KG REDDY College of Engineering & Technology
Engineering India's Changemakers

KG REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

(Approved by AICTE, New Delhi, Affiliated to JNTUH, Hyderabad)
Chilkur (Village), Moinabad (Mandal), R. R Dist, TS-501504

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

DATABASE MANAGEMENT SYSTEM

LAB MANUAL

Subject Code : **CS407PC**
Regulation : R18/JNTUH
Academic Year : 2020-2021

II B. TECH II SEMESTER

COMPUTER SCIENCE AND ENGINEERING

KG REDDY COLLEGE OF ENGINEERING AND TECHNOLOGY

Affiliated o JNTUH, Chilkur,(V), Moinabad(M) R. R Dist, TS-501504

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISION AND MISSION OF THE INSTITUTION

VISION

To become self-sustainable institution this is recognized for its new age engineering through innovative teaching and learning culture, inculcating research and entrepreneurial ecosystem, and sustainable social impact in the community.

MISSION

- To offer undergraduate and post-graduate programs that is supported through industry relevant curriculum and innovative teaching and learning processes that would help students succeed in their professional careers.
- To provide necessary support structures for students, this will contribute to their personal and professional growth and enable them to become leaders in their respective fields.
- To provide faculty and students with an ecosystem that fosters research and development through strategic partnerships with government organisations and collaboration with industries.
- To contribute to the development of the region by using our technological expertise to work with nearby communities and support them in their social and economic growth.

VISION AND MISSION OF CSE DEPARTMENT

VISION

To be recognized as a department of excellence by stimulating a learning environment in which students and faculty will thrive and grow to achieve their professional, institutional and societal goals.

MISSION

- To provide high quality technical education to students that will enable life-long learning and build expertise in advanced technologies in Computer Science and Engineering.
- To promote research and development by providing opportunities to solve complex engineering problems in collaboration with industry and government agencies.
- To encourage professional development of students that will inculcate ethical values and leadership skills while working with the community to address societal issues.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PROGRAM EDUCATIONAL OBJECTIVES (PEOS):

A graduate of the Computer Science and Engineering Program should:

PEO1	**Program Educational Objective1: (PEO1)** The Graduates will provide solutions to difficult and challenging issues in their profession by applying computer science and engineering theory and principles.
PEO2	**Program Educational Objective2 :(PEO2)** **The** Graduates have successful careers in computer science and engineering fields or will be able to successfully pursue advanced degrees.
PEO3	**Program Educational Objective3: (PEO3)** The Graduates will communicate effectively, work collaboratively and exhibit high levels of Professionalism, moral and ethical responsibility.
PEO4	**Program Educational Objective4 :(PEO4)** **The** Graduates will develop the ability to understand and analyse Engineering issues in a broader perspective with ethical responsibility towards sustainable development.

PROGRAM OUTCOMES (POS):

PO1	**Engineeringknowledge**:Applytheknowledgeofmathematics,science,engineering Fundamentals andanengineeringspecializationtothesolutionofcomplexengineeringproblems.
PO2	**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.
PO3	**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.
PO4	**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.

PO5	**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.
PO6	**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.
PO7	**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.
PO8	**Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO9	**Individual and team work**: Function effectively as an individual, and as a member or leader In diverse teams, and in multi-disciplinary settings.
PO10	**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.
PO11	**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.
PO12	**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.

PROGRAM SPECIFIC OUTCOMES(PSOS):

PSO1	**Problem Solving Skills** – Graduate will be able to apply computational techniques and software principles to solve complex engineering problems pertaining to software engineering.
PSO2	**Professional Skills** – Graduate will be able to think critically, communicate effectively, and collaborate in teams through participation in co and extra-curricular activities.
PSO3	**Successful Career** – Graduates will possess a solid foundation in computer science and engineering that will enable them to grow in their profession and pursue lifelong learning through post-graduation and professional development.

COURSE OBJECTIVES, COURSE OUTCOMES

COURSEOBJECTIVES :

1. To understand the basic concepts and the applications of database systems.
2. To master the basics of SQL and construct queries usingSQL.
3. To understand the relational database designprinciples.
4. To become familiar with the basic issues of transaction

processing and concurrencycontrol.

COURSE OUTCOMES

1. Implement the basic knowledge of SQL queries and relationalalgebra.
2. Construct database models for different databaseapplications.
3. Apply normalization techniques for refining of databases.
4. Practice various triggers, procedures, and cursors usingPL/SQL.

COURSEPREREQUISITES

- Basic concepts of files, data structures and design of database systems

CO'S, PO'SMAPPING

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	√		√		√							
CO2	√	√			√							
CO3	√	√			√							
CO4	√		√		√	√						

1. LOW 2.MEDIUM 3.HIGH

COURSE INFORMATION SHEET(CIS)

a) CourseDescription

COURSE TITLE	DATABASE MANAGEMENT SYSTEM LAB			
COURSE CODE	**CS407ES**			
REGULATION	R18			
COURSE STRUCTURE	LECTURES	TUTORIALS	PRACTICALS	CREDITS
	-	-	3	1.5
COURSE COORDINATOR	K. RAMAKRISHNA REDDY, ASSISTANT PROFESSOR, CSE			

b) COURSEPLAN

Division of Experiments	List of experiments	Course outcomes
WEEK1	Concept Design With E-R Model	PO3,PO5
WEEK 2	Relational Model	PO1,PO3,PO5
WEEK 3	Normalization	PO1,PO2,PO5
WEEK 4	Practicing DDL Commands	PO3,PO4,PO5,PO6
WEEK 5	Practicing DML Commands	PO3,PO4,PO5
WEEK 6	Querying (using ANY, ALL, IN, EXISTS, NOT EXISTS, UNION, INTERSECT, Constraints Etc)	PO3,PO4,PO5
WEEK 7	Queries using Aggregate functions, GROUP BY, HAVING and Creation and dropping of Views.	PO3,PO4,PO5
WEEK 8	Triggers (Creation of insert trigger, delete trigger, update trigger)	PO1,PO2,PO3
WEEK 9	Procedures	PO2,PO3,PO4,PO5
WEEK 10	Usage of Cursors	PO2,PO3,PO4,PO5
WEEK 11	Case Study: Book Publishing Company	PO2,PO3,PO4,PO5
WEEK 12	Case Study: General Hospital	PO2,PO3,PO4,PO5
WEEK 13	Case Study: Car Rental Company	
WEEK 14	Case Study: Student Progress Monitoring System	

c) ADDITIONALEXPERIMENTS

d) MarksDistribution

Sessional marks	End semester exam	Internal marks
There shall be a continuous evaluation during the semester for 25 marks. Day-to-day work in the laboratory shall be evaluated for 15 marks and internal practical examination conducted by the concerned teacher shall be evaluated for 10 marks.	75	25

e) EvaluationScheme

s.no	Component	Duration	Marks
1	Day-to-day Evaluation	-	15
2	Internal Practical Examination	3hours	10
3	End Semester Examination	3 hours	75

f)Text books &Referencebooks

1. RamezElmasri, Shamkant, B. Navathe, "Database Systems", Pearson Education, 6thEdition,2013.
2. Peter Rob, Carles Coronel, "Database System Concepts", Cengage Learning, 7thEdition,2008.
3. M L Gillenson, "Introduction to Database Management", Wiley Student Edition,2012.

DATABASE MANAGEMENT SYSTES LAB

LIST OF EXPERIMENTS

INTRODUCTION

Database Management System

This model is like a hierarchical tree structure, used to construct a hierarchy of records in the form of nodes and branches. The data elements present in the structure have Parent-Child relationship. Closely related information in the parent-child structure is stored together as a logical unit. A parent unit may have many child units, but a child is restricted to have only one parent.

The drawbacks of this model are:

The hierarchical structure is not flexible to represent all the relationship proportions, which occur in the real world.
It cannot demonstrate the overall data model for the enterprise because of the non-availability of actual data at the time of designing the data model.
It cannot represent the Many-to-Many relationship.

Network Model

It supports the One-To-One and One-To-Many types only. The basic objects in this model are Data Items, Data Aggregates, Records and Sets.

It is an improvement on the Hierarchical Model. Here multiple parent-child relationships are used. Rapid and easy access to data is possible in this model due to multiple access paths to the data elements.

Relational Model

Does not maintain physical connection between relations Data is organized

in terms of rows and columns in a table

The position of a row and/or column in a table is of no importance The intersection of

a row and column must give a single value

Features of an RDBMS

The ability to create multiple relations and enter data into them An

attractive query language

Retrieval of information stored in more than one table

An RDBMS product has to satisfy at least Seven of the 12 rules of Codd to be

accepted as a full- fledged RDBMS.

Relational Database Management System

RDBMS is acronym for Relation Database Management System. Dr. E. F. Codd first introduced

the Relational Database Model in 1970. The Relational model allows data to be represented in a simple row- column. Each data field is considered as a column and each record is considered as a row. Relational Database is more or less similar to Database Management System. In relational model there is relation between their data elements. Data is stored in tables. Tables have columns, rows and names. Tables can be related to each other if each has a column with a common type of information. The most famous RDBMS packages are Oracle, Sybase and Informix.

Simple example of Relational model is as follows :

Student Details Table

Roll_no	Sname	S_Address
1	Rahul	Satelite
2	Sachin	Ambawadi
3	Saurav	Naranpura

Student Marksheet Table

Rollno	Sub1	Sub2	Sub3
1	78	89	94
2	54	65	77
3	23	78	46

Here, both tables are based on students details. Common field in both tables is Rollno. So we can say both tables are related with each other through Rollno column.

Degree of Relationship

One to One (1:1)

One to Many or Many to One (1:M / M: 1)

Many to Many (M: M)

The Degree of Relationship indicates the link between two entities for a specified occurrence of each.

One to One Relationship : (1:1)

1 1

Student Has Roll No.

One student has only one Rollno. For one occurrence of the first entity, there can be, at the most one related occurrence of the second entity, and vice-versa.

One to Many or Many to One Relationship: (1:M/M: 1)

1 M

Course Contains Students

As per the Institutions Norm, One student can enroll in one course at a time however, in one course, there can be more than one student.

For one occurrence of the first entity there can exist many related occurrences of the second entity and for every occurrence of the second entity there exists only one associated occurrence of the first.

Many to Many Relationship: (M:M)

M M

Students Appears Tests

The major disadvantage of the relational model is that a clear-cut interface cannot be determined. Reusability of a structure is not possible. The Relational Database now accepted model on which major database system are built.

Oracle has introduced added functionality to this by incorporated object-oriented capabilities. Now it is known is as Object Relational Database Management System (ORDBMS). Object-oriented concept is added in Oracle8.

Some basic rules have to be followed for a DBMS to be relational. They are known as Codd's rules, designed in such a way that when the database is ready for use it encapsulates the relational theory to its full potential. These twelve rules are as follows.

E. F. Codd Rules

1. **The Information Rule**

   All information must be store in table as data values.

2. **The Rule of Guaranteed Access**

   Every item in a table must be logically addressable with the help of a table name.

3. **The Systematic Treatment of Null Values**

The RDBMS must be taken care of null values to represent missing or inapplicable information.

4. **The Database Description Rule**

A description of database is maintained using the same logical structures with which data was defined by the RDBMS.

5. **Comprehensive Data Sub Language**

According to the rule the system must support data definition, view definition, data manipulation, integrity constraints, authorization and transaction management operations.

6. **The View Updating Rule**

All views that are theoretically updateable are also updateable by the system.

7. **The Insert and Update Rule**

This rule indicates that all the data manipulation commands must be operational on sets of rows having a relation rather than on a single row.

8. **The Physical Independence Rule**

Application programs must remain unimpaired when any changes are made in storage representation or access methods.

9. **The Logical Data Independence Rule**

The changes that are made should not affect the user's ability to work with the data.The change can be splitting table into many more tables.

10. **The Integrity Independence Rule**

The integrity constraints should store in the system catalog or in the database.

11. **The Distribution Rule**

The system must be access or manipulate the data that is distributed in other systems.

12. **The Non-subversion Rule**

If a RDBMS supports a lower level language then it should not bypass any integrity constraints defined in the higher level.

Object Relational Database Management System

Oracle8 and later versions are supported object-oriented concepts. A structure once created can be reused is the fundamental of the OOP's concept. So we can say Oracle8 is supported Object Relational model, Object – oriented model both. Oracle products are based on a concept known as a

client-server technology. This concept involves segregating the processing of an application between two systems. One performs all activities related to the database (server) and the other performs activities that help the user to interact with the application (client). A client or front-end database application also interacts with the database by requesting and receiving information from database server. It acts as an interface between the user and the database.

The database server or back end is used to manage the database tables and also respond to client requests.

Introduction to ORACLE

ORACLE is a powerful RDBMS product that provides efficient and effective solutions for major database features. This includes:

Large databases and space management control

Many concurrent database users

High transaction processing performance

High availability

Controlled availability

Industry accepted standards

Manageable security

Database enforced integrity

Client/Server environment

Distributed database systems

Portability

Compatibility

Connectivity

An ORACLE database system can easily take advantage of distributed processing by using its Client/ Server architecture. In this architecture, the database system is divided into two parts:

A front-end or a client portion

The client executes the database application that accesses database information and interacts with the user.

A back-end or a server portion

The server executes the ORACLE software and handles the functions required for concurrent, shared data access to ORACLE database.

ROADWAY TRAVELS

"Roadway Travels" is in business since 1977 with several buses connecting differentplaces in India. Its main office is located in Hyderabad.

The company wants to computerize its operations in the following areas:

Reservations

Ticketing

Cancellations

Reservations :

Reservations are directly handeled by booking office.reservations can be made 60 days in advance in either cash or credit. In case the ticket is not available,a wait listed ticket is issued to the customer. This ticket is confirmed against the cancellation.

Cancellation and modification:

Cancellations are also directly handed at the booking office. Cancellation charges will be charged.

Wait listed tickets that do not get confirmed are fully refunded.

Exp No: 1 **Date: _ _/_ _/ _ _**

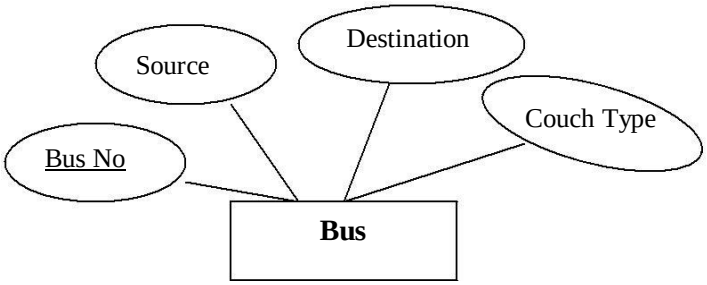
AIM: Analyze the problem and come with the entities in it. Identify what Data has to be persisted in the databases.

The Following are the entities:

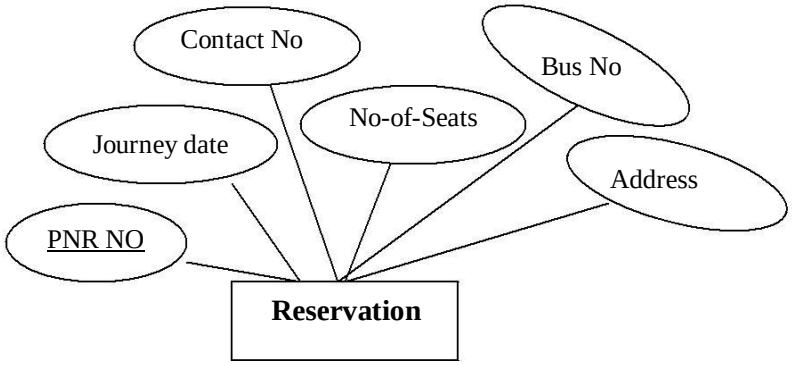
1 .Bus
2. Reservation
3. Ticket
4. Passenger
5. Cancellation

The attributes in the Entities:

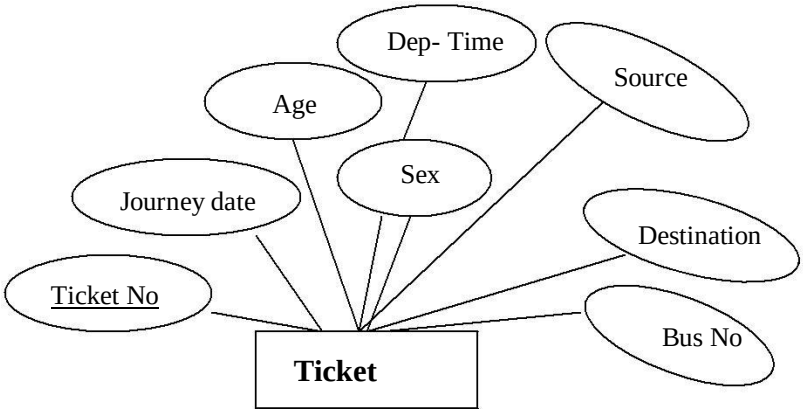
Bus:(Entity)



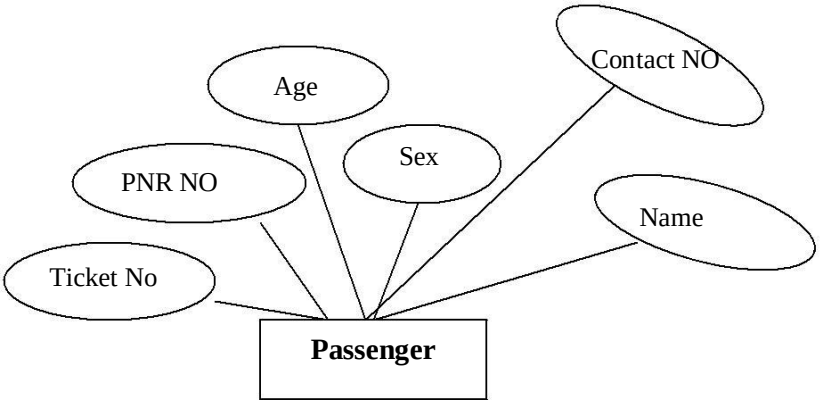
Reservation (Entity)



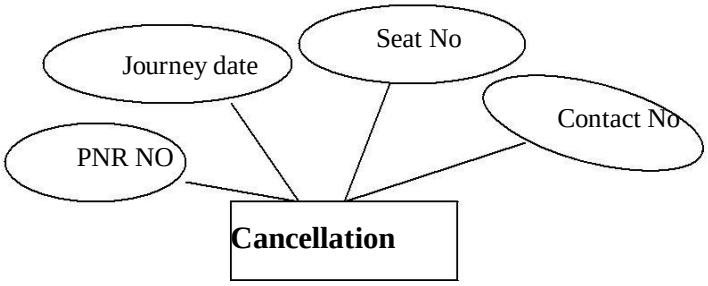
Ticket :(Entity)



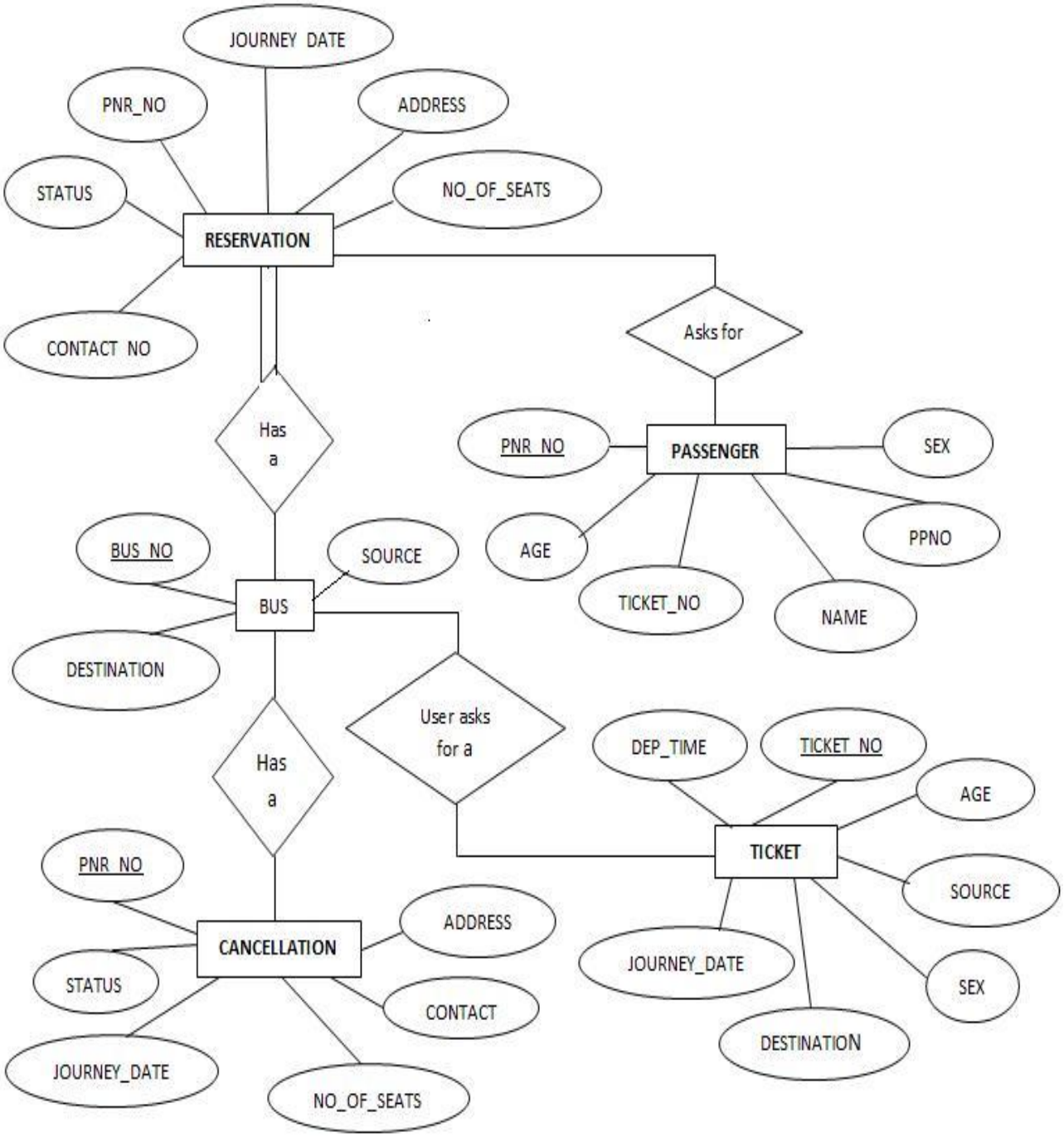
Passenger:



Cancellation (Entity)



Concept design with E-R Model:



WEEK-1

CREATION OF TABLES

1) Create a table called Employee with the followingstructure.

Name	Type
Empno	Number
Ename	Varchar2(10)
Job	Varchar2(10)
Mgr	Number
Sal	Number

a. **Add a column commission with domain to the Employeetable.**
b. Insert any five records into thetable.
c. Update the column details ofjob
d. Rename the column of Employ table using altercommand.
e. Delete the employee whose Empnois105.

SOLUTION:

SQL> create table employee(empnonumber,ename varchar2(10),job varchar2(10),mgrnumber,sal number);

Table created.

SQL> desc employee;

Name Null? Type

--- --- ---

EMPNO

NUMBER

ENAME VARCHAR2(10)

JOB VARCHAR2(10)

MGR NUMBER

SAL NUMBER

a. Add a column commission with domain to the Employeetable.

SQL> alter table employee add(commission number); Table altered.

```
SQL> desc employee;
Name                                      Null?    Type
----------------------------------------- -------- ---------
------------------

EMPNO                                              NUMBER
ENAME                                              VARCHAR2(10)
JOB                                               VARCHAR2(10)
MGR                                               NUMBER
SAL                                               NUMBER
COMMISSION                                        NUMBER
```

b. Insert any five records into thetable.

```
SQL> insert into employee values(&empno,'&ename','&job',&mgr,&sal,'&commission');
Enter value for empno: 101
Enter value for ename: abhi Enter value for job: manager

Enter value for mgr: 1234 Enter value for sal: 10000 Enter value for commission:70
old 1: insert into employee values(&empno,'&ename','&job',&mgr,&sal,'&commission') new
1: insert into employee values(101,'abhi','manager',1234,10000,'70')
1 row created.

SQL> /
Enter value for empno: 102 Enter value for ename: rohith Enter value for job: analyst Enter
value for mgr: 2345 Enter value for sal: 9000
Enter value for commission: 65
old 1: insert into employee values(&empno,'&ename','&job',&mgr,&sal,'&commission') new
1: insert into employee values(102,'rohith','analyst',2345,9000,'65')

1 row created.
SQL> /Enter value for empno: 103 Enter value for ename: david Enter value for job: analyst
Enter value for mgr: 3456 Enter value for sal: 9000
Enter value for commission: 65
old 1: insert into employee values(&empno,'&ename','&job',&mgr,&sal,'&commission') new
1: insert into employee values(103,'david','analyst',3456,9000,'65')
1 row created.

SQL> /Enter value for empno: 104 Enter value for ename: rahul Enter value for job: clerk
Enter value for mgr: 4567 Enter value for sal: 7000
Enter value for commission: 55
old 1: insert into employee values(&empno,'&ename','&job',&mgr,&sal,'&commission') new
1: insert into employee values(104,'rahul','clerk',4567,7000,'55')
1 row created.
```

SQL> / Enter value for empno: 105 Enter value for ename: pramod Enter value for job: salesman Enter value for mgr: 5678 Enter value for sal:5000
Enter value for commission: 50
old 1: insert into employee values(&empno,'&ename','&job',&mgr,&sal,'&commission') new 1: insert into employee values(105,'pramod','salesman',5678,5000,'50')
1 row created.

SQL> select * from employee;

EMPNOENAME JOB MGRSALCOMMISSION
---------- ---------- ---------- ---------- ---------- ----------

101 abhi	Manager	1234	10000	70
102 rohith	Analyst	2345	9000	65
103 david	Analyst	3456	9000	65
104 rahul	Clerk	4567	7000	55
105 pramod	salesman	5678	5000	50

c. Update the column details ofjob

SQL> update employee set job='trainee' where empno=103; 1 row updated.

SQL> select * from employee;

EMPNOENAME JOBMGR SALCOMMISSION
---------- ---------- ---------- ---------- ---------- ----------

101 abhi	manager	1234	10000	70
102 rohith	analyst	2345	9000	65
103 david	trainee	3456	9000	65
104 rahul	clerk	4567	7000	55
105 pramod	salesman	5678	5000	50

d. Rename the column of Employ table using altercommand.

SQL> alter table employee rename column mgr to manager_no; Table altered.
SQL> desc employee;

Name	Null?	Type
EMPNO		NUMBER
ENAME		VARCHAR2(10)
JOB		VARCHAR2(10)
MANAGER_NO		NUMBER
SAL		NUMBER
COMMISSION		NUMBER

e. Delete the employee whose Empno is105.

SQL> delete employee where empno=105; 1 row deleted.

SQL> select * from employee;

EMPNO	ENAME	JOB	MANAGER_NO	SAL	COMMISSION
101	abhi	manager	1234	10000	70
102	rohith	analyst	2345	9000	65
103	david	trainee	3456	9000	65
104	rahul	clerk	4567	7000	55

2) Create department table with the followingstructure.

Name	**Type**
Deptno	Number
Deptname	Varchar2(10)
Location	Varchar2(10)

a. Add column designation to the departmenttable.
b. Insert values into thetable.
c. List the records of dept table grouped bydeptno.
d. Update the record where deptnois9.
e. Delete any column data from thetable.

SOLUTION:

SQL> create table department(deptnonumber,deptname varchar2(10),locationvarchar2(10)); Table created.

SQL> desc department;

Name	Null?	Type
DEPTNO		NUMBER
DEPTNAME		VARCHAR2(10)
LOCATION		VARCHAR2(10)

a. Add column designation to the departmenttable.

SQL> alter table department add(designation varchar2(10)); Table altered.

SQL> desc department;

Name	Null?	Type
DEPTNO		NUMBER
DEPTNAME		VARCHAR2(10)
LOCATION		VARCHAR2(10)
DESIGNATION		VARCHAR2(10)

b. Insert values into thetable.

SQL> insert into department values(&deptno,'&deptname','&location','&designation'); Enter value for deptno: 9
Enter value for deptname: accounting
Enter value for location: hyderabad
Enter value for designation: manager
old 1: insert into department values(&deptno,'&deptname','&location','&designation') new 1: insert into department values(9,'accounting','hyderabad','manager')

1 row created.

SQL> /

Enter value for deptno: 10
Enter value for deptname: research Enter value for location: chennai Enter value for designation: professor
old 1: insert into department values(&deptno,'&deptname','&location','&designation') new 1: insert into department values(10,'research','chennai','professor')

1 row created.
SQL> /
Enter value for deptno: 11 Enter value for deptname:sales
Enter value for location: banglore Enter value for designation: salesman
old 1: insert into department values(&deptno,'&deptname','&location','&designation') new 1: insert into department values(11,'sales','banglore','salesman')
1 row created.
SQL> /
Enter value for deptno: 12
Enter value for deptname: operations Enter value for location: mumbai Enter value for designation: operator
old 1: insert into department values(&deptno,'&deptname','&location','&designation') new

1: insert into department values(12,'operations','mumbai','operator')
1 row created.

SQL> insert into department values(&deptno,'&deptname','&location','&designation');
Enter value for deptno: 9
Enter value for deptname: accounting Enter value for location: chennai Enter value for designation: manager
old 1: insert into department values(&deptno,'&deptname','&location','&designation') new 1: insert into department values(9,'accounting','chennai','manager')
1 row created.

SQL> select * from department ;

DEPTNO	DEPTNAME	LOCATION	DESIGNATION
9	accounting	hyderabad	manager
10	research	chennai	professor
11	sales	banglore	salesman
12	operations	mumbai	operator
9	accounting	chennai	manager

a. List the records of dept table grouped bydeptno.

SQL> select deptno,deptname from department group by deptno,deptname;

DEPTNO	DEPTNAME
9	accounting
12	operations
10	research
11	sales

c. Update the record where deptno is9.

SQL> update department set designation='accountant' where deptno=9; 2 rows updated.

SQL> select * from department;

DEPTNO DEPTNAME LOCATION DESIGNATION

9	Accounting	hyderabad	accountant
10	Research	chennai	professor
11	Sales	banglore	salesman
12	Operations	mumbai	operator
9	Accounting	chennai	accountant

d. Delete any column data from thetable.

SQL> alter table department drop(designation); Table altered.

SQL> select * from department;

DEPTNO	DEPTNAME	LOCATION
9	accounting	hyderabad
10	research	chennai
11	sales	banglore
12	operations	mumbai
9	accounting	Chennai

WEEK -2
QUERIES USING DDL AND DML

1. **a. Create a user and grant all permissions to theuser.**
   b. Insert the any three records in the employee table and use rollback. Check the result.
   c. Add primary key constraint and not null constraint to the employeetable.
   d. Insert null values to the employee table and verify theresult.

SOLUTION:

a) create a user and grant all permissions to theuser.

CONNECT <USER-NAME>/<PASSWORD>@<DATABASE NAME>;

--Create user query

CREATE USER <USER NAME> IDENTIFIED BY <PASSWORD>;

--Provide roles

GRANT CONNECT,RESOURCE,DBA TO <USER NAME>;

--Assigning privileges

GRANT CREATE SESSION GRANT ANY PRIVILEGE TO <USER NAME>; GRANT UNLIMITED TABLESPACE TO <USER NAME>;

--Provide access to tables.

GRANT SELECT, UPDATE, INSERT, DELETE ON <TABLE NAME> TO <USER NAME>;

b) Insert the any three records in the employee table and use rollback. Check the result.

SQL> SELECT * FROM EMPLOYEE;

EMPNO	ENAME	JOB	MANAGER_NO	SAL	COMMISSION
101	**abhi**	**manager**	**1234**	**1100**	**70**
102	**rohith**	**analyst**	**2345**	**9000**	**65**
david	trainee	3456	9000	65	
104	rahul	clerk	4567	7000	55

SQL> insert into employee
values(&empno,'&ename','&job',&manager_no,&sal,&commission); Enter value for empno: 105
Enter value for ename: aravind Enter value for job: salesman Enter value for manager_no: 5678 Enter value for sal: 5000
Enter value for commission: 50
old 1: insert into employee
values(&empno,'&ename','&job',&manager_no,&sal,&commission) new 1: insert into employee values(105,'aravind','salesman',5678,5000,50)
1 row created.

SQL> rollback; Rollback complete.

SQL> SELECT * FROM EMPLOYEE;

EMPNO	ENAME	JOB	MANAGER_NO	SAL	COMMISSION
101	**abhi**	**manager**	**1234**	**1100**	**70**
102	**rohith**	**analyst**	**2345**	**9000**	**65**
103	**david**	**trainee**	**3456**	**9000**	**65**
104	**rahul**	**clerk**	**4567**	**7000**	**55**

c) Add primary key constraint and not null constraint to the employeetable.

SQL> alter table employee modify(empno number primary key, ename varchar2(10) not null); Table altered.

SQL> desc employee;

Name	Null?	Type
- EMPNO	NOTNULL	NUMBER
ENAME	NOT NULL	VARCHAR2(10)
JOB		VARCHAR2(10)
MANAGER_NO	NUMBER	
SALNUMBER		
COMMISSION		NUMBER

d) Insert null values to the employee table and verify theresult.

SQL> desc employee;

Name	Null?	Type
EMPNO	NOTNULL	NUMBER
ENAME	NOTNULL	VARCHAR2(10)
JOB	NOTNULL	VARCHAR2(10)
MANAGER_NO		NUMBER
SAL	NOTNUL	
L	NUMBER	
COMMISSION		NUMBER

SQL> insert into employee values(&empno,'&ename','&job',&manager_no,&sal,&commission); Enter value for empno: 105
Enter value for ename: mohith Enter value for job: salesman Enter value for manager_no: 5678 Enter value for sal: null
Enter value for commission: 50
old 1: insert into employee values(&empno,'&ename','&job',&manager_no,&sal,&commission) new 1: insert into employee values(105,'mohith','salesman',5678,null,50)
insert into employee values(105,'mohith','salesman',5678,null,50)
*

a. create a user and grant all permissions to theuser.
Insert values in the department table and usecommit.
Add constraints like unique and not null to the departmenttable.
Insert repeated values and null values into thetable.

SOLUTION:

create a user and grant all permissions to theuser.

CONNECT <USER-NAME>/<PASSWORD>@<DATABASE NAME>;

--Create user query

CREATE USER <USER NAME> IDENTIFIED BY <PASSWORD>;

--Provide roles

GRANT CONNECT,RESOURCE,DBA TO <USER NAME>;

--Assigning privileges

GRANT CREATE SESSION GRANT ANY PRIVILEGE TO <USER NAME>;
GRANT UNLIMITED TABLESPACE TO <USER NAME>;

--Provide access to tables.

GRANT SELECT, UPDATE, INSERT, DELETE ON <TABLE NAME> TO <USER NAME>;

Insert values in the department table and usecommit.

SQL> insert into department values(&deptno,'&deptname','&location'); Enter value for deptno: 13
Enter value for deptname: sales Enter value for location: delhi
old 1: insert into department values(&deptno,'&deptname','&location') new 1: insert into department values(13,'sales','delhi')
1 row created.

SQL> commit; Commit complete.

SQL> select * from department;

DEPTNO	DEPTNAME	LOCATION
accounting	hyderabad	
research	chennai	
sales	banglore	
operations	mumbai	
9	accounting	chennai
13	sales	delhi

6 rows selected.

a) Add constraints like unique and not null to the departmenttable.

SQL> alter table department modify(deptnonumberunique); Table altered.
SQL> alter table department modify(location varchar2(10) notnull); Table altered.
SQL> DESC DEPARTMENT;

Name	Null?	Type
DEPTNO		NUMBER
DEPTNAME		VARCHAR2(10)
LOCATION	NOT NULL	VARCHAR2(10)

b) Insert repeated values and null values into thetable.

SQL> insert into department values(&deptno,'&deptname','&location'); Enter value for deptno: 10
Enter value for deptname: research Enter value for location:
old 1: insert into department values(&deptno,'&deptname','&location') new 1: insert into department values(10,'research',")
insert into department values(10,'research',")
SQL> insert into department values(&deptno,'&deptname','&location'); Enter value for deptno: 10
Enter value for deptname: research Enter value for location: hyderabad
old 1: insert into department values(&deptno,'&deptname','&location') new 1: insert into department values(10,'research','hyderabad')

WEEK -3

QUERIES USING AGGREGATE FUNCTIONS

AIM :-

Queriesusingaggregatefunctions(COUNT,AVG,MIN,MAX,SUM),Group by,Orderby,Having.

E_id	E_name	Age	Salary
101	Anu	22	9000
102	Shane	29	8000
103	Rohan	34	6000
104	Scott	44	10000
105	Tiger	35	8000
106	Alex	27	7000
107	Abhi	29	8000

(i) Create Employee table containing allRecords.

SQL> create table emp(eidnumber,ename varchar2(10),age

number,salary

number); Table created.

SQL> desc emp;

Name	Null?	Type
----------	--------	----------
EID		-- NUMBER
ENAME		VARCHAR2(10)
AGE		NUMBER
SALARY		NUMBER

(ii) Count number of employee names from employeetable.

SQL> select count(ename) from emp; COUNT(ENAME)

7

(iii) Find the Maximum age from employeetable.

SQL> select max(age) from emp;

MAX(AGE)
---------- 44

(iv) Find the Minimum age from employeetable.

SQL> select min(age)
from emp;
MIN(AGE)
---------- 22

Display the Sum of age employeetable.

SQL> select sum(age) from emp;

SUM(AGE)
---------- 220

Display the Average of age from Employeetable.

SQL> select avg(age) from emp; AVG(AGE)
---------- 31.4285714

Create a View for age in employeetable.

SQL> create or replace view A as select age from emp where age<30; View created.

Displayviews

SQL> select * from A;

AGE
---------- 22
29
27
29

(v) Find grouped salaries of employees.(group byclause)

```
SQL> select salary from emp group
     by salary; SALARY
---------------
9000
10000
8000
6000
7000
```

(x).Find salaries of employee in Ascending Order.(order by clause)

```
SQL> select ename,salary from
     emp order by salary;
ENAME        SALARY
----------   ----------
rohan        6000
alex         7000
shane        8000
abhi         8000
tiger        8000
anu          9000
scott        10000
```

7 rowsselected.

(xi) Find salaries of employee in DescendingOrder.SQL> select ename,salary from emp order by salary desc;

```
ENAME          SALARY
--------------  ---------------
scott          10000
anu            9000
Shane          8000
Abhi           8000
Tiger          8000
Alex           7000
Rohan          6000
```

7 rows selected.

(xii)HavingClause.

SQL> select ename,salary from emp where age<29 group by ename,salary having salary<10000;

```
ENAME        SALARY
-----------  --------------
alex                 7000
anu                  9000
```

WEEK – 4

PROGRAMS ON PL/SQL

1 a) Write a PL/SQL block to find the maximum number from given threenumbers.

```
Declareanumber;
bnumber;
cnumber;
begin
  a:=&a;
  b:=&b;
  c:=&c;
  if (a>b and a>c) then
          dbms_output.put_line('a       is
  maximum ' || a); elsif (b>a and b>c)
  then
          dbms_output.put_line('b is maximum ' || b);
  else

  end if; end;
  /
```

1b) write a PL/SQL program for swapping 2 numbers.

```
Declarea number(3);
b number(3);
begin
      a:=&a;
      b:=&b;
      dbms_output.put_line(„Beforeswappinga=„||a
      ||‟andb=„||b); a:=a+b;
      b:=a-b;
      a:=a-b;
      dbms_output.put_line(„Afterswappinga=„||a
      ||‟andb=„||b); end;
  /
```

2 a) Write a PL/SQL program to find the total and average of 4 subjects and display the grade

```
declarejava number(10);

dbmsnumber(10);

co number(10);

mfcsnumber(10);

total number(1);

avgsnumber(10);

per number(10);

begindbmsutput.put_line('ENTER THE MARKS');
java:=&java;
 dbms:=&dbms;
co:=&co;
 mfcs:=&mfcsl;
total:=(java+dbms+co+mfcs);
 per:=(total/600)*100;
        if java<40 or dbms<40 or co<40 or mfcs<40
         then dbms_output.put_line('FAIL');
        if per>75 then
            dbms_output.put_lin
        e('GRADE    A');    elsif
        per>65 and per<75then
            dbms_output.put_lin
        e('GRADE    B');    elsif
        per>55 and per<65then
            dbms_output.put_line('GRADE C');
        else
            dbms_output.put_line('INVALID INPUT');
        end if;
            dbms_output.put_line('PERCENTAGE IS '||per);
    end;
    /
```

2 b) Write a program to accept a number and find the sum of thedigits

```
declaren number(5):=&n;
s number:=0;
rnumber(2):=0;
begin
while n !=0 loop r:=mod(n,10);
s:=s+r; n:=trunc(n/10);
end loop;
dbms_output.put_line('sum of digits of given number is '||s);
end;
/
```

3 a) PL/SQL Program to accept a number from user and print number in reverseorder.

```
Declarenum1 number(5);
num2 number(5);
rev number(5);
begin num1:=&num1;
rev:=0;
whilenum1>0 loop
num2:=num1 mod 10;
rev:=num2+(rev num1:=floor(num1/10);
end loop;
dbms_output.put_line('Reverse number is: '||rev); end;
/
```

3b) Write a PL / SQL program to check whether the given number is prime or not.

```
Declarenum number;
inumber:=1;
c number:=0;
begin
num:=&num;
for i in 1..num loop
if((mod(num,i))=0) then
c:=c+1;
end if;
end loop;
if(c>2) then
dbms_output.put_line(num||' not a prime');
else
dbms_output.put_line(num||' is prime');
end if;
end;
/
```

4 a) Write a PL/SQL program to find the factorial of a givennumber.

```
declare
inumber(4):=1;
n number(4):=&n;
f number(4):=1;
begin
for i in 1..n loop
f:=f*i;
end loop;
Dbms_output.put_line('the factorial of '||n||' is:'||f);
end;
/
```

4 b) calculate the area of a circle for a value of radius varying from 3 to 7. Store the radius and the corresponding values of calculated area in table areas. Consisting of two columns radius andarea

```
Declare
pi constant number(4,2) := 3.14;
 radius number(5);
area number(14,2);
Begin radius := 3;
While radius <=7 Loop
area := pi* power(radius,2);
Insert into areas values (radius, area);
radius:= radius+1; end loop;
end;
/
```

5a) Write a PL/SQL program to accept a string and remove the vowels from the string. (When 'hello' passed to the program it should display 'Hll' removing e and o from the world Hello).

```
set server output on set verify off
accept v string prompt "Please enter your string: ";
declare v new string varchar2(100);
begin
v new string := reg exp replace('&v string', '[aeiou AEIOU]',''); dbms output .put_ line('The new string
is: ' || v new string);
end;
        /
```

5 b) Write a PL/SQL program to accept a number and a divisor. Make sure the divisor is less than or equal to 10. Else display an error message. Otherwise Display theremainder.

```
select remainder(37,5) "remainder" from dual ;
```

WEEK -5

PROCEDURESANDFUNCTIONS

1) calculate the net salary and year salary if da is 30% of basic, hra is 10% of basic and pf is 7% if basic salary is less than 8000, pf is 10% if basic sal between 8000 to160000.

```
declare
e name varchar2(15);
basic number;
d a number; h r a number;
pf number;
net salary number;
year salary number;
begin
e name:='&e name';
basic:=&basic;
da:=basic * (30/100);
hra:=basic * (10/100);
if (basic < 8000) then
pf:=basic * (8/100);
elsif (basic >= 8000 and basic <= 16000) then
pf:=basic * (10/100);

end if;
netsalary:=basic + da + hra - pf; yearsalary := netsalary*12;

dbms_output.put_line('Employee name : ' || ename); dbms_output.put_line('Providend Fund : ' || pf);
dbms_output.put_line('Net salary : ' || netsalary); dbms_output.put_line('Year salary : '|| yearsalary);
end;
/
```

2) Create a function to find the factorial of a given number and hence findNCR.

```
SQL> create or replace function fact(n number) return number isa number:=n; f number:=1;
i number;
begin
for i in 1..n loop
f:=f*a;
a:=a-1;
end loop;
return f;
end;
/
SQL> create or replace function ncr(n number ,r number) return number isn1
number:=fact(n);
r1 number:=fact(r);
nr1 number:=fact(n-r);
resultnumber;
begin result:=(n1)/(r1*nr1); return result;
end;
/
```

3) Print Fibonacci series using localfunctions.

```
sql>create or replace function fib (n positive) return integer is begin
if (n = 1) or (n = 2) then -- terminating condition return 1;
else
return fib(n - 1) + fib(n - 2); -- recursive call end if;
end fib;
        /
```

-- Test Fibonacci Series:

```
SQL>SELECT fib(1), fib(2), fib(3), fib(4), fib(5) FROM dual;
```

4) write a pl/sql function accept date of birth as "dd-mm-yyyy" and sum all digits tillyou get single digit number to show as he luckynumber.

```
SQL> set serverout on SQL> declare
l_input varchar2(20) := '31/01/1978';
l_output int;
begin
loop
dbms_output.put_line('  ');
dbms_output.put_line('l_input='||l_input); l_output := 0;
for i in 1 .. length(l_input) loop
if substr(l_input,i,1) between '0' and '9' then
l_output := l_output + to_number(substr(l_input,i,1));
end if;
end loop; dbms_output.put_line('l_output='||l_output); exit when l_output< 10;
l_input := to_char(l_output);
 end loop;
dbms_output.put_line('  ');
dbms_output.put_line('Lucky='||l_output); end;
/

------------------
l_input=31/01/1978 l_output=30
------------------
l_input=30 l_output=3
------------------
Lucky=3

PL/SQL procedure successfully completed
```

WEEK-6
TRIGGERS

1. Create a row level trigger for the customers table that would fire for INSERT or UPDATE or DELETE operations performed on the CUSTOMERS table. This trigger will display the salary difference between the old values and new values:

CUSTOMERS table:

ID	NAME	AGE	ADDRESS	SALARY
1	Alive	24	Khammam	2000
2	Bob	27	Kadappa	3000
3	Catri	25	Guntur	4000
4	Dena	28	Hyderabad	5000
5	Eeshwar	27	Kurnool	6000
6	Farooq	28	Nellur	7000

```
CREATE OR REPLACE TRIGGER display_salary_changes BEFORE DELETE OR INSERT OR
UPDATE ON customers FOR EACH ROW
WHEN (NEW.ID > 0) DECLARE
sal_diff number; BEGIN
sal_diff := :NEW .salary - :OLD .salary; dbms_output.put_line('Old salary: ' || :OLD.salary);
dbms_output.put_line('New salary: ' || :NEW.salary); dbms_output.put_line('Salary difference: ' ||
sal_diff);
    END;
    /
```

Trigger created.

Here following two points are important and should be noted carefully:

OLD and NEW references are not available for table level triggers, rather you can use them for record leveltriggers.

If you want to query the table in the same trigger, then you should use the AFTER keyword, because triggers can query the table or change it again only after the initial changes are applied and the table is back in a consistent state.

Above trigger has been written in such a way that it will fire before any DELETE or INSERT or UPDATE operation on the table, but you can write your trigger on a single or multiple operations, for example BEFORE DELETE, which will fire whenever a record will be deleted using DELETE operation on the table.

Let us perform some DML operations on the CUSTOMERS table. Here is one INSERT statement, which will create a new record in the table:

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY) VALUES (7, 'Kriti', 22, 'HP', 7500.00);

When a record is created in CUSTOMERS table, above create trigger display_salary_changeswill be fired and it will display the following result:

Old salary:
New salary: 7500 Salary difference:

2) Convert employee name into uppercase whenever an employee record is inserted or updated. Trigger to fire before the insert orupdate.

```
SQL> create table Employee(
2   ID              VARCHAR2(4 BYTE) NOTNULL,
3   First_Name      VARCHAR2(10BYTE),
4   Last_Name       VARCHAR2(10BYTE),
5   Start_Date      DATE,
6   End_Date        DATE,
7   Salary          NUMBER(8,2),
8   City            VARCHAR2(10BYTE),
9   Description
    VARCHAR2(15
BYTE) 10)
11 /

Table created.

SQL> CREATE OR REPLACE TRIGGER employee_ insert_ update
BEFORE INSERT OR UPDATE ON employee
FOR EACH ROW 4DECLARE
5       dup_ flag INTEGER; 6BEGIN
--Force all employee names toupper case.
:NEW. first_ name :=UPPER(:NEW.  first_name); 9END;
10 /

Trigger created.

SQL> insert into Employee(ID, First_Name, Last_Name, Start_Date, End_Date, Salary, City,
Description)
2       values('01','Jason',        'Martin', to_date('19960725','YYYYMMDD'),
to_date('20060725','YYYYMMDD'), 1234.56, 'Toronto', 'Programmer')
3 /

1 row created.
```

```
SQL> insert into Employee(ID, First_Name, Last_Name, Start_Date, End_Date, Salary, City,
Description)
2       values('02','Alison', 'Mathews', to_date('19760321','YYYYMMDD'),
to_date('19860221','YYYYMMDD'), 6661.78, 'Vancouver','Tester')
3 /

1 row created.

SQL> insert into Employee(ID, First_Name, Last_Name, Start_Date, End_Date, Salary, City,
Description)
2       values('03','James',      'Smith', to_date('19781212','YYYYMMDD'),
to_date('19900315','YYYYMMDD'), 6544.78, 'Vancouver','Tester')
3 /

1 row created.

SQL> insert into Employee(ID, First_Name, Last_Name, Start_Date, End_Date, Salary, City,
Description)
2       values('04','Celia',      'Rice',    to_date('19821024','YYYYMMDD'),
to_date('19990421','YYYYMMDD'), 2344.78,'Vancouver','Manager')
3 /

1 row created.

SQL> insert into Employee(ID, First_Name, Last_Name, Start_Date, End_Date, Salary,
City, Description)
  2       values('05','Robert',                   'Black',
to_date('19840115','YYYYMMDD'),
to_date('19980808','YYYYMMDD'),                 2334.78,
'Vancouver','Tester')
  3 /
1 row created.

SQL> insert into Employee(ID, First_Name, Last_Name, Start_Date, End_Date, Salary,
City, Description)
  2       values('06','Linda',    'Green',
to_date('19870730','YYYYMMDD'),
to_date('19960104','YYYYMMDD'),         4322.78,'New
York','Tester')
  3 /
1 row created.

SQL> insert into Employee(ID, First_Name, Last_Name, Start_Date, End_Date, Salary,
City, Description)
  2       values('07','David',    'Larry',
to_date('19901231','YYYYMMDD'),
to_date('19980212','YYYYMMDD'),         7897.78,'New
York','Manager')
  3 /
1 row created.
```

```
SQL> insert into Employee(ID, First_Name, Last_Name, Start_Date, End_Date, Salary,
City, Description)
  2             values('08','James',    'Cat',
                to_date('19960917','YYYYMMDD'),
to_date('20020415','YYYYMMDD'),
1232.78,'Vancouver','Tester')
  3 /

1 row created.

SQL>   select
  *         from
 Employee  2
 /
```

ID	FIRST_NAME	LAST_NAME	START_DAT	END_DATE	SALARY	CITY DESCRIPTION
01	JASON	Martin	25-JUL-96	25-JUL-06	1234.56	Toronto Programmer
02	ALISON	Mathews	21-MAR-76	21-FEB-86	6661.78	Vancouver Tester
03	JAMES	Smith	12-DEC-78	15-MAR-90	6544.78	Vancouver Tester
04	CELIA	Rice	24-OCT-82	21-APR-99	2344.78	Vancouver Manager
05	ROBERT	Black	15-JAN-84	08-AUG-98	2334.78	Vancouver Tester
06	LINDA	Green	30-JUL-87	04-JAN-96	4322.78	New YorkTester
07	DAVID	Larry	31-DEC-90	12-FEB-98	7897.78	New York Manager
08	JAMES	Cat	17-SEP-96	15-APR-02	1232.78	Vancouver Tester

```
8 rows selected.

SQL> drop table Employee 2 /

Table dropped.
```

3) Trigger before deleting a record from emp table. Trigger will insert the row to be deleted into another table and also record the user who has deleted therecord.

```
SQL>    CREATE OR REPLACE TRIGGERemployee_before_delete
 2    BEFOREDELETE
 3      ON employee
 4      FOR EACHROW
 5    DECLARE
 6      v_usernamevarchar2(10);
 7    BEGIN
 8     -- Find username of person performing the DELETE on thetable
 9      SELECT user INTOv_username
10       FROMdual;
11    -- Insert record into audit table
12      INSERT INTO employee_audit (id, salary, delete_date,deleted_by)
13                    VALUES (:old.id,:old.salary, sysdate, v_username);
14    END;
15    /
```

Trigger created.

SQL> delete from employee; 8 rows deleted.
SQL> select * from employee_audit;

```
ID          SALARY DELETE     _DADELETED_BY
----  ---------- --------- ---------------- 01        1234.56 09-SEP-06JAVA2S
02    6661.78 09-SEP-06JAVA2S
03    6544.78 09-SEP-06JAVA2S
04    2344.78 09-SEP-06JAVA2S
05    2334.78 09-SEP-06JAVA2S
06    4322.78 09-SEP-06JAVA2S
07    7897.78 09-SEP-06JAVA2S
08    1232.78 09-SEP-06JAVA2S

8 rows selected.
```

SQL>drop tableemployee_audit;
Table dropped.

WEEK-7
PROCEDURES

1) Create a procedure to reverse astring.

```
CREATE OR REPLACE PROCEDURE ReverseOf(input IN
varchar2(50)) IS DECLARE
      reverse
varchar2(50);
BEGIN
      FOR      i      in      reverse
            1..length(input)   LOOP
            reverse                :=
            reverse||''||substr(input, i,
            1);
      END LOOP;

      dbms_output.put
_line(reverse); END;
/
```

WEEK-8
CURSORS

DEFINITION OF A CURSOR

1. Cursor can be created to store the values from tabletemporally.
2. In execution these values fetch from cursor for access the database
   - Create cursor fetch the values from thetable
   - Declare thevariables
   - Open thecursor
   - Fetch the values from thecursor
   - Close thecursor

CURSOR EXAMPLE:

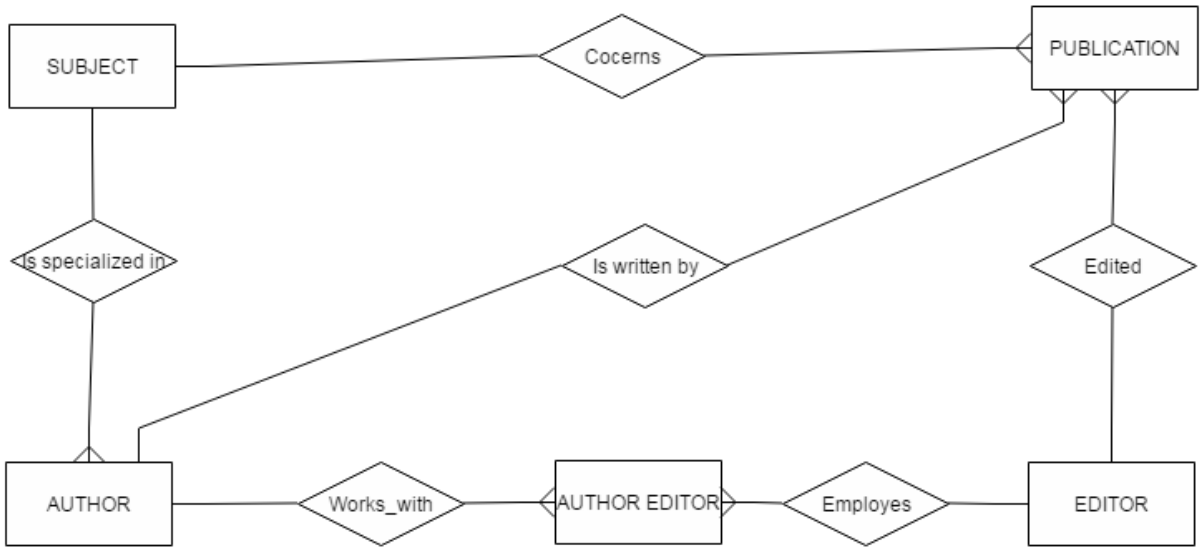
```
Declarecursor xx is select empno,ename,sal from emp26;
a_empno emp26.empno%type;
a_ename emp26.ename%type;
a_sal emp26.sal%type;
begin
open xx;
loop
fetch xx into a_empno,a_ename,a_sal;
exit when xx% not found;
dbms_output.put_line(a_empno||' '||a_ename||' '||a_sal);
end loop;
close xx;
end;
      SQL> /
```

WEEK-9

CASE STUDY: BOOK PUBLISHING COMPANY

AIM: A publishing company produces scientific books on various subjects. The books are written by authors who specialize in one particular subject. The company employs editors who, not necessarily being specialists in a particular area, each take sole responsibility for editing one or more publications.

A publication covers essentially one of the specialist subjects and is normally written by a single author. When writing a particular book, each author works with on editor, but may submit another work for publication to be supervised by other editors. To improve their competitiveness, the company tries to employ a variety of authors, more than one author being a specialist in a particular subject.

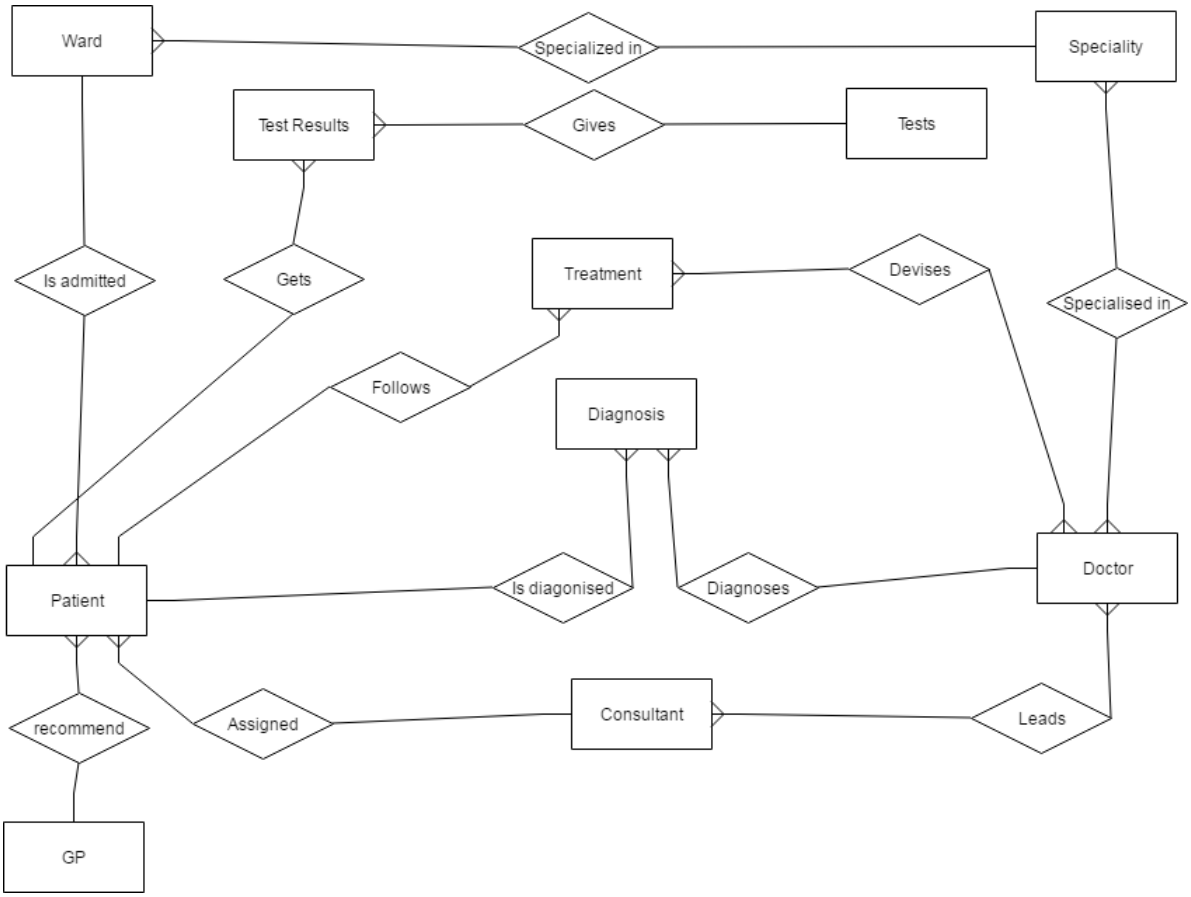


WEEK -10

CASE STUDY: GENERAL HOSPITAL

AIM: A General Hospital consists of a number of specialized wards (such as Maternity, Paediatry, Oncology, etc). Each ward hosts a number of patients, who were admitted on the recommendation of their own GP and confirmed by a consultant employed by the Hospital. On admission, the personal details of every patient are recorded.

A separate register is to be held to store the information of the tests undertaken and the results of a prescribed treatment. A number of tests may be conducted for each patient. Each patient is assigned to one leading consultant but may be examined by another doctor, if required. Doctors are specialists in some branch of medicine and may be leading consultants for a number of patients, not necessarily from the sameward.

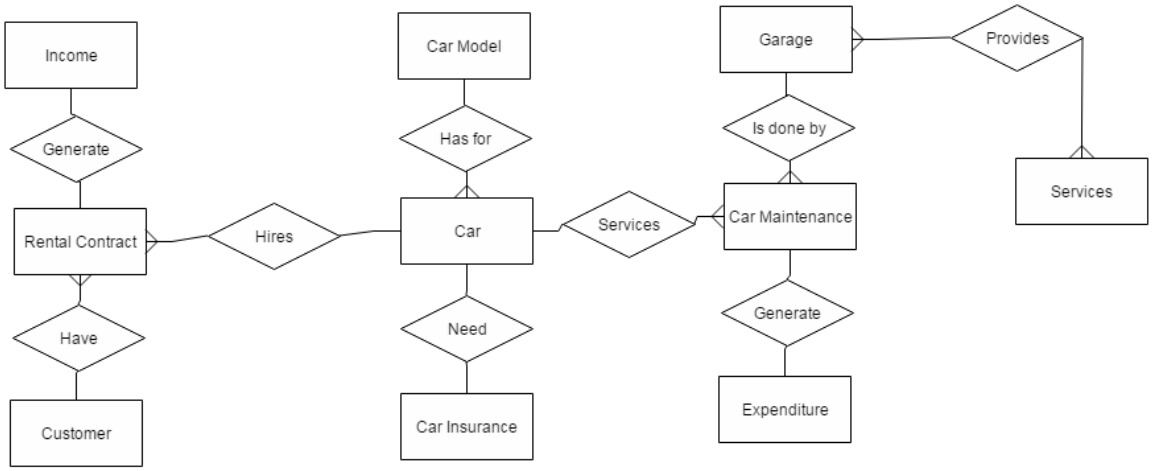


WEEK -11
CASE STUDY: CAR RENTAL COMPANY

AIM: A database is to be designed for a Car Rental Co. (CRC). The information required includes a description of cars, subcontractors (i.e. garages), company expenditures, company revenues and customers. Cars are to be described by such data as: make, model, year of production, engine size, and fuel type, number of passengers, registration number, purchase price, purchase date, rent price and insurance details. It is the company policy not to keep any car for a period exceeding one year.

All major repairs and maintenance are done by subcontractors (i.e. franchised garages), with whom CRC has long-term agreements. Therefore the data about garages to be kept in the database includes garage names, addresses, range of services and the like. Some garages require payments immediately after a repair has been made; with others CRC has made arrangements for credit facilities. Company expenditures are to be registered for all outgoings connected with purchases, repairs, maintenance, insurance etc.

Similarly the cash inflow coming from all sources - car hire, car sales, insurance claims - must be kept of file.CRCmaintainsa reasonably stable client base. For this privileged category of customers special credit card facilities are provided. These customers may also book in advance a particular car. These reservations can be made for any period of time up to one month. Casual customers must pay a deposit for an estimated time of rental, unless they wish to pay by credit card. All major credit cards are accepted. Personal details (such as name, address, telephone number, driving license, number) about each customer are kept in thedatabase.

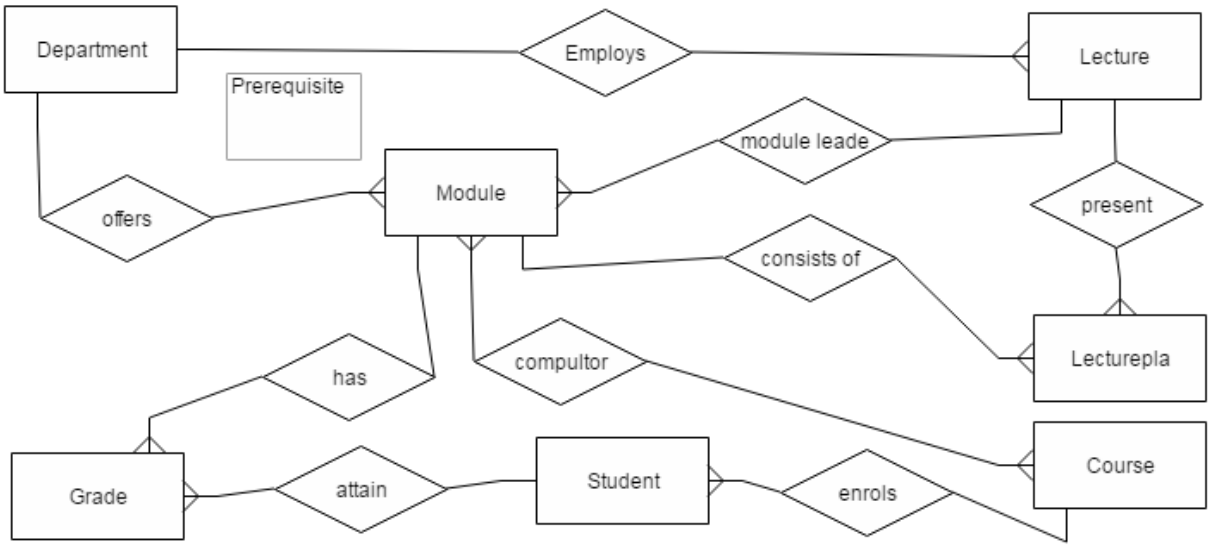


WEEK-12

CASE STUDY: STUDENT PROGRESS MONITORING SYSTEM

AIM: A database is to be designed for a college to monitor students' progress throughout their course of study. The students are reading for a degree (such as BA, BA(Hons) MSc, etc) within the framework of the modular system. The college provides a number of module, each being characterised by its code, title, credit value, module leader, teaching staff and the department they come from. A module is co- ordinated by a module leader who shares teaching duties with one or more lecturers.

A lecturer may teach (and be a module leader for) more than one module. Students are free to choose any module they wish but the following rules must be observed: some modules require pre-requisites modules and some degree programmes have compulsory modules. The database is also to contain some information about students including their numbers, names, addresses, degrees they read for, and their past performance (i.e.modules taken and examination results).



VIVA QUESTIONS WITH ANSWERS

A database is a logically coherent collection of data with some inherent meaning, representing some aspect of real world and which is designed, built and populated with data for a specific purpose.

1. What isDBMS?

It is a collection of programs that enables user to create and maintain a database. In other words it is general-purpose software that provides the users with the processes of defining, constructing and manipulating the database for various applications.

2. What is a Databasesystem?

The database and DBMS software together is called as Database system.

3. Advantages ofDBMS?

- Redundancy iscontrolled.
- Unauthorized access isrestricted.
- Providing multiple userinterfaces.
- Enforcing integrityconstraints.
- Providing backup andrecovery.

4. Disadvantage in File ProcessingSystem?

- Data redundancy &inconsistency.
- Difficult in accessingdata.
- Dataisolation.
- Dataintegrity.
- Concurrent access is notpossible.
- SecurityProblems.

5. Describe the three levels of dataabstraction?

Three levels of abstraction:

Physical level: The lowest level of abstraction describes how data are stored.

Logical level: The next higher level of abstraction, describes what data are stored in database and what relationship among those data.

View level: The highest level of abstraction describes only part of entire database.

6. Define the "integrityrules"

There are two Integrity rules.

Entity Integrity: States that Primary key cannot have NULL value

Referential Integrity: States that Foreign Key can be either a NULL value or should be Primary Key value of other relation.

7. What is extension andintension?

Extension: It is the number of tuples present in a table at any instance. This is time dependent.

Intension: It is a constant value that gives the name, structure of table and the constraints laid on it.

8. What is DataIndependence?

Data independence means that "The application is independent of the storage structure and access strategy of data". In other words, the ability to modify the schema definition in one level should not affect the schema definition in the next higher level.

Two types of Data Independence:

Physical Data Independence: Modification in physical level should not affect the logical level.

Logical Data Independence: Modification in logical level should affect the view level.

9. What is a view? How it is related to dataindependence?

A view may be thought of as a virtual table, that is, a table that does not really exist in its own right but is instead derived from one or more underlying base table. In other words, there is no stored file that direct represents the view instead a definition of view is stored in data dictionary. Growth and restructuring of base tables is not reflected in views. Thus the

View can insulate users from the effects of restructuring and growth in the database. Hence accounts for logical data independence.

10. What is DataModel?

A collection of conceptual tools for describing data, data relationships data semantics and constraints.

11. What is E-Rmodel?

This data model is based on real world that consists of basic objects called entities and of relationship among these objects. Entities are described in a database by a set of attributes.

12. What is Object Orientedmodel?

This model is based on collection of objects. An object contains values stored in instance variables within the object. An object also contains bodies of code that operate on the object. These bodies of code are called methods. Objects that contain same types of values and the same methods are grouped together into classes.

13. What is anEntity?

It is a 'thing' in the real world with an independent existence.

14. What is an Entitytype?

It is a collection (set) of entities that have same attributes.

15. What is an Entityset?

It is a collection of all entities of particular entity type in the database.

16. What is an Extension of entitytype?

The collections of entities of a particular entity type are grouped together into an entity set.

17. What is Weak Entityset?

An entity set may not have sufficient attributes to form a primary key, and its primary key compromises of its partial key and primary key of its parent entity, then it is said to be Weak Entity set.

18. What is anattribute?

It is a particular property, which describes the entity.

19. What is aRelation?

A relation is defined as a set of tuples.

20. What is degree of aRelation?

It is the number of attribute of its relation schema.

21. What isRelationship?

It is an association among two or more entities.

22. What is Relationshipset?

The collection (or set) of similar relationships.

23. What is Relationshiptype?

Relationship type defines a set of associations or a relationship set among a given set of entity types.

24. What is degree of Relationship type?

It is the number of entity type participating.

25. What is DDL (Data DefinitionLanguage)?

A data base schema is specifies by a set of definitions expressed by a special language called DDL.

26. What is VDL (View DefinitionLanguage)?

It specifies user views and their mappings to the conceptual schema.

27. What is DML (Data ManipulationLanguage)?

This language that enable user to access or manipulate data as organized by appropriate data model.

28. What is DMLCompiler?

It translates DML statements in a query language into low-level instruction that the query evaluation engine canunderstand.

29. What is Query evaluationengine?

It executes low-level instruction generated by compiler.

30. What is DDLInterpreter?

It interprets DDL statements and records them in tables containing metadata.

31. What is aquery?

A query with respect to DBMS relates to user commands that are used to interact with a data base. The query language can be classified into data definition language and data manipulation language.

32. What do you mean by Correlated subquery?

A correlated sub query can be easily identified if it contains any references to the parent sub querycolumns in its WHERE clause. Columns from the sub query cannot be referenced anywhere else in the parent query.

33. Are the resulting relations of PRODUCT and JOIN operation the same?

No.

PRODUCT: Concatenation of every row in one relation with every row in another.

JOIN: Concatenation of rows from one relation and related rows from another.

34. What is databaseTrigger?

A database trigger is a PL/SQL block that can defined to automatically execute for insert, update, and delete statements against a table. The trigger can be defined to execute once for the entire statement or once for every row that is inserted, updated, or deleted. For any one table, there are twelve events for which you can define database triggers. A database trigger can call database procedures that are also written inPL/SQL.

35. What are stored-procedures? What are the advantages of usingthem?

Stored procedures are database objects that perform a user defined operation. A stored procedure can have a set of compound SQL statements. A stored procedure executes the SQL commands and returns the result tothe client. Stored procedures are used to reduce networktraffic.

36. Define super key and give example to illustrate the superkey?

Set of one or more attributes taken collectively, allowing to identify uniquely an entity in the entity set.Eg1. {SSN} and {SSN, Cust_name} of customer table are super keys.Eg2. {Branch name}and
{Branch name, Branch city} of Branch table re super keys.

37. Define candidate key and give example to illustrate the candidatekey?

Super keys with no proper subset are called the candidate keys. Otherwise it is called minimal super key. Candidate key is nothing but the primary key used in SQL. Eg1. {SSN} is the candidate keyfor the super keys {SSN} and {SSN, Cust_name} of customer table.Eg2. {Branch name} is the candidate key for the super keys {Branch name} and {Branch name, Branch city} of Branch table.

38. What is Primarykey?

A key chosen to act as the means by which to identify tuples in a relation.

39. What is foreignkey?

A foreign key of relation R is a set of its attributes intended to be used (by each tuple in R) for identifying/referring to a tuples in some relation S. (R is called the referencing relation and S the referenced relation.) For this to make sense, the set of attributes of R forming the foreign key should "correspond to" some superkeyofS. Indeed, by definition we require this superkeytobe the primary key of S.

40. What is a Cursor?

A cursor is a pointer to this context area. PL/SQL controls the context area through a cursor. A cursor holds the rows (one or more) returned by a SQL statement. The set of rows the cursor holds is referred to as the activeset.

41. What is FunctionalDependency?

A Functional dependency is denoted by X-> Y between two sets of attributes X and Y that are subsets of R specifies a constraint on the possible tuples that can form a relation state r of R. The constraint is for any two tuples t1 and t2 in r if t1[X] = t2[X] then they have t1[Y] = t2[Y].

42. What is 1 NF (NormalForm)?

The domain of attribute must include only atomic (simple, indivisible) values.

43. What is Fully Functionaldependency?

It is based on concept of full functional dependency. A functional dependency X-> Y is fully functional dependency if removal of any attribute A from X means that the dependency does not hold any more.

44. What is2NF?

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

45. What is3NF?

A relation schema R is in 3NF if it is in 2NF and for every FD X A either of the following is true X is a Super-key of R.

46. What is BCNF (Boyce-Codd NormalForm)?

A relation schema R is in BCNF if it is in 3NF and satisfies an additional constraint that for every FD X A, X must be a candidate key.

47. What is4NF?

A relation schema R is said to be in 4NF if for every multivalued dependency X Y that holds over R, one of following is true X is subset or equal to (or) XY = R. X is a superkey.

48. What is5NF?

A Relation schema R is said to be 5NF if for every join dependency {R1, R2... Rn} that holds R, one the following is true

i) Ri = R for somei.

ii) The join dependency is implied by the set of FD, over R in which the left side is key ofR.

49.What is dependencypreservation?

Dependency Preservation Property enables us to enforce a constraint on the original relation from corresponding instances in the smaller relations.

50. What is Lossless joinproperty?

Lossless join property enables us to find any instance of the original relation from corresponding instances in the smaller relations