CONTENTS

TITLE

1. Institute Vision and Mission
2. School/Department Vision and Mission
3. PEOs
4. Program Specific Outcomes (PSO)
5. Program Outcomes (PO)
6. Course Outcome (CO) Statements and Course Articulation Matrix
7. Rubrics for Evaluation (CA and Semester End Assessment)
8. List of experiment with CO Mapping
9. List of Tools used and Reference books
10. Experiments with Solutions

Experiment with Solutions

Experiment 1: Execute different Data Definition Language (DDL), Data Manipulation Language (DML), DCL, TCL commands on a sample database.

DDL: CREATE, DROP, TRUNCATE,ALTER AND RENAME

DML: INSERT, UPDATE AND DELETE

The Create Table Command

CREATE TABLE Student (Reg_novarchar2(10),

Name char(30),
DOB date,
Address varchar2(50));

Syntax : DROP TABLE <tablename>

Syntax : TRUNCATE TABLE <tablename>

The create table command defines each column of the table uniquely. Each column has minimum of three

attributes.

Table name is Student

Column name	Data type	Size
Reg_no	varchar2	10
Name	char	30
DOB	date	
Address	varchar2	50

The DROP Command

It will destroy the table and all data which will be recorded in it.

DROP TABLE Student;

The TRUNCATE Command

It will delete all the records but the schema is retained

Syntax : Eg.,

RENAME <OldTableName> TO <NewTableName>

RENAME <Student> TO <Stu>

The old name table was Student now new name is the Stu.

The ALTER Table Command

By The use of ALTER TABLE Command we can modify our exiting table.

Adding New Columns

ALTER TABLE Student ADD (Age number(2), Marks number(3));

Dropping a Column from the Table

Syntax:ALTER TABLE <table_name> DROP COLUMN <column_name>

ALTER TABLE Student DROP COLUMN Age;

Modifying Existing Table

11

Syntax:

ALTER TABLE <table_name> MODIFY (<column_name><NewDataType>(<NewSize>))

ALTER TABLE Student MODIFY (Name Varchar2(40));

DML(Data Manipulation Language) Commands - Select - Insert - Update - Delete

Viewing Data in the Table (Select Command)

Once data has been inserted into a table, the next most logical operation would be to view what has been inserted. The SELECT SQL verb is used to achieve this.

All Rows and All Columns
Syntax: SELECT * FROM Table_name;
eg: Select * from Student; It will show all the table records.

SELECT First_name, DOB FROM STUDENT WHERE Reg_no = 'S101';
Cover it by single inverted comma if its datatype is varchar or char.

Eliminating Duplicates:

A table could hold duplicate rows. In such a case, you can eliminate duplicates. Syntax: SELECT DISTINCT col, col, .., FROM table_name;

eg : SELECT DISTINCT * FROM Student; Sorting DATA:

The Rows retrieved from the table will be sorted in either Ascending or Descending order depending on the condition specified in select statement, the Keyword has used ORDER BY.

SELECT * FROM Student ORDER BY First_Name;

INSERT Command

INSERT INTO student VALUES('A101', 'Mohd', 'Imran', '01-MAR-89','Allahabad', 211001);

Inserting data into a table from another table:

In addition to inserting data one row at a time into a table, it is quite possible to populate a table with data

that already exist in another table. You can store same record in a table that already stored in another table.

Eg : suppose you want to insert data from course table to university table then use this example: INSERT INTO university SELECT course_id, course_name FROM course;

DELETE Operation:

The DELETE command can remove all the rows from the table or a set of rows from the table.

eg: DELETE FROM student; It will DELETE all the rows from student table.

eg: DELETE FROM student WHERE reg_no='A101'; If condition will be satisfied then it will delete a row from the table

12

UPDATE Operation:

The UPDATE command is used to change or modify data values in a table and UPDATE command can Update all the rows from the table or a set of rows from the table.
eg : UPDATE Student SET course='MCA';

Course is a column name, suppose ant time you want to update something like that in the student table course should be MCA for all students then you can use this type of query. It will update all the rows in the table all rows will have MCA course.
Now, if you want update particular row then see below.

UPDATE Student SET course='MCA' where reg_no='A101';

DCL(Data Control Language) Commands - Grant – Revoke

GRANTING AND REVOKING PERMISSIONS:

Oracle provides extensive security features in order to safeguard information stored in its tables from unauthorized viewing and damage. Depending on a user's status and responsibility, appropriate rights on Oracle's resources can be assigned to the user by the DBA. The rights that allow the use of some or all of oracle's resources on the Server are called PRIVILEGES.

Objects that are created by a user are owned and controlled by that user. If a user wishes to access any of the objects belonging to another user, the owner of the object will have to give permissions for such access. This is called GRANTING of PRIVILEGES. Privileges once given can be taken back by the owner of the object. This is called REVOKING of PRIVILEGES.

GRANT Statement:

Syntax:

GRANT <object privileges>

ON <object_name>
TO <User_Name>
[WITH GRANT OPTION]

eg:

GRANT ALL
ON Student TO Mohd Imran WITH GRANT OPTION

Syntax:
REVOKE <Object_Privileges> ON <Object_Name>
FROM <User_Name>

Eg:

REVOKE UPDATE ON Student
FROM Fareen;

TCL (Transaction Control Language) Commands - Commit - Rollback - Savepoint

ORACLE TRANSACTION:

A series of one or more SQL statements that are logically related, or a series of operation performed on Oracle table data is termed as a Transaction. Oracle treats changes to table data as a two step process. First the changes requested are done. To make these changes permanent a COMMIT statement has to be given at the SQL prompt. A ROLLBACK statement given at the SQL prompt can be used to undo a part of or the entire Transaction.

Using COMMIT:

A COMMIT ends the current transaction and makes permanent any changes made during the transaction. All transaction locks acquired on tables are released.

Syntax:

COMMIT;

Using ROLLBACK:

A ROLLBACK does exactly the opposite of COMMIT. It ends the transaction but undoes any changes made during the transaction. All transaction locks acquired on tables are released.

Syntax:
ROLLBACK [WORK] [TO SAVEPOINT] < Save-point_Name>

WORK: It is optional and is provided for ANSI compatibility
SAVEPOINT : It is optional and it is used to rollback a partial transaction, as far as

the specified savepoint.
SAVEPOINTNAME: It is a savepoint created during the current transaction

Crating a SAVEPOINT:

SAVEPOINT marks and saves the current point in the processing of a transaction. When a SAVEPOINT is used with a ROLLBACK statement, parts of a transaction can be undone. An active savepoint is one that is specified since the last COMMIT or ROLLBACK.

Syntax:
SAVEPOINT <SavePointName>

Experiment 2: Consider the following Student Database consisting of four relations Students, Teachers , subjects and Grades. Primary keys are underlined.

Students(Sid int, Name string)
Teachers(Tidint, Name string)
Subjects(Subidint, Name string)
Grades(studentIDint, teachersIDint, subjectIDint, grade string)

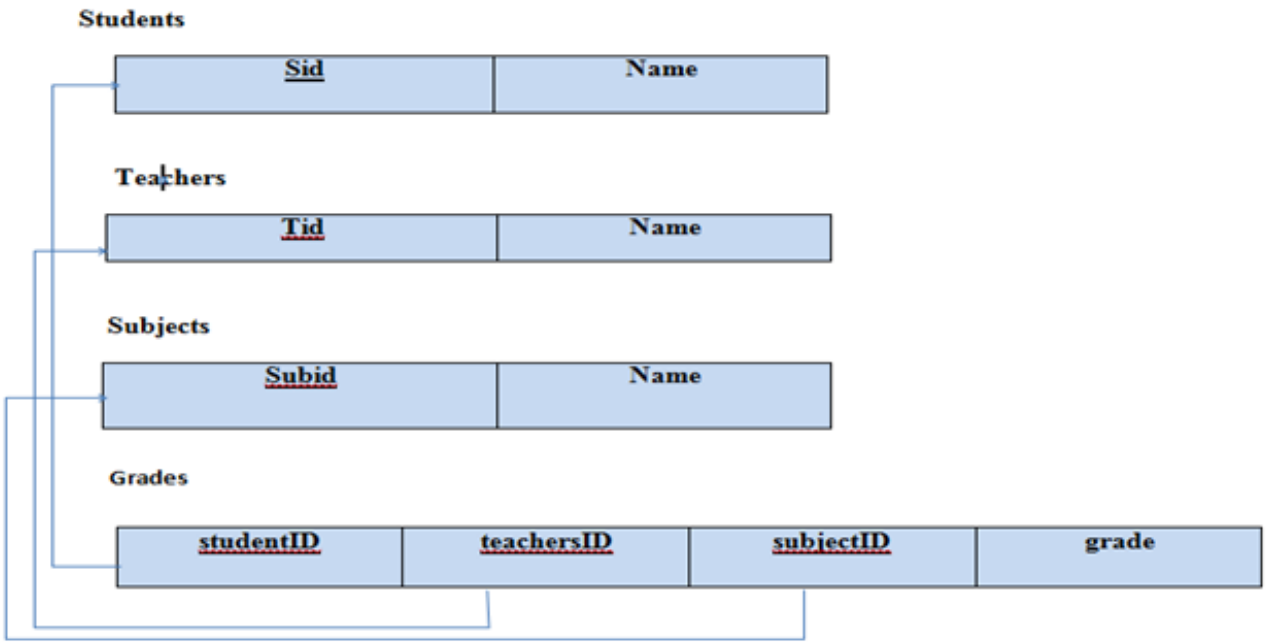
1. Create all the tables with necessary constraints

CREATE TABLE Students(sidnumber(5) not null, namevarchar(15) not null, primary key(sid)

);
CREATE TABLE teachers(tidnumber(5) not null, namevarchar(15) not null, primary key(tid)
);

CREATE TABLE Subjects(subidnumber(5) not null, name varchar(15) not null, primary key(subid)

);
CREATE TABLE Grades(
studentID number(5) not null references students(sid),
teacherIDnumber(5) not null references teachers(tid), subjectID number(5) not null references subjects(subid), gradevarchar(3),
primary key(studentID, teacherID, subjectID)
);



Experiment 3: Insert at least five rows to every table

INSERT INTO students (sid, name) VALUES(1, 'Simon'); INSERT INTO students (sid, name) VALUES(2, 'Alvin'); INSERT INTO students (sid, name) VALUES(3, 'Theo'); INSERT INTO students (sid, name)

VALUES(4, 'Brittany'); INSERT INTO students (sid, name) VALUES(5, 'Jenette'); INSERT INTO students (sid, name) VALUES(6, 'Elenor'); INSERT INTO students (sid, name) VALUES(7, 'Stu');

INSERT INTO teachers (tid, name) VALUES (1, 'Washington'); INSERT INTO teachers (tid, name) VALUES (2, 'Adams'); INSERT INTO teachers (tid, name) VALUES (3, 'Jefferson'); INSERT INTO teachers (tid, name) VALUES (4, 'Lincoln');

INSERT INTO subjects (subid, name) VALUES (1, 'History'); INSERT INTO subjects (subid, name) VALUES (2, 'Biology'); INSERT INTO subjects (subid, name) VALUES (3, 'SF');

INSERT INTO grades (studentID, teacherID, subjectID, grade) INSERT INTO grades (studentID, teacherID, subjectID, grade) INSERT INTO grades (studentID, teacherID, subjectID, grade) INSERT INTO grades (studentID, teacherID, subjectID, grade) INSERT INTO grades (studentID, teacherID, subjectID, grade) INSERT INTO grades (studentID, teacherID, subjectID, grade) INSERT INTO grades (studentID, teacherID, subjectID, grade)

VALUES (1, 2, 1, 'A'); VALUES (1, 2, 2, 'B'); VALUES (7, 4, 3, 'C+'); VALUES (7, 3, 2, 'F'); VALUES (6, 2, 1, 'B+'); VALUES (2, 4, 3, 'C'); VALUES (3, 4, 3, 'B');

Experiment 4: Execute the queries on Student Database

1. List the students as per their alphabetic order

select * from students order by name ASC;

Output

2. List the names of students in any class taught by Adams:

select name from students where sid in
(select studentID from grades where teacherID in

(select tid from teachers where name = 'Adams'));

Output

4. Names of teachers who taught Biology:

SID	NAME
2	Alvin
4	Brittany
6	Elenor
5	Jenette
1	Simon
7	Stu
3	Theo

NAME
Simon
Elenor

select namefrom teacherswhere tid in

Output

(select teacherID from grades wheresubjectID in

(selectsubid from subjects

where name = 'Biology'));

NAME
Adams
Jefferson

5.Retrieve the names of teachers who have not yet taught:

select namefrom teacherswhere tid not in (selectteacherID

from grades);

Output

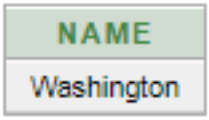
6. List the names of students who have not yet taken any classes:

select namefrom students where sid not in (select studentID

NAME
Adams
Jefferson

Output

from grades);



17

Experiment 5: Consider the Employee Database consisting of two relations Employees and department. Create all the tables with necessary constraints.

1. Create all the tables with necessary constraints

CREATE TABLE employees (emp_id number(5), emp_namevarchar(15) NOT NULL, job_name VARCHAR(10), manager_id number(5) ,

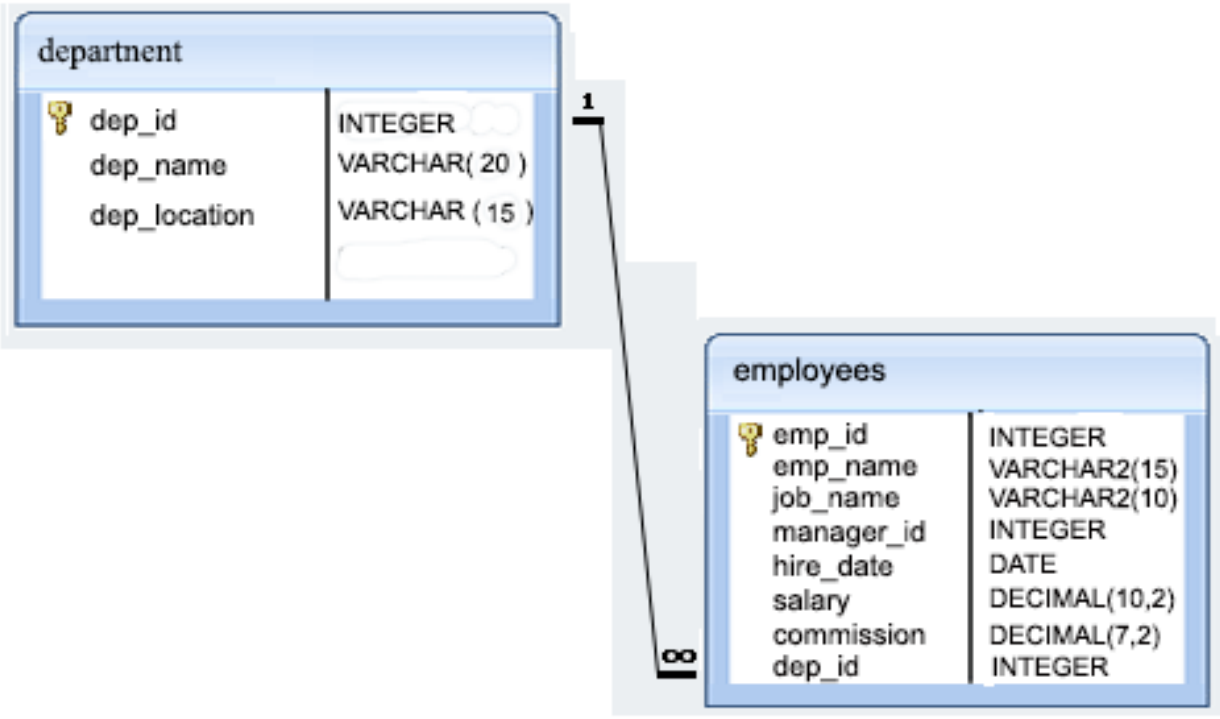
hire_date DATE NOT NULL,
salary number(10,2),
commission number(7,2),
dept_id number(5) references department(dept_id),

PRIMARY KEY (emp_id));

CREATE TABLE department (
dept_id number(5) NOT NULL, dept_name VARCHAR(20) NOT NULL,
dept_location VARCHAR(15),

PRIMARY KEY (dept_id), UNIQUE(dept_name)

);



Experiment 6: Insert at least five tuples to all the relations of the Employee database

Insert into department values(2001, 'inventary', 'central'); Insert into department values(2002,'purchase', 'East');
Insert into department values(2003,'delivery', 'West');
Insert into department values(2004,'orders', 'North');
Insert into department values(2005,'Human Resource', 'South');

Insert into Employees(emp_id,emp_name,job_name,hire_date, salary,dept_id) values(101,'James', 'President', '18-Jan-91', 1000000, 2001)
Insert into Employees values(102,'Adarsh', 'Analyst', 101,'18-Jan-95', 10000,400, 2001)
Insert into Employees values(103,'Akash', 'Clarke', 102,'18-Jan-96', 1000, 200, 2002)

Insert into Employees values(104,'Aditya', 'programmer', 102,'18-Mar-96', 1000, 200, 2002)
Insert into Employeesvalues(105,'Amith', 'Analyst', 102,'18-Apr-96', 1000, 200, 2002)
Insert into Employees(emp_id,emp_name,job_name,manager_id,hire_date, salary,dept_id) values(106,'Arun', 'Analyst', 102,'18-Apr-96', 1000, 2003)

Insert into
Employees(emp_id,emp_name,job_name,manager_id,hire_date,
salary,dept_id) values(107,'Arjun', 'Clarke', 102,'18-Apr-96', 2000, 2004)
Insert into
Employees(emp_id,emp_name,job_name,manager_id,hire_date,
salary,dept_id) values(108,'Arya', 'Clarke', 102,'21-Apr-97', 2000, 2005)

Experiment 7: Execute queries on Employee Database- Biginner Level

2. Write a query in SQL to display the unique designations for the employees SELECT DISTINCT job_name FROM employees;

3. Write a query in SQL to produce the output of employees name and job name as a format of "Employee & Job"
   SELECT emp_name ||' '|| job_name AS "Employee & Job "FROM employees ;

4. Write a query in SQL to list the employees with Hire date in the format like February 22, 1991 SELECT emp_id, emp_name,

salary,to_char(hire_date,'MONTH DD,YYYY') FROM employees;

5. Write a query in SQL to count the no. of characters with out considering the spaces for each name. SELECT length(trim(emp_name)) FROM employees;

6. Write a query in SQL to list the employees who does not belong to department 2001. SELECT*
   FROM employees
   WHERE dep_id NOT IN (2001);

7.

8.

Experiment 8: Execute queries on Employee Database- Intermediate Level

9. Write a query in SQL to display the average salaries of all the employees who works as Analyst.

```
SELECT avg(salary)
FROM employees
WHERE job_name='Analyst';
```

10. Write a query in SQL to list the employees whose salary will be more than 10000 after giving 25% increment.

```
SELECT*
FROM employees
WHERE (1.25*salary)>10000;
```

10. Write a query in SQL to list the employees who have joined in the year 1995. SELECT*

```
FROM employees
WHERE to_char(hire_date,'YYYY')='1991';
```

11. Write a query in SQL to list the employees along with department name.

```
SELECT e.emp_id, e.emp_name, e.job_name, e.manager_id,
e.hire_date,
```



```
e.salary,
e.commission,
e.dept_id,
d.dept_name
```

```
FROM employees e, department d
WHERE e.dept_id = d.dept_id Or

SELECT *
FROM employees e, department d
WHERE e.dept_id = d.dept_id
```

12. Write a query in SQL to list the employees, joined in the month FEBRUARY with a salary range between 1001 to 20000.

```
SELECT*
FROM employees
WHERE to_char(hire_date,'MON')='FEB' AND salary BETWEEN 1000
AND 2000;
```

13. Write a query in SQL to list the name, job name, manager id, salary, manager name, manager's salary for those employees whose salary is greater than the salary of their managers

```
SELECT w.emp_name, w. ,
w. , w.salary,

m. "Manager", m. , m.salary"Manager_Salary" FROM
                ,

WHERE w.              =m.emp_id AND w.salary>m.salary;
```

Experiment 9 : Execute queries on Employee Database-Advanced Level-1

14. Write a query in SQL to list the employees whose manager name is Adarsh.

```
SELECT w.emp_id, w.emp_name, w.job_name, w.manager_id,
w.hire_date, w.salary,

w.dept_id,
m.emp_name
FROM employees w,
employees m
WHERE w.manager_id = m.emp_id
```

job_name

manager_id

emp_name

emp_id

employees w

employees m

manager_id

21

AND m.emp_name = 'Adarsh';

15. Write a query in SQL to list the employees who are working either MANAGER or HR with a salary range between 20000 to 100000 without any commission.

```
SELECT*
FROM employees
WHERE job_name IN('MANAGER',
'HR')
AND salary BETWEEN 20000 AND 100000 AND commission IS NULL;
```

16. Write a query in SQL to list the employees who are senior to their own manager. SELECT*

FROM employees w,
employees m WHEREw.manager_id=m.emp_id
ANDw.hire_date<m.hire_date;

17. Write a query in SQL to display the location of Adarsh. SELECT dep_location

FROM department d, employees e WHEREe.emp_name='Adarsh' AND e.dept_id=d.dept_id;

18.

location of all the employees working under marketing and HR in the ascending department no.

```
SELECT *
FROM employees e,
department d
WHERE d.dept_name IN ('marketing',

'HR')
AND e.dept_id = d.dept_id

ORDER BY e.dept_id ASC;
```

19. Write a query in SQL to list the details of the employees along with the details of their departments.

```
SELECT*
FROM employees e, department d WHEREe.dept_id=d.dept_id;
```

22

Write a query in SQL to list the total information of employees table along with department, and

Experiment 10 : Execute queries on Employee Database-Advanced Level-2

20.

commission) for each type of job.

```
SELECTjob_name, avg(salary), avg(salary+commission) FROM
employees GROUPBYjob_name;
```

21.

managers in ascending order on manager id.

```
SELECTw.manager_id,
count(*)
FROM employees w,
employees m WHEREw.manager_id=m.emp_id GROUPBYw.manager_id
```

ORDERBYw.manager_idASC;
22. Write a query in SQL to list the department where at least two employees are working.

SELECTdep_id, count(*)
FROM employees GROUPBYdep_id HAVINGcount(*)>=2;

23. Write a query in SQL to display the number of employee for each job in each department. SELECTdept_id,

job_name,
count(*)
FROM employees GROUPBYdept_id, job_name;

24.

Write a query in SQL to find the average salary and average total remuneration(salary and

Write a query in SQL to list the manager no and the number of employees working for those

Write a query in SQL to find the total annual salary distributed against each job in the

year 1995

SELECT job_name,
sum(12*salary)
FROM employees
WHERE to_char(hire_date,'YYYY') = '1995' GROUP BY job_name;

25.

Write a query in SQL to list the employee id, name, salary, and department id of the employees in

ascending order of salary who works in the department 1001

SELECTe. , e.emp_name, e.salary,
e.dept_id

FROM employees E WHEREe.dept_id=2001 ORDERBYe.salaryASC;

emp_id

24

Experiment 11: Consider the LibraryDatabase consisting of six relations Publisher, Book, Book_Authors, Book_Copies, Book_lending and Library_Branch. Create all the tables with necessary constraints and insert at least five tuples to each relation

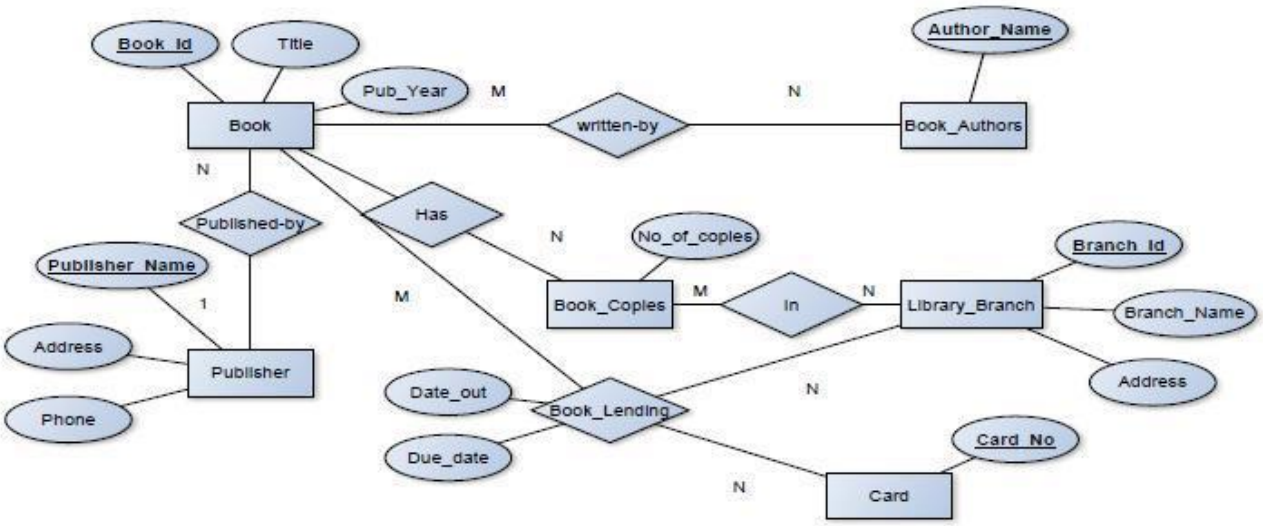
1)

Consider the following schema for a Library Database

PUBLISHER (Name, Address, Phone)

BOOK (Book_id, Title, Publisher_Name, Pub_Year)
BOOK_AUTHORS (Book_id, Author_Name)
BOOK_COPIES (Book_id, Branch_id, No-of_Copies) BOOK_LENDING (Book_id, Branch_id, Card_No, Date_Out, Due_Date)
LIBRARY_BRANCH (Branch_id, Branch_Name, Address)

ER-Diagram:



Schema Diagram

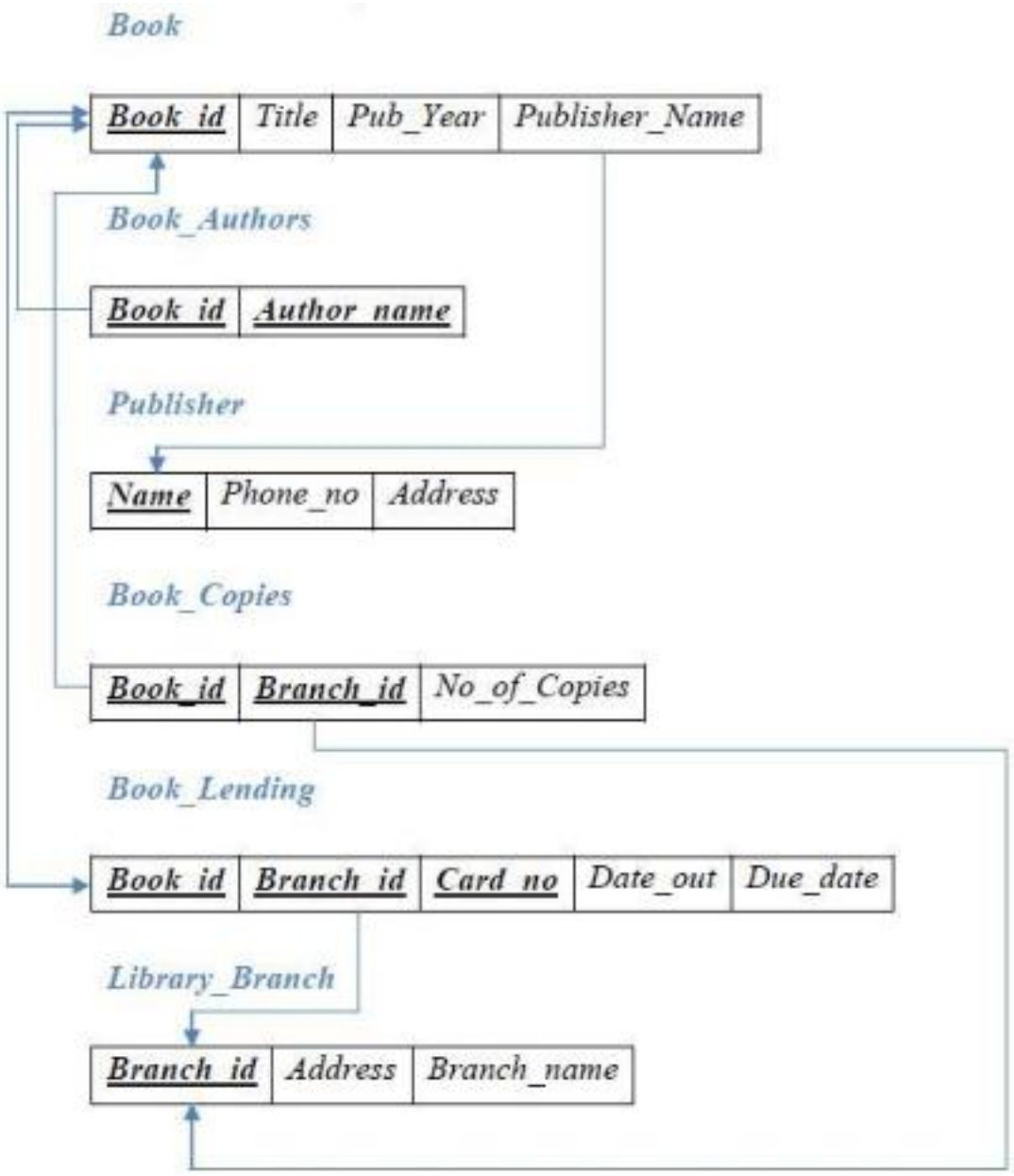


Table Creation:

PUBLISHER

CREATE TABLEPUBLISHER(
NAME VARCHAR(18) PRIMARY KEY, ADDRESS VARCHAR(10),
PHONE VARCHAR(10));

BOOK

PUB_YEAR NUMBER(4));

BOOK_AUTHORS

CREATETABLE BOOK_AUTHORS(
BOOK_ID NUMBER(5)REFERENCES BOOK(BOOK_ID) ON
DELETE CASCADE, AUTHOR_NAME VARCHAR(20),

PRIMARY KEY(BOOK_ID, AUTHOR_NAME));

CREATETABLEBOOK(
BOOK_IDNUMBER(5)PRIMARY KEY,
PUBLISHER_NAME VARCHAR(20)REFERENCES
PUBLISHER(NAME)ON DELETE

TITLE VARCHAR(20), CASCADE,

LIBRARY_BRANCH

CREATETABLE LIBRARY_BRANCH(BRANCH_ID
NUMBER(5)PRIMARY KEY, BRANCH_NAME VARCHAR(18),
ADDRESS VARCHAR(15));

BOOK_COPIES

CREATE TABLE BOOK_COPIES(
BOOK_ID INTEGER REFERENCESBOOK(BOOK_ID) ON DELETE
CASCADE, BRANCH_ID NUMBER(5) REFERENCES
LIBRARY_BRANCH(BRANCH_ID) ON DELETE

CASCADE,
NO_OF_COPIES NUMBER(5),

PRIMARY KEY(BOOK_ID,BRANCH_ID));

BOOK_LENDING

CREATE TABLE BOOK_LENDING(
BOOK_ID NUMBER(5) REFERENCES BOOK(BOOK_ID) ON
DELETE CASCADE, BRANCH_ID NUMBER(5) REFERENCES
LIBRARY_BRANCH(BRANCH_ID) ON

DELETE CASCADE, CARD_NO NUMBER(5), DATE_OUT DATE, DUE_DATE DATE,

PRIMARY KEY(BOOK_ID,BRANCH_ID,CARD_NO));
Values for tables:
PUBLISHER
INSERT INTO PUBLISHER
VALUES('PEARSON','BANGALORE','9875462530');

INSERT INTO PUBLISHER
VALUES('MCGRAW','NEWDELHI','7845691234');

INSERT INTO PUBLISHER
VALUES('SAPNA','BANGALORE','7845963210');

BOOK

INSERT INTO BOOK VALUES(1111,'SE','PEARSON',2005); INSERT INTO BOOK VALUES(2222,'DBMS','MCGRAW',2004);

INSERT INTO BOOK VALUES(3333,'ANOTOMY','PEARSON',2010);
INSERT INTO BOOK
VALUES(4444,'ENCYCLOPEDIA','SAPNA',2010);

BOOK_AUTHORS

INSERT INTO BOOK_AUTHORS VALUES(1111,'SOMMERVILLE');
INSERT INTO BOOK_AUTHORS VALUES(2222,'NAVATHE'); INSERT INTO BOOK_AUTHORS VALUES(3333,'HENRY GRAY');

INSERT INTO BOOK_AUTHORS VALUES(4444,'THOMAS');

LIBRARY_BRANCH

INSERT INTO LIBRARY_BRANCH VALUES(11,'CENTRAL TECHNICAL','MG ROAD'); INSERT INTO LIBRARY_BRANCH VALUES(22,'MEDICAL','BH ROAD');
INSERT INTO LIBRARY_BRANCH VALUES(33,'CHILDREN','SS PURAM');
INSERT INTO LIBRARY_BRANCH

VALUES(44,'SECRETARIAT','SIRAGATE'); INSERT INTO LIBRARY_BRANCH VALUES(55,'GENERAL','JAYANAGAR');

BOOK_COPIES

INSERT INTO BOOK_COPIES VALUES(1111,11,5); INSERT INTO BOOK_COPIES VALUES(3333,22,6); INSERT INTO BOOK_COPIES VALUES(4444,33,10); INSERT INTO BOOK_COPIES VALUES(2222,11,12); INSERT INTO BOOK_COPIES VALUES(4444,55,3);

BOOK_LENDING

INSERT INTO BOOK_LENDING VALUES(2222,11,1,'10-JAN-2017','20-AUG-2017'); INSERT INTO BOOK_LENDING VALUES(3333,22,2,'09-JUL-2017','12-AUG-2017'); INSERT INTO BOOK_LENDING VALUES(4444,55,1,'11-APR-2017','09-AUG-2017'); INSERT INTO BOOK_LENDING VALUES(2222,11,5,'09-AUG-2017','19-AUG-2017'); INSERT INTO BOOK_LENDING VALUES(4444,33,1,'10-JUN-2017','15-AUG-2017'); INSERT INTO BOOK_LENDING VALUES(1111,11,1,'12-MAY-2017','10-JUN-2017'); INSERT INTO BOOK_LENDING VALUES(3333,22,1,'10-JUL-2017','15-JUL-2017');

Experiment 12 : Execute Complex queries on Employee Database

1) Retrieve details of all books in the library – id, title, name of publisher, authors, number ofcopies in each branch, etc.

SELECT LB.BRANCH_NAME, B.BOOK_ID,TITLE, PUBLISHER_NAME, AUTHOR_NAME, NO_OF_COPIES FROM BOOK B, BOOK_AUTHORS BA, BOOK_COPIES BC, LIBRARY_BRANCH LB WHERE B.BOOK_ID = BA.BOOK_ID AND

BA.BOOK_ID = BC.BOOK_ID AND BC.BRANCH_ID = LB.BRANCH_ID

GROUP BY LB.BRANCH_NAME, B.BOOK_ID, TITLE, PUBLISHER_NAME, AUTHOR_NAME, NO_OF_COPIES;

2) Get the particulars of borrowers who have borrowed more than 3 books, but from Jan 2017 2017.

SELECT CARD_NO
FROM BOOK_LENDING
WHERE DATE_OUT BETWEEN '01-JAN-2017' AND '30-JUN-2017'
GROUPBY CARD_NO
HAVINGCOUNT(*) > 3;

3) Delete a book in BOOK table. Update the contents of other tables to reflect this data operation.

DELETE FROM BOOKWHERE BOOK_ID = '3333';

to Jun

4) Partition the BOOK table based on year of publication. Demonstrate its working with a simple query.
SELECT BOOK_ID, TITLE, PUBLISHER_NAME, PUB_YEAR
FROM BOOK

GROUPBY PUB_YEAR, BOOK_ID, TITLE, PUBLISHER_NAME;
5)Create a view of all books and its number of copies that are currently available in the

Library.

CREATEVIEW BOOKS_AVAILABLE AS
SELECT B.BOOK_ID, B.TITLE, C.NO_OF_COPIES
FROM LIBRARY_BRANCH L, BOOK B, BOOK_COPIES C WHERE B.BOOK_ID = C.BOOK_ID AND

L.BRANCH_ID=C.BRANCH_ID;

Experiment 13: Mini project on database application using any programming language for front end design and Oracle as back end.

manipulation

29