## What is Data?

// Data is an implicit meaning.

// The data known as facts abouts any entity fore object.

## What is information?

// When we combine all the data it create a meaningful data ~~and the rr~~ is known as information.

## What is entity?

// Anything exist in the real world is known as one entity which is used for a specific purpose.

// Entity are two types :- (i) strong entity
(ii) Weak entity

// Those entities are uses the unique value is known as strong entity and those entities are not used the unique value is known as weak entity.

## What is database?

// Database is a collection of datas, or which has implicit meanings.

## Other classification of Database :-

There are 7 types of database are present.

(i) Commercial database
(ii) Multimedia database
(iii) Document Database
(iv) GIS database
(v) Active database
(vi) Temperal database
(vii) Deductive Database

### (i) Commercial Database -

In commercial database, the information are stored in terms of character, integer, number, floating point.

Ex:- College database

### (ii) Multimedia Database :-

In multimedia database, the information are stored in the form of audio files, video files, images.

Ex:- MP3 . PNG . JPEG

## Document Database :-

In document DB, if it's not give exact data then it gives related datas.

Ex:- Wiki-pedia

## GIS Database :-

// GIS stands for Geographic information system.

In GIS DB, We can get the detail picture of earth and the picture will be capture by satelite.

## Active Database :-

In active DB, it's use in real time applicatio

Ex:- Weather podcasting

## Temperal Database :-

If the time aspects will be attached to the database is known as temperal database.

// In temperal DB it use the certain time limit.

Ex:- Railway reservation.

## Deductive Database :-

In deductive database, the information are

present by providing certain rules and regulation, we've to access the data items from the database.

## Database Management System:-

Database management system is a software i.e used create and maintain the data.

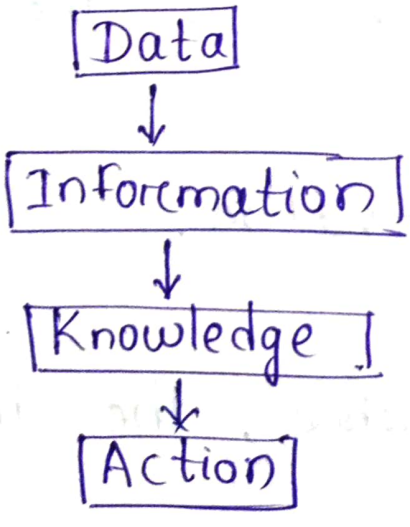
or

Database management system is a general purpose software system that provides the process of defining, constructing, manipulation shearing the database among various user applications.

## Purpose of Database System:-

The purpose of database system is;

(i) Data into information
(ii) Information into knowledge
(iii) Knowledge to action.



- Database are used for storing, maintaining, accessing any string of data.
- It can collect the information from the users, the information is gathered in one place, so it can be observed and analysised, The database system provides a safe and affective platform to manage vast amount of data.
- The database system is used in different ways
  (i) effecient storage and retrival process
  (ii) It provides accuracy data.
  (iii) It can protecting the confidential data.
  (iv) Security very strong.
  (v) Data sharing process is available.
  (vi) Provides data backup
  (vii) Eleminated redundency data. ~~(dup~~

* Data integrity:

Data integrity refers to the accuracy and consistency of data in a database.

→ Database integrity can be used to describe a state, process or function

→ Data integrity is mainly used for ~~error~~ error checking method and validation procedures.

→ data integrity is classified into 4 catagories.

(i) entity integrity.
(ii) domain integrity.
(iii) referanced integrity
(iv) distributed / user define

(1) entity integrity:

In entity integrity defines a row is a unique entity for a specific table.

→ an entity integrity can uses the primary key concept

(2) Domain integrity:

It is validating for entire specific column

→ It uses the Foreign key concept.

(3) Referanced ~~/ user define~~: integrity:

~~It is mainly used for business application and find out the needs of the customers~~. it defines the relation between table where the row are inserted or deleted

(4) Distributed / user define:

It is mainly used for business application and find out the needs. ~~out the needs~~ of the customers.

2 marks

## Database Administrator :-

• // Database Administrator is a person or users, he is the overall controller the data of the system.

• // It has some responsibilities :-

(i) Deciding the information contain i.e identifying the entities.

(ii) ~~Disc~~ Deciding the storage structure how the data is to be represented by writing a storage structure definition.

(iii) Defining the authorization check and validation procedures.

(iv) Define for ~~bree~~ backup and recovery process.

(v) Providing the priviliege to excess the database.

(vi) Responding to the change and acceording to the user.

## ✓ Data Redundency :-

• // Storing the information several times that leads ~~needs~~ to wastage of storage place is called as data redundency.

• // Data redundency is a term used about database that some data fields Appeals more than one in that data field.

// Data redundency is wastepool and inefficient for several reasons.

// To avoid the data redundenay concept we are using normalization process.

## // Database Instance and Schema :-

// The actual data stored in a database at a particulare moment in time is called database instance.

// It changes frequently.

// The descaription of the database is known as database schema.

| Dept | Id |
|---|---|
| CSE | 1 |
| ME | 2 |
| ECE | 3 |

// It doesn't changes frequently.

// The database schema is specified during the DB Design.

| Dept. Name | Id |
|---|---|
| | |

## Components of DBMS :-

It is a type of software by which we can save and retrieve the users data with the security process.

// In database management system we can manipulate the database with the help of group of programe.

// In DBMS, it gives the permission to the users to use the data according to their need.

// There are basically 5 types of components present in DBMS :-

(i) Hardware
(ii) Software
(iii) Data
(iv) Procedure
(v) Users

(i) Hardware :-

// It is the physical components of DBMS.

// It includes o/p objects like a printer, monitor and the storage device like hard disk.

// With the help of hardware, it can access and update the Database.

(ii) Software :-

// It is the main component of DBMS.

- It is defined as collection of programs that are used to instruct the computer about its work.
- We can says that computer software is a set of instruction i.e used to instruct the computer hardware for the operations of the computer.
- Some examples of DBMS software are MySQL, Oracle, Diff. version of Oracle, DB2, File maker, Clipper, Fox Pro and Microsoft XS.

(iii) Data :-

- The term data means rough facts of any entities which can find out the meaning of information
- By using the data we can create and construct the DBMS.
- After the creation of database we have to excess and updates database.

(iv) Procedure :-

- It is the general instruction to use the database. This instruction includes how to

set up the database, how to install the database, how to login the DB, how to log out the DB, how to manage the DB, how to take the back up of DB and how to generate the reports.

// By using the procedures we can validating the data items.

(v) Users

// There are no. of users who can access and retrieve the data on demand using the applications.

// There are diff. type of users are present in DBMS :-

(i) Native Users
(ii) Online "
(iii) Sophisticated "
(iv) Specialized "
(v) Applications "
(vi) Database Administrators

# Data Dictionary :-

- Data Dictionary stores the information about the data in the database.
- It is implemented in the business resources.
- The data dictionary uses the meta data concept.
- Data dictionary is a database which can stores the informations of the other DB.
- By using the DB, we can identify how the data is stored, how the data is used, who is the owner of the data, who can access the data and where the data is stored.
- In Data Dictionary, it provides 3 benefits

(i) Reduce the redundency or avoid the redundency

(ii) avoids inconsistency data

(iii) provides the quality of data

- Data dictionary are of 2 types :-

(i) Active DD

(ii) Passive "

## (i) Active DD :-

In active data dictionary, it is update automatically when data base structure is changed.

## (ii) Passive DD :-

In passive data dictionary, it is changed structure/ the object of the data.

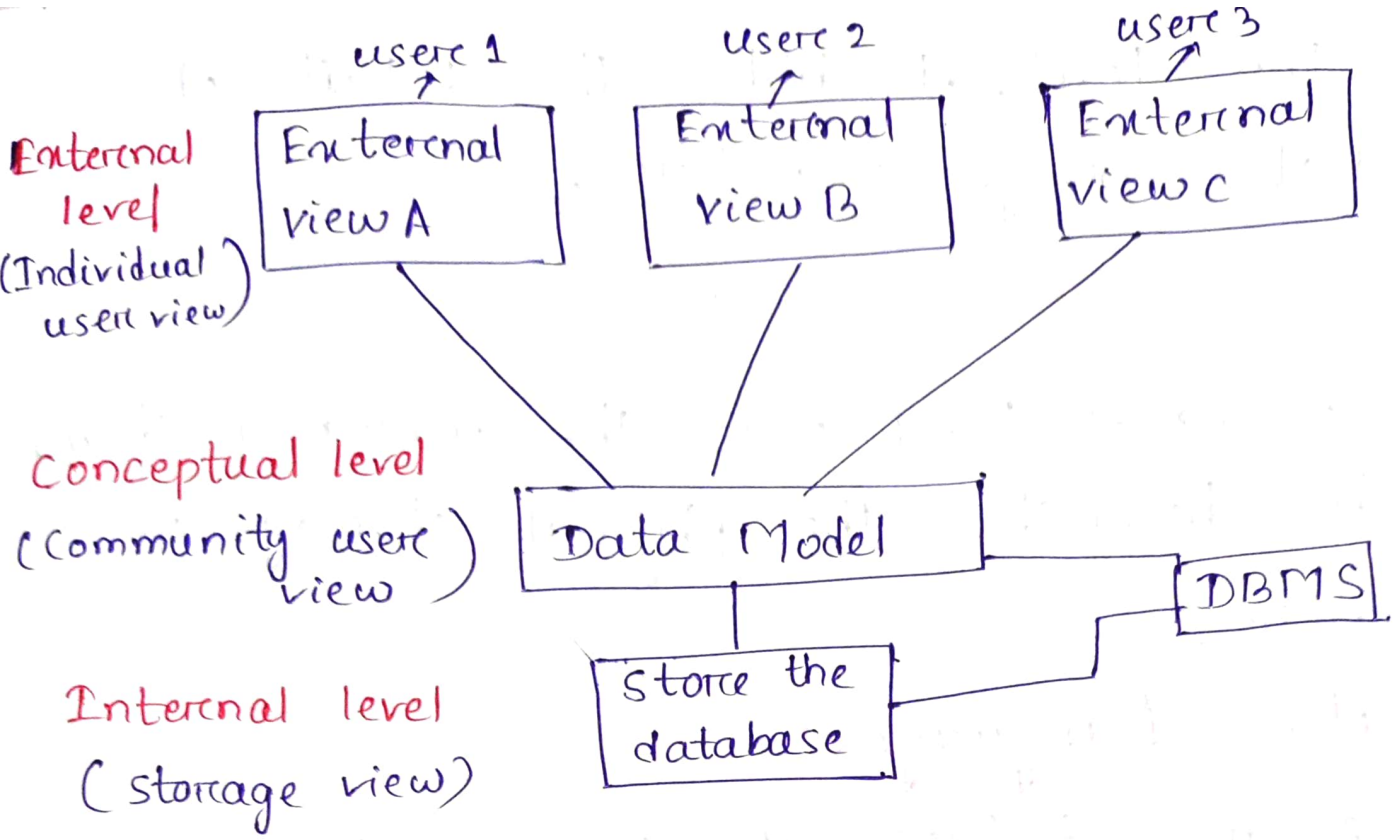
IMP

## Database Architecture / 3-tyre Architecture/ 3-Schema architecture / ANSI / SPARC Architecture

// In 3-tyre architecture, it can uses the schem concept.

// The main goal of 3-tyre architecture is separate the usere applications.

// The 3-tyre architecture uses 3-levels or 3-schema i.e (i) external level or schema
(ii) conceptual " " "
(iii) internal " " "




## (i) External level :-

- The external level can uses the external schema
- In this level, the individual users only see the database, ~~there~~ they are not accessing the "".
- After giving the privilege by the database user can access some part of DB and they are not accessing the whole part of DB.

## (ii) Conceptual level :-

- The conceptual level uses, the conceptual schema which describes the structure of the database for a community of users.

* In this level, it hides the details of physical storage structure.

(iii) Internal level :-

The internal level uses the internal schema which describes the physical storage of database.

## Database Languages :-

There are 6 types of database languages are present. i.e (i) Data definition
(ii) Data manipulation (DML)
(iii) Data control
(iv) Transaction control
(v) View defination
(vi) Standard defination

### (i) Data Defination

* This language is mainly accesed by database administrator or designer to specify the str. of DB.
* In this language, it can uses 6 statements
(i) Create DB
(ii) Create Table

(iii) Alter DB
(iv) Alter Table
(v) Drop DB
(vi) " Table

## (ii) Data manipulation (DML)

// It is managing the data within the schema object.
// The DBM uses 4GL approach.
// Once the database schema are compiled, the database is copulated with the data.
// In database manipulation, we use 4 type of opr$^{n}$ :- (i) Select (retrive the data from database
(ii) Insert (insert the data into the table)
(iii) update (update existing data into table
(iv) delete (delete all the records from the table)

// DML are of 2 types :- (i) Procedural DML
(ii) Non-procedural DML

### (i) Procedural DML

// In this language, the user can specify what data is needed and how to get it.

### (ii) Non-Procedural DML

In this language, the users can only specify

what data is needed.

(iii) Data Control :-

The data control language consists of some commands ——— that controls user access

// It prevents the unauthorized accessed to the data.

// The data base administrator has the power to give and take the privileage. to a specific user.

// The data control language uses 2 staments i.e

(i) Grant :- Give the privileage to the users to access the database.

(ii) Revoke :- Withdraw the privileage from user access.

(iv) Transaction Control :-

// It is used to manage the changes made by data manipulation language statements.

// The transaction control language uses 2 statements i.e

(i) Commit :- Save the work.

(ii) Role back :- Restore the database.

what data is needed.

(iii) Data Control :-

The data control language consists of some commands ——— that controls user access

// It prevents the unauthorized accessed to the data.

// The data base administrator has the power to give and take the privileage. to a specific user.

// The data control language uses 2 staments i.e

(i) Grant :- Give the privileage to the users to access the database.

(ii) Revoke :- Withdraw the privileage from user access.

(iv) Transaction Control :-

// It is used to manage the changes made by data manipulation language statements.

// The transaction control language uses 2 statements i.e

(i) Commit :- Save the work.

(ii) Role back :- Restore the database.

(v) Star View defination :-

// It is used to define the internal and external schema.

(vi) Standard defination :-

// It uses the internal schema.

// It is used when a cleare separcation is maintained between the conceptual and physical level.

ER Diagram :-

ER Diagram means entity-relationship diagram.

// The ER Diagram is a pictorial presentation, and it is classified into 3 categories i.e

(i) entity
(ii) relationship
(iii) attributes

(i) Entity :- Anything exist in real world is known an entity.

// According to the ER Diagram the entity is classified into 2 types :-

(i) entity set
(ii) entity type

## Entity Set :-

// A set of entity of same type.

Ex:- A user entity either employee or customer or businessman.

## Entity type:-

// Any entity type defines collection of entities that have some attributes.

// In entity type we can define the entity name and its corresponding attributes.

Student → entity name

| Name | Ph. no | Address | Reg. no |
|---|---|---|---|

→ attributes

## (ii) Relationship :-

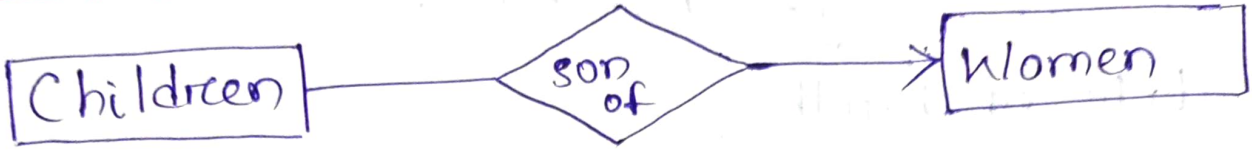
Combination of more than one entity is known as relationship.

// Relationship are of 4 types :- one to one
one to many
many to one
" " many

## (i) One to One :-

(ii) One to many :-


(iii) Many to One :-




(iv) Many to many :-


(iii) Attributes :-

- Columns are present in the table is known as attributes.
- Attributes are of 3 types :-
  - (i) simple vs Composite
  - (ii) Single vs Multivalue
  - (iii) Stored vs derived.

(i) Simple vs Composite :-

- If the attribute can not divided in further is known as simple.

  ex :- age, roll no.
- If the attribute is divided in further is known as composite attribute.

ex:- name.

(ii) Single vs Multivalue:-

• If the attribute uses one value then it is known as single attribute.

ex:- date of birth.

• If the atribute uses more than one value is known as multivalue.

ex:- Phone no.

(iii) Stored vs Derived :-

• If the 2 attributes are related with each other is known as stored.

ex:- DOB, age.

• If the 2 atributes are not related with each other is known as derived.

ex:- name, address.

Data manipulation Opercations :-

• The DMO are uses the schema concept.

• Once the schema is compiled it is populated in the database and providing to the users.

* The user can modify according to their requirements.
* Data manipulation oprⁿ are of 4 types:-

(i) Select
(ii) insert
(iii) update
(iv) delete

## (i) Select :-

The select oprⁿ is used to retrieve the data from the database.

syntax :-

```
select * from tablename
```

## (ii) Insert :-

The insert oprⁿ is used to providing data to the table.

syntax :-

```
insert into tablename values ( );
```

## (iii) update :-

The update oprⁿ is used to update the existing information from the table.

syntax :-

```
update tablename set fieldname =valu
where fieldname = value;
```

(iv) **Delete :-**

The delete opern is used to remove the specific row from the table and also remove all the information from the table.

Syntax:- Delete table tablename
where attribute name = value ;

16 marks imp

## Data Models :-

- // Data Model is a mathematical description of a system.
- // Data Model can hides the storage and implementation details.
- // Data model is a type of data abstracti i.e used to provide the conceptual representation.
- // Data models are of 5 types:-
  - (i) Entity - Relationship Data model
  - (ii) Relational ,,
  - (iii) Hierarchical ,,
  - (iv) Network ,,
  - (v) Object Oriented ,,

## (i) E-R DM :-

// In the entity relationship datamodel, it uses one to one, one to many, many to one and many to many relationship.

// This is called as entity - relationship data model because it uses both entity set and relationship set.

// The entity set are represented by respective tablename, fieldname and along with their definition.

// In relationship set are represented by the field name along with their respective table name to which they belongs.

// It uses 2 types of representation :-

(i) structural represention

(ii) E-R "

### (i) Structural Representation :-

// In structural representation, it defines the structure of the table and it is represented is given below.

entity set <table name>

field 1 : Datatype (size)
field 2 : Datatype (size)
end ;

student → Table name

Field name ←

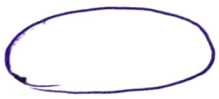
| Name | Ph. no |
|---|---|
| | |

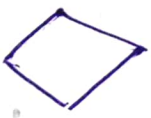
(ii) E-R - Representation :-

// It is a pictorial representation of the entity set and their atribute and their relationship set.

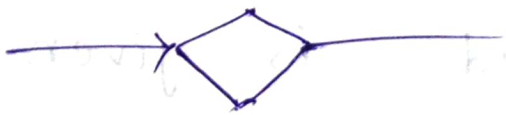
// In E-R representation, it uses some symbols :-

(i) Entity set 

(ii) atributes 

(iii) relationship 

(iv) one to one 

(v) one to many 

(vi) many to one 

(vii) many to many 

## (ii) Relational Datamodel :-

// In the relational datamodel, it uses one to one relationship.

// This model gives the empasis on the relationship set and its various opr$^{n}$ on it.

// In relational data model, it uses 2 types of representation,

(i) formal representation

(ii) Tabular "

### (i) Formal Representation

// In formal representation, it provides the tablename and its respective fieldname are specified.

Ex:- Student ⟶ tablename

Roll no, branch, name, ph. no ⟶ field name

### (ii) Tabular Representation :-

// It is used to display the data items in a respective table.

Student

| Roll no | branch | name | ph. no |
|---|---|---|---|
| | | | |

Advantages :-

Design, implementation and maintainance is quite easy.

// It is a structural independence.

Disadvantages :-

// Hardware cost is very high.

// Easy of design can be lead to a bad design.

## (iii) Hierarchical Data Model :-

// This model is used for one to many relationship i.e parent-child relationship.

// This representation is done in a tree like manner i.e known as hierarchical data model.

// It uses 2 types of data representation

(i) Formal

(ii) E-R Diagram

### (i) Formal

// In formal representation, it can identify the table name and field name are specified.

Ex:- Student → table name
branch, regd. no, name, ph. no → field name

(ii) E-R Diagram

It is a pictorial representation of the relationship set along with entity set.

Advantages:-

// Data sharing.
// Data security.
// Data integrity.

Disadvantages:-

// Implementation is very complex.
// Database management problems are arises.
// In-flexibility.

(iv) Network Data Model :-

// The network data model is quite similar to the hierarchial data model except record can have multiple parents.
// It is of 2 types :- (i) Record type
(ii) Set type

(i) Record type :-

// It represents either a single record or a collection of records.

// The record type is again divided into 2 types :- (i) Owner record type
(ii) Master " "

(i) Owner

// The owner record type represents an independent relation.

(ii) Master

// The master record type represents a dependent relation.

Set type :-

It represents either a single domain or collection of domains.

Advantages :-

// Data independence.
// Database integrity.

Disadvantages :-

// System is very complex.
// not a user-friendly.

## (v) Object Oriented Data Model

- // OO data model uses the concept of entity, attribute and relationship.
- // An entity is a distinct object in the organisation that represents in the data base.
- // An attribute is the property that describe the records and a relationship is an association between the entities.
- // The common type object-oriented data model are categorised 3-different parts.
  - (i) entity
  - (ii) OO
  - (iii) functional
- // The object oriented data models extend the defn of entity and only its attributes that describes the states of the object also associated with the object and its behaviour.
- // The object is said to be encapsulate both its state and behaviour.

Q//1. What is degree of rel°ship?

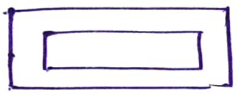
Ans:- How many entities are participating in a relationship is known as degree of rel°ship.

Q//2. What is tuple?

Ans:- Collection of information about the attribute of the table for a single instance is known as tuple.

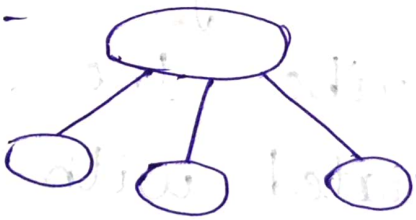
## Notations used for the E-R Diagram :-

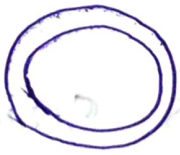
(i) Entity Set :- 

(ii) Weak Entity :- 

(iii) Plain Attribute :- 

(iv) Plain key Attribute :- 

(v) Composite Attribute :- 

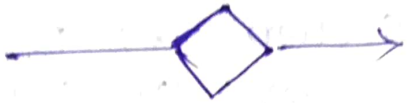
(vi) Multivalue Attribute :- 

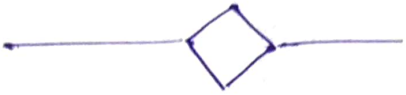
(vi) Derived Attribute :- 

(vii) Relationship :- 

(viii) One to one :- 

(ix) One to many :- 

(x) many to one :- 

(xi) many to many :- 

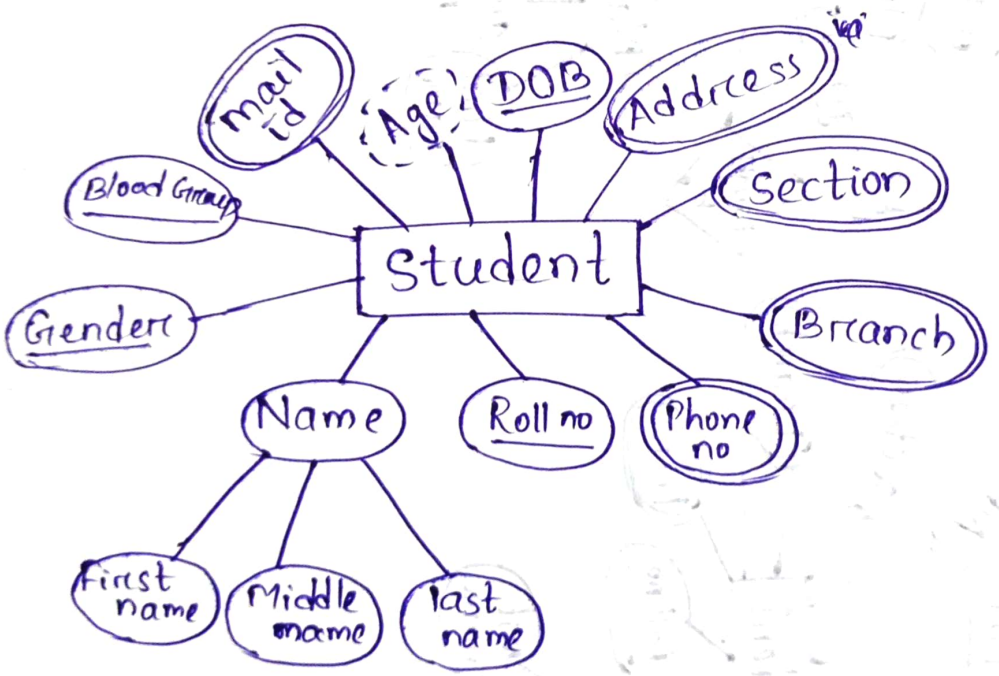
Q. Draw the E-R Diagram of a student

Ans:- Entity

Student

Attributes

(i) Name
(ii) Roll no
(iii) Ph. no
(iv) Branch
(v) Section
(vi) Address
(vii) DOB
(viii) Age
(ix) mail id
(x) Blood group
(xi) gender



Q/2. Draw the E-R diagram of student and teacher

Ans:- Entity

Student

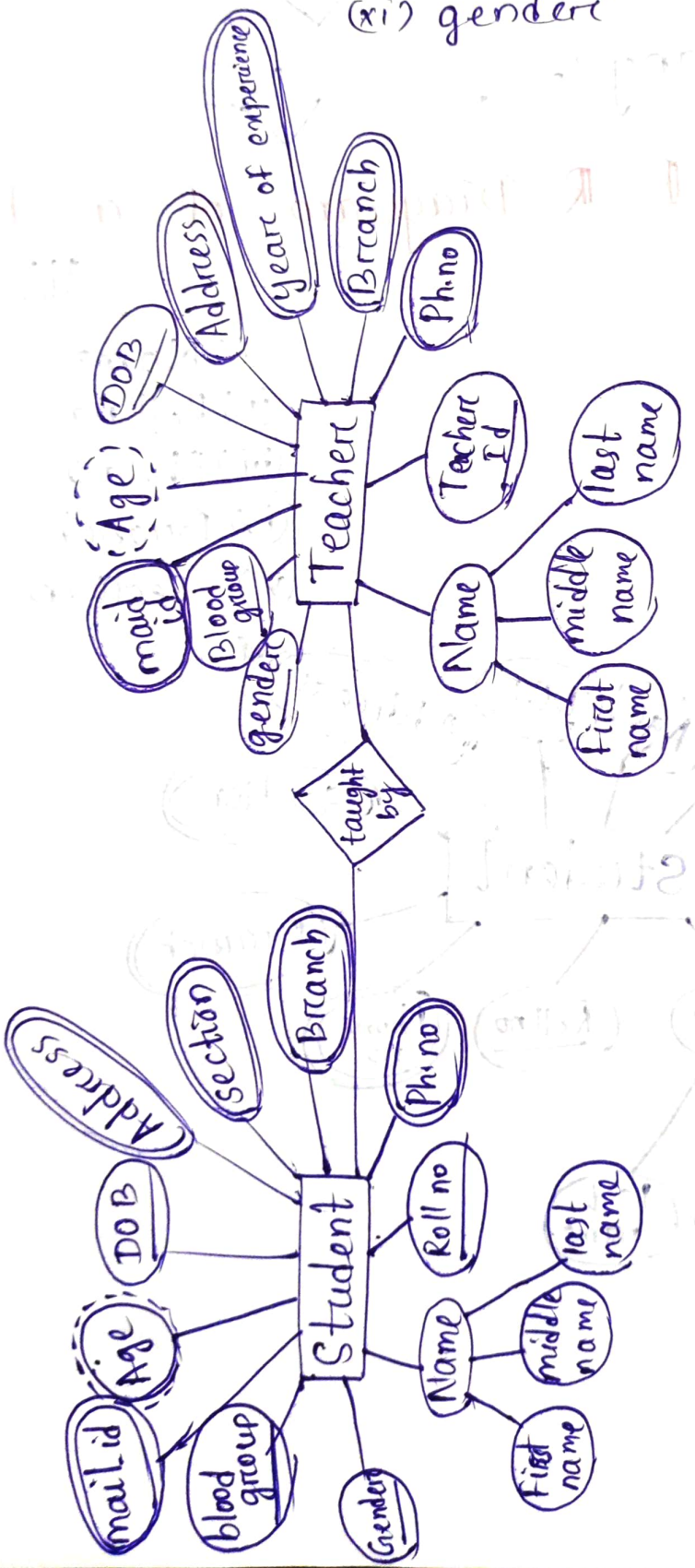
Attributes

(i) Name
(ii) Roll no
(iii) Ph. no
(iv) Branch
(v) Section
(vi) Address
(vii) DOB
(viii) Age
(ix) mail.id
(x) Blood-group
(xi) gender

Teacher

(i) Name
(ii) Teacher Id
(iii) Ph. no
(iv) Branch
(v) year of experience
(vi) Address
(vii) DOB
(viii) Age
(ix) mail. id
(x) Blood group
(xi) gender.

~~Entity~~



Q#3. Draw the E-R Digram of Hospital ~~mang~~ management system.

Entity
(i) Hospital
(ii) Doctor
(iii) Patient
(iv) Staff
(v) Test
(vi) Department

Hospital
(i) Name
(ii) Type (govt. or private
(iii) Address
(iv) Fax no.
(v) no. of doctors
(vi) no. of staff
(vii) no. of beds

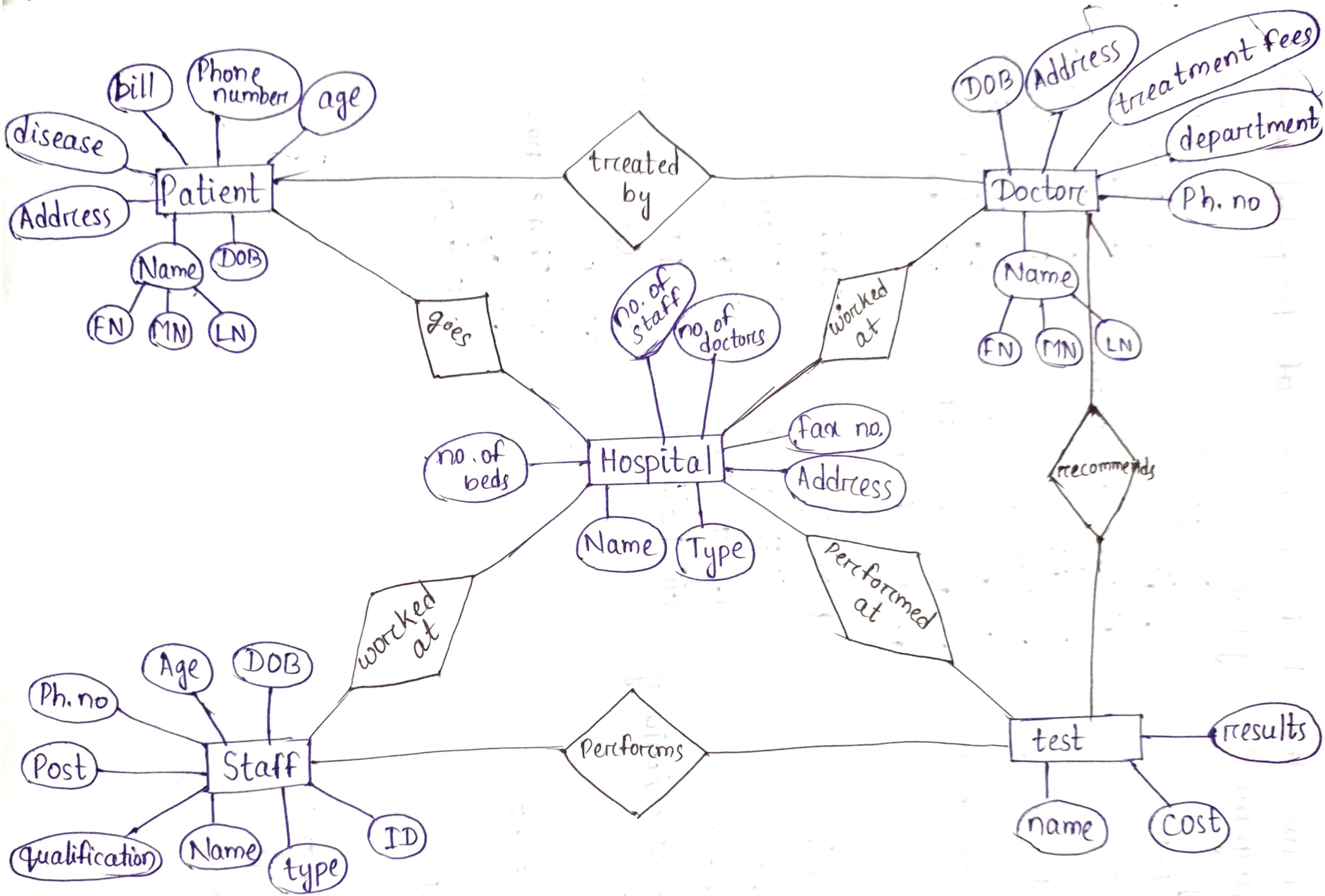
Doctor
(i) Name
(ii) Phone no
(iii) department
(iv) treatment ~~fees~~ fees
(v) Address
(vi) DOB

Patient
(i) Name
(ii) Address
(iii) diesease / injury
(iv) Test
(v) bill
(vi) phone no.
(vii) age
(viii) DOB

Entity
(i) Hospital
(ii) Doctor
(iii) Patient

Staff
(i) Name
(ii) Type
(iii) ID
(iv) Qualification
(v) Post
(vi) Phone no.
(vii) Age
(viii) DOB

Test
Test name
result
cost


Patient
bill
Phone number
age
disease
Address
Name
DOB
FN
MN
LN
treated by
goes
Doctor
DOB
Address
treatment fees
department
Ph. no
Name
FN
MN
LN
worked at
recommends
Hospital
no. of staff
no of doctors
no. of beds
fax no.
Address
Name
Type
performed at
worked at
Staff
Age
DOB
Ph. no
Post
Qualification
Name
type
ID
Performs
test
results
name
Cost

Q/4. Draw the E-R Diagram of university management system.

Q/5. Draw the E-R Diagram for a specific company.

Q/6. Draw a E-R Diagram for car insurance company.

Q/7. Draw a E-R Diagram for employee managment system.

4. Entity

University
Professors
Students
Staffs
Department

Professors

Name
Id
DOB
Age
Phone no
Qualification
Department
Salary

Students

Name
roll no
phone no
Branch
address
DOB
Age
Gender

Staff

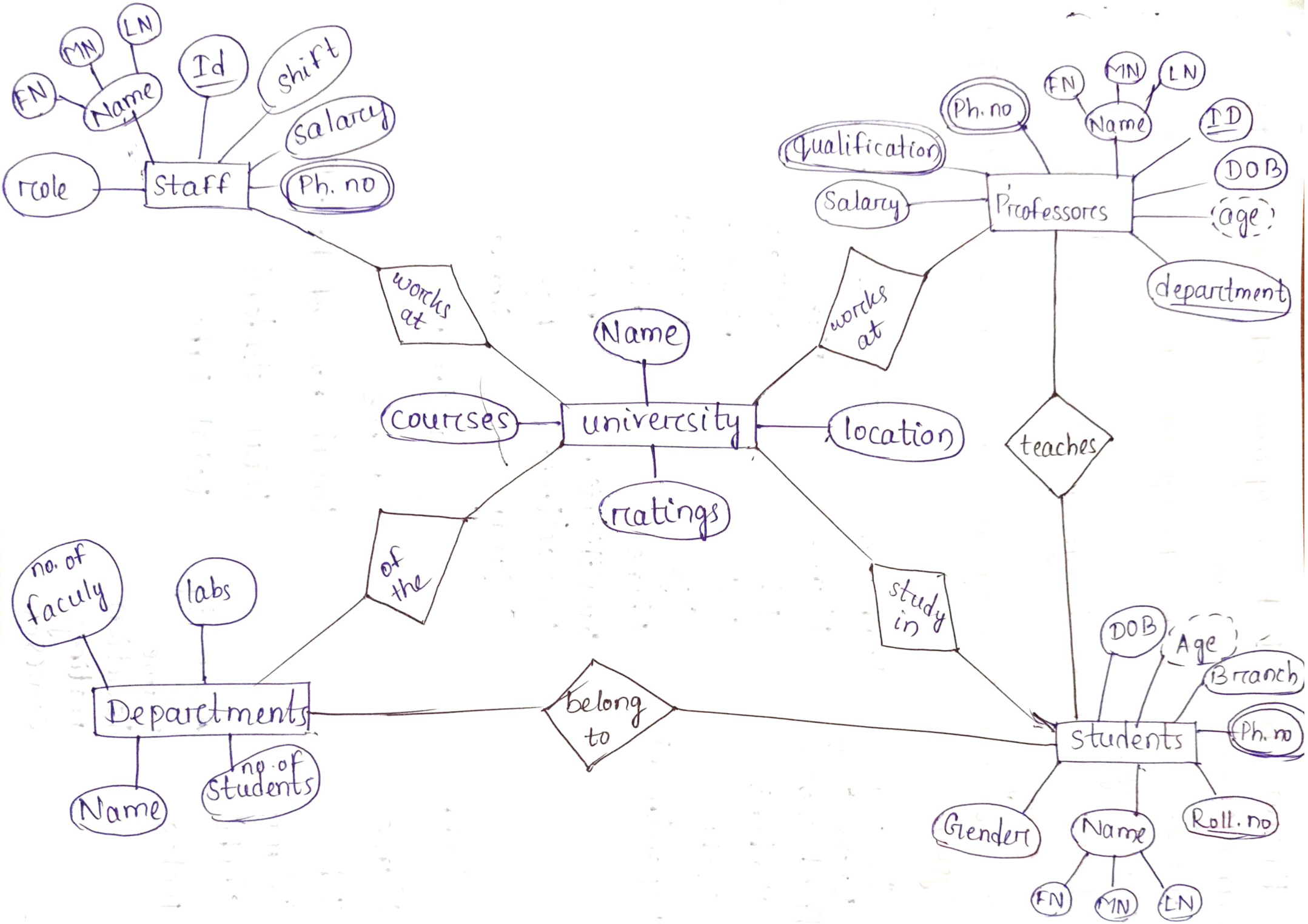
Name
Id
Salary
shift
roles

Department

name
total no. of faculty
total student
total labs

University

Name
location
ratings
courses


FN
MN
LN
Name
Id
shift
Salary
role
Staff
Ph. no
works at
Name
courses
university
location
ratings
works at
Qualification
Ph. no
FN
MN
LN
Name
ID
DOB
Salary
Professors
age
department
teaches
no. of faculy
labs
of the
Departments
Name
no. of students
belong to
study in
DOB
Age
Branch
Ph. no
Students
Grender
Name
Roll. no
FN
MN
LN

5. Entity

Employees
Customers
Product

Employees

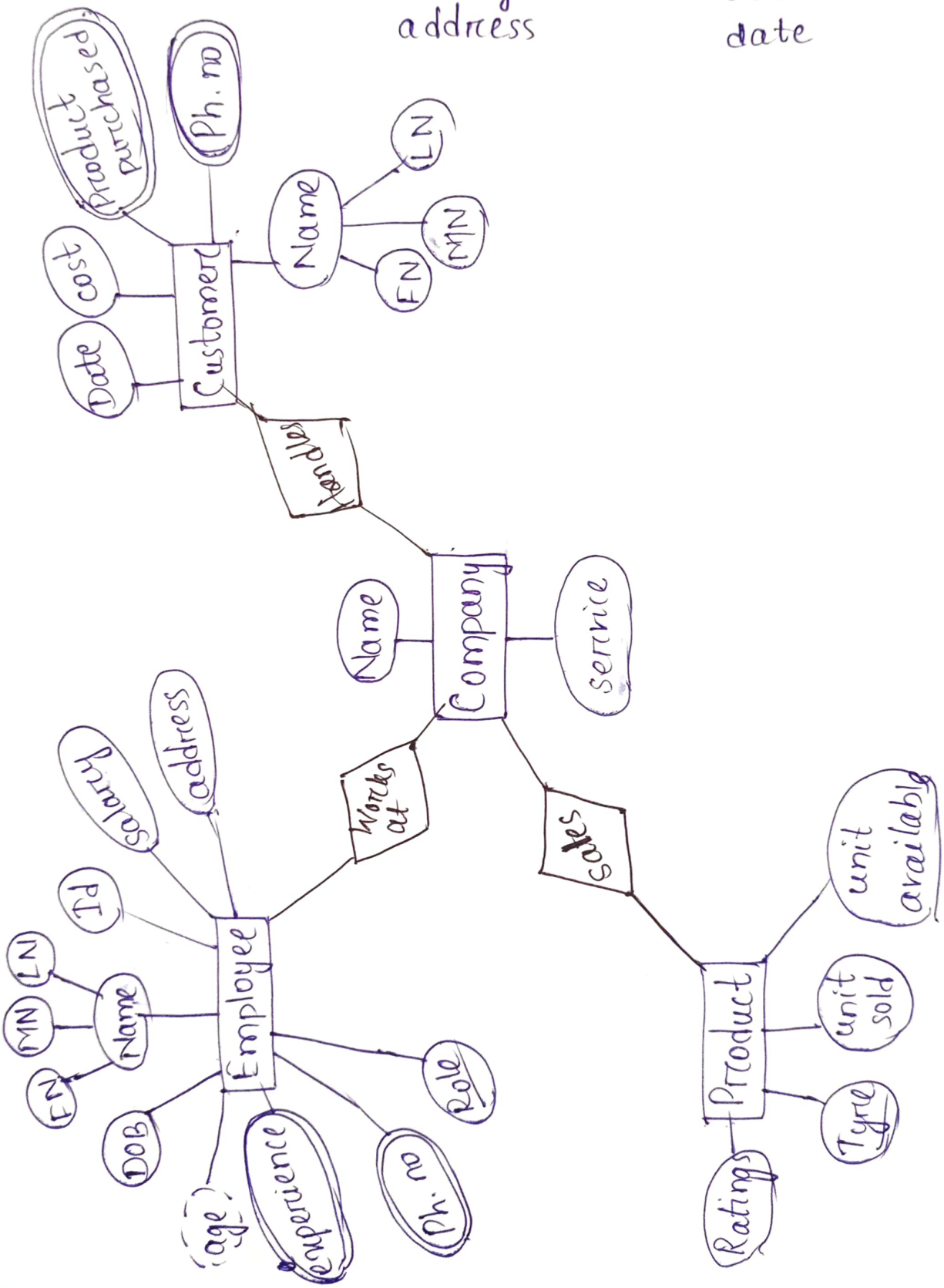
Name
Id
age
DOB
experience
role
salary
address

Customer

Name
Ph. no
email id
product Purchased
cost
date

Product

Type
Sold
availability
ratings



Draw the ER Diagram for a car insurance company.

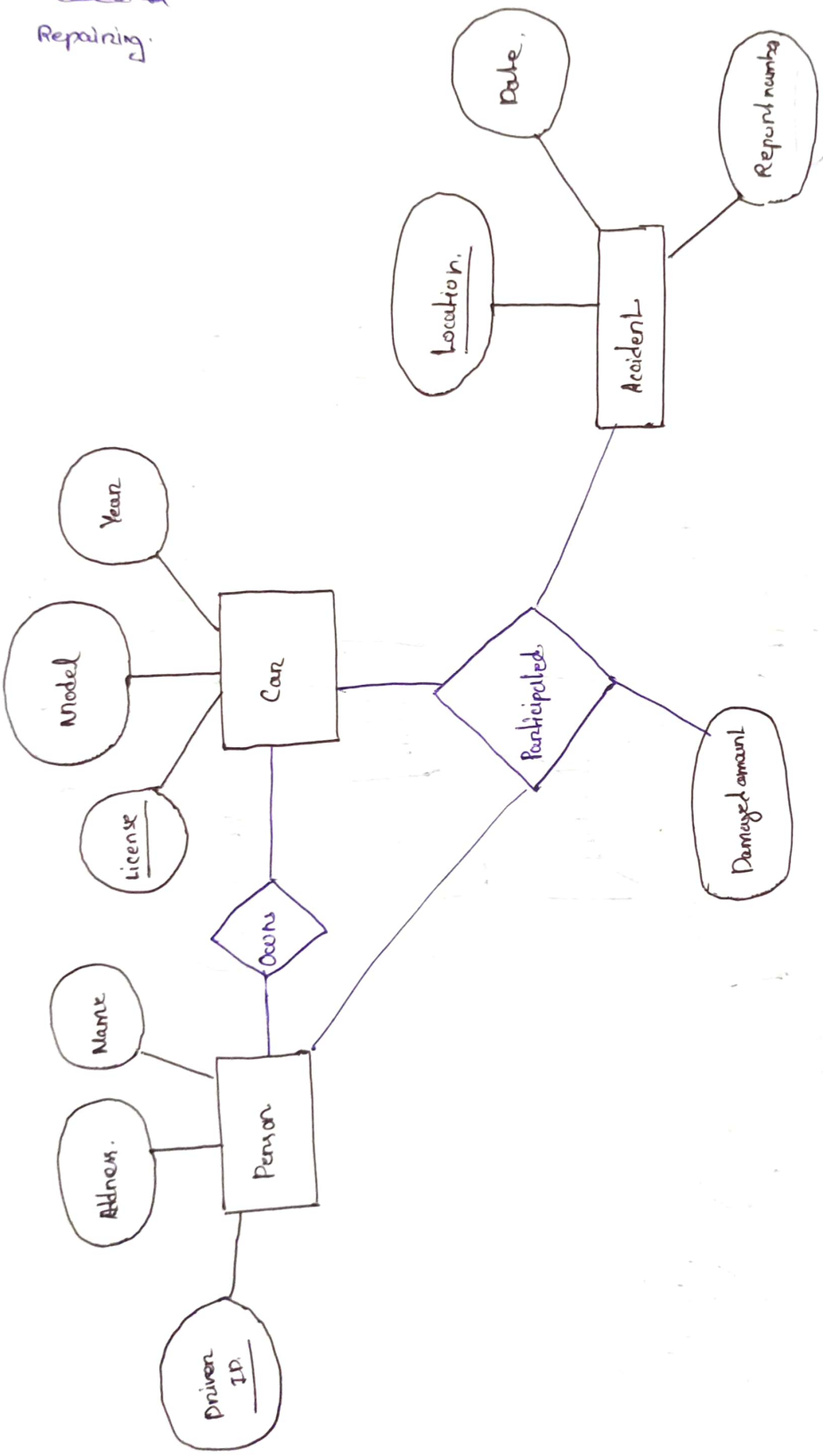
entity

Car

Owner

[redacted]

Repairing.



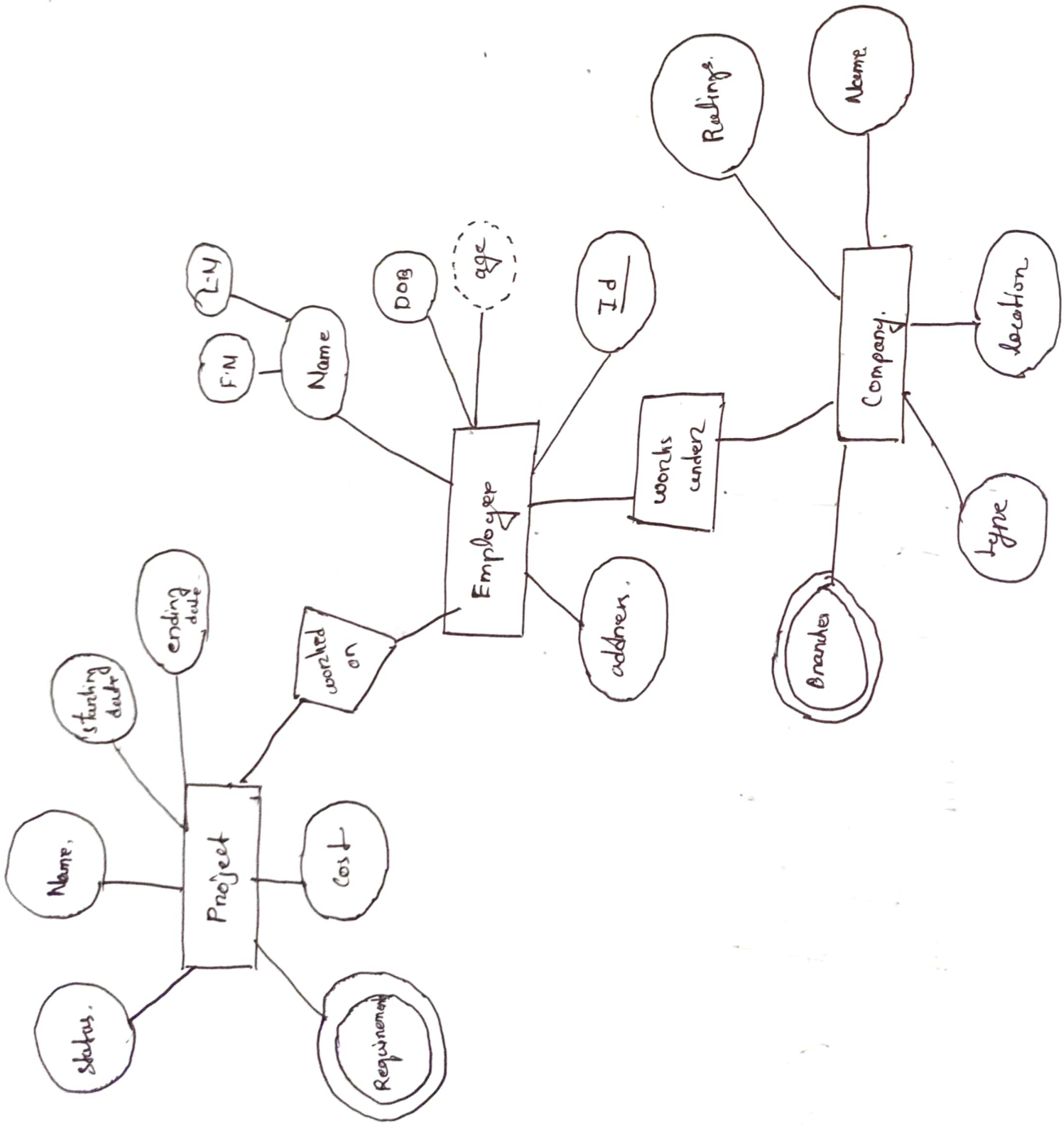
Draw the E-R diagram for employee management system

→ Employee
→ project
→ company

**employee**
Name.
age
DB
address
Id.

**project**
Name
Starting date
ending date
status.
Requirements
Cost

**Company**
Name
location
type
Ratings
Branch



* Database users / end users :

An end users are the people whose job require to access to the database for inserting, updating and generating the reports.

→ An users are of four types. (i) Casual end user.
(ii) Naive end user.
(iii) Sofesticated end user.
(iv) standalone end user.

(i) Casual end user :

Casual end user to occasionaly access the database but they may need different information each time.

→ They use a safesticated database query language to specify there request and they are the top level managements.

Ex →
Bank mannegen

(ii) Naive end User :

The end user uses different portion of the database.

→ there main job is to update the database.

Ex →
account users.

(iii) Sofesticated end User :

They are the engineers, scientist, doctors and business analyst there main job is in order to impliment the new applications by using the complex requirement.

→ The safesticated end users are developing the new concepts according to the market requirements.

(iv) Stand alone end user :

It maintainces the personal database by using the readimade software packages that provides easy to use.

## Key

Key is an ~~set~~ o attribute or an set of attributes at a rel$^n$ having capability to identity is tuple uniquely.

// The keys can be categorised into the following types :-

(i) Primary Key
(ii) Foreign Key
(iii) Alternate Key
(iv) Candidate Key
(v) Super Key
(vi) Composite Key

## Key Integrity Constraint :-

// Every rel$^n$ in a database should have atleast one set of attributes that defines the tuple uniquely.

// These set of attributes are called Keys.

// A Key has two properties :-

(i) It should be unique for all tuples.

(ii) It cann't have null values.

## (i) Primary Key :-

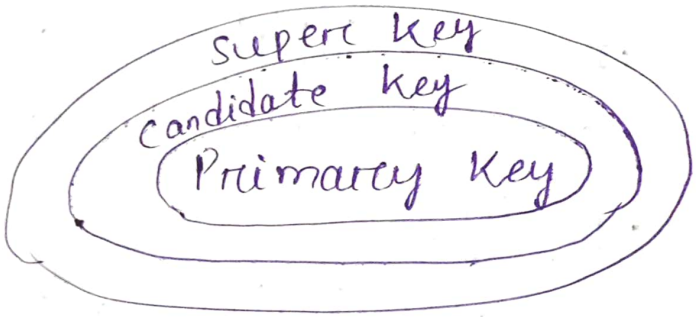
// Primary Key in SQL is a column that uniquely identifies the records / tuples in that table.

// There can only be one primary key in that table, but that primary key can be of one or more columns.

// A primary key automatically has a unique constraint defined on and it ensures that there are no duplicate or null values in that column.

SYNTAX :-

```
CREATE TABLE table_name (column1 datatype
                         NOT NULL PRIMARY KEY)
```




## (ii) Candidate Key :-

// The minimum set of attributes that can uniquely identify or tuple is known as candidate key.

// ex :- serial no. in any relation

### Features :-

// It is a minimal super key.

// Super key with ~~not~~ no repeated data is called candidate key.
// It must contain unique values.
// It can contain null values.
// Every table must have atleast a single candidate key.
// A table can have multiple candidate keys but only one primary key.
// The value of candidate key is unique and may be null for some tupples.

(iii) Super Key :-

// The set of attributes that can uniquely identify a tuple is known as super key.
// A super key is a group of single or multiple keys that identifies rows in a table.
// It supports null values.

Features :-

// Adding 0 or more attributes to the candidate key generates the super key.
// A " " is a super key but the vice-versa is not true.

Ex:-

| Sl. no | Name | Course | Marks |
|---|---|---|---|

Super Key:

(Sl. no + name)

(Sl. no + course)

(sl. no + mareks)

(Sl. no + name + marks)

(Sl. no + course + marks)

(sl. no + name + course + marks)

(iv) Foreign Key:-

// A foreign key column is one table always refers primary key of another table.

// A table can have multiple foreign key.

// It is used to ~~integr~~ maintain referential integrity.

// Whenever we are using the foreign concept it required min^m 2 tables.

P.key

| Did | Dname |
|---|---|
| 1 | CSE |
| 2 | ME |
| 3 | MCA |

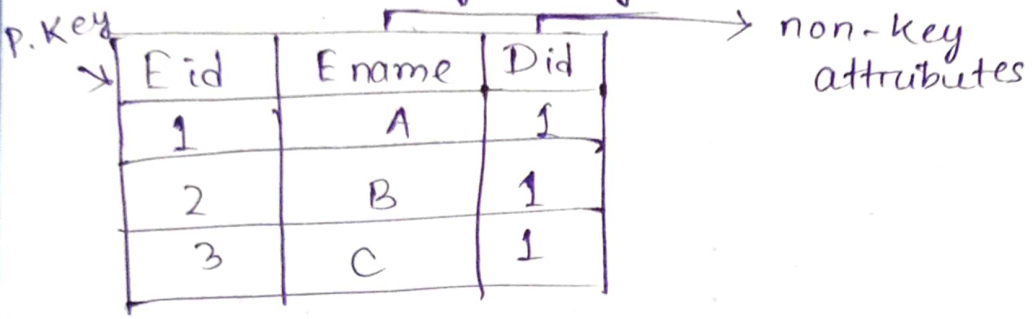
P.key

| Eid | Ename | Eid |
|---|---|---|
| 1 | Sami | 1 |
| 2 | Tejas | 1 |
| 3 | Roji | 1 |

F.K

## (v) Composite Key :-

In many cases we design a database and the database creates the multiple tables that uses more than one column as the part of the primary key is known as composite key.




| Eid | Ename | Did |
|---|---|---|
| 1 | A | 1 |
| 2 | B | 1 |
| 3 | C | 1 |

## (vi) Alternate Key :-

// An alternate key is otherwise known as secondary key that contains all property of a candidate key but is an alternate option.

// An alternative is a candidate key i.e capable to identifying a row uniquely however such key is not used as a primary key.

// Only one key is selected as the primary key and the other remaining keys are known as alternate key or secondary key.

# Mapping ER Model to Relational Model :-

The mapping ER model to relational model uses 3 concept :- (i) Strong Entity Set
(ii) Weak " "
(iii) Relation

## (i) Strong Entity Set :-

// In strong entity set it uses the key constants. (Primary key).
// It represents each entity set as a relation
// In strong entity set in the ER Model creates a relation of the same name with the same set of attributes.

## (ii) Weak Entity Set :-

// In weak entity set it is not used the key constraints.
// It represents each entity set as a relation
// In weak entity set it ~~dosr~~ doesn't includes the key attributes.

## (iii) Relation :-

// The relation is denoted as 'R'.
// In the ER model it creates a relation of same name with the key attributes of each entity set in the rel$^{n}$ (R) as the

attributes of the relation.

Q. Draw the E R diagram of Hospital Management system and convert it into Relational model.

Ans:- (i) E-R Draw

(ii) Convert it into Relation

Entities

Hospital , Patient, Doctor, Staff , Test

Hospital → Entity name (Table)

| Name | Type | Address | Fax no. | no. of doctors | no. of staff | no. of beds |
|---|---|---|---|---|---|---|
| | | | | | | |

Patient → E. name (Table)

| Name | Address | diesease/ injury | Test | bill | Ph. no | age | Id |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

Doctor → E. name (Table)

| Name | Phone. no | Dept. | treatment fees | Address | Id |
|---|---|---|---|---|---|
| | | | | | |

Test → E. name

| Name | Result | cost |
|---|---|---|
| | | |

6 marks

# Integrity Constant

- It is a condⁿ that can be applied in the database to restrict the data according to the need.
- If the condⁿ is satified then the data is stored into the database.
- The integrity constraint can be applied in the database when the database administrator and the end users defines the db schema.
- It is classified into 3 categories :-
  (i) Entity Integrity Contraints.
  (ii) Referential " "
  (iii) Domain " "

## (i) Entity IC :-

- In entity ic, it uses the concept of the primary key.
- It states that no primary key value can be null.
- This is because the primary key value is used to identify individual tuple in in a relⁿ.

// If we provide the null value for the primary key that can't identify the tuple.

## (ii) Referential IC :-

// The referential integrity constraint uses the concept of foreign key.

// It is a condition that uses a value in one relation for a given set of attributes also uses for a certain set of " in the another rel$^{n}$.

// It can ~~satisfiel~~ satisfied bet$^{n}$ two rel$^{n}$.

## (iii) Domain IC :-

// In domain ic, It uses the domain values for a particular field.

// The domain IC. is classified into 2 categories :- (i) not null constraints
(ii) check "

### (i) Not null constraints :-

// It is associated with a field that can not accept any null type values during the insertion.

// If a blank values given then the entire record is discarded.

(ii) Check constraints:-

- It is used to specify a user defined condn
- Whenever this constraint is specified with a condn then every insert oprn is possible and it avoids the blank values.

Q// What is domain?

Ans:- Domain is denoted as D which provides the atomic value . . .

- Atomic means each value in the domain is individuals.

Eg:-

| Ph.no | Name |
|---|---|
| 0-9 | A to Z<br>a to z |

- A domain can uses concept of relational schema and it is denoted as 'R' and it is returned as $R(A_1, A_2, A_3 \ldots A_n)$ where R is known as Relational schema, $A_1, A_2, A_3 \ldots A_n$ is known as attributes
- The domain can find out the degree of relational schema.

Q) What is degree of relational schema?

Ans:- The degree of relational schema is count how many attributes are present in relational schema.

Student

| Regd.no | Name | Ph.no | Address |
|---|---|---|---|
| | | | |

In the above example the degree of relnal schema is 4 because the student table is uses 4 attributes.

Tuples:-
Collection ~~in~~ of information ~~is k~~ about attributes is known as Tuples.

## Data Independence:-

The data independence is a process which can be defined as the capacity to change the schema at one level of database system without having to change the schema at next higher level.

There are 2 types of data independence are present :- (i) Logical Data Independence
(ii) Physical " "

## (i) Logical data independence :-

It is a process to change the conceptual schema without having change the external schema.

## (ii) Physical data independence :-

It is the capacity to change the internal schema without having to change the conceptual schema.

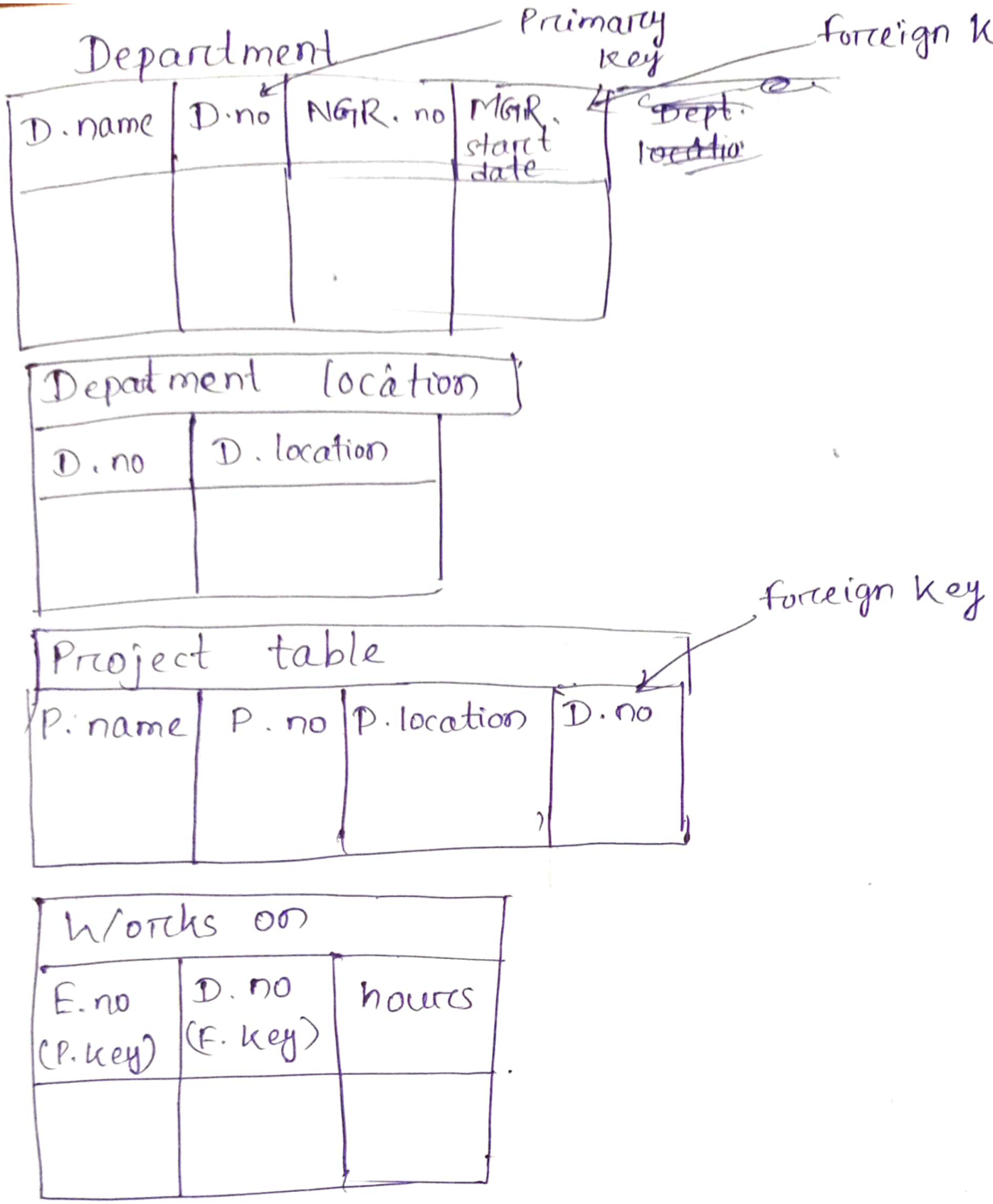
## Relational Algebra :-

- The relational algebra is a procedural query language.
- It consists of set of operation that takes one or more relation as i/p and produce a new relation as a o/p.
- In relational algebra is identify how the data stored in the DB and also describe how the database is operates.

Primary key

Foreign key

| Employee | | | | | | | |
|---|---|---|---|---|---|---|---|
| E.name | E.no | DOB | Address | Gender | Salary | Dept.no | Supervisor no. |
| | | | | | | | |




## Unary Relational Operation :-

// It is of 2 types :- (i) Select oprⁿ
(ii) Project oprⁿ

(i) Select oprⁿ :- (retrive row value from table)
The select oprⁿ is used to select a subset of the tuples from a relation that satisfied the select condⁿ.

- Those tuple satisfy the cond$^n$ that are selected and those tuple doesn't satisfy the cond$^n$ that are discarded.
- The select opr$^n$ uses boolean operators (and, or, not).
- It also uses the comparison operators ($>, <, =, \leq, \geq, \neq$).
- The select operation is denoted as

$$\sigma_{\langle \text{select cond}^n \rangle}(R)$$

where, select operator is denoted as $\sigma$.

- $\langle$select cond$^n\rangle$ :- it is an boolean expression.
- R :- It is a relational algebra or table name.

Q/1. To select the employee ~~tuples~~ table whose dept. no. is 4.

Ans:-

$$\sigma_{\langle D\_no = 4 \rangle}(\text{Employee})$$

Q#2. To select the employee tuples whose salary is greater than 30,000.

Ans:-

$$\sigma \langle \text{salary} > 30,000 \rangle \quad (\text{Employee})$$

Q#3. Select the tuples of all employee who either work in dept. no = 4 and make over 25,000 salary or work in dept. no = 5 and make over salary 30,000.

$$\sigma \langle D.no = 4 \text{ AND } Salary > 25,000 \rangle \text{ OR } \langle D.no = 5 \text{ AND } salary > 30,000 \rangle \quad (\text{Employee})$$

(ii) Project Opr$^n$ :- (retrieve column value)

Project operation is used to select contain columns from the table those column satisfy the condition those column doesnot satisfies the condition. that value are discarded.

" Project operation is denoted us.

$$\pi \langle \text{Select condition} \rangle \quad (R)$$

where, $\pi$ is denotes project operation.

⟨select condition⟩ It denotes boolean expression and operation.

(R) denotes ~~ret~~ relational algebra for table name.

Q//1. ~~It~~ d Find Birthdate, salary, address and Ename.

↑ ⟨ Birthdate, salary, address, Employee name ⟩ (Employee)

Q//2. Find out Employee number, gender, birthdate and ~~dependent~~ department name.

↑ ⟨ employee no, gender, birtdate ⟩ employee

(iii) Sequence of Operation :-

Sequence of operation means it was row and column of a specific table.

// It is the combination of select and project operation.

// It provides a intermediate result as a relation.

Q//.1 Retrive the employee name, birthdate, salary of employee who works on Department number seven.

(employee)

↑ ⟨ Ename, B.date, Salary ⟩ ⟨ $\sigma_{D.no=7}$ ⟩

Q#2. Retrieve Department Name, Department No, manager no. where the manager start date is 23 January 1986.

$\pi$ ⟨D.Name, D.No, Manager no⟩ $\sigma$ ⟨M.startdat = 23 Jan 1986⟩ (employee)

(iv) Rename Operation :-

In rename operation we can rename either the relationname or attribute name.

The rename operation is denoted as

$$\rho_s \langle B_1, B_2, B_3 --- B_n \rangle (R)$$

where, $\rho$ : denotes rename operation.

s : denotes new relation name.

⟨$B_1, B_2 --- B_n$⟩ : are known as new attribute name.

R : Relational algebra or table name.

Q#1. Retrieve employee name, birthdate, salary of employee who works on dept. number 5.

A ← $\sigma$ ⟨D.no = 5⟩ (employee)

| E-name | E.no | D.no | DOB | Email | address |
|---|---|---|---|---|---|
| | | | | | |

$B \leftarrow \pi \langle Ename, DoB, \sout{\quad\quad}, Salary \rangle$ (A)

B

| Ename | Salary | DOB |
|---|---|---|
| | | |

$\sigma_{(E, salary, DOB)}$

(6) Relational

* Relational Algebra Operation from set theory:

These are the binary operation. i.e applayed in two sets., it is a standard mathemetical operation on sets.

→ It is off three types. (i) union.
(ii) Intersection.
(iii) set difference or minus.

(i) Union:

It uses two relation. $R(A_1, A_2, \ldots A_n)$ and another relation $S(B_1, B_2, \ldots B_n)$ are said to be union operation.

→ The result of the operation is denoted by $R \cup S$, in the relation that includes all the tuples that are either in the relational R or in the relation S or both the relation R and S

in union operation duplicate tuples are eleminated

(Q.1) Retrive the emplee number of all employee who either works in Dept no = 5 or directly supervisionnumber who works in Deptno = 5

$A \leftarrow \sigma_{<Dept = 5>}(Employee)$

| E.name | E.id | DOB | Address | No. |
|---|---|---|---|---|
| | | | | |

$B \leftarrow \pi_{<E.no>}(A)$

| E.no. |
|---|
| |

$C \leftarrow \pi_{<S.no>}(A)$

| S.no. |
|---|
| |

B ∪ C

| E.no and S.no. |
|---|
| |

(ii) Intersection : (∩)

The intersection uses two relation $R(A_1 A_2 \dots A_n)$ and $S(B_1 \dots B_n)$ are said to be intersection.

→ The intersection is denoted as $R \cap S$

→ It is a relation that includes all the tuples that are in both R and S.

(Q.1) Retrive the employee number of all employee who either work in Dept no : 3, or directly supervise number who work in department : 3.

$A \leftarrow \delta_{\langle Dept.no = 3 \rangle}(Employee)$

| E.name | E.no | DOB | Deptno | Salary |
|---|---|---|---|---|
| | | | | |

$B \leftarrow \pi_{\langle E.no \rangle}(A)$

| E.no |
|---|
| |

$C \leftarrow \pi_{\langle S.no \rangle}(A)$

| S.no. |
|---|
| |

$D \leftarrow B \cap C$

| E.no | S.no |
|---|---|
| | |

(iii) Set Difference :

The result of the operation is denoted as R - S. where R is one relation, and S is one relation.

→ In set Difference, the relation that includes all the tuples that are present in R but not in S.

(Q.1)

$R = \{1, 2, 3, 4\}$

$S = \{3, 4, 5, 6\}$ find out the set difference.

$R - S = \{1, 2\}$

## * Cross product / Cartetian Product :

It is denoted as $\times$ symbol and uses binary set of operation.

→ It uses two relation $R(A_1 A_2 \dots A_n)$ and the second relation $S(B_1, B_2 \dots A_n)$

→ And the general result is $R \times S = R(A_1, A_2 \dots A_n) \times S(B_1, B_2 \dots B_n)$

Student table

| Name | Branch |
|---|---|
| A | EE |
| B | EEE |
| C | CSE |

Teacher

| Id |
|---|
| 12 |
| 13 |
| 14 |

Student / teacher

| Name | Branch | Id. |
|---|---|---|
| A | EE | 12 |
| A | EE | 13 |
| A | EE | 14 |
| B | EEE | 12 |
| B | EEE | 13 |
| B | EEE | 14 |
| C | CSE | 12 |
| C | CSE | 13 |
| C | CSE | 14 |

## * Binary Relational Operator :

Binary relational operator use join condition, it is used to combine related tuples from two relation into a single tuple.

→ Join operation is denoted as. ⋈

→ This operation is very important for any relational Database which contains with more than a single relation because it knows relationship among relation.

→ It uses two relations. $R(A_1 A_2 \dots A_n)$ and $s(B_1, B_2 \dots B_n)$

$B \leftarrow {}^{*}\langle Ename, DOB, Salary \rangle^{(A)}$

| Ename | Salary | DOB |
|---|---|---|
| | | |

→ The general form of join condition is written as $R \bowtie_{(condition)} S$.

The join operation are of three types (i) equijoin.
(ii) Natural Join.
(iii) Outer join.

* Equijoin :

The most case of join condition with equality comparision.

→ On equijoin it uses a comparation operator i.e equal to. (=)

→ The equijoin uses the statement

equijoin = Cartention product + Select operation

→ If uses more than one pair of attribute that have identify the value of every tuple.

Employee

| E. No | E.name | E.loc. |
|---|---|---|
| 7369 | SMITH | DALLAS |
| 7499 | SMITH | CHICAGO |
| 7566 | JONES | DALLAS. |
| 7698 | BLAKE | CHICAGO |
| 7782 | CLARK | NEWYORK |
| 7788 | SCOTT | CHICAGO. |

Department

| Dept no | Dept. Loc |
|---|---|
| 10 | NEWYORK. |
| 20 | DALLAS. |
| 30 | CHICAGO |
| 40 | DALLAS. |

Retrive the employee and Department whose E. loc and Dept. loc is same.

| E.NO | E.name | E.loc | Dept no | Dept loc |
|---|---|---|---|---|
| | | | | |

→ The general form of join condition is written as $R \bowtie_{(condition)} S$.

The join operation are of three types (i) equijoin.
(ii) Natural Join.
(iii) Outer join.

* Equijoin :

The most case of join condition with equality comparision.

→ In equijoin it uses a comparision operator i.e equal to. (=)

→ The equijoin uses the statement

equijoin = Cartention product + Select operation

→ It uses more than one pair of attribute that have identify the value of every tuple.

Employee

| E. No | E.name | E.loc. |
|---|---|---|
| 7369 | SMITH | DALLAS |
| 7499 | SMITH | CHICAGO |
| 7566 | JONES | DALLAS. |
| 7698 | BLAKE | CHICAGO |
| 7782 | CLARK | NEWYORK |
| 7788 | SCOTT | CHICAGO. |

Department

| Dept no | Dept. Loc |
|---|---|
| 10 | NEWYORK. |
| 20 | DALLAS. |
| 30 | CHICAGO |
| 40 | DALLAS. |

Retrive the employee and Department whose E. loc and Dept. loc. are same.

| E.NO | E.name | E.loc | Dept no | Dept loc |
|---|---|---|---|---|
| | | | | |

| E.name | E.no | E.location | D.no | D.loc |
|---|---|---|---|---|
| SMITH | 7369 | DALLAS | 20 | DALLAS. |
| SMITH | 7499 | CHICAGO | 30 | CHICAGO |
| JONES | 7566 | DALLAS | 20 | DALLAS. |
| BLAKE | 7698 | CHICAGO | 30 | CHICAGO. |
| CLARK | 7782 | NEWYORK | 10 | NEWORK |
| SCOTT | 7788 | CHICAGO | 30 | CHICAGO |
| SMITH | 7869 | DALLAS. | 40 | DALLAS. |
| JONES | 7566 | DALLAS | 40 | DALLAS. |

$\delta_{E.loc = Dept.loc}$ (Employee × Department)

## Functional Dependency :-

It describe association among attributes and it is written as $A \rightarrow B$.

// If B can't determine C then $B \rightarrow$ B tends to C isn't possible. $B \not\rightarrow C$

Ex :- Regd. no $\rightarrow$ name, branch, Ph. no.

name $\not\rightarrow$ Reg. no, branch

## Characteristics of Functional Dependency :-

// It deals with one-one relnship among the attributes.

// It must depends on schema but not instance.

// It should be non-trivial or completely non-trivial.

// It X determines Y if X is superset of Y ($X \supseteq Y$)

ex :- AB $\rightarrow$ B (Possible)

AB $\rightarrow$ BC (not possible)

## Inference rule for FD :-

Inference rule that can be used a new dependency from a given set of dependency. There are 6 types of rules are present.

(i) Reflexive Rule (if $x \supseteq y$ then $x \to y$)

(ii) Augumentation Rule (if $x \to y$ then $xz \to yz$)

(iii) Transitive Rule (if $x \to y$ and $y \to z$ then $x \to z$)

(iv) Decomposition Rule (if $x \to yz$ then $x \to y$ and $x \to z$)

(v) Union Rule (if $x \to y$, $x \to z$ then $x \to yz$)

(vi) Pseudo Transitivity Rule (if $x \to y$, $wy \to z$ then $xw \to z$)

(i) Reflexive Rule :-

Q// If $\alpha$ is a set of attributes and $\beta$ is subset of $\alpha$ then $\alpha \to \beta$ is a FD or not.

Proof :-

Let $t_1$ and $t_2$ are the 2 tuples of the relation then it is written as $t_1$ and

$t_1[\alpha] = t_2[\alpha]$

then

$\beta \subseteq \alpha$ is given.

So, it will be written as $t_1[\beta] = t_2[\beta]$

If So,

$\alpha \to \beta$ is a Functional Dependency.

(ii) Augumentation rule :-

Q// If $\alpha \to \beta$ is a FD and 'r' is set of attribute then $r\alpha \to r\beta$ is a FD or not.

Proof :-

Let $t_1$ and $t_2$ be the 2 tuples of the relⁿ R.

Let us assume that the FD $r\alpha \to r\beta$ is not a FD in the relⁿ R.

Since $\alpha \to \beta$ is satisfied in the relation

So we can get that $t_1[\alpha] = t_2[\alpha]$ --- (i)

then $t_1[\beta] = t_2[\beta]$ ---- (ii)

Since $r\alpha \longrightarrow r\beta$ doesn't hold in the relⁿ R and it is return as $t_1[r\alpha] = t_2[r\alpha]$ --(iii)

then $t_1[r\beta] \neq t_2[r\beta]$ --- (iv)

then

Combine eqⁿ (i) and (iii),

$t_1[r] = t_2[r]$ --- (v)

By using eqⁿ (ii) and (iv)

$t_1[r\beta] = t_2[r\beta]$ --- (vi)

From eqⁿ (iii) and eqⁿ (vi), We get

$r\alpha \to r\beta$

(iii) Transitive Rule :-

Q// If $\alpha \to \beta$ and $\beta \to \gamma$ is a FD then $\alpha \to \gamma$ is a FD or not.

Proof :-

Let $t_1$ and $t_2$ be the 2 tuple of the relation R. Since $\alpha \to \beta$ is satisfied on the reln.

$$t_1[\alpha] = t_2[\alpha] \quad \text{---(i)}$$

$$t_1[\beta] = t_2[\beta] \quad \text{---(ii)}$$

Since $\beta \to \gamma$ is satisfied in the relation So, we can get

$$t_1[\beta] = t_2[\beta] \quad \text{---(iii)}$$

$$t_1[\gamma] = t_2[\gamma] \quad \text{---(iv)}$$

By using eqn (i) and (iv), We can get

$$\alpha \to \gamma$$

(iv) Decomposition / Projectivity rule :-

Q// If $\alpha \to \beta\gamma$ is a FD then $\alpha \to \beta$ and $\alpha \to \gamma$ is FD or not.

Ans :- $\alpha \to \beta\gamma$ is given.

$\beta\gamma \to \beta$ (by using reflexive rule)

$\beta\gamma \to \gamma$ (by using reflexive rule)

$\alpha \to \beta\gamma$ , $\beta\gamma \to \beta$

then $\alpha \to \beta$ (by using transitive rule)

$\alpha \to \beta\gamma$ , $\beta\gamma \to \gamma$

then $\alpha \to \gamma$ (by using transitive rule)

Since, $\alpha \to \beta\gamma$ is a FD then $\alpha \to \beta$ then $\alpha \to \gamma$ is also a FD.

(v) Union Rule or (

Q// If $\alpha \to \beta$ and $\alpha \to \gamma$ is a FD then $\alpha \to \beta\gamma$ is a FD or not.

Proof :- $\alpha \to \beta$ is given.

$\alpha \to \gamma$ is also given.

$\alpha \to \beta \Rightarrow \alpha \to \alpha\beta$ (by using augmented rule)

we can add $\alpha$ in R.H.S

$\alpha \to \gamma \Rightarrow \alpha\beta \to \gamma\beta$ (by using augumentation rule we can add $\beta$ in both H.S)

$\alpha \to \alpha\beta$ , $\alpha\beta \to \gamma\beta$

We can find out

$\alpha \to \gamma\beta$

so that we can get $\alpha \to \beta$ and $\alpha \to \gamma$ is the FD then $\alpha \to \beta\gamma$ is also a FD.

## (v) Pseudo Transitivity Rule:-

// Q If $\alpha \to \beta$ is a FD then $\gamma\beta \to \sigma$ is a FD then $\alpha\gamma \to \sigma$ is a FD or not.

Proof :-

$\alpha \to \beta$ is given.

$\gamma\beta \to \sigma$ is given.

$\alpha \to \beta \Rightarrow \alpha\gamma \to \gamma\beta$ (by using augumentation rule we can add $\gamma$ in both L.H.S)

$\alpha\gamma \to \gamma\beta$ , $\gamma\beta \to \sigma \Rightarrow \alpha\gamma \to \sigma$

(by using transitive rule)

So that, $\alpha \to \beta$ is a FD and $\gamma\beta \to \sigma$ is also a FD the $\alpha \to \sigma$ is also a FD.

IMP

## Armstrong Axioms (Amstercom Exioms) :-

It is sound and complete. They are sound because it uses a given set of FD. They are complete because all FD should be implemented by using give set of FD.

// It can satisfied 3 rules :- (i) Reflexive Rule
(ii) Augumentation Rule
(iii) Transitive Rule

## Dependency Preservation :-

If R is decomposed into $R_1, R_2, \ldots, R_n$ then the decomposition is said to be dependency preservation and it is returned as

$$(F_1^+ \cup F_2^+ \cup \ldots F_n^+)$$

where F is known as functional dependency in R.

Q/1. Let R(A, B, C, D) be a rel$^n$al schema with a set of functional dependency.

$F = \{A \to B, B \to C, C \to D, D \to A\}$. Suppose R is decomposed into

$R_1(A, B), R_2(B, C), R_3(C, D)$

Determine whether the decomposition is dependency preservation or not.

Ans:- Given, R (A, B, C, D) ~~also~~

also given the Functional Dependency

$F = \{A \to B, B \to C, C \to D, D \to A\}$

The decomposition values are

$R_1(A, B), R_2(B, C), R_3(C, D)$

Now, the FD

$F_1 = \{A \to B, B \to A\}$

$F_2 = \{ B \to C , C \to B \}$

$F_3 = \{ C \to D , D \to C \}$

∴ The $D \to A$ can be obtained by using FD $F_1$, $F_2$ & $F_3$. Hence, this decomposition is dependency preservation.

Q.2. Let $R(A, B, C, D, E)$ be a relational schema with a set of FD

$F = \{ A \to B , B \to C , C \to D , D \to B \}$ suppose R is decomposed into ~~$R_1$, $R_2$~~,

$R_1(A, B)$, $R_2(B, C)$, $R_3(D, B)$. Determine whether the decomposition is dependency preservation or not.

Ans:- Given $R(A, B, C, D, E)$

also given FD

$F = \{ A \to B , B \to C , C \to D , D \to B \}$

The decomposition value are

$R_1(A, B)$, $R_2(B, C)$, $R_3(D, B)$

Now the FD

$F_1 = \{ A \to B , B \to A \}$

$F_2 = \{ B \to C , C \to B \}$

$F_3 = \{D \to B, B \to D\}$

∴ ~~The~~ The $C \to D$ can be obtained by using FD $F_1, F_2, F_3$. Hence this decomposition is dependency preservation.

IMP **Normalization / Normal Forms :-**

It is the process to reduce the redundency by using functional dependency and key attributes.

OR

Normalization of data is a process of analyzing the given relational schema based on their FD and primary key to achiere the desire property.

// If we are not using the normalization it creates 3 types of problems :- (i) insertion
(ii) modification
(iii) deletion

// These are of 6 types :- (i) 1st Normal form (1NF)
(ii) 2nd " " (2NF)
(iii) 3rd " " (3NF)
(iv) Bg Cut NF (BCNF)
(v) 4th Normal Form (4NF)
(vi) 5th " " (5NF)

## (i) 1st Normal Form :-

A relational schema R is said to be 1NF if the domain of all attributes of R must be atomic.

* In 1NF, there shouldn't be a repeating attributes.

Student

| Regd.no | Subject |
|---|---|
| 1 | C, C++, DS |
| 2 | OS – DBMS |
| 3 | Math |

not present in 1NF

convert it into 1NF

| Regd.no | Subject |
|---|---|
| 1 | C |
| 1 | C++ |
| 1 | DS |
| 2 | OS |
| 2 | DBMS |
| 3 | Math |

* 1NF is mandatory where as other normal forms are optional.

## (ii) 2nd Normal form :-

The rel$^n$ is in 2nd Normal form if and only if it satisfy 1st Normal form and every non-key attributes are fully functionally depends on the whole key.

In 2NF, there is no repeating attributes are present.

non-key

| key Id no. | name | address | Course no. | course name | Grade |
|---|---|---|---|---|---|

In 2NF, we check all the non-key attributes against each part of the key to ensure their functionally depend upon it.

// Id no. determines student name and address but not course name.

// Course no. determines course name but not student name and address. So we can split the tables.

// If we combine Id no. and course no. then we can easily calculate the grade.

| Id no. | name | address |
|---|---|---|
| | | |

| C. no | C. name |
|---|---|
| | |

| Id no. | C. no. | Grade |
|---|---|---|
| | | |

After split the rel$^n$, all of the attributes in each of the rel$^n$ will be fully depend upon the primary key.

## (iii) 3rd Normal Form :-

A relation is said to be 3rd NF if and only if it satisfies the 2NF and there should not any transitive dependency on the primary key.

// According to this form, the basic technique is to eliminate the transitive dependency.

// To eliminate it $A \longrightarrow B \longrightarrow C$ will need to break the part $B \longrightarrow C$.

// The rel$^n$ is decomposed into 2 tables when (i) B, (ii) C and it is returned as

$R_1 (A, B)$ $\{A \rightarrow B\}$

$R_2 (A, C)$ $\{A \rightarrow C\}$

In 3rd NF we can find out the primary key.

Q// 1. Consider the following rel$^n$ $R(A, B, C, D, E, F)$ with a set of FD, $F = \{A \rightarrow C, B \rightarrow D, AB \rightarrow E, C \rightarrow F, be \rightarrow f\}$ Represent it into the 3rd NF.

Primary Key = A, B i.e A, B or AB

| | | |
|---|---|---|
| $R_1 (AB, E, F)$ | $f_1 (AB \rightarrow E \rightarrow F)$ | 2NF |
| $R_2 (A, C, F)$ | $F_2 (A \rightarrow C \rightarrow F)$ | 2NF |
| $R_3 (B, D)$ | $F_3 (B \rightarrow D)$ | 3NF |

We have to convert the considering $R_1$

Decomposition $R_{11}$ (AB, E) $F_{11}$ {AB → E} 3NF

$R_{12}$ (AB, F) $F_{12}$ {AB → F} 3NF

considering $R_2$ :- A → C → F

Decomposition

$R_{21}$ (A, C) $F_{21}$ {A → C} 3NF

$R_{22}$ (A, F) $F_{22}$ {A → F} 3NF

Q.2. Consider the following relation R(A, B, C, D, E, F) with a set of FD, F = {A → C, B → D, A → E, C → F, E → F}. Represent it into 3NF.

Primary Key = AB i.e A, B

$R_1$ (A, C, F) $f_1$ (A → C → F) 2NF

$R_2$ (B, D) $f_2$ (B → D) 3NF

$R_3$ (A, E, F) $f_3$ (A → E → F) 2NF

We have to convert the $R_1$ into 3NF

considering $R_1$ : A → C → F

$R_{11}$ (A, C) $f_{11}$ (A → C) 3NF

$R_{12}$ (A, F) $f_{12}$ (A → F) 3NF

$R_3$ : A → E → F

$R_{31}$ (A, E) $f_{31}$ (A → E) 3NF

$R_{32}$ (A, F) $F_{32}$ (A → E) 3NF

(iv) Boyes codd Normal Form : ( BCNF)

The basic problem in 3NF is overlapping of the candidate key.

→ If two candidate key like AB and BC are available, with overlapping. person B then the dependency B → D is derived.

→ To avoid this type of problem BCNF is ~~derived~~ created.

→ A relations is said to be BCNF if every function dependency in the form, $x \rightarrow A$ on, ~~a~~ each and every x it either a candidate key on a superkey.

(Q.) Consider the following relation R(A, B, C, D, E, F) with the set of functional dependency { A → C, B → D, AB → E, C → E, D → F }. represent it into BCNF.

On the above relation the candidate keys are

CK : { AB }.

(i) FOR A → C

Here A is not a candidate key on superkey so it needs to be decomposed.

$R_1$ ( A, C )

$R_2$ (A, B, D, E, F)

(ii) FOR B → D

B is not a candidate key on superkey so.

$R_3$ ( B, D ) $R_4$ ( A, B, E, F)

For $C \rightarrow E$ :

C is also not a candidate or ~~primary~~ super key i.e.

$R_5 (C, E)$

$R_6 (A B, F)$

For $D \rightarrow F$ :

D is also not a candidate or super key

$R_7 (D, F)$

$R_8 (A B)$

* Final Relation in BCNF is

$R_1 (A, C)$ $R_8 (A, B, E)$

$R_3 (B, D)$

$R_5 (D, F)$

$R_4 (C, E)$

## 4NF :-

A relation is said to be 4NF (4th Normal form) if every multiple value dependency must be in the form of $\alpha \twoheadrightarrow A$

where, $\alpha$ is superkey and A is non-key attribute.

The superkey is denoted as S.

Q/ Consider the following relation are contain the value of A, B, C, D, E, F, with the set of dependency $A \twoheadrightarrow C$, $B \twoheadrightarrow D$, $C \twoheadrightarrow E$, $D \twoheadrightarrow F$ and $D \twoheadrightarrow EF$ and convert into 4NF.

Ans:- Superkey = ~~A, B~~ AB

A, B = S

$A \twoheadrightarrow C \Rightarrow S \twoheadrightarrow C$

$B \twoheadrightarrow D \Rightarrow S \twoheadrightarrow D$

$C \twoheadrightarrow DE \Rightarrow E \twoheadrightarrow F$

$D \twoheadrightarrow EF$

$\alpha \twoheadrightarrow A$

$R_1$ (S, C) $f_1$ ($S \twoheadrightarrow C$) 4NF

$R_2$ (S, D) $f_2$ ($S \twoheadrightarrow D$) 4NF

$R_3$ (C, D, E) $f_3$ ($C \twoheadrightarrow DE$) 4NF

$R_4$ (E, F) $F_4$ ($E \twoheadrightarrow F$)

$R_5$ (D, E, F) $F_5$ ($D \twoheadrightarrow EF$)

$R_1$ & $R_3$ = (S, D, E) $F_6$ $S \twoheadrightarrow DE$

$R_2$ & $R_5$ = (S, E, F) $F_7$ ($S \twoheadrightarrow EF$)

$R_4$ & $R_5$ = (S, F) $f_8$ ($S \twoheadrightarrow F$)

## 5th NF :-

It is a NF in which we use the join dependency. It also provides loss less joint decomposition.

Acc. to the def$^n$, a rel$^n$ is in 5th NF if every joint dependency can be taken on the superkey.

Ex:- $R_1$ (AB, C)

$R_2$ (BC, D)

$R_3$ (ABC) = { $\overline{AB}$ & $\overline{BC}$ }

Here, $R_3$ contain the superkey i.e ABC.

and it has 2 projection AB and BC. AB and BC joined together, they form the original set i.e ABC is a lossless joint decomposition.

## SQL :-

- SQL stands for structured Query Language.
- It is an international standard language for manipulating Relational DBMS.
- It is based on IBM (International Business ~~Con~~ Machine).
- SQL can create the table, delete them and change them.
- It is the combination of data definition and data manipulation language.

## SQL Basics

There are 6 types of basics are present i.e:-

(i) Create
(ii) insert
(iii) select
(iv) update
(v) delete
(vi) drop

Q How to create table ?

Syntax :-

```
Create table tablename (
        attributename datatype (size),
            "           "      (size)
        );
```

Create the table and the name of the table is student and attribute (name, address, city, roll no)

```
create table student (
        name varchar (40),
        address varchar (40),
        city varchar (40),
        roll_no int
    );
```

To see the structure of the table.

```
desc student ;
```

## Difference betn BCNF & 3NF

| BCNF | 3NF |
|---|---|
| // In BCNF, we are using superkey concept. | // IN 3NF, we are using primary key concept. |
| // It is more strict than 3NF. | // It is less strict than BCNF. |
| // Redundency is low as compared to 3NF. | // Redundency is high as compared to BCNF. |
| // It doesn't use over-lapping concept. | // It uses overlapping concept. |
| // All BCNF are 3NF. | // All 3NF aren't BCNF. |

## Modify the table structure.

It uses the alter command. The alter command is used to add or modify the table structure.

SYNTAX:-

alter table tablename (add/modify) attributename datatype (size);

To add a new attribute i.e branch into the student table.

SYNTAX :-

```
alter table student add
branch varchar(40);
```

## Update :-

SYNTAX :-

```
update table tablename set attributname=
                                "value"
where attributename= value ;
```

Ex :-

```
update table student set branch='CSE' where
name = 'A';
```

## Rename :-

In rename operation, it can rename the table as well as rename the column

SYNTAX :-

```
rename oldtablename to newtablename;
```

or

```
alter table rename oldtablename to
newtablename ;
```

Ex:- rename student to student_info;

alter table rename student to student_info;

For rename the column :-

SYNTAX

alter table tablename rename oldcolumnname to newcolumnname datatype (size);

Ex:-

alter table student rename name to student_name varchar(40);

Drop:-

In drop command, it can drop the table structure as well as the information the table contain.

SYNTAX:-

drop table tablename;

To drop student table.

drop table student;

Delete:-

The delete command is used to delete a specific row in a table.

SYNTAX:-

delete table tablename where attributename = value;

Ex:-

delete table student where ph_no = 1;

## Aggregate Function:- (imp)

The aggregate function are of 5 types :-

(i) average function
(ii) sum function
(iii) max function
(iv) min function
(v) count fun$^n$.

### (i) avg() :-

SYNTAX:-

select avg(attributename) from tablename;

### (ii) sum() :-

SYNTAX:-

select sum(attributename) from tablename;

### (iii) max() :-

select max(attributename) from tablename;

(iv) min() :-

select min(attributename) from tablename;

(v) count() :-

select count(attributename) from tablename;

Q/1. To create a table college and the attributes are C_name, C_address, C_phno, city and price. After creating the table insert 5 information to the table and retrive the table. After retrieving we have to add another attribute i.e location in given table then update the location.

NOT NULL :-

- By default, a column can hold a null value.
- In table, when we mention NOT NULL constraint we must providing the values.
- If we doesn't ~~ess~~ assign the value the record is not accepted.

SYNTAX :-

```
create table student (
    id int NOT NULL,
```

```
name    varchar2(20),
age int
);
```

## Unique :-

- It provides the unique value.
- If We are using multiple uniques in a single table.

SYNTAX :-

```
create table tablename (
    id int NOT NULL UNIQUE,
    name varchar(20) UNIQUE,
    age int
    );
```

## Primary Key :-

- In a table, we are using one primary key.
- It always gives unique value.
- If we provided duplicate value, it doesn't accept it.

SYNTAX :-

```
create table tablename
(
  id int,
  name varchar(10),
  city varchar2(20),
  age int,
  primary key (id)
);
```

or

```
create table tablename
(
  id int primary key,
  name varchar(10),
  city varchar2(20),
  age int,
);
```

## Foreign Key :-

// In foreign key, it required more than one table.

The first table primary key is enter into the second table that attribute uses the foreign key, then it provides the references of the table.

1st table

Person

| Pid | Pname | location |
|---|---|---|

P.key

2nd table

Orderc

| Oid | Oname | Price | Pid |
|---|---|---|---|

P.key

F.key

SYNTAX :-

```
create table orderc
(
  Oid int primary key,
  Oname varchar2(20),
  Price decimal(8,2),
  foreign key (Pid) Reference person(Pid)
);
```

## Auto increement :-

SYNTAX :-

```
create table orderc
(
  Oid int AUTO_INCREMENT,
  Oname varchar2(20),
  price decimal(8,2)
);
```

## Default constraint :-

// The default constraint is used to set a default value for all.

// The value will be added to all new records if no other value are specified.

SYNTAX :-

```
create table ~~tablename~~ orders
(
  Oid int,
  city varchar(200) DEFAULT 'BBSR',
  price decimal(8,2)
);
```

## Check constraint :-

// It is used to limit the range value that can be placed in a column.

SYNTAX :-

```
create table person
(
  id int NOT NULL,
  name varchar2(20) UNIQUE,
  age int,
  check (age>=18)
);
```

## Order By :-

It is used to sort the result set in ascending or descending order.

SYNTAX :-

```
select * from tablename order by
attributename ;
```

ex :-

```
select * from person order by price ;
```

IMP

## Decomposition :-

It is a process in which a relation is divided into no. of smaller relation.

// It is classified into 3 types :-

(i) Loss-less Joint Decomposition

(ii) Lossy Join Decomposition

(iii) Dependency ~~Joint~~ preserving decomposition

### (i) Lossless Join Demposition :-

// In loss-less joint decomposition the relation is decomposed into no. of smaller relations.

// Then perform the join operation on the smaller relation to obtain the resultant

table.

// When we compare the resultant table with the initial table if both the table are same then we can say the relation is loss-less joint decomposition.

R

| A | B | C |
|---|---|---|
| a | 1 | 10 |
| b | 2 | 20 |
| c | 3 | 30 |

$R_1$

| A | B |
|---|---|
| a | 1 |
| b | 2 |
| c | 3 |

$R_2$

| B | C |
|---|---|
| 1 | 10 |
| 2 | 20 |
| 3 | 30 |

$R_1 \bowtie R_2$

$R_{12}$

| A | B | C |
|---|---|---|
| a | 1 | 10 |
| b | 2 | 20 |
| c | 3 | 30 |

(ii) Lossy Join Decomposition:-

In this process there is some loss of data in the resultant table after the join operation i.e the content of resultant table and original table is not same.

R

| A | B | C |
|---|---|---|
| a | 1 | 10 |
| b | 2 | 20 |
| c | 3 | 30 |
| a | 4 | 40 |

$R_1$

| A | B |
|---|---|
| a | 1 |
| b | 2 |
| c | 3 |

$R_2$

| B | C |
|---|---|
| 1 | 10 |
| 2 | 20 |
| 4 | 40 |

$R_{12}$

| A | B | C |
|---|---|---|
| a | 1 | 10 |
| b | 2 | 20 |

(iii) Dependency Preserving Decomposition:-

Let F be the funtional dependency be the orriginal. If the FD in the original table is equal to the union of set of the dependency on the smaller table due to the decomposition process so it is known as dependency preserving ~~decomp~~ decomposition.

Mathematically,

$$F = F_1 \cup F_2$$

Where,

$F_1$ and $F_2$ are the dependency present in the relation $R_1$ and $R_2$

Q. Decompose the relation R can contain the value R(A, B, C, D, E), $R_1$(A, B, C) & $R_2$(A, D, E) having FD $F = \{A \to BC, CD \to E, B \to D, E \to A\}$. Show that the above decomposition is loss-less or lossy joint decomposition.

ns:- Decomposition Matrix

~~step-1~~

step-1

| | A | B | C | D | E |
|---|---|---|---|---|---|
| $R_1$ | A | B | C | | |
| $R_2$ | A | | | D | E |

step 2

Consider the function depending $A \rightarrow BC$

| | A | B | C | D | E |
|---|---|---|---|---|---|
| $R_1$ | A | B | C | | |
| $R_2$ | A | (B) | (C) | D | E |

Step-3

Consider the function depending $CD \rightarrow E$

| | A | B | C | D | E |
|---|---|---|---|---|---|
| $R_1$ | A | B | C | | (E) |
| $R_2$ | A | B | C | D | E |

Step-4

Consider the function depending $B \rightarrow D$

| | A | B | C | D | E |
|---|---|---|---|---|---|
| $R_1$ | A | B | C | (D) | E |
| $R_2$ | A | B | C | D | E |

∴ Hence there is no empty set in the matrix So it is lossless join decomposition.

Q// Decompose the relation R (A, B, C, D, E), $R_1$ (A, B, C), $R_2$ (C, D, A) having the functional dependency $A \rightarrow BC$, $CD \rightarrow E$,

$B \rightarrow D$ and $A \rightarrow E \rightarrow A$ by using the decomposition matrix to check the decomposition is loss-less or lossy join decomposition.

| | A | B | C | D | E |
|---|---|---|---|---|---|
| $R_1$ | A | B | C | | |
| $R_2$ | A | | C | D | |

$A \rightarrow BC$

Step-2

| | A | B | C | D | E |
|---|---|---|---|---|---|
| $R_1$ | A | B | C | | |
| $R_2$ | A | (B) | C | D | |

Step-3 $CD \rightarrow E$

| | A | B | C | D | E |
|---|---|---|---|---|---|
| $R_1$ | A | B | C | | (E) |
| $R_2$ | A | B | C | D | |

Step-4 $B \rightarrow D$

| | A | B | C | D | E |
|---|---|---|---|---|---|
| $R_1$ | A | B | C | (D) | E |
| $R_2$ | A | B | C | D | |

Step-5 $A \rightarrow E$

| | A | B | C | D | E |
|---|---|---|---|---|---|
| $R_1$ | A | B | C | D | |
| $R_2$ | A | B | C | D | (E) |

$B \to D$ and $A \to E \to A$ by using the decomposition matrix to check the decomposition is loss-less or lossy join decomposition.

| | A | B | C | D | E |
|---|---|---|---|---|---|
| $R_1$ | A | B | C | | |
| $R_2$ | A | | C | D | |

$A \to BC$

Step-2

| | A | B | C | D | E |
|---|---|---|---|---|---|
| $R_1$ | A | B | C | | |
| $R_2$ | A | (B) | C | D | |

Step-3 $CD \to E$

| | A | B | C | D | E |
|---|---|---|---|---|---|
| $R_1$ | A | B | C | | (E) |
| $R_2$ | A | B | C | D | |

Step-4 $B \to D$

| | A | B | C | D | E |
|---|---|---|---|---|---|
| $R_1$ | A | B | C | (D) | E |
| $R_2$ | A | B | C | D | |

Step-5 $A \to E$

| | A | B | C | D | E |
|---|---|---|---|---|---|
| $R_1$ | A | B | C | D | |
| $R_2$ | A | B | C | D | (E) |

Hence there is a 'empty set in the matrix, so it is lossy joint decomposition.

Q// What is Synthesis?

// Synthesis is the reverse decomposition.

// It is the process in which no. of smaller relation is combine to form large relation.