S.Y. Diploma : Sem. III
[CO/CM/CW]

Database Management System [DBMS]

Time: 3 Hrs.] Prelim Question Paper Solution **[Marks : 70**

Q.1 Attempt any FIVE of the following: [10]

Q.1(a) List any four applications of DBMS. [2]

Ans.: Data base system applications:

(1) Banking: For customer information, accounts, and loans, and banking transactions.

(2) Airlines: For reservations and schedule information. Airlines were among the first to use databases in a geographically distributed manner-terminals situated around the world accessed the central database system through phone lines and other data networks.

(3) Universities: For student information, course registrations, and grades.

(4) Credit card transactions: For purchases on credit cards and generation of monthly statements.

(5) Telecommunication: For keeping records of calls made, generating monthly bills, maintaining balances on prepaid calling cards, and storing information about the communication networks.

(6) Finance: For storing information about holdings, sales, and purchases of financial instruments such as stocks and bonds.

(7) Sales: For customer, product, and purchase information.

(8) Manufacturing: For management of supply chain and for tracking production of items in factories, inventories of items in warehouses / stores, and orders for items.

(9) Human resources: For information about employees, salaries, payroll taxes and benefits, and for generation of paychecks.

Q.1(b) State the four database users. [2]

Ans.: Database Users

There are four different types of Database users:

Application Programmers: Are computer professionals who interact with the system through DML calls, which are embedded in a program written in a host language. These programs are commonly referred to as application programs.

Sophisticated Users : They interact with the system through their requests written using a database query language. This request is referred to as a query.

Specialized Users: They are sophisticated users who write specialized database applications that do not fit into the traditional data processing framework. Examples of specialized database applications are:
Knowledge based and expert system, computer aided design system.

Naive Users : They are unsophisticated users who interact with the system by involving one of the permanent application programs that have been written previously.

Q.1(c) Define normalization. Enlist its type. **[2]**

Ans.: Normalization

Normalization is a process of analyzing given relational schema based on its functional dependencies and primary keys/candidate keys to achieve the following desired properties.

Following are the two objectives of normalization:
(i) Minimize data redundancy.
(ii) Minimize insert, update, delete anomalies

Types :
1 NF,
2 NF
3 NF, BCNF, 4NF, 5NF

Q.1(d) Enlist DDL and DML commands. **[2]**

Ans.: DDL commands

CREATE ALTER DROP TRUNCATE RENAME

DML commands

INSERT UPDATE DELETE

Q.1(e) Define the primary and foreign keys. **[2]**

Ans.: Primary key

Within a relation there is always one attribute which has values that are unique in a relation and thus can be used to uniquely identify tuples of that relation. Such a unique tuple identifier is called the primary key. For example, the student relation given below shows the attribute rollno and name of the students. Here the attribute rollno of the student has the property of primary key. Here each student tuple contains a distinct rollno value and this value may be used to uniquely identify the tuple from all the other tuples in the relation.

Rollno	Name
1	Ajay
2	Vijay
3	Raj

The student relation

Foreign Keys

A foreign key is defined as an attribute or a set of attributes of a relation such that each attribute value in this set is that of a primary key of another relation or the same relation.

For example, consider the following tables:

Sales Order Table

S_order_no	S_order_date	Client_date
019001	12 - Oct. – 95	C0001
019002	13 - Oct. – 95	C0002
019003	14 - Oct. – 95	C0003

Sales Order Details

S_order_no	Product_no	Qty_ordered
019001	1	10
019001	2	20
019001	3	30
019002	4	30

The s_order_no column from the SALES ORDER table is the primary key. In the table SALES ORDER DETAILS the s_order_no is a foreign key that references the s_order_no values in the table SALES ORDER.

Q.1 (f)Define the following: [2]

(i) Instance (ii) Schema

Ans.: (i) Instance: The data stored in database at a particular moment of time is called instance of database.

(ii) Schema: Design of a database is called the schema. Schema is of three types: Physical schema, logical schema and view schema.

Q.1(g) State four advantages of DBMS over file processing system. [2]

Ans.: 1. The Information rule: All information in an RDBMS is represented logically in just one way - by values in tables.

2. **The Guaranteed Access rule**: Each item of data in an RDBMS is guaranteed to be logically accessible by resorting to a combination of table name, primary key value, and column name.
3. **The Systematic Treatment of Null Values rule**: Null values (distinct from an empty character string or a string of blank characters and distinct from zero or any other number) are supported in a fully relational DBMS for representing missing.
4. **The Dynamic Online Catalog Based on the Relational Model rule**: The database description is represented at the logical level in the same way as ordinary data, so that authorized users can apply the same relational database.
5. **The Comprehensive Data Sublanguage rule:** A relational system may support several languages and various modes of terminal for data definition, view definition, data manipulation etc.
6. **The View Updating rule**: All views of the data which are theoretically updatable must be updatable in practice by the DBMS.
7. **The High-level Insert, Update, and Delete rule:** The capability of handling a base relation or a derived relation as a single database to perform all DML operations.
8. **The Physical Data Independence rule**: Application programs and terminal activities remain logically unchanged whenever any changes are made in either storage representations or access methods.
9. **The Logical Data Independence rule:** Application programs and terminal activities remain logically unchanged when information preserving changes of any kind are made to the base tables.
10. **The Integrity Independence rule:** Integrity constraints must be definable in the RDBMS sub-language and stored in the system catalogue and not within individual application programs.
11. **The Distribution Independence rule:** An RDBMS has distribution independence. Distribution independence implies that users should not have to be aware of whether a database is distributed.
12. **The No subversion rule:** If the database has any means of handling a single record at a time that low-level language must not be able avoid the integrity rules which are expressed in a higher-level language that handles multiple records at a time.

Q.2Attempt any THREE of the following: [12]

Q.2(a) Explain PL/SQL block structure. [4]

Ans.: PL/SQL Block Structure

Declare

Declaration of memory variables

BEGIN (Mandatory)

SQL executable statements

Exception

Handling errors

END; (Mandatory)

Explanation of PL/SQL Block Strucure:

Declaration section

A block begins with declarative section where variables, cursors are declared. It is an Optional block.

Execution section

Executable SQL or PL/SQL Statements are needed to write here for the execution. It is mandatory block.

Exception section

It is used to handles the exceptions. It is an Optional block.

End Statement

It is used to indicate termination of PL/SQL block. It is mandatory.

Q.2(b) Write and explain syntax for creating procedure. [4]

Ans.: A procedure is created with the CREATE OR REPLACE PROCEDURE statement. The simplified syntax for the CREATE OR REPLACE PROCEDURE statement is as follows :

```
CREATE [OR REPLACE] PROCEDURE procedure_name
[(parameter_name [IN | OUT | IN OUT] type [, ...])]
{IS | AS}
BEGIN
    < procedure_body >
END procedure_name;
```

Where,

procedure-name specifies the name of the procedure.

[OR REPLACE] option allows the modification of an existing procedure.

The optional parameter list contains name, mode and types of the parameters. IN represents the value that will be passed from outside and OUT represents the parameter that will be used to return a value outside of the procedure.

procedure-body contains the executable part.

The AS keyword is used instead of the IS keyword for creating a standalone procedure.

Example: Creating Procedure and calling it using EXEC

```
CREATE OR REPLACE PROCEDURE welcome_msg (p_name IN VARCHAR2)
IS
BEGIN
dbms_output.put_line ('Welcome '|| p_name);
END;
/
EXEC welcome_msg ('world');
```

Q.2(c) Explain grant and revoke command with syntax and example. [4]

Ans.: Granting Permissions Using The Grant Statement

The grant statement provides various types of access to database objects such as tables, views and sequences.

Syntax: **Grant** {object privileges}
ON object name
To < user list >
[With Grant Option]

Explanation of Syntax:

Object Privileges: Each object privilege that is granted-authorizes the grantee to perform some operation on the object. The user can grant all the privileges or grant only specific object privileges.

The Object name refers to a relation name or a view name.

The with grant option allows the grantee to grant object privileges to other users.

Examples:

(1) To grant a privileges on the table product to the user Ajay:

Grant ALL
On Product
To Ajay;

(2) To grant select and update privileges on the table product to Vijay.

Grant select, update
On product
To Vijay;

Revoking Permissions Using The Revoke Statement:

To revoke an authorization we use the revoke statement. It takes the following form:

Syntax: Revoke < object privilege list > **on** < Object name > **from** < user list> **[Restrict / Cascade];**

One cannot use the Revoke command to perform the following operation:

(1) Revoke the object privileges that one didn't grant to the revoke.

(2) Revoke the object privileges granted through the operating system.

The revocation of a privilege from a user may cause other users also to lose that privilege. This behavior is called as cascading of the revoke.

E.g.:

Revoke select **on** product **From** Ajay **cascade;**

The revoke statement may also specify restrict:

E.g.:

Revoke select **on** branch **from** Ajay, Vijay **restrict;**

In this case, an error is returned if there are any cascading revokes and the revoke action is not carried out. The following revoke statement revokes only the grant option, rather than the actual select privilege.

E.g.: Revoke grant option for select on branch **From** Raj;

Q.2(d) Distinguish any four points between network model and hierarchical model. [4]

Ans.: Hierarchical Model

The hierarchical database organizes data in a tree structure. Data is structured downward in a hierarchy of tables. Any level in the hierarchy can have unlimited children, but any child can have only one parent. Hierarchical DBMS have not gained any noticeable acceptance for use within GIS. They are oriented for data sets that are very stable, where primary relationships among the data change infrequently or never at all. Also, the limitation on the number of parents that an element may have is not always conducive to actual geographic phenomenon.

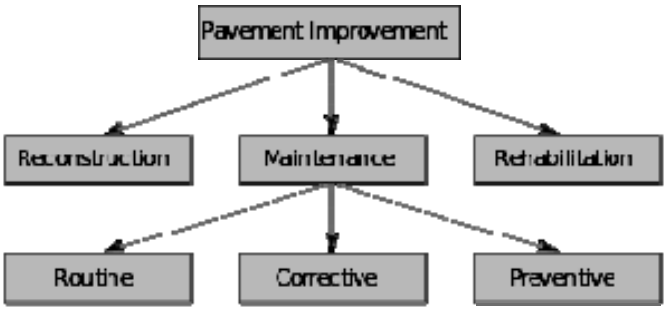


Fig. 1 : Hierarchical Model

Network Model

The network database organizes data in a network or plex structure. Any column in a plex structure can be linked to any other. Like a tree structure, a plex structure can be described in terms of parents and children. This model allows for children to have more than one parent.

Network DBMS have not found much more acceptance in GIS than the hierarchical DBMS. They have the same flexibility limitations as hierarchical databases; however, the more powerful structure for representing data relationships allows a more realistic modelling of geographic phenomenon. However, network databases tend to become overly complex too easily. In this regard it is easy to lose control and understanding of the relationships between elements.

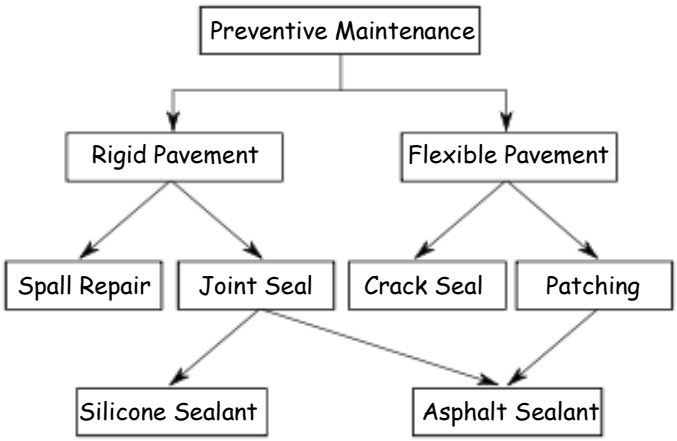


Fig. 2 : Network Model

Q.3 Attempt any Three of the following: **[12]**

Q.3(a) Explain aggregate functions with example **[4]**

Ans.: Aggregate Functions

Aggregate functions are the functions that take a collection of values as input and return a single value.

Avg function : The avg function computes the column's average value.

Min & Max functions : The min and max functions return the minimum and maximum values for the specified columns
e.g.: SQL > select max (salary), min (salary) from employee;

Sum function : The sum function computes the column's total value.
SQL> **Select sum** (salary) **from** employee;

Count function : The count function counts the number of rows. They are of two forms
Count (*): It counts all the rows in a table that satisfy any specified criteria.
Count (column name): It counts all rows in a table that have a non-null value for the column name and satisfy the specified criteria.
e.g. : **Select count (*) from** employee.

Q.3(b) Explain set operators in SQL **[4]**

Ans.: Set operators combine the results of two component queries into a single result. Queries containing set operators are called as compound queries. Set operators in SQL are represented with following special keywords as: Union, Union all, intersection & minus. Consider data from two tables emp and employee as

Emp

Ename
a
b
c
d

Employee

Ename
c
e

(i) Union: The Union of two or more sets contains all elements, which are present in either or both. Union works as or.
E.g. select ename from emp union select ename from employee;
The output considering above data is :

Output:

Ename
a
b
c
d
e

(ii) Union all: The Union of 2 or more sets contains all elements, which are present in both, including duplicates.

E.g. select ename from emp union all select ename from employee;

The output considering above data is:

Output

Ename
a
b
c
c
d
e

(iii) **Intersection:** The intersection of two sets includes elements which are present in both. E.g. select ename from emp intersect select ename from employee;

The output considering above data is:

Output

Ename
c

(iv) **Minus:** The minus of two sets includes elements from set1 minus elements of set2.

E.g. select ename from emp minus select ename from employee:

The output considering above data is:

Ename
a
b
d

Q.3(c) Explain predefined and user defined exception handling with the help of example. [4]

Ans.: Predefined Exceptions

These exceptions have a unique exception name and error number. These exceptions are already defined in the 'STANDARD' package in Oracle. In code, we can directly use these predefined exception name to handle them.

Below are the few predefined exceptions

Exception	Exception Reason
INVALID_NUMBER	Conversion of character to a number failed due to invalid number character
NO_DATA_FOUND	When 'SELECT' statement that contains INTO clause fetches no rows.
ROW_MISMATCH	When cursor variable data type is incompatible with the actual cursor return type
TOO_MANY_ROWS	When a 'SELECT' statement with INTO clause returns more than one row
VALUE_ERROR	Arithmetic or size constraint error (eg: assigning a value to a variable that is larger than the variable size)
ZERO_DIVIDE	Dividing a number by '0'

Example predefined exception

```
DECLARE
    v_num1 NUMBER := 5;
    v_num2 NUMBER := 0;
    myResult NUMBER;
BEGIN
    myResult := v_num1/v_num2;
    DBMS_OUTPUT.PUT_LINE('The quotient is '||TO_CHAR(myResult));
EXCEPTION
    WHEN ZERO_DIVIDE THEN
    DBMS_OUTPUT.PUT_LINE('You cannot divide by zero');
    END;
```

User Defined Exception

In user defined exception, the programmer can create their own exception and handle them. They can be created at a subprogram level in the declaration part. These exceptions are visible only in that subprogram.

Example

- Divide non-negative integer x by y such that the result is greater than or equal to 1. From the given question we can conclude that there exist two exceptions.
  - Division be zero.
  - If result is greater than or equal to 1 means y is less than or equal to x.

```
DECLARE
    x int:=&x; /*taking value at run time*/
    y int:=&y;
    div_r float;
    exp1 EXCEPTION;
    exp2 EXCEPTION;
BEGIN
    IF y=0 then
        raise exp1;
ELSEIF y > x then
        raise exp2;
ELSE
    div_r:= x / y;
    dbms_output.put_line('the result is '||div_r);

END IF;

EXCEPTION
    WHEN exp1 THEN
        dbms_output.put_line('Error');
        dbms_output.put_line('division by zero not allowed');

    WHEN exp2 THEN
        dbms_output.put_line('Error');
        dbms_output.put_line('y is greater than x please check the
        input');

    END;
```

Q.3(d) Explain ACID properties of transaction. **[4]**

Ans.: Transaction

Definition:

A collection of operations that forms a single logical unit of work is called transaction. A transaction is thus a program unit whose execution accesses

and possibly updates the contents of a database. After every transaction, the database should be in a consistent state. A transaction is the result of execution of a user program, written in a high-level data manipulation language or programming language.

To ensure integrity of the data, the database system must maintain the following properties of the transaction :

(i) Atomicity

The atomicity property ensures that at the end of the transaction, either no changes have occurred to the database or the database has been changed in a consistent manner i.e. all the operations of the transaction are reflected properly in the database, or none are.

(ii) Consistency

The consistency property of transaction implies that if the database was in a consistent state before the start of a transaction, then on termination of the transaction, the database will also be in a consistent state. Execution of a transaction in isolation preserves the consistency of the database.

(iii) Isolation

Even though multiple transactions may execute concurrently, each transaction is unaware of the other transactions executing concurrent in the system. This property of transaction indicates that the action performed by a transaction will be hidden from the other transactions until the transaction terminates.

(iv) Durability

The durability property of a transaction ensures that once a transaction completes successfully, the changes it has made to the database remains, even if there are system failures. These properties are often called as the ACID properties.

Q.4 Attempt any THREE of the following: [12]

Q.4(a) Write and explain the syntax for creating and dropping synonyms with an example. [4]

Ans.: Synonym

A synonym is an alternative name for objects such as tables, views, sequences, stored procedures, and other database objects.

Creating or replacing a synonym

The syntax for creating a synonym is:

```
create [or replace] [public] synonym [schema .] synonym_name
for [schema .] object_name [@ dblink];
```

The or replace phrase allows you to recreate the synonym (if it already exists) without having to issue a DROP synonym command.

The public phrase means that the synonym is a public synonym and is accessible to all users. Remember though that the user must first have the appropriate privileges to the object to use the synonym.

The schema phrase is the appropriate schema. If this phrase is omitted, Oracle assumes that you are referring to your own schema.

The object_name phrase is the name of the object for which you are creating the synonym. It can be one of the following:

- table
- view
- sequence
- stored procedure
- function
- package
- materialized view
- java class schema object
- user-defined object
- synonym

Example

```
create public synonym suppliers
for app.suppliers;
```

Q.4(b) Write and explain syntax for creating function. [4]

Ans.: Creating a Function

A standalone function is created using the CREATE FUNCTION statement. The simplified syntax for the CREATE OR REPLACE PROCEDURE statement is as follows:

```
CREATE [OR REPLACE] FUNCTION function_name
[(parameter_name [IN | OUT | IN OUT] type [, ...])]
RETURN return_datatype
{IS | AS}
BEGIN
   < function_body >
END [function_name];
```

Where,

- *function-name* specifies the name of the function.
- [OR REPLACE] option allows modifying an existing function.

- The optional parameter list contains name, mode and types of the parameters. IN represents that value will be passed from outside and OUT represents that this parameter will be used to return a value outside of the procedure.
- The function must contain a **return** statement.
- *RETURN* clause specifies that data type you are going to return from the function.
- *function-body* contains the executable part.
- The AS keyword is used instead of the IS keyword for creating a standalone function.

Q.4(c) Explain cursor with example [4]

Ans.: A cursor is a temporary work area created in system memory when a SQL statement is executed. A cursor is a set of rows together with a pointer that identifies a current row. It is a database object to retrieve data from a result set one row at a time. It is useful when we want to manipulate the record of a table in a singleton method, in other words one row at a time. In other words, a cursor can hold more than one row, but can process only one row at a time. The set of rows the cursor holds is called the active set.

Each cursor contains the followings 4 steps,

1. Declare Cursor: In this part we declare variables and return a set of values.
2. Open: This is the entering part of the cursor.
3. Fetch: Used to retrieve the data row by row from a cursor.
4. Close: This is an exit part of the cursor and used to close a cursor.
5. Eg:

```
Declare
enumemp.eno%type;
enemp.ename%type;
Cursor cur is select eno, ename from emp where jobname = "mgr";
Begin
Open cur;
Loop Fetch cur into enum,en;
Exit when cur%NOTFOUND;
Dbms_output.put_line(„emp num "||enum||" emp name „||en);
End loop;
Close cur;
End; /
```

The example shows fetching multiple records using cursor. A cursor is a temporary work area created in system memory when a SQL statement is

executed. A cursor is a set of rows together with a pointer that identifies a current row.

In the example, the cursor is defined to hold the rows as defined by the select query. Once the cursor is defined, the next step is to open the cursor. When the cursor is opened, it is ready to retrieve the rows. This is done using the fetch statement. Since there are many rows, a loop is used to display the values of all the rows. Once the rows are fetched, the cursor should be closed.

Q.4(d) Explain 2NF with example. **[4]**

Ans.: 2NF

A database is in second normal form if it satisfies the following conditions:

- It is in first normal form.
- All non-key attributes are fully functional dependent on the primary key.

In a table, if attribute B is functionally dependent on A, but is not functionally dependent on a proper subset of A, then B is considered fully functional dependent on A. Hence, in a 2NF table, all non-key attributes cannot be dependent on a subset of the primary key. Note that if the primary key is not a composite key, all non-key attributes are always fully functional dependent on the primary key. A table that is in 1st normal form and contains only a single key as the primary key is automatically in 2nd normal form.

2nd Normal Form Example

Consider the following example:

TABLE_PURCHASE_DETAIL

Customer ID	Store ID	Purchase Location
1	1	Los Angeles
1	3	San Francisco
2	1	Los Angeles
3	2	New York
4	3	San Francisco

This table has a composite primary key [Customer ID, Store ID]. The non-key attribute is [Purchase Location]. In this case, [Purchase Location] only depends on [Store ID], which is only part of the primary key. Therefore, this table does not satisfy second normal form.

To bring this table to second normal form, we break the table into two tables, and now we have the following:

TABLE_PURCHASE

Customer ID	Store ID
1	1
1	3
2	1
3	2
4	3

TABLE_STORE

Store ID	Purchase Location
1	Los Angeles
2	New York
3	San Francisco

Now, in the table [TABLE_STORE], the column [Purchase Location] is fully dependent on the primary key of that table, which is [Store ID].

Q.4(e) Explain the four roles of database administrator. [4]

Ans.: Roles of database administrator

A Database Administrator is a person or a group of person who are responsible for managing all the activities related to database system. This job requires a high level of expertise by a person or group of person.

Following are the roles or functions of Database Administrator :

(i) Schema Definition

The DBA creates the original database schema by executing a set of data definition statements in the DDL.

(ii) Granting of authorization for data access

By granting different types of authorization, the database administrator can regulate which parts of the database various users can access. The authorization information is kept in a special system structure that the database system consults whenever someone attempts to access the data.

(iii) Security

A DBA needs to know potential weaknesses of the database software and the company's overall system and work to minimize risks.In the case of a security breach or irregularity, the DBA can consult audit logs to see who has done what to the data. Audit trails are also important when working with regulated data.

(iv) Performance Monitoring

Monitoring databases for performance issues is part of the on-going system maintenance a DBA performs. If some part of the system is slowing down processing, the DBA may need to make configuration changes to the software or add additional hardware capacity.

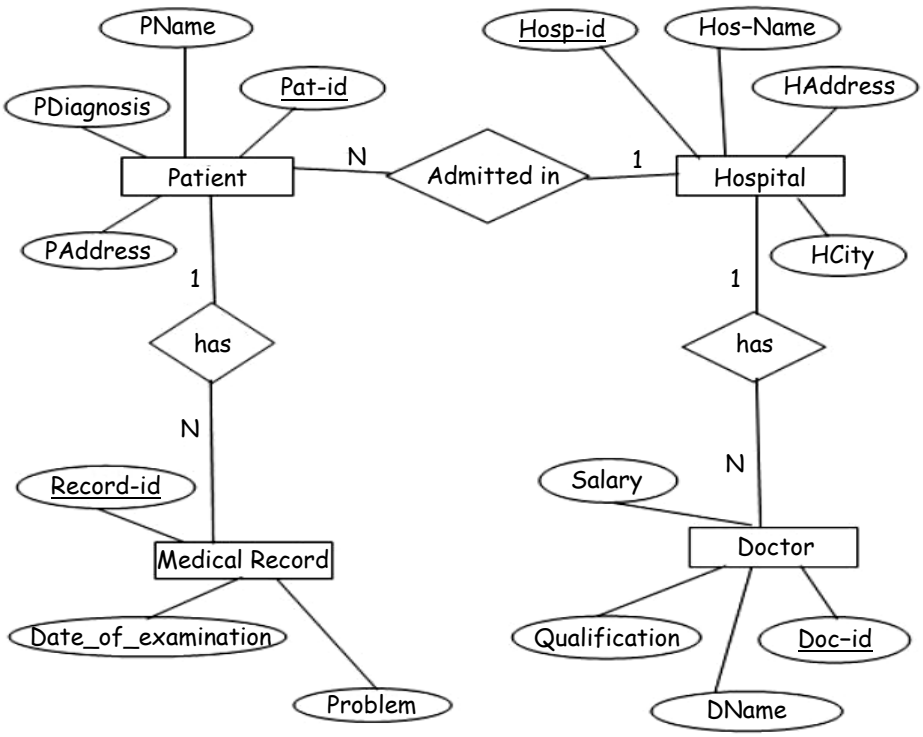
(v) Capacity Issues

All the databases have their limits of storing data in it and the physical memory also has some limitations. DBA has to decide the limit and capacity of database and all the issues related to it.

Q.5 Attempt any TWO of the following: [12]

Q.5(a) Draw ER diagram for Hospital Management System [6]
(Use DOCTOR,PATIENT,HOSPITAL and MEDICAL_RECORD Entity). Identify Primary Key and Foreign Key.

Ans.:



Q.5(b) Normalize database [6]
Employee(emp_id,emp_name,phone,skill,salary,deptno,dept_name,jobno ,job_title) upto 3NF

Ans.: To check for 1 NF, it has multiple values attribute phones,skills and jobs. So it must be in separate table after 1 NF decomposition, the tables will be Employee_phone (emp_id, phone) with key as (emp_id,phone) and no FD, so it is in BCNF.

Employee_skill(emp_id,skill) with key as (emp_id, skill) and no FD, so it is in BCNF

Employee_job(emp_id, jobno, job_title) with key as (emp_id, jobno) and FD jobno->job_title which is a partial dependency, decomposing further, we get,
Employee_job(emp_id, jobno) with key (emp_id, jobno) and Job **(jobno, job_title)** with key jobno
Employee(emp_id, emp_name, salary, deptno, dept_name) with FDs
emp_id -> emp_name, salary, deptno, dept_name (as emp_id is key for this table)
dept_no->dept_name (as each department no will have unique name)

This relation is not in 3NF because of transitive dependency, so decomposing further:
Employee(emp_id, emp_name, salary, deptno) and Department(deptno, dept_name)

Final tables have been highlighted.

Q.5(c) Write SQL query for following consider table [6]
EMP(empno, deptno, ename ,salary, Designation, joiningdate, DOB,city)
(i) Display names of employees whose experience is more than 10 years
(ii) Display age of employees
(iii) Display average salary of all employee
(iv) Display name of employee who earned highest salary

Ans.: (i) Display names of employees whose experience is more than 10 years

```
Select ename from EMP where to_char(sysdate,'YYYY')-
to_char(joiningdate,'YYYY')>10;
```

OR

```
Select ename from EMP where DATEDIFF(YEAR, joiningdate,
GETDATE())>10
```

(ii) Display age of employees

```
Select emp_no,ename, to_char(sysdate,'YYYY')-to_char(dob,'YYYY')as
Age from EMP
```

OR

```
Select emp_no,emp_name, DATEDIFF(YEAR, DOB, GETDATE()) as Age
from EMP
```

(iii) Display average salary of all employee

```
Select avg(salary) as Average_salary from EMP
```

(iv) Display name of employee who earned highest salary

Select emp_name,salary from EMP where salary =(select max(salary) from EMP).

Q.6 Attempt any TWO of the following: [12]

Q.6(a) Create table [6]

EMP(empno , deptno, ename ,salary, Designation, joiningdate, DOB,city).

(i) Insert one row into the table

(ii) Save the data

(iii) Insert second row into the table

(iv) Undo the insertion of second row

(v) Insert two rows into the table

(vi) Create Savepoint s1

(vii)Insert one row into the table

(viii) Undo upto savepoint s1

Ans.:

```
CREATE TABLE EMP
(EMPNO INT, DEPTNO,ENAME VARCHAR(20),SALARY FLOAT,DESIGNATION
VARCHAR(100),JOININGDATE DATE,DOB DATE,CITY VARCHAR(20));
```

(i) Insert one row into the table

```
INSERT INTO EMP VALUES(1,10,'ANIL',10000,'HR','01-02-
2006','12-03-1990','MUMBAI');
```

(ii) Save the data

```
COMMIT;
```

(iii) Insert second row into the table

```
INSERT INTO EMP VALUES (2,10,'SUJA',12000,'SALES
REPRESENTATIVE','01-02-2009','12-03-1987','MUMBAI');
```

(iv) Undo the insertion of second row

```
ROLLBACK;
```

(v) Insert two rows into the table

```
INSERT INTO EMP VALUES (2,10,'SUJA',12000,'SALES
REPRESENTATIVE'','11-08-2009','21-03-1987','MUMBAI');

INSERT INTO EMP VALUES(3,20,'LEENA',15000,'PR MANAGER',
'01-02-2010','12-05-1974','MUMBAI');
```

(vi) Create Savepoint s1

```
SAVEPOINT S1;
```

(vii) Insert one row into the table

```
INSERT INTO EMP VALUES(4,20,'AJAY',18000,'CLERK','25-06-2010','18-08-1974','MUMBAI');
```

(viii) Undo upto savepoint s1

```
ROLLBACK TO S1
```

Q.6(b) Write a PL/SQL program to check whether specified employee is present in EMP table or not. Accept empno from user. If employee does not exist display message using exception handling. [6]

Ans.:

```
Accept x number prompt 'Please enter emp no:
DECLARE
Emp_no number;
BEGIN

Emp_no:=&x;
Select * from EMP where emp_no=Emp_no;

EXCEPTION
    WHEN no_data_found THEN
        dbms_output.put_line('No such employee!');

END;
/
```

Q.6(c) Write SQL query for following consider table EMP(empno, deptno, ename, salary, Designation, joiningdate, DOB, city) [6]

(i) Display employees name and number in an increasing order of salary
(ii) Display employee name and employee number dept wise
(iii) Display total salary of all employee
(iv) Display number of employees dept wise
(v) Display employee name having experience more than 3 years
(vi) Display employee name staring with "S" and working in deptno 1002

Ans.: (i) Display employees name and number in an increasing order of salary

```
Select ename from EMP order by salary;
```

(ii) Display employee name and employee number dept wise

Select empno,ename, from EMP order by deptno

(iii) Display total salary of all employee

Select sum(salary) as TotalSalary from EMP

(iv) Display number of employees dept wise

SELECT dept_no, COUNT(*) as Number_of_Employees FROM EMP GROUP BY dept_no;

(v) Display employee name having experience more than 3 years

Select ename from EMP where DATEDIFF(YEAR, joiningdate, GETDATE())>3

(vi) Display employee name staring with "S" and working in deptno 1002

Select ename from EMP where ename like 'S%' and dept_no=1002

❑ ❑ ❑ ❑ ❑