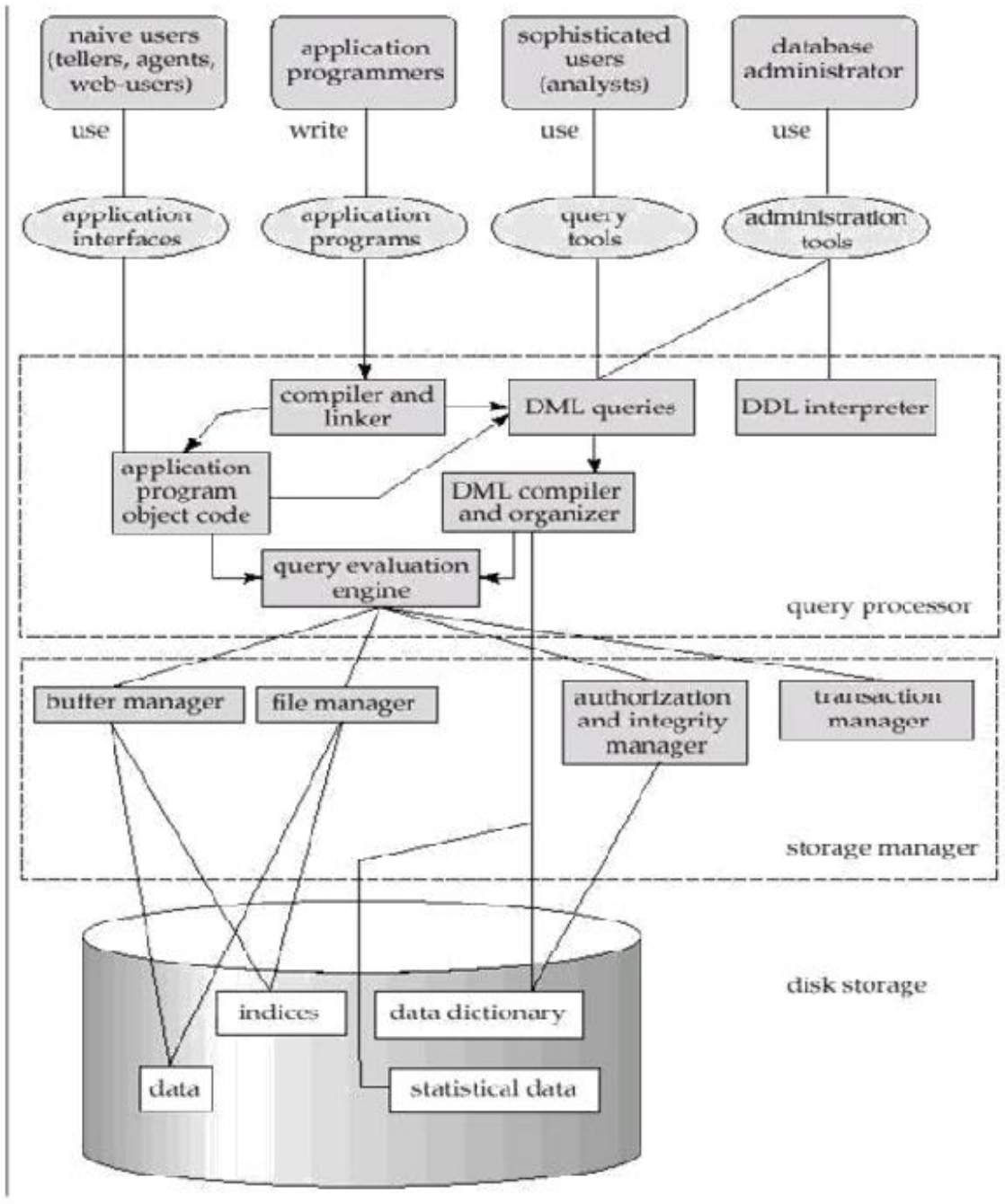
Sub:	Database Management System					Sub Code:	17CS53	
Date:	07/09/19	Duration:	90 mins	Max Marks:	50	Sem /Sec:	V/A,B,C	OBE

Answer any FOUR FULL Questions from Q1 to Q5,Q6 is COMPULSORY.

		CO	RBT
1	**With reference to database system environment, describe the component of DBMS and their interaction, with the help of a diagram.**	CO1	L1

Answer: Diagram



A database system is partitioned into modules that deal with each of the responsibilities of the overall system. The functional components of a database system can be broadly divided into the storage manager and the query processor components.

The storage manager is important because databases typically require a large amount of storage space. Some Big organizations Database ranges from Giga bytes to Tera bytes. So the main memory of computers cannot store this much information, the information is stored on disks. Data are moved between disk storage and main memory as needed.

The query processor also very important because it helps the database system simplify and facilitate access to data. So quick processing of updates and queries is important. It is the job of the database system to translate updates and queries written in a nonprocedural language.

StorageManager:

A storage manager is a program module that provides the interface between the low level data stored in the database and the application programs and queries submitted to the system. The storage manager is responsible for the interaction with the file manager. The storage manager translates the various DML statements into low-level file-system commands. Thus, the storage manager is responsible for storing, retrieving, and updating data in the database.

Storage Manager Components:

Authorization and integrity manager which tests for the satisfaction of integrity constraints and checks the authority of users to access data.

Transaction manager which ensures that the database itself remains in a consistent state despite system failures, and that concurrent transaction executions proceed without conflicting.

File manager: which manages the allocation of space on disk storage and the data structures used to represent information stored on disk.

Buffer manager which is responsible for fetching data from disk storage into main memory. Storage manager implements several data structures as part of the physical system implementation. Data files are used to store the database itself. Data dictionary is used to stores metadata about the structure of the database, in particular the schema of the database.

Query Processor Components:

DDL interpreter: It interprets DDL statements and records the definitions in the data dictionary.

DML compiler: It translates DML statements in a query language into an evaluation plan consisting of low-level instructions that the query evaluation engine understands.

Query evaluation engine: It executes low-level instructions generated by the DML compiler

2(a)	**What is the difference between logical independence and physical data independence?** There are two kinds: 1. Physical Data Independence 2. Logical Data Independence **Physical Data Independence:** The ability to modify the physical schema without causing application programs to be rewritten. **Logical Data Independence:** The ability to modify the conceptual schema without causing application programs to be rewritten Usually done when logical structure of database is altered Logical data independence is harder to achieve as the application programs are usually heavily dependent on the logical structure of the data. Modifications at this level are usually to improve performance.	CO1	L1
2(b)	**What are different database schema languages and interfaces?** **Database Language** o A DBMS has appropriate languages and interfaces to express database queries and updates. o Database languages can be used to read, store and update the data in the database. Types of Database Language: DDL DML DCL TCL 1.Data Definition Language o **DDL** stands for **D**ata **D**efinition **L**anguage. It is used to define database structure or pattern. o It is used to create schema, tables, indexes, constraints, etc. in the database. o Using the DDL statements, you can create the skeleton of the database. o Data definition language is used to store the information of metadata like the number of tables and schemas, their names, indexes, columns in each table, constraints, etc. 2. Data Manipulation Language **DML** stands for **D**ata **M**anipulation **L**anguage. It is used for accessing and manipulating data in a database. It handles user requests. Here are some tasks that come under DML: o **Select:** It is used to retrieve data from a database. o **Insert:** It is used to insert data into a table. o **Update:** It is used to update existing data within a table. o **Delete:** It is used to delete all records from a table. o **Merge:** It performs UPSERT operation, i.e., insert or update operations.	CO1	L1

- **Call:** It is used to call a structured query language or a Java subprogram.
- **Explain Plan:** It has the parameter of explaining data.
- **Lock Table:** It controls concurrency.

3. Data Control Language

- **DCL** stands for **D**ata **C**ontrol **L**anguage. It is used to retrieve the stored or saved data.
- The DCL execution is transactional. It also has rollback parameters.

  (But in Oracle database, the execution of data control language does not have the feature of rolling back.)

Here are some tasks that come under DCL:

- **Grant:** It is used to give user access privileges to a database.
- **Revoke:** It is used to take back permissions from the user.

There are the following operations which have the authorization of Revoke:

CONNECT, INSERT, USAGE, EXECUTE, DELETE, UPDATE and SELECT.

4. Transaction Control Language

TCL is used to run the changes made by the DML statement. TCL can be grouped into a logical transaction.

Here are some tasks that come under TCL:

- **Commit:** It is used to save the transaction on the database.

- **Rollback:** It is used to restore the database to original since the last Commit.

Interfaces in DBMS

A database management system (DBMS) interface is a user interface which allows for the ability to input queries to a database without using the query language itself.

User-friendly interfaces provide by DBMS may include the following:

1. **Menu-Based Interfaces for Web Clients or Browsing –** These interfaces present the user with lists of options (called menus) that lead the user through the formation of a request. Basic advantage of using menus is that they removes the tension of remembering specific commands and syntax of any query language, rather than query is basically composed step by step by collecting or picking options from a menu that is basically shown by the system. Pull-down menus are a very popular technique in *Web based interfaces*. They are also often used in *browsing interface* which allow a user to look through the contents of a database in an exploratory and unstructured manner.
2. **Forms-Based Interfaces –** A forms-based interface displays a form to each user. Users can fill out all of the form entries to insert a new data, or they can fill out only certain entries, in which case the DBMS will redeem same type of data for other remaining entries. This type of forms are usually designed or created and programmed for the users that have no expertise in operating system. Many DBMSs have *forms specification languages* which are special languages that help specify such forms.
   Example: SQL* Forms is a form-based language that specifies queries using a form designed in conjunction with the relational database schema.b>

3. **Graphical User Interface –**
A GUI typically displays a schema to the user in diagrammatic form.The user then can specify a query by manipulating the diagram. In many cases, GUI's utilize both menus and forms. Most GUIs use a pointing device such as mouse, to pick certain part of the displayed schema diagram.

4. **Natural language Interfaces –**
These interfaces accept request written in English or some other language and attempt to understand them. A Natural language interface has its own schema, which is similar to the database conceptual schema as well as a dictionary of important words.
The natural language interface refers to the words in its schema as well as to the set of standard words in a dictionary to interpret the request.If the interpretation is successful, the interface generates a high-level query corresponding to the natural language and submits it to the DBMS for processing, otherwise a dialogue is started with the user to clarify any provided condition or request. The main disadvantage with this is that the capabilities of this type of interfaces are not that much advance.

5. **Speech Input and Output –**
There is an limited use of speech say it for a query or an answer to a question or being a result of a request it is becoming commonplace Applications with limited vocabularies such as inquiries for telephone directory, flight arrival/departure, and bank account information are allowed speech for input and output to enable ordinary folks to access this information.
The Speech input is detected using a predefined words and used to set up the parameters that are supplied to the queries. For output, a similar conversion from text or numbers into speech take place.

6. **Interfaces for DBA –**
Most database system contains privileged commands that can be used only by the DBA's staff. These include commands for creating accounts, setting system parameters, granting account authorization, changing a schema, reorganizing the storage structures of a databases.

3(a)	**Explain how the different update operations deal with constraint violations.**	CO1	L2

There are three basic operations that can change the states of relations in the database: Insert, Delete, and Update (or Modify). There are three basic operations that can change the states of relations in the database:Insert, Delete, and Update (or Modify). They insert new data, delete old data,or modify existing data records, respectively. **Insert** is used to insert one or more new tuples in a relation, **Delete** is used to delete tuples, and **Update** (or **Modify**) is used to change the values of some attributes in existing tuples.

1.Insert Operation

Insert can violate any of the four types of constraints. Domain constraints can be violated if an attribute value is given that does not appear in the corresponding domain or is not of the appropriate data type. Key constraints can be violated if a key value in the new tuple t already exists in another tuple in the relation r(R). Entity integrity can be violated if any part of the primary key of the new tuple t is NULL. Referential integrity can be violated if the value of any foreign key in t refers to a tuple that does not exist in the referenced relation.

Option to deal with violation.

If an insertion violates one or more constraints, the default option is to reject the insertion. Another option is to attempt to correct the reason for rejecting the insertion, but this is typically not used for violations caused by Insert.

2.The Delete Operation

The **Delete** operation can violate only referential integrity. This occurs if the tuple being deleted is referenced by foreign keys from other tuples in the database. To specify deletion, a condition on the attributes of the relation selects the tuple (or tuples) to be deleted.

Option to deal with violation.

Several options are available if a deletion operation causes a violation. The first option, called **restrict,** is to reject the deletion. The second option, called **cascade,** is to attempt to cascade (or propagate) the deletion by deleting tuples that reference the tuple that is being deleted. A third option, called **set null** or **set default,** is to modify the referencing attribute values that cause the violation; each such value is either set to NULL or changed to reference another default valid tuple. Notice that if a referencing attribute that causes a violation is part of the primary key, it cannot be set to NULL; otherwise, it would violate entity integrity. Combinations of these three options also possible.

3.The Update Operation

Updating an attribute that is neither part of a primary key nor part of a foreign key usually causes no problems; the DBMS need only check to confirm that the new value is of the correct data type and domain. Modifying a primary key value is similar to deleting one tuple and inserting another in its place because we use the primary key to identify tuples.So all violations that can happen in insert and delete operations apply in case of update also. If a foreign key attribute is modified, the DBMS must make sure that the new value refers to an existing tuple in the referenced relation (or is set to NULL).

Options to deal with violation

Same as the options available if a insertion operation and deletion operation causes violation.

3(b)	**Explain the basic constraints that can be specified in SQL as part of table creation with example.**	CO1	L2

Constraints are used to make sure that the integrity of data is maintained in the database. Following are the most used constraints that can be applied to a table.

- NOT NULL
- PRIMARY KEY
- FOREIGN KEY
- CHECK
- DEFAULT

The following constraints are commonly used in SQL:

- NOT NULL - Ensures that a column cannot have a NULL value
- UNIQUE - Ensures that all values in a column are different
- PRIMARY KEY - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
- FOREIGN KEY - Uniquely identifies a row/record in another table
- CHECK - Ensures that all values in a column satisfies a specific condition
- DEFAULT - Sets a default value for a column when no value is specified
- INDEX - Used to create and retrieve data from the database very quickly
- UNIQUE -Used to create and retrieve data from the database very quickly

NOT NULL Constraint

NOT NULL constraint restricts a column from having a NULL value. Once NOT NULL constraint is applied to a column, you cannot pass a null value to that column. It enforces a column to contain a proper value.

One important point to note about this constraint is that it cannot be defined at table level.

Example using NOT NULL constraint

```
CREATE TABLE Student(s_id int NOT NULL, Name varchar(60), Age int);
```

The above query will declare that the s_id field of Student table will not take NULL value.

UNIQUE Constraint

UNIQUE constraint ensures that a field or column will only have unique values. A UNIQUE constraint field will not have duplicate data. This constraint can be applied at column level or table level.

Using UNIQUE constraint when creating a Table (Table Level)

Here we have a simple CREATE query to create a table, which will have a column s_idwith unique values.

```
CREATE TABLE Student(s_id int NOT NULL UNIQUE, Name varchar(60), Age int);
```

The above query will declare that the s_id field of Student table will only have unique values and won't take NULL value.

Using UNIQUE constraint after Table is created (Column Level)

```
ALTER TABLE Student ADD UNIQUE(s_id);
```

The above query specifies that s_id field of Student table will only have unique value.

Primary Key Constraint

Primary key constraint uniquely identifies each record in a database. A Primary Key must contain unique value and it must not contain null value. Usually Primary Key is used to index the data inside the table.

Using PRIMARY KEY constraint at Table Level

```
CREATE table Student (s_id int PRIMARY KEY, Name varchar(60) NOT NULL, Age int);
```

The above command will creates a PRIMARY KEY on the s_id.

Using PRIMARY KEY constraint at Column Level

```
ALTER table Student ADD PRIMARY KEY (s_id);
```

The above command will creates a PRIMARY KEY on the s_id.

Foreign Key Constraint

FOREIGN KEY is used to relate two tables. FOREIGN KEY constraint is also used to restrict actions that would destroy links between tables. To understand FOREIGN KEY, let's see its use, with help of the below tables:

Customer_Detail Table

c_id	Customer_Name	Address
101	Adam	Noida
102	Alex	Delhi
103	Stuart	Rohtak

Order_Detail Table

Order_id	Order_Name	c_id
10	Order1	101
11	Order2	103
12	Order3	102

In Customer_Detail table, c_id is the primary key which is set as foreign key in Order_Detail table. The value that is entered in c_id which is set as foreign key in Order_Detail table must be present in Customer_Detail table where it is set as primary key. This prevents invalid data to be inserted into c_id column of Order_Detail table.

If you try to insert any incorrect data, DBMS will return error and will not allow you to insert the data.

Using FOREIGN KEY constraint at Table Level

```
CREATE table Order_Detail(
    order_id int PRIMARY KEY,
    order_name varchar(60) NOT NULL,
    c_id int FOREIGN KEY REFERENCES Customer_Detail(c_id)
);
```

In this query, c_id in table Order_Detail is made as foriegn key, which is a reference of c_id column in Customer_Detail table.

Using FOREIGN KEY constraint at Column Level

```
ALTER table Order_Detail ADD FOREIGN KEY (c_id) REFERENCES Customer_Detail(c_id);
```

CHECK Constraint

CHECK constraint is used to restrict the value of a column between a range. It performs check on the values, before storing them into the database. Its like condition checking before saving data into a column

Using CHECK constraint at Table Level

```
CREATE table Student(
    s_id int NOT NULL CHECK(s_id > 0),
    Name varchar(60) NOT NULL,
    Age int
);
```

The above query will restrict the s_id value to be greater than zero.

Using CHECK constraint at Column Level

```
ALTER table Student ADD CHECK(s_id > 0);
```

4(a)	**Consider the relations.** **EMPLOYEE(<u>emp_id</u>,name)** **ASSIGNED_TO(<u>projectno</u>,emp_id)** **PROJECT(<u>projectno</u>,project_name)** **Express the following queries in Relational Algebra.** **i)Get details of employee working on both P354 and P345 project numbers.** **Answer:** Π emp_id,name(∂project_no.=P354 (EMPLOYEE*ASSIGNED_TO)) ∩ Π emp_id,name (∂project_no.=P345 (EMPLOYEE*ASSIGNED_TO)) **ii)Find the employee number of employee who do not work on project P678** **Answer:** Π emp_id(∂project_no.<>P678 (EMPLOYEE*ASSIGNED_TO))	CO2	L3

4b)	**Explain create, insert, delete and update, drop, alter statements in SQL with example.** **Answer:** **i)Create** Format: create table table_name(att1 datatype,...primary key(att1),foreign key(att3) references TABLE2(att2) on delete cascade on update cascade) Example CREATE TABLE EMPLOYEE(Name varchar(15),Ssn int,Dno INT NOT NULL DEFAULT 1, Super_ssn, CONSTRAINT EMPPK PRIMARY KEY (Ssn), CONSTRAINT EMPSUPERFK FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn) ON DELETE SET NULL ON UPDATE CASCADE, CONSTRAINT EMPDEPTFK FOREIGN KEY(Dno) REFERENCES DEPARTMENT(Dnumber) ON DELETE SET DEFAULT ON UPDATE CASCADE); **ii)Insert** Format: To insert records into a table/relation:- Insert into table_name values (value list); Example Insert into EMPLOYEE values('abe',1002,21,1001); **iii)Delete** Format To delete a record a set of records:- Delete from table_name where condition; Example Delete from WORKS_ON where Essn = '999887777' and Pno = 10; To delete full table content:- Delete from table; Delete from WORKS_ON; **iv)Update** Format To update an attribute value:- Update table_name set att1='new value' where condition; Example Update EMPLOYEE set salary = new value where Ssn = '999887777'; **v)Drop**	CO3	L2

Format
To drop a table:-
Drop table table_name;

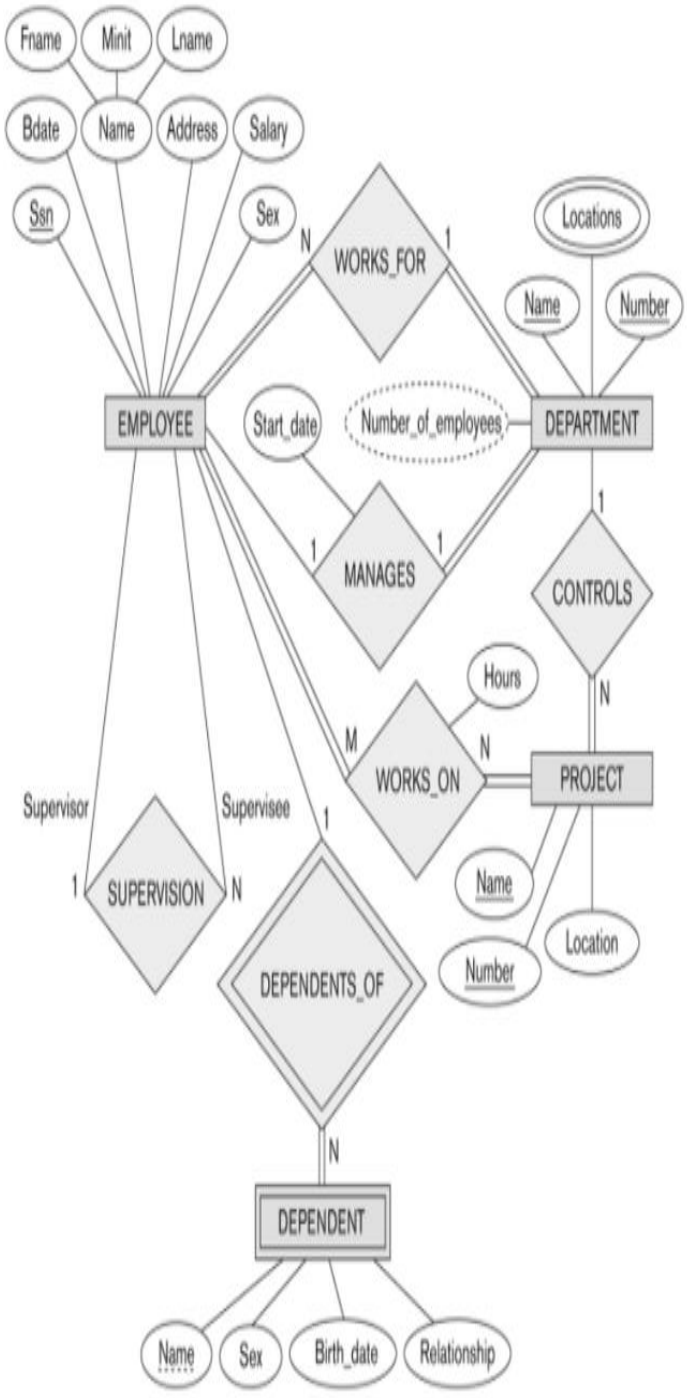
Example
DROP TABLE DEPENDENT;

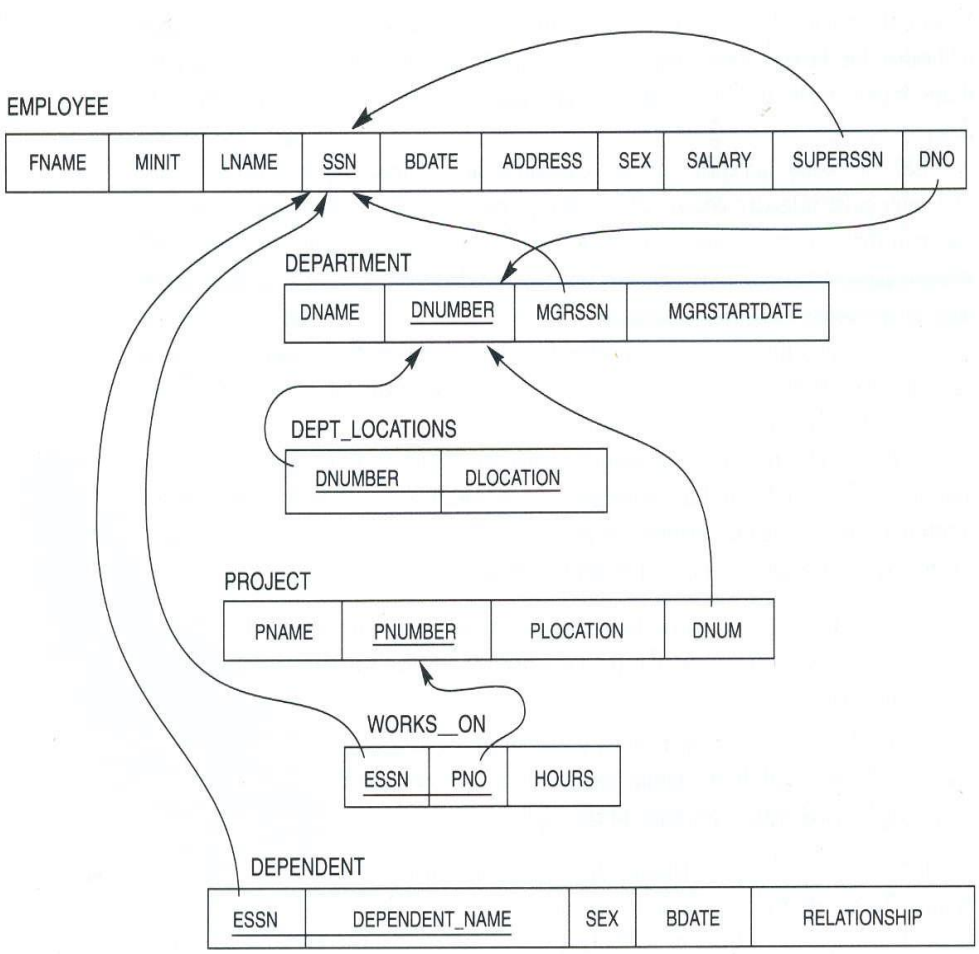
vi)Alter

Format
To add a new column:-
Alter table table_name add new_column_name datatype;

Example

To drop a column:-
ALTER TABLE EMPLOYEE DROP COLUMN Address ;
To add a new column:-
ALTER TABLE EMPLOYEE ADD column Address;
To add primary key to a table:-
Alter table EMPLOYEE add primary key(pnumber);

5(a)	**EMPLOYEE(Name,Ssn,Bdate,Address,Sex,Salary,Supervisor_ssn,Dno)** **DEPARTMENT(Dname,Dnumber,Mgr_ssn,Mgr_start_date)** **DEPT_LOCATION(Dnumber,Dlocation)** **PROJECT(Pname,Pnumber,Plocation,Dnum)** **WORKS_ON(Essn,Pno,Hours)** **DEPENDENT(Essn,Dependent_name,sex,Bdate,Relationship)** **Note: Attribute 'Dependent_name' is a partial key.** **Draw an ER diagram for the above "COMPANY" database** Answer:  	CO1	L3

5(b)	Draw schema diagram for above database. Answer:  	CO2	L3

5(c)	**Write an SQL query to find the details of employees who do not work on project P3** **Answer:** i)select * from employee where Ssn NOT IN (select Essn from WORKS_ON where Pno='P3'); **Write an SQL query to list the employee names whose name contains the letter 'b'** **Answer** ii)select Name from EMPLOYEE where Name LIKE '%b%';	CO3	L3

6	For the Company database given in question number 5 ,Write SQL query for the following.	CO3	L3
(a)	List female employees from DNo =20 earning more than 50000. Answer: Select Name from EMPLOYEE where sex='F' and salary>50000 and Dno=20;		
(b)	Retrieve the names and ssn of employees who have no dependents Answer: select Name,Ssn from EMPLOYEE where Ssn NOT IN (select distinct Essn from DEPENDENT);		
(c)	Each project on which more than 2 employees work, retrieve the project number ,project name and number of employees who work on the project. Answer select Pnumber,Pname,COUNT(*) from PROJECT,WORKS_ON where Pnumber=Pno group by Pnumber,Pname Having COUNT(*)>2;		
(d)	For each dept,retrieve the dept number,number of employees in dept,and their average salary. Answer: select Dno,COUNT(*),AVG(Salary) from EMPLOYEE Group by Dno;		
(e)	For each dept that has more than five employees,retrieve the dept number and the number of employees who are making more than $40000. select Dno,COUNT(*) from EMPLOYEE where Salary>40000 AND Dno IN (select Dno from EMPLOYEE Group by Dno Having COUNT(*)>5) Group by Dno;		