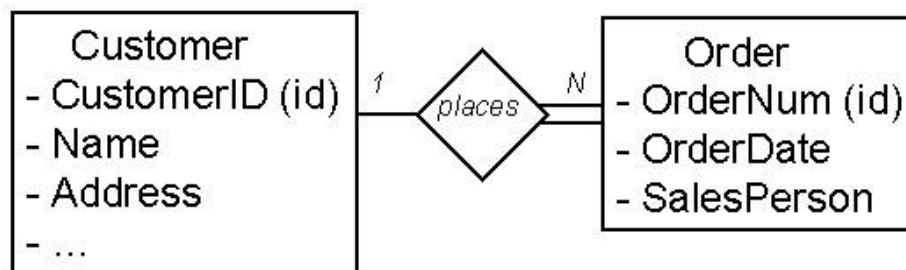
ER diagram:

Chen Notation



- **ORDER** (OrderNum (key), OrderDate, SalesPerson)
- **ORDERITEMS** (OrderNum (key)(fk), ItemNum (key), PartNum, Quantity, Cost)
- In the above example, in the ORDERITEMS Relation: OrderNum is the *Foreign Key* and OrderNum plus ItemNum is the *Composite Key*.

Chen Notation



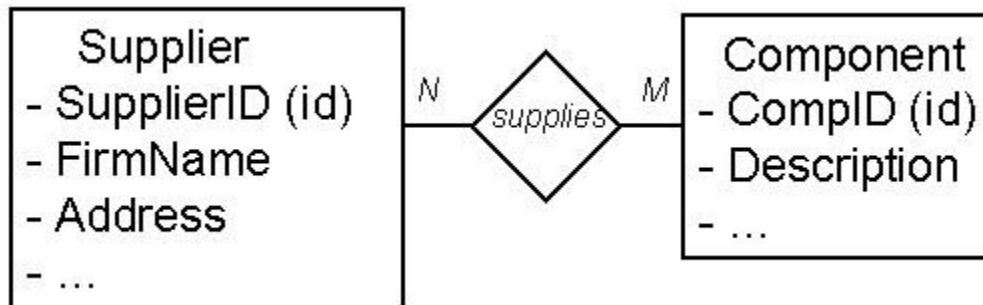
In the ORDER Relation: OrderNum is the *Key*.

### Representing Relationships

- **1:1 Relationships.** The key of one relation is stored in the second relation. Look at example queries to determine which key is queried most often.
- **1:N Relationships.**
  - Parent** - Relation on the "1" side.
  - Child** - Relation on the "Many" side.
- Represent each Entity as a relation.

- Copy the key of the parent into the child relation.
- CUSTOMER (CustomerID (key), Name, Address, ...)**  
**ORDER (OrderNum (key), OrderDate, SalesPerson, CustomerID (fk))**
- M:N Relationships.** Many to Many relationships can not be directly implemented in relations.
- Solution: Introduce a third *Intersection relation* and copy keys from original two relations.

### Chen Notation



- SUPPLIER (SupplierID (key), FirmName, Address, ...)**  
**COMPONENT (CompID (key), Description, ...)**  
**SUPPLIER\_COMPONENT (SupplierID (key), CompID (key))**
- Note that this can also be shown in the ER diagram. Also, look for potential added attributes in the intersection relation.

### RESULT:

Thus, the ER Database design using E-R model and Normalization was implemented successfully.

---

**EX NO: 8    DATABASE DESIGN AND IMPLEMENTATION PAY ROLL  
PROCESSING**

---

**STEPS:**

1. Create a database for payroll processing which request the using SQL
2. Establish ODBC connection
3. In the administrator tools open data source ODBC
4. Click add button and select oracle in ORA home 90, click finish
5. A window will appear given the data source home as oracle and select TNS source name as lion and give the used id as SWTT
6. ADODC CONTROL FOR SALARY FORM:-
7. The above procedure must be follow except the table , A select the table as salary
8. Write appropriate Program in form each from created in VB from each from created in VB form project.

SQL>create table emp(eno number primary key,enamr varchar(20),age number,addr varchar(20),DOB date,phno number(10)); Table created.

SQL>create table salary(eno number,edesig varchar(10),basic number,da number,hra number,pf number,mc number,met number,foreign key(eno) references emp); Table created.

TRIGGER to calculate DA,HRA,PF,MC

SQL> create or replace trigger employ

2 after insert on salary

3 declare

4 cursor cur is select eno,basic from salary;

5 begin

6 for cur1 in cur loop

7 update salary set

8 hra=basic\*0.1,da=basic\*0.07,pf=basic\*0.05,mc=basic\*0.03 where hra=0; 9 end loop;

10 end;

11 / Trigger created.

PROGRAM FOR FORM 1

Private Sub emp\_Click() Form

2.Show End Sub

Private

```

Sub exit_Click() Unload
Me
End Sub Private
Sub salary_Click()
Form3.Show
End Sub
PROGRAM FOR FORM 2
Private Sub add_Click()
Adodc1.Recordset.AddNew MsgBox "Record added"

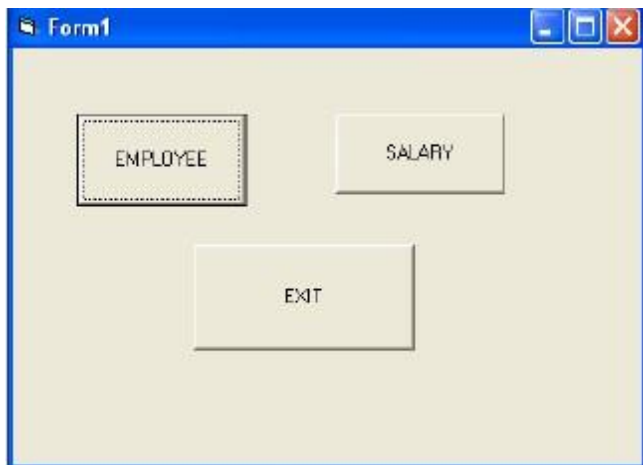
End Sub Private
Sub clear_Click()
Text1.Text = ""
Text2.Text = ""
Text3.Text = ""
Text4.Text = ""
Text5.Text = ""
Text6.Text = ""
End Sub Private Sub delte_Click()
Adodc1.Recordset.Delete MsgBox "Record Deleted"
If Adodc1.Recordset.EOF = True
Then Adodc1.Recordset.MovePrevious
End If
End
Sub Private Sub exit_Click()
Unload Me
End Sub
Private Sub main_Click()
Form1.Show
End Sub
Private Sub modify_Click()
Adodc1.Recordset.Update
End Sub
PROGRAM FOR FORM 3
Private Sub add_Click()
Adodc1.Recordset.AddNew MsgBox "Record added"
End Sub Private
Sub
clear_Click()
Text1.Text = ""
Text2.Text = ""
Text3.Text = ""
Text4.Text = ""
Text5.Text = ""
Text6.Text = ""
End Sub

```

```

Private Sub delte_Click()
Adodc1.Recordset.Delete MsgBox "Record Deleted"
If Adodc1.Recordset.EOF = True
Then Adodc1.Recordset.MovePrevious
End If
End Sub
Private Sub exit_Click()
Unload Me
End Sub
Private Sub main_Click()
Form1.Show
End Sub Private Sub
modify_Click()
Adodc1.Recordset.Update
End Sub Output:

```



Form2

Emp No	<input type="text" value="1"/>	
Emp Name	<input type="text" value="Rose"/>	
Age	<input type="text" value="20"/>	<input type="button" value="⏮"/> <input type="button" value="⏪"/> <input type="text" value="Adcdc1"/> <input type="button" value="⏩"/> <input type="button" value="⏭"/>
Address	<input type="text" value="Trichy"/>	
D.O.B	<input type="text" value="3/12/1988"/>	
Ph.No	<input type="text" value="98145789"/>	

Form3

EMP ID	<input type="text" value="2"/>	HRA	<input type="text" value="850"/>
DESIGNATION	<input type="text" value="S.LECT"/>	PF	<input type="text" value="425"/>
BASIC	<input type="text" value="8500"/>	MC	<input type="text" value="255"/>
DA	<input type="text" value="585"/>	NET SALARY	<input type="text" value="10625"/>

### RESULT:

Thus, payroll system was designed and implemented successfully.

**DETAILS OF THE STEP**

- 1.Create the DB for banking system source request the using SQL
- 2.Establishing ODBC connection
- 3.Click add button and select oracle in ORA home 90 click finished
- 4.A window will appear give the data source name as oracle and give the user id as scott
- 5.Now click the test connection a window will appear with server and user name give user as scott and password tiger Click ok
- 6.VISUAL BASIC APPLICATION:-
  - Create standard exe project in to and design ms from in request format
  - To add ADODC project select component and check ms ADO data control click ok
  - Now the control is added in the tool book
  - Create standard exe project in to and design ms from in request format
- 7ADODC CONTEOL FOR ACCOUNT FROM:- Click customs and property window and window will appear and select ODBC data source name as oracle and click apply as the some window.

**CREATE A TABLE IN ORACLE**

SQL>create table account(cname varchar(20),accno number(10),balance number);

Table Created

SQL> insert into account values('&cname',&accno,&balance);

Enter value for cname: Mathi

Enter value for accno: 1234 Enter

value for balance: 10000

old 1: insert into account values('&cname',&accno,&balance) new

1: insert into emp values('Mathi',1234,10000) 1 row created.

**SOURCE CODE FOR FORM1**

```
Private Sub ACCOUNT_Click()
```

```
Form2.Show
```

```
End Sub
```

```
Private Sub
```

```
EXIT_Click()
```

```
Unload Me
```

```
End Sub
```

```
Private Sub
```

```
TRANSACTION_Click() Form3.Show
```

```
End Sub
```

**SOURCE CODE FOR FORM 2**

```
Private Sub CLEAR_Click()
```

```

Text1.Text = "" Text2.Text
= ""
Text3.Text = ""
End Sub
Private Sub
DELETE_Click()
Adodc1.Recordset.DELETE MsgBox "record deleted"
Adodc1.Recordset.MoveNext If Adodc1.Recordset.EOF = True Then
Adodc1.Recordset.MovePrevious
End If
End Sub
Private Sub EXIT_Click()
Unload Me
End Sub
Private Sub
HOME_Click()
Form1.Show
End Sub
Private Sub
INSERT_Click() Adodc1.Recordset.AddNew
End Sub
Private Sub
TRANSACTION_Click()
Form3.Show
End Sub
Private Sub UPDATE_Click() Adodc1.Recordset.UPDATE MsgBox "record updated
successfully"
End Sub
SOURCE CODE FOR FORM 3
Private Sub ACCOUNT_Click()
Form2.Show
End Sub
Private Sub CLEAR_Click()
Text1.Text = ""
Text2.Text = ""
End Sub
Private Sub
DEPOSIT_Click()
Dim s As String s = InputBox("enter the amount to be deposited")
Text2.Text = Val(Text2.Text) + Val(s) A = Text2.Text MsgBox "CURRENT BALANCE IS
Rs" + Str(A) Adodc1.Recordset.Save Adodc1.Recordset.UPDATE
End Sub
Private Sub
EXIT_Click()
Unload Me
End Sub

```



```

Private Sub
HOME_Click()
Form1.Show End
Sub Private Sub
WITHDRAW_Click()
Dim s As String s = InputBox("enter the amount to be deleted")
Text2.Text = Val(Text2.Text) - Val(s) A = Text2.Text MsgBox "current balance is Rs" +
Str(A)
Adodc1.Recordset.Save
Adodc1.Recordset.UPDATE
End Sub

```

Form1: BANK MANAGEMENT DETAILS

Buttons: ACCOUNT details, TRANSACTION, EXIT

Form2

NAME: malathi

ACCNO: 116

BALANCE: 10900

Buttons: INSERT, UPDATE, DELETE

Record Selector: Adodc1

Buttons: HOME, TRANSACTION, EXIT, CLEAR

Form3

Accno: 112

balance: 20000

DEPOSIT WITHDRAW CLEAR

ACCOUNT HOME EXIT

Adodc1

Project1

enter the amount to be deposited

OK Cancel

200

Project1

CURRENT BALANCE IS Rs 20200

OK

Result:

Thus, the banking system was designed and implemented successfully.

**STEPS:**

1. Create a database for library which request the using SQL
2. Establish ODBC connection
3. In the administrator tools open data source ODBC
4. Click add button and select oracle in ORA home 90, click finish
5. A window will appear given the data source home as oracle and select TNS source name as lion and give the used id as SWTT
6. ADODC CONTROL FOR library FORM:-
7. The above procedure must be follow except the table , A select the table as library
8. Write appropriate Program in form each from created in VB from each from created in VB form project.

Relational Database Schema							
Status	code	description					
Media	media_id	code					
Book	ISBN	title	author	year	dewey	price	
BookMedia	media_id	ISBN					
Customer	ID	name	addr	DOB	phone	username	password
Card	num	fin	ID				
Checkout	media_id	num	since	until			
Location	name	addr	phone				
Hold	media_id	num	name	until	queue		
Stored_In	media_id	name					
Librarian	eid	ID	Pay	name	since		
Video	title	year	director	rating	price		
VideoMedia	media_id	title	year				

```
CREATE TABLE Status ( code INTEGER, description CHAR(30), PRIMARY KEY
(code) );
```

```
CREATE TABLE Media( media_id INTEGER, code INTEGER, PRIMARY KEY
(media_id),
```

```
FOREIGN KEY (code) REFERENCES Status );
```

```
CREATE TABLE Book(ISBNCHAR(14), title CHAR(128), author CHAR(64), year
INTEGER, dewey INTEGER, price REAL, PRIMARY KEY (ISBN) );
```

```
CREATE TABLE BookMedia( media_id INTEGER, ISBN CHAR(14), PRIMARY
KEY (media_id),
```

```
FOREIGN KEY (media_id) REFERENCES Media,
```

```
FOREIGN KEY (ISBN) REFERENCES Book);
```

```
CREATE TABLE Customer( ID INTEGER, name CHAR(64), addr CHAR(256),
DOB CHAR(10), phone CHAR(30), username CHAR(16), password CHAR(32),
PRIMARY KEY
(ID),
```

```
UNIQUE (username) );
```

```
CREATE TABLE Card( num INTEGER, fines REAL, ID INTEGER, PRIMARY
KEY (num),
```

```
FOREIGN KEY (ID) REFERENCES Customer );
```

```
CREATE TABLE Checkout( media_id INTEGER, num INTEGER, since CHAR(10),
until CHAR(10), PRIMARY KEY (media_id), FOREIGN KEY (media_id)
```

```
REFERENCES Media,
```

```
FOREIGN KEY (num) REFERENCES Card );
```

```
CREATE TABLE Location( name CHAR(64), addr CHAR(256), phone CHAR(30),
PRIMARY KEY (name) );
```

```
CREATE TABLE Hold( media_id INTEGER, num INTEGER, name CHAR(64),
until CHAR(10),
```

```

queue INTEGER, PRIMARY KEY (media_id, num), FOREIGN
KEY (name) REFERENCES Location, FOREIGN KEY (num)
REFERENCES Card,

FOREIGN KEY (media_id) REFERENCES Media );

CREATE TABLE Stored_In( media_id INTEGER, name char(64), PRIMARY KEY
(media_id),

FOREIGN KEY (media_id) REFERENCES Media ON DELETE CASCADE,

FOREIGN KEY (name) REFERENCES Location );

CREATE TABLE Librarian( eid INTEGER, ID INTEGER NOT NULL, Pay REAL,
Loc_name CHAR(64) NOT NULL, PRIMARY KEY (eid),

FOREIGN KEY (ID) REFERENCES Customer ON DELETE CASCADE,

FOREIGN KEY (Loc_name) REFERENCES Location(name) );

CREATE TABLE Video( title CHAR(128), year INTEGER, director CHAR(64),
rating REAL, price REAL, PRIMARY KEY (title, year) );

CREATE TABLE VideoMedia( media_id INTEGER, title CHAR(128), year
INTEGER,

PRIMARY KEY (media_id), FOREIGN KEY (media_id) REFERENCES Media,

FOREIGN KEY (title, year) REFERENCES Video );

INSERT INTO Customer(ID, name, addr, DOB, phone, username, password)
VALUES

(60201, 'Jason L. Gray', '2087 Timberbrook Lane, Gypsum, CO 81637',
'09/09/1958', '970-273-9237', 'jlgray', 'password1');

INSERT INTO Customer(ID, name, addr, DOB, phone, username, password)
VALUES

(89682, 'Mary L. Prieto', '1465 Marion Drive, Tampa, FL 33602',
'11/20/1961', '813-487-4873', 'mlprieto', 'password2');

INSERT INTO Customer(ID, name, addr, DOB, phone, username, password)
VALUES

(64937, 'Roger Hurst', '974 Bingamon Branch Rd, Bensenville, IL 60106',
'08/22/1973', '847-221-4986', 'rhurst', 'password3');

```

```
INSERT INTO Customer(ID, name, addr, DOB, phone, username, password)
VALUES
```

```
(31430, 'Warren V. Woodson', '3022 Lords Way, Parsons, TN 38363',
'03/07/1945', '731-845-0077', 'wvwoodson', 'password4');
```

```
INSERT INTO Customer(ID, name, addr, DOB, phone, username, password)
VALUES
```

```
(79916, 'Steven Jensen', '93 Sunny Glen Ln, Garfield Heights, OH 44125',
'12/14/1968', '216-789-6442', 'sjensen', 'password5');
```

```
INSERT INTO Customer(ID, name, addr, DOB, phone, username, password)
VALUES
```

```
(93265, 'David Bain', '4356 Pooh Bear Lane, Travelers Rest, SC 29690',
'08/10/1947', '864-610-9558', 'dbain', 'password6');
```

```
INSERT INTO Customer(ID, name, addr, DOB, phone, username, password)
VALUES
```

```
(58359, 'Ruth P. Alber', '3842 Willow Oaks Lane, Lafayette, LA 70507',
'02/18/1976', '337-316-3161', 'rpalber', 'password7');
```

```
INSERT INTO Customer(ID, name, addr, DOB, phone, username, password)
VALUES
```

```
(88564, 'Sally J. Schilling', '1894 Wines Lane, Houston, TX 77002',
'07/02/1954', '832-366-9035', 'sjschilling', 'password8');
```

```
INSERT INTO Customer(ID, name, addr, DOB, phone, username, password)
VALUES
```

```
(57054, 'John M. Byler', '279 Raver Croft Drive, La Follette, TN 37766',
'11/27/1954', '423-592-8630', 'jmbyler', 'password9');
```

```
INSERT INTO Customer(ID, name, addr, DOB, phone, username, password)
VALUES
```

```
(49312, 'Kevin Spruell', '1124 Broadcast Drive, Beltsville, VA 20705',
'03/04/1984', '703-953-1216', 'kspruell', 'password10');
```

```
INSERT INTO Card(num, fines, ID) VALUES ( 5767052, 0.0, 60201);
```

```
INSERT INTO Card(num, fines, ID) VALUES ( 5532681, 0.0, 60201); INSERT
INTO Card(num, fines, ID) VALUES ( 2197620, 10.0, 89682);
```

```
INSERT INTO Card(num, fines, ID) VALUES ( 9780749, 0.0, 64937);
```

```
INSERT INTO Card(num, fines, ID) VALUES ( 1521412, 0.0, 31430);
INSERT INTO Card(num, fines, ID) VALUES ( 3920486, 0.0, 79916);
INSERT INTO Card(num, fines, ID) VALUES ( 2323953, 0.0, 93265);
INSERT INTO Card(num, fines, ID) VALUES ( 4387969, 0.0, 58359);
INSERT INTO Card(num, fines, ID) VALUES ( 4444172, 0.0, 88564);
INSERT INTO Card(num, fines, ID) VALUES ( 2645634, 0.0, 57054);
INSERT INTO Card(num, fines, ID) VALUES ( 3688632, 0.0, 49312);
INSERT INTO Location(name, addr, phone) VALUES ('Texas Branch',
'4832 Deercove Drive, Dallas, TX 75208', '214-948-7102');
INSERT INTO Location(name, addr, phone) VALUES ('Illinois Branch',
'2888 Oak Avenue, Des Plaines, IL 60016', '847-953-8130');
INSERT INTO Location(name, addr, phone) VALUES ('Louisiana Branch',
'2063 Washburn Street, Baton Rouge, LA 70802', '225-346-0068');
INSERT INTO Status(code, description) VALUES (1, 'Available');
INSERT INTO Status(code, description) VALUES (2, 'In Transit');
INSERT INTO Status(code, description) VALUES (3, 'Checked Out');
INSERT INTO Status(code, description) VALUES (4, 'On Hold');
INSERT INTO Media( media_id, code) VALUES (8733, 1);
INSERT INTO Media( media_id, code) VALUES (9982, 1);
INSERT INTO Media( media_id, code) VALUES (3725, 1);
INSERT INTO Media( media_id, code) VALUES (2150, 1);
INSERT INTO Media( media_id, code) VALUES (4188, 1);
INSERT INTO Media( media_id, code) VALUES (5271, 2);
INSERT INTO Media( media_id, code) VALUES (2220, 3);
INSERT INTO Media( media_id, code) VALUES (7757, 1);
INSERT INTO Media( media_id, code) VALUES (4589, 1);
INSERT INTO Media( media_id, code) VALUES (5748, 1);
INSERT INTO Media( media_id, code) VALUES (1734, 1);
```

```

INSERT INTO Media( media_id, code) VALUES (5725, 1);
INSERT INTO Media( media_id, code) VALUES (1716, 4);
INSERT INTO Media( media_id, code) VALUES (8388, 1);
INSERT INTO Media( media_id, code) VALUES (8714, 1);
INSERT INTO Book(ISBN, title, author, year, dewey, price) VALUES
('978-0743289412', 'Lisey's Story', 'Stephen King',
2006, 813, 10.0);
INSERT INTO Book(ISBN, title, author, year, dewey, price) VALUES
('978-1596912366', 'Restless: A Novel', 'William Boyd',
2006, 813, 10.0);
INSERT INTO Book(ISBN, title, author, year, dewey, price) VALUES
('978-0312351588', 'Beachglass', 'Wendy Blackburn',
2006, 813, 10.0);
INSERT INTO Book(ISBN, title, author, year, dewey, price) VALUES
('978-0156031561', 'The Places In Between', 'Rory Stewart',
2006, 910, 10.0);
INSERT INTO Book(ISBN, title, author, year, dewey, price) VALUES
('978-0060583002', 'The Last Season', 'Eric Blehm',
2006, 902, 10.0);
INSERT INTO Book(ISBN, title, author, year, dewey, price) VALUES
('978-0316740401', 'Case Histories: A Novel', 'Kate Atkinson',
2006, 813, 10.0);
INSERT INTO Book(ISBN, title, author, year, dewey, price) VALUES
('978-0316013949', 'Step on a Crack', 'James Patterson, et al.',
2007, 813, 10.0);
INSERT INTO Book(ISBN, title, author, year, dewey, price) VALUES
('978-0374105235', 'Long Way Gone: Memoirs of a Boy Soldier',
'Ishmael Beah', 2007, 916, 10.0);

```



```

INSERT INTO Book(ISBN, title, author, year, dewey, price) VALUES
('978-0385340229', 'Sisters', 'Danielle Steel', 2006, 813, 10.0);

INSERT INTO BookMedia(media_id, ISBN) VALUES (8733, '978-0743289412');
INSERT INTO BookMedia(media_id, ISBN) VALUES (9982, '978-1596912366');
INSERT INTO BookMedia(media_id, ISBN) VALUES (3725, '978-1596912366');
INSERT INTO BookMedia(media_id, ISBN) VALUES (2150, '978-0312351588');
INSERT INTO BookMedia(media_id, ISBN) VALUES (4188, '978-0156031561');
INSERT INTO BookMedia(media_id, ISBN) VALUES (5271, '978-0060583002');
INSERT INTO BookMedia(media_id, ISBN) VALUES (2220, '978-0316740401');
INSERT INTO BookMedia(media_id, ISBN) VALUES (7757, '978-0316013949');
INSERT INTO BookMedia(media_id, ISBN) VALUES (4589, '978-0374105235');
INSERT INTO BookMedia(media_id, ISBN) VALUES (5748, '978-0385340229');

INSERT INTO Checkout(media_id, num, since, until) VALUES
(2220, 9780749, '02/15/2007', '03/15/2007');

INSERT INTO Video(title, year, director, rating, price) VALUES
('Terminator 2: Judgment Day', 1991, 'James Cameron', 8.3, 20.0);

INSERT INTO Video(title, year, director, rating, price) VALUES
('Raiders of the Lost Ark', 1981, 'Steven Spielberg', 8.7, 20.0);

INSERT INTO Video(title, year, director, rating, price) VALUES
('Aliens', 1986, 'James Cameron', 8.3, 20.0);

INSERT INTO Video(title, year, director, rating, price) VALUES
('Die Hard', 1988, 'John McTiernan', 8.0, 20.0);

INSERT INTO VideoMedia(media_id, title, year) VALUES
( 1734, 'Terminator 2: Judgment Day', 1991);

INSERT INTO VideoMedia(media_id, title, year) VALUES
( 5725, 'Raiders of the Lost Ark', 1981);

INSERT INTO VideoMedia(media_id, title, year) VALUES
( 1716, 'Aliens', 1986);

```

```

INSERT INTO VideoMedia(media_id, title, year) VALUES
( 8388, 'Aliens', 1986);

INSERT INTO VideoMedia(media_id, title, year) VALUES
( 8714, 'Die Hard', 1988);

INSERT INTO Hold(media_id, num, name, until, queue) VALUES
(1716, 4444172, 'Texas Branch', '02/20/2008', 1);

INSERT INTO Librarian(eid, ID, pay, Loc_name) Values
(2591051, 88564, 30000.00, 'Texas Branch');

INSERT INTO Librarian(eid, ID, pay, Loc_name) Values
(6190164, 64937, 30000.00, 'Illinois Branch');

INSERT INTO Librarian(eid, ID, pay, Loc_name) Values
(1810386, 58359, 30000.00, 'Louisiana Branch');

INSERT INTO Stored_In(media_id, name) VALUES(8733, 'Texas Branch');
INSERT INTO Stored_In(media_id, name) VALUES(9982, 'Texas Branch');
INSERT INTO Stored_In(media_id, name) VALUES(1716, 'Texas Branch');
INSERT INTO Stored_In(media_id, name) VALUES(1734, 'Texas Branch');
INSERT INTO Stored_In(media_id, name) VALUES(4589, 'Texas Branch');
INSERT INTO Stored_In(media_id, name) VALUES(4188, 'Illinois Branch');
INSERT INTO Stored_In(media_id, name) VALUES(5271, 'Illinois Branch');
INSERT INTO Stored_In(media_id, name) VALUES(3725, 'Illinois Branch');
INSERT INTO Stored_In(media_id, name) VALUES(8388, 'Illinois Branch');
INSERT INTO Stored_In(media_id, name) VALUES(5748, 'Illinois Branch');
INSERT INTO Stored_In(media_id, name) VALUES(2150, 'Louisiana Branch');

INSERT INTO Stored_In(media_id, name) VALUES(8714, 'Louisiana Branch');
INSERT INTO Stored_In(media_id, name) VALUES(7757, 'Louisiana Branch');
INSERT INTO Stored_In(media_id, name) VALUES(5725, 'Louisiana Branch');

```

```

SELECT C.ID, C.name, C.addr, C.DOB, C.phone, C.username, nvl((SELECT
'Librarian'

```

```

FROM Librarian L

WHERE L.ID = C.ID), 'Customer') AS role

FROM Customer C

WHERE C.username = <user input> AND C.password = <user input>;

/* Book search for customers */

SELECT B.ISBN, B.title, B.author, B.year,

(SELECT COUNT(*)

FROM BookMedia BM

WHERE BM.ISBN = B.ISBN AND BM.code = 1) AS num_available

FROM Book B

WHERE B.title LIKE '%<user input>%' AND B.author LIKE '%<user input>%' AND

B.year <= <user input> AND B.year >= <user input>;

/* Find all copies of a book (used for placing holds or viewing detailed information).

*/

SELECT BM.media_id, S.description, nvl((SELECT

SI.name

FROM Stored_In SI

WHERE SI.media_id = BM.media_id), 'none') AS name

FROM BookMedia BM, Media M, Status S

WHERE BM.ISBN = <user input> AND M.media_id = BM.media_id AND S.code =

M.code;

/* Video search for customers */

SELECT V.title, V.year, V.director, V.rating

(SELECT COUNT(*)

FROM VideoMedia VM

WHERE VM.ID = V.ID AND VM.code = 1) AS num_available

FROM Video V

WHERE V.title LIKE '%<user input>%' AND V.year <= <user input> AND V.year <= <user

input>

```

```
AND V.director LIKE '%" <user input> %' AND V.rating >= <user input>; /*
```

Find all copies of a video (used for placing holds or viewing detailed information). \*/

```
SELECT VM.media_id, S.description, nvl((SELECT
SI.name
FROM Stored_In SI
WHERE SI.media_id = VM.media_id), 'none') AS name
FROM VideoMedia VM, Media M, Status S
WHERE VM.title = <user input> AND VM.year = <user input> AND
M.media_id = VM.media_id AND S.code = M.code;
```

/\* Find the status of a given media item \*/

```
SELECT S.description
FROM Status S, Media M
WHERE S.code = M.code AND M.media_id = <user input>;
```

/\* Create a new Hold \*/

```
INSERT INTO Hold(media_id, num, name, until, queue) VALUES
(<user input>, <user input>, <user input>, <user input>,
nvl((SELECT MAX(H.queue)
FROM Hold H
WHERE H.media_id = <user input>), 0) + 1 );
```

/\* Cancel Hold, Step 1: Remove the entry from hold \*/

```
DELETE FROM Hold
WHERE media_id = <user input> AND num = <user input>
```

/\* Cancel Hold, Step 2: Update queue for this item \*/

```
UPDATE Hold
SET queue = queue-1
WHERE media_id = <user input> AND queue > <user input>;
```

/\* Functions needed to view information about a customer \*/

/\* View the customer's card(s) \*/

SELECT CR.num, CR.fines

FROM Card CR

WHERE CR.ID = <user input>;

/\* View media checked out on a given card \*/

SELECT B.title, B.author, B.year, BM.media\_id, CO.since, CO.until

FROM Checkout CO, BookMedia BM, Book B

WHERE CO.num = <user input> AND CO.media\_id = BM.media\_id AND B.ISBN =  
BM.ISBN

UNION

SELECT V.title, V.director, V.year, VM.media\_id, CO.since, CO.until

FROM Checkout CO, VideoMedia VM, Book B

WHERE CO.num = <user input> AND CO.media\_id = VM.media\_id AND

VM.title = V.title AND VM.year = V.year;

/\* View media currently on hold for a given card \*/

SELECT B.title, B.author, B.year, BM.media\_id, H.until, H.queue, SI.name

FROM Hold H, BookMedia BM, Book B, Stored\_In SI

WHERE H.num = <user input> AND H.media\_id = BM.media\_id AND B.ISBN =  
BM.ISBN

AND SI.media\_id = H.media\_id

UNION

SELECT V.title, V.director, V.year, VM.media\_id, H.until, H.queue, SI.name

FROM Hold H, VideoMedia VM, Book B, Stored\_In SI

WHERE H.num = <user input> AND H.media\_id = VM.media\_id AND

VM.title = V.title AND VM.year = V.year AND SI.media\_id = H.media\_id;

/\* View the total amount of fines the customer has to pay \*/

SELECT SUM(CR.fines)

FROM Card CR

WHERE CR.ID = <user input>;

```
/* *\
```

Functions reserved for librarians

```
\* */
```

```
/* Add new customer */
```

```
INSERT INTO Customer(ID, name, addr, DOB, phone, username, password) VALUES  
(<user input>, <user input>, <user input>, <user input>, <user input>,  
<user input>, <user input>, );
```

```
/* Find a customer */
```

```
SELECT C.ID, C.name, C.addr, C.DOB, C.phone, C.username, nvl((SELECT  
'Librarian'  
FROM Librarian L  
WHERE L.ID = C.ID), 'Customer') AS role  
FROM Customer C
```

```
WHERE C.username = <user input> AND C.name LIKE '%<user input>%';
```

```
/* Add new card and assign it to a customer */
```

```
INSERT INTO Card(num, fines, ID) VALUES ( <user input>, 0, <user input>);
```

```
/* Create an entry in Checkout */
```

```
INSERT INTO Checkout(media_id, num, since, until) VALUES  
(<user input>, <user input>, <user input>, <user input>);
```

```
/* Remove the entry for Stored_In */
```

```
DELETE FROM Stored_In  
WHERE media_id = <user input>;
```

```
/* Change the status code of the media */
```

```
UPDATE Media  
SET code = <user input>  
WHERE media_id = <user input>;
```

```
/* Remove the entry from Checkout */
```

```
DELETE FROM Checkout
```

```

WHERE media_id = <user input>;

/* Create the entry in Stored_In */

INSERT INTO Stored_In(media_id, name) VALUES (<user input>, <user input>);

/* Find the next Hold entry for a given media */

SELECT H.num, H.name, H.until

FROM Hold H

WHERE H.queue = 1 AND H.media_id = <user input>;

/* Change the Stored_In entry to the target library branch */

UPDATE Stored_In

SET name = <user input>

WHERE media_id = <user input>;

/* Find the customer that should be notified about book arrival */

SELECT C.name, C.phone, CR.num

FROM Customer C, Card CR, Hold H

WHERE H.queue = 1 AND H.name = <user input> AND H.media_id = <user input> AND

CR.num = H.num AND C.ID = CR.ID;

/* Add a new entry into the Book table */

INSERT INTO Book(ISBN, title, author, year, dewey, price) VALUES

(<user input>, <user input>, <user input>, <user input>, <user input>,

<user input>);

/* Add a new entry into the Video table */

INSERT INTO Video(title, year, director, rating, price) VALUES

(<user input>, <user input>, <user input>, <user input>, <user input>);

/* Add a new Media object */

INSERT INTO Media( media_id, code) VALUES (<user input>, 1);

/* Add a new BookMedia object */

INSERT INTO BookMedia(media_id, ISBN) VALUES (<user input>, <user input>);

/* Add a new VideoMedia object */

```

```
INSERT INTO VideoMedia(media_id, title, year) VALUES
```

```
(<user input>, <user input>, <user input>);
```

```
/* Remove an entry from the BookMedia table */
```

```
DELETE FROM BookMedia
```

```
WHERE media_id = <user input>;
```

```
/* Remove an entry from the VideoMedia table */
```

```
DELETE FROM VideoMedia
```

```
WHERE media_id = <user input>;
```

```
/* Remove an entry from the Media table */
```

```
DELETE FROM Media
```

```
WHERE media_id = <user input>;
```

```
/* Remove an entry from the Book table */
```

```
DELETE FROM Book
```

```
WHERE ISBN = <user input>;
```

```
/* Remove an entry from the Video table */
```

```
DELETE FROM Video
```

```
WHERE title = <user input> AND year = <user input>;
```

```
/* Update the customer's fines */
```

```
UPDATE Card
```

```
SET fines = <user input>
```

```
WHERE num = <user input>
```



MDIForm1 - [BOOKS]

Books Entry User Entry Issues of Books Exit

## *Books*

Book No.	3
ISBN No.	2568956
Subject	Mathematics
Name Of The Book	Trigonometry
Author	Loni
Publisher	Moon Light
Edition	2003
Copies	5
Cost	150

MDIForm1 - [Form1]

Books Entry User Entry Issues of Books Exit

## *ISSUES OF BOOKS*

Book No.	3
Student ID.	2
Current No. of Copies Available	500
Issue Date	13 JUN 2004
Due date	20 JUN 2004

**STEPS:**

1. Create a database for library which request the using SQL
2. Establish ODBC connection
3. In the administrator tools open data source ODBC
4. Click add button and select oracle in ORA home 90, click finish
5. A window will appear given the data source home as oracle and select TNS source name as lion and give the used id as SWTT
6. ADODC CONTROL FOR library FORM:-
7. The above procedure must be follow except the table , A select the table as library
8. Write appropriate Program in form each from created in VB from each from created in VB form project.

**i.ADMINISTRATOR Table**

This table holds the profile information of the application super users otherwise known as system administrators. They have control of the software meaning that they can perform additional tasks that other users cannot ordinarily perform. Every software of this nature has such users and this one is no

exception. The table contains the following columns; ADMIN\_ID, TITLE, FRIST\_NAME, LAST\_NAME, and DEPARMENT\_ID. The column ADMIN\_ID is the primary key column (primary key disallows duplicate values and nulls in a column) every table should have a primary key column, as this acts like table indexing.

**ii. ALL\_COURCES Table**

This table keeps the courses offered by students in different departments in the school. The table contains the following columns; COURCE\_ID, COURCE\_TITLE, and COURCE\_CODE.

The

COURCE\_ID is the primary key column. iii.

**APP\_USER\_A Table**

This table contains application login details for application administrators. The table columns are;

USERNAME, PASSWD and ADMIN\_ID. The column ADMIN\_ID is the primary key column.

iv. APP\_USER\_L Table

This table contains application login details for application lecturers. The table columns are;

USERNAME, PASSWD and LECTURER\_ID. The column LECTURER\_ID is the primary key column.

v. APP\_USER\_S Table

This table contains application login details for application students. The table columns are;

USERNAME, PASSWD and MATRIG\_NO. The column MATRIG\_NO is the primary key column.

vi. DEPARTMENTS Table

This table holds information about the schools departments. The table contains the following columns;

DEPARTMENT\_ID and DEPARTMENT\_NAME. The column DEPARTMENT\_ID is the primary key column.

vii. GRADES Table

This is more like the main table in the database as all other tables relate to this table directly or in some other way. This table holds students examination records. The table contains the following

columns; GRADES\_ID, SESSION1, REG\_NUMBER, DEPARTMENT\_ID, LEVEL1,

MATRIG\_NO, FRIST\_NAME, LAST\_NAME, COURCE\_CODE, GRADE, CREDIT\_UNIT,

SCORE, LECTURER\_ID and GRADE\_POINT. The column GRADES\_ID is the primary key column.

viii. LECTURERS Table

This table holds the profile information of the application lecturers. The table contains the following columns; LECTURER\_ID, TITLE, FRIST\_NAME, LAST\_NAME, and DEPARMENT\_ID. The column LECTUTER\_ID is the primary key column.

ix. REG\_TABLE Table

This table contains student's registration details i.e. if a student is registered for the semester this table

is used to store that information. The table contains the following columns; REG\_ID,

REG\_NUMBER, MATRIG\_NO, FRIST\_NAME, LAST\_NAME, LEVEL1, DEPARTMENT\_ID and

SESSION1. The column REG\_ID is the primary key column. x.

#### STUDENTS Table

This table holds the profile information of the application students. The table contains the following columns; MATRIG\_NO, TITLE, FRIST\_NAME, LAST\_NAME, and DEPARMENT\_ID. The column MATRIG\_NO is the primary key column.

The screenshot shows a web browser window titled 'Oracle Application Server Forms Services'. The main content area has a teal background. At the top center is the Demo University logo, which consists of a large black 'V' with a white leaf-like shape inside, and the text 'Demo University' below it, with the tagline 'knowledge is power.' underneath. Below the logo, the text reads: 'Welcome to Demo University's Exam Officer.' followed by 'Kindly click the check box if you are a student.' and 'Leave unchecked if not and continue.' At the bottom, there is a checkbox labeled 'Student' which is currently checked, and a 'Continue' button to its right. A status bar at the very bottom indicates 'Record: 1/1'.

The screenshot shows a web browser window titled 'Window' with 'ORACLE' in the top right corner. The main content area has a teal background. At the top center is the Demo University logo, which consists of a large black 'V' with a white leaf-like shape inside, and the text 'Demo University' below it, with the tagline 'knowledge is power.' underneath. Below the logo, the text reads: 'Kindly click the check box if you wish to sign in as an Administrator, leave unchecked if you wish to sign in as a Lecturer.' At the bottom, there is a checkbox labeled 'Administrator' which is currently checked, and a 'Continue' button to its right.

Window ORACLE

LOGIN

USER LOGIN

Username

Password

Enter Exit

Demo University  
Knowledge to power

07-OCT-14

**Student Result**

First Name(s): INNOCENT AGRI Last Name: IDU

Matric Number: NSU/WAS/294/08/09

Course Code	Grade	Credit Unit	Score
PHY 211	C	3	55
PHY 212	C	3	51
PHY 215	B	2	60
PHY 217	C	2	55
MTH 211	B	3	64
MTH 212	B	3	60
MTH 213	B	3	66
MTH 214	E	3	40

GPA : 03.23

## RESULT:

Thus, the student information system was designed and implemented successfully.

**AIM:**

To study about automatic backup of files and recovery.

**INTRODUCTION:**

Because data is the heart of the enterprise, it's crucial to protect it. And to protect organization's data, one need to implement a data backup and recovery plan. Backing up files can protect against accidental loss of user data, database corruption, hardware failures, and even natural disasters. It's our job as an administrator to make sure that backups are performed and that backup tapes are stored in a secure location.

**Creating a Backup and Recovery Plan**

Data backup is an insurance plan. Important files are accidentally deleted all the time. Mission-critical data can become corrupt. Natural disasters can leave office in ruin. With a solid backup and recovery plan, one can recover from any of these.

**Figuring Out a Backup Plan**

It takes time to create and implement a backup and recovery plan. We'll need to figure out what data needs to be backed up, how often the data should be backed up, and more. To help we create a plan, consider the following:

- How important is the data on systems? The importance of data can go a long way in helping to determine if one need to back it up—as well as when and how it should be backed up. For critical data, such as a database, one'll want to have redundant backup sets that extend back for several backup periods. For less important data, such as daily user files, we won't need such an elaborate backup plan, but 'll need to back up the data regularly and ensure that the data can be recovered easily.
- What type of information does the data contain? Data that doesn't seem important to we may be very important to someone else. Thus, the type of information the data contains can help we determine if we need to back up the data—as well as when and how the data should be backed up.
- How often does the data change? The frequency of change can affect our decision on how often the data should be backed up. For example, data that changes daily should be backed up daily.
- How quickly do we need to recover the data? Time is an important factor in creating a backup plan. For critical systems, we may need to get back online swiftly. To do this, we may need to alter our backup plan.
- Do we have the equipment to perform backups? We must have backup hardware to perform backups. To perform timely backups, we may need several backup devices and several sets of backup media. Backup hardware includes tape drives, optical drives, and

removable disk drives. Generally, tape drives are less expensive but slower than other types of drives.

- Who will be responsible for the backup and recovery plan? Ideally, someone should be a primary contact for the organization's backup and recovery plan. This person may also be responsible for performing the actual backup and recovery of data.
- What is the best time to schedule backups? Scheduling backups when system use is as low as possible will speed the backup process. However, we can't always schedule backups for off-peak hours. So we'll need to carefully plan when key system data is backed up.
- Do we need to store backups off-site? Storing copies of backup tapes off-site is essential to recovering our systems in the case of a natural disaster. In our off-site storage location, we should also include copies of the software we may need to install to reestablish operational systems.

### **The Basic Types of Backup**

There are many techniques for backing up files. The techniques we will use will depend on the type of data we're backing up, how convenient we want the recovery process to be, and more.

If we view the properties of a file or directory in Windows Explorer, we'll note an attribute called Archive. This attribute often is used to determine whether a file or directory should be backed up. If the attribute is on, the file or directory may need to be backed up. The basic types of backups we can perform include

- **Normal/full backups** All files that have been selected are backed up, regardless of the setting of the archive attribute. When a file is backed up, the archive attribute is cleared. If the file is later modified, this attribute is set, which indicates that the file needs to be backed up.
- **Copy backups** All files that have been selected are backed up, regardless of the setting of the archive attribute. Unlike a normal backup, the archive attribute on files isn't modified. This allows us to perform other types of backups on the files at a later date.
- **Differential backups** Designed to create backup copies of files that have changed since the last normal backup. The presence of the archive attribute indicates that the file has been modified and only files with this attribute are backed up. However, the archive attribute on files isn't modified. This allows to perform other types of backups on the files at a later date.
- **Incremental backups** Designed to create backups of files that have changed since the most recent normal or incremental backup. The presence of the archive attribute indicates that the file has been modified and only files with this attribute are backed up. When a file is backed up, the archive attribute is cleared. If the file is later modified, this attribute is set, which indicates that the file needs to be backed up.
- **Daily backups** Designed to back up files using the modification date on the file itself. If a file has been modified on the same day as the backup, the file will be backed up. This technique doesn't change the archive attributes of files.

In our backup plan we'll probably want to perform full backups on a weekly basis and supplement this with daily, differential, or incremental backups. We may also want to create an

extended backup set for monthly and quarterly backups that includes additional files that aren't being backed up regularly.

**Tip** We'll often find that weeks or months can go by before anyone notices that a file or data source is missing. This doesn't mean the file isn't important. Although some types of data aren't used often, they're still needed. So don't forget that we may also want to create extra sets of backups for monthly or quarterly periods, or both, to ensure that we can recover historical data over time.

### Differential and Incremental Backups

The difference between differential and incremental backups is extremely important. To understand the distinction between them. As it shows, with differential backups we back up all the files that have changed since the last full backup (which means that the size of the differential backup grows over time). With incremental backups, we only back up files that have changed since the most recent full or incremental backup (which means the size of the incremental backup is usually much smaller than a full backup).

Once we determine what data we're going to back up and how often, we can select backup devices and media that support these choices. These are covered in the next section.

### Selecting Backup Devices and Media

Many tools are available for backing up data. Some are fast and expensive. Others are slow but very reliable. The backup solution that's right for our organization depends on many factors, including

- **Capacity** The amount of data that we need to back up on a routine basis. Can the backup hardware support the required load given our time and resource constraints?
- **Reliability** The reliability of the backup hardware and media. Can we afford to sacrifice reliability to meet budget or time needs?
- **Extensibility** The extensibility of the backup solution. Will this solution meet our needs as the organization grows?
- **Speed** The speed with which data can be backed up and recovered. Can we afford to sacrifice speed to reduce costs?
- **Cost** The cost of the backup solution. Does it fit into our budget?

### Common Backup Solutions

Capacity, reliability, extensibility, speed, and cost are the issues driving our backup plan. If we understand how these issues affect our organization, we'll be on track to select an appropriate backup solution. Some of the most commonly used backup solutions include

- **Tape drives** Tape drives are the most common backup devices. Tape drives use magnetic tape cartridges to store data. Magnetic tapes are relatively inexpensive but aren't highly reliable. Tapes can break or stretch. They can also lose information over time. The average capacity of tape cartridges ranges from 100 MB to 2 GB. Compared with other backup solutions, tape drives are fairly slow. Still, the selling point is the low cost.
- **Digital audio tape (DAT) drives** DAT drives are quickly replacing standard tape drives as the preferred backup devices. DAT drives use 4 mm and 8 mm tapes to store data. DAT



drives and tapes are more expensive than standard tape drives and tapes, but they offer more speed and capacity. DAT drives that use 4 mm tapes can typically record over 30 MB per minute and have capacities of up to 16 GB. DAT drives that use 8 mm tapes can typically record more than 10 MB per minute and have capacities of up to 36 GB (with compression).

- **Auto-loader tape systems** Auto-loader tape systems use a magazine of tapes to create extended backup volumes capable of meeting the high-capacity needs of the enterprise. With an auto-loader system, tapes within the magazine are automatically changed as needed during the backup or recovery process. Most auto-loader tape systems use DAT tapes. The typical system uses magazines with between 4 and 12 tapes. The main drawback to these systems is the high cost.
- **Magnetic optical drives** Magnetic optical drives combine magnetic tape technology with optical lasers to create a more reliable backup solution than DAT. Magnetic optical drives use 3.5-inch and 5.25-inch disks that look similar to floppies but are much thicker. Typically, magnetic optical disks have capacities of between 1 GB and 4 GB.
- **Tape jukeboxes** Tape jukeboxes are similar to auto-loader tape systems. Jukeboxes use magnetic optical disks rather than DAT tapes to offer high-capacity solutions. These systems load and unload disks stored internally for backup and recovery operations. Their key drawback is the high cost.
- **Removable disks** Removable disks, such as Iomega Jaz, are increasingly being used as backup devices. Removable disks offer good speed and ease of use for a single drive or single system backup. However, the disk drives and the removable disks tend to be more expensive than standard tape or DAT drive solutions.
- **Disk drives** Disk drives provide the fastest way to back up and restore files. With disk drives, you can often accomplish in minutes what takes a tape drive hours. So when business needs mandate a speedy recovery, nothing beats a disk drive. The drawbacks to disk drives, however, are relatively high costs and less extensibility.

Before we can use a backup device, we must install it. When we install backup devices other than standard tape and DAT drives, we need to tell the operating system about the controller card and drivers that the backup device uses. For detailed information on installing devices and drivers, see the section of Chapter 2 entitled "Managing Hardware Devices and Drivers."

### **Buying and Using Tapes**

Selecting a backup device is an important step toward implementing a backup and recovery plan. But we also need to purchase the tapes or disks, or both, that will allow us to implement our plan. The number of tapes we need depends on how much data we'll be backing up, how often we'll be backing up the data, and how long we'll need to keep additional data sets.

The typical way to use backup tapes is to set up a rotation schedule whereby we rotate through two or more sets of tapes. The idea is that we can increase tape longevity by reducing tape usage and at the same time reduce the number of tapes we need to ensure that we have historic data on hand when necessary.

One of the most common tape rotation schedules is the 10-tape rotation. With this rotation schedule, we use 10 tapes divided into two sets of 5 (one for each weekday). As shown in Table 14-2, the first set of tapes is used one week and the second set of tapes is used the next week.

On Fridays, full backups are scheduled. On Mondays through Thursdays, incremental backups are scheduled. If we add a third set of tapes, we can rotate one of the tape sets to an off-site storage location on a weekly basis.

Tip The 10-tape rotation schedule is designed for the 9 to 5 workers of the world. If we're in a 24 x 7 environment, we'll definitely want extra tapes for Saturday and Sunday. In this case, use a 14-tape rotation with two sets of 7 tapes. On Sundays, schedule full backups. On Mondays through Saturdays, schedule incremental backups.

**RESULT:**

Thus, the study of automatic backup of files was performed successfully.