Unit I : Introduction

Database

- A repository for collection of related data and facts.
- is an organized collection of data so that it can be easily accessed and managed.

Data and Information

→ Data is a raw, unorganized facts that need to be processed.

→ when data is processed, organized, structured or presented in given content so as to make it useful, it is called information.

DBMS

→ A collection of programs that enables users to perform certain actions on a particular database

- 'Define' the structure of db information.
- 'populate' the database with appropriate information.
- 'Manipulate the database.
- 'Protect' the db contents against ~~accien~~ accidential or deliberate corruption

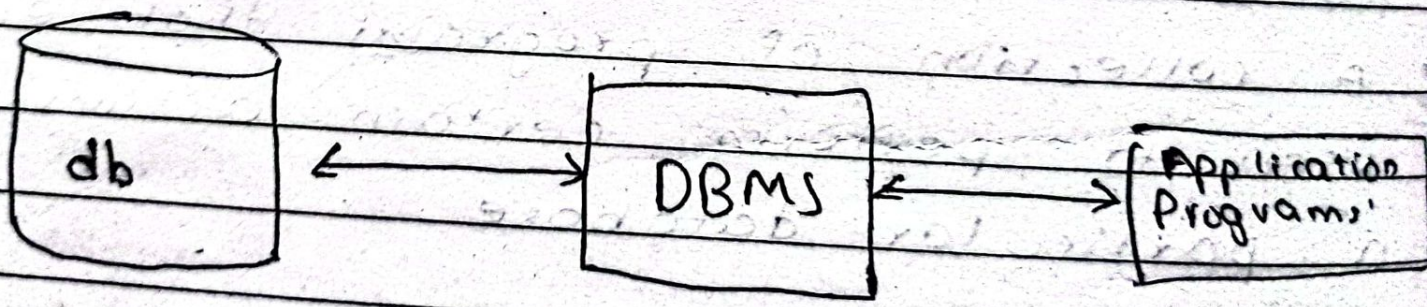
of contents

- 'share' the db among multiple users.

→ Is a software that defines a database, store data, supports query language, produce reports and create data entry ~~tect~~ screens.

→ Provides an ~~otter~~ environment that is both convenient and efficient to use in storing and retrieving db information.

→ e.g. ORACLE, MYSQL, SQL SERVER.



Applications of DBMS

- Banking
- Airlines
- Human Resource
- Education
- Hospitals
- E-commerce
- Sales

Objectives of DBMS

- Providing mass storage of relevant data.
- Making easy access to data for authorized users.
- Providing prompt response to user's request for data.
- Making latest modification visible for all database users immediately.
- Eliminating data redundancy.
- Allowing multiple users to be active at one time.
- ~~Allowing multiple users to be~~
- Allowing the growth of database system
- Providing data integrity
- Protecting data from physical harm and unauthorized access.
- Serving diff. types of users
- Providing security with user access privilege

→ Combining inter related data to generate reports.

Advantages of DBMS

→ Data sharing
→ Reduced data redundancy
→ Improved data integrity
→ Increased security
→ Time saving
→ Report generation.

Disadvantages of DBMS

→ Costly
→ Fast changing technology
→ Requires trained manpower
→ Chance of data leakage and hacking.

Drawbacks of File System

- Redundancy
- Inconsistency
- Maintenance problem
- Security
- ~~Different~~ Difficulty in combining data.

7th Bhadra

Data Abstraction

→ It hides details of data storage that are not needed by most db users and applications

→ ensures easy, smooth and efficient data structures in such a way that every type of db user is able to access its desired information efficiently.

→ Types of data abstraction,

i) Internal / Physical level

→ is the lowest level of data abstractio that is close to physical storage

→ is describes
- how data are actually stored in storage device?
- what will be storage techniques?

ii) Conceptual / Logical level

→ Is the next higher level.

→ it describes
- what data are actually stored in db?

- what are relationship between data entities?

iii) External / view level

→ is close to users

→ it describes what is the way of viewing information to concerned us

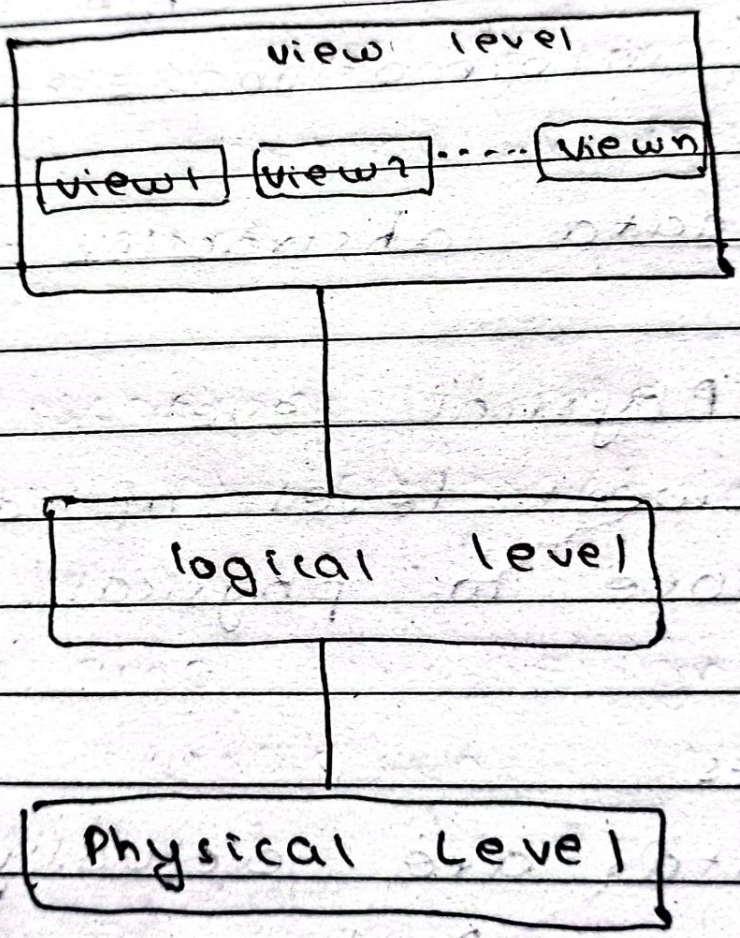


fig: three levels of data abstraction.

Date
Page

Data Independence

is the ability to modify a schema definition in one level without affecting a schema definition in next high level

Each higher level of data architecture is immuned to the changes of next lower level of architecture.

we can change the structure of a database without affecting data required by users or programs.

Two types of data independence.

Physical data independence:
refers to ability to modify the physical schema without affecting conceptual schema

Logical data independence:
refers to ability to modify conceptual schema without affecting view schema

it is more difficult to achieve than

physical data independence because application programs are dependent on logical structure of data that user access.

Schema and Instance

→ schema is overall design of the database.

→ The collection of information stored in db at particular moment is called instance.

→ Physical schema is design at physical level of abstraction.

→ Logical schema is design at logical level of abstraction.

→ Subschema is schema at new level

e.g. employee(eid, name, job, salary) is schema.

eid	name	job	salary
1	Ram	Manager	75,000
2	Sita	Designer	45,000
3	Hari	Analyst	60,000

Concept of DDL, DML and DCL

① DDL (Data Definition Language)
→ used to define database strict structure or schema
→ commands are:
- CREATE
- DROP
- ALTER
- TRUNCATE

② DML (Data Manipulation Language)
→ used for managing data within schema object
→ commands are:
- INSERT
- DELETE
- UPDATE
- SELECT → DQL (Data Query or DRL (Data Retrieval

③ DCL (Data control language)
→ used to control the database
→ commands are:
- GRANT
- REVOKE

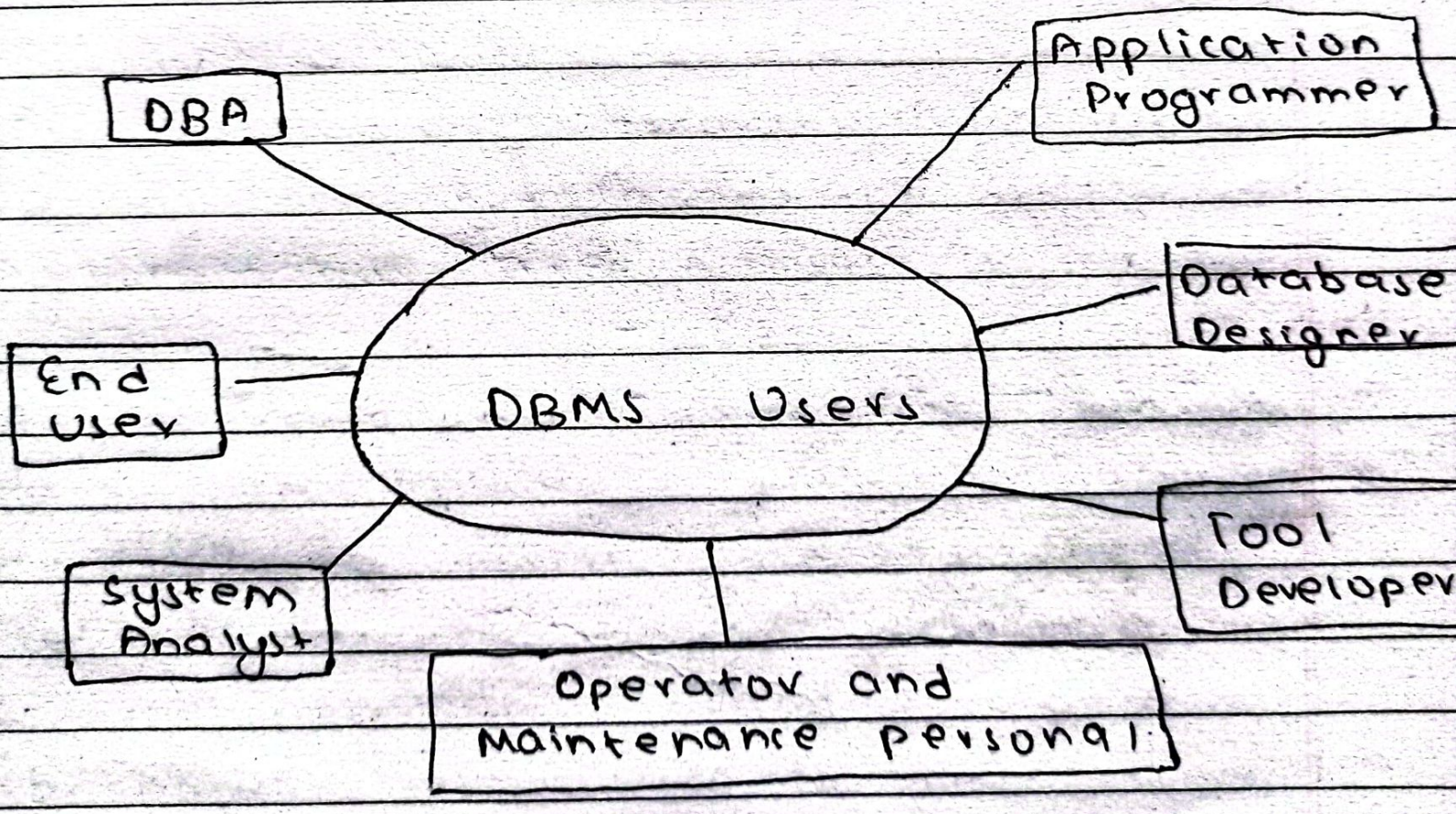
4) TCL (Transaction Control language)
→ used to manage the changes made by DML statements.
→ commands are:
- COMMIT
- ROLLBACK

Database Users:
① DBA (Database Administrator)
② Application Programmer
③ End User (clients)

DBA:
→ creates or modify database schema uses DDL.
→ responsible for providing security to the db and allows only the authorized users to access/ modify db.
→ monitors the recovery and backup and provide technical support.

Application Programmer
interact with database system through DML.

End User (Client)
Invoke one of the permanent application programs that have been written previously.



Unit 2: ER Model and Relational Model

Data Model

is a collection of conceptual tools for describing data, data relationships, semantics, etc

is a simple & representation of complex real world data structure.

is a communication tool to facilitate among the designer, application programmer and end users.

defines how data is connected to each and how it will be processed and stored inside the system.

3 level of data modeling
Conceptual Data Model
Logical Data Model
Physical Data Model

1) Conceptual Data Model

→ identifies the highest level relationship between different entities.

→ includes important entities and their relationships.

→ no attributes and primary key is specified.

ii) Logical Data Model

→ describes the data in as much detail as possible - without saying much about their physical implementation.

→ includes all entities and rel^ship a them.

→ all attributes for each entity are specified.

→ primary key and foreign key are a specified

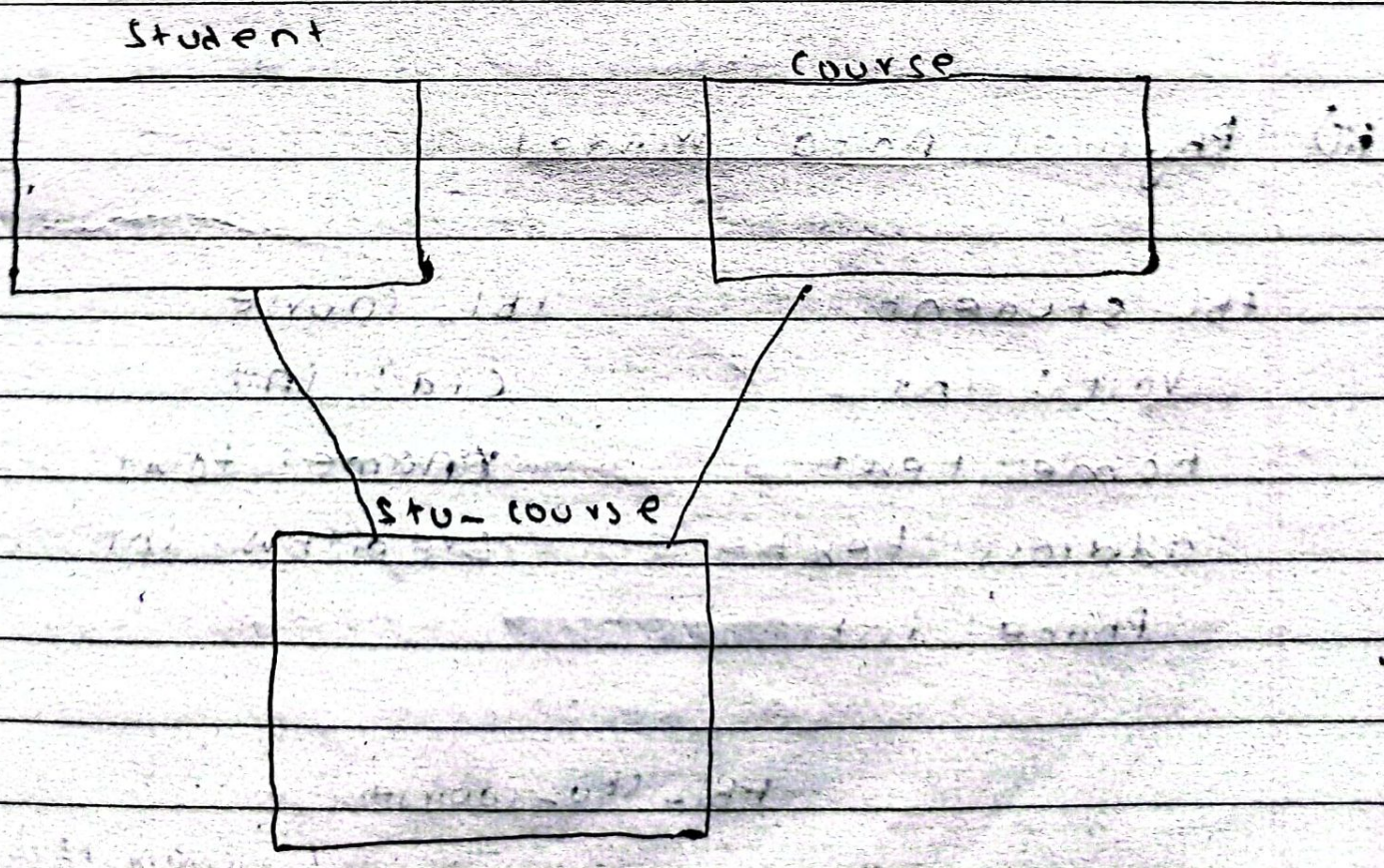
iii) Physical Data Model

→ represents how the model will be built in db.

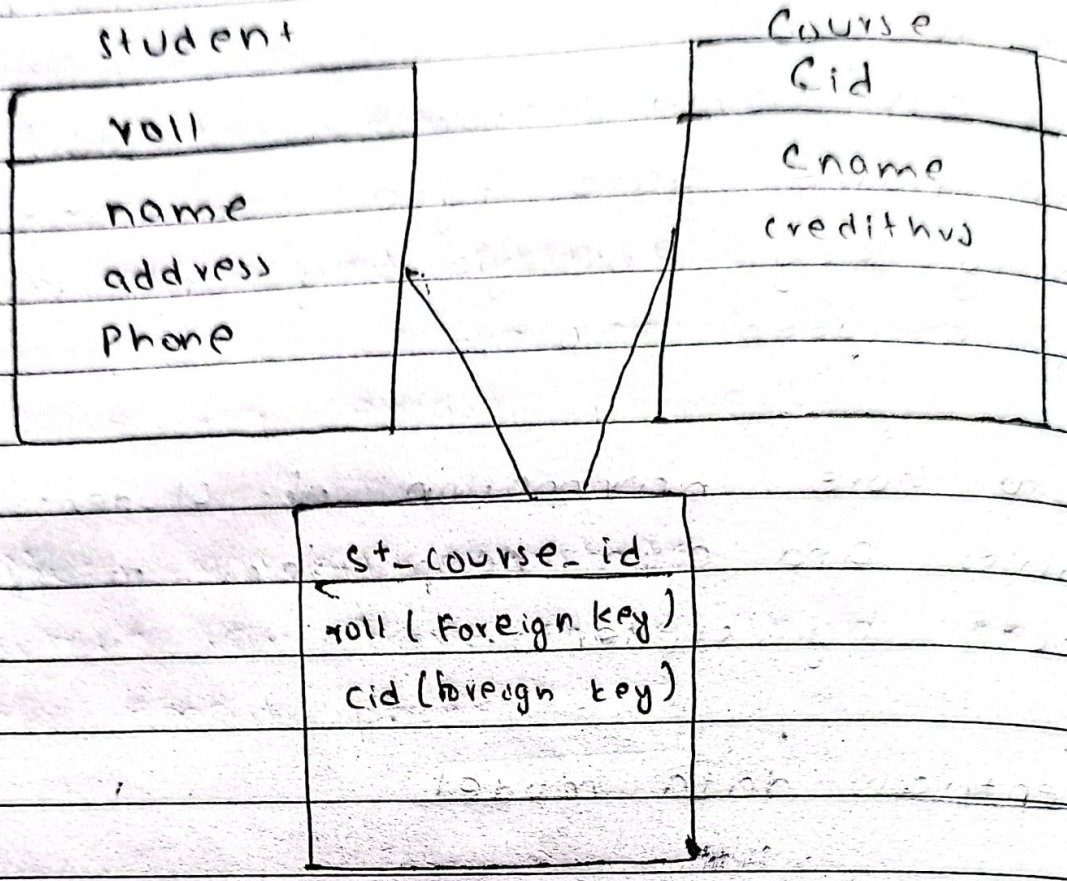
→ shows all table structures, including column name, data type constraints, primary key, foreign key and rel^n-ship between tables.

e.g. to store information of students, courses and also keep track of courses taken by students.

i) Conceptual data model



ii) Logical Data Model



iii) Physical Data Model

tbl_student
roll : int
name : text
address : text
Phone : int

tbl_course
Cid : int
Cname : text
creditHrs : int

tbl_stu_course
stu_courseid : int (primary key)
roll : int (foreign key)
Cid : int (foreign key)

ER (Entity Relationship) Model

→ defines the conceptual view of database.
→ work around real world objects called entities and relationship among them.

Entity

→ a real world thing that is easily identifiable and distinguishable
→ entity set is a collection of similar types of entities.
→ entity sets do not need to be disjoint. e.g. entity set of employee and customer may have common members.

Attributes

→ are properties of entities.

a) Simple attribute
- which cannot be divided further
e.g. LicenceNo

b) Composite attribute
- made of more than one simple attributes
eg. name

c) Derived attribute:
- do not exist in physical database but value is derived from other attributes. e.g. age

d) Single valued attribute
- which can have ~~more than~~ only one value e.g. ~~email, phoneNo~~ citizenship No

e) Multi valued attribute
- which can have more than one value e.g. email, phone.

⇒ Domain of attribute is set of permitted values.
eg. mobileNo; must have 10 digits.

Relationship
→ is association among the entities.
→ Degree of relationship refers the ~~multiple~~ number of entities that participate in relationship.

→ Unary Relationship is a degree 1 that involves only one entity
→ Binary relationship is of degree 2.
→ Ternary relationship is of degree 3.
→ Relationships in database are often binary.
→ It is possible to replace a non binary (n-nary, n>2) relationship ser by number of distinct binary relationship.

ASSIGNMENT 1

1) What is database? Explain different types of databases.

2) To store data related to your system, among file base system and database management system which one do you prefer and why?

3) Explain different types of database architecture.

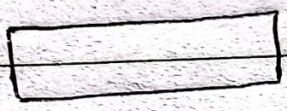
7/25 Sunday

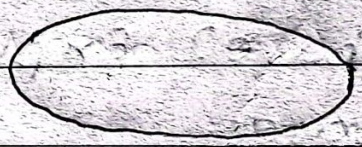
ER Diagram

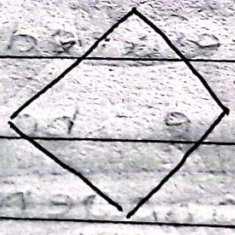
- is a data modeling technique that creates a graphical representation of entities and the relationship betw entities, within an information syste

→ it illustrates the logical structure of db.

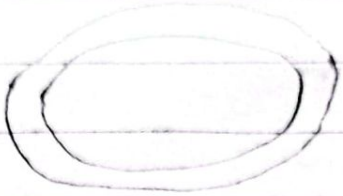
→ Major components are.

i) Rectangle  ⇒ to represent entity set

ii) ellipse  ⇒ to represent attributes.

iii) Diamonds  ⇒ to represent relationship.

iv) Lines ——————— → to link attri to entity and entity set to relationship

v) Double Ellipse  => to represent multi-valued attribute.

vi) Dashed Ellipse  => to represent derived attribute

vii) Double Rectangle.  => to represent weak entity.

eg.

Entity:-

i) Driver

ii) Tani

Attributes for:

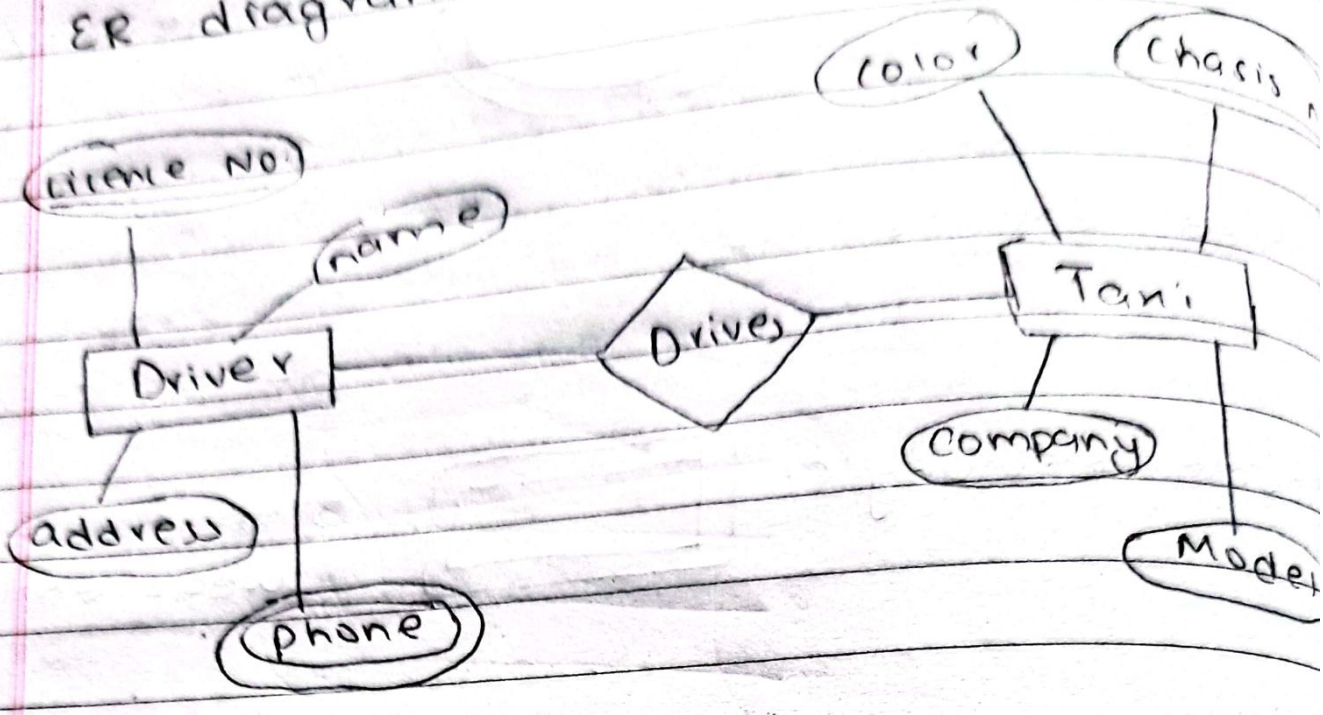
driver
- Licence no
- name
- address
- phone

tani
- Chasis n
- company
- model
- color.

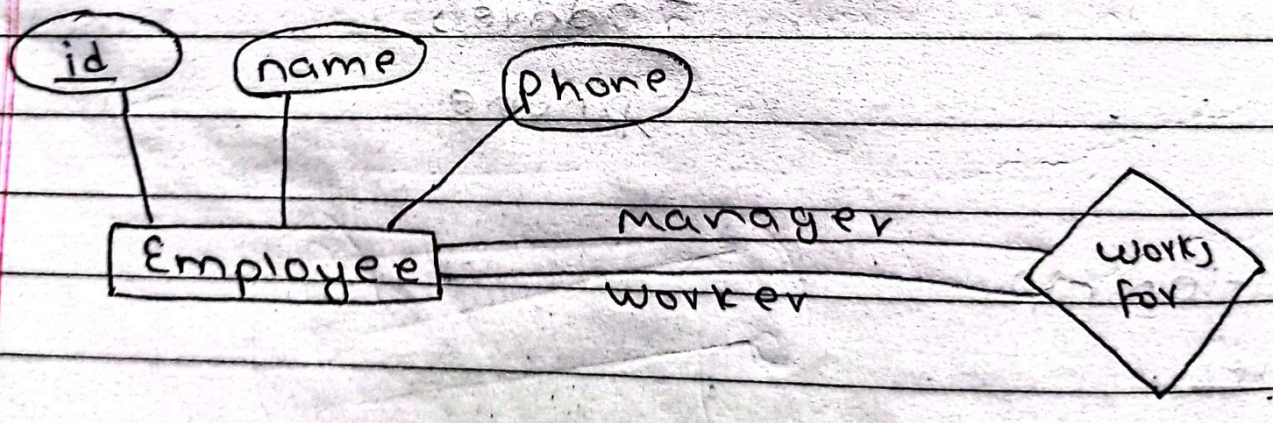
Relationship

- drives

ER diagram:



→ The role of an entity is the fun it plays in relationship



- Here, 'Manager' and 'Worker' are roles of entity 'employee' in 'works for' relationship

- roles are optional and clarify semantics of relationship.

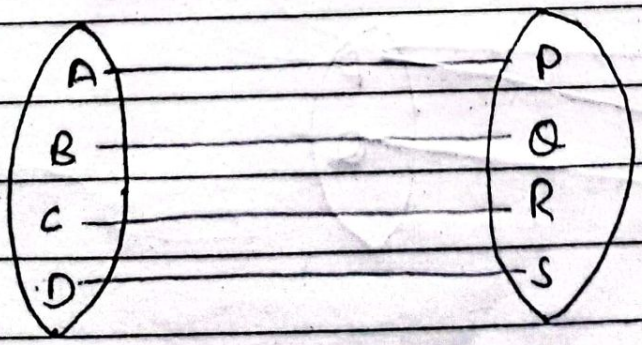
Mapping Cardinalities

→ express the number of entities to which another entity can be associated through a relationship set.

→ For binary relationship set R bet^{n} entity set A and B, the mapping cardinality must be one of the following.

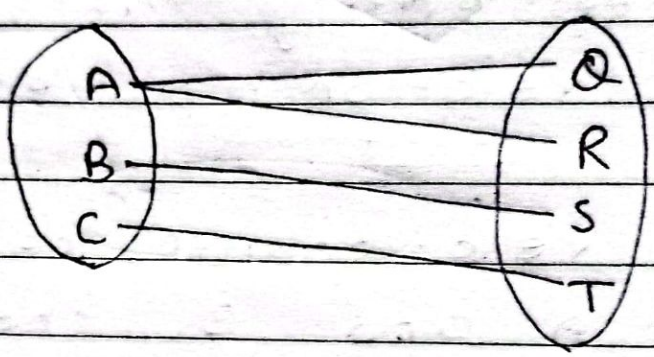
i) One to One

→ An entity in A is associated with at most one entity in B and vice versa.



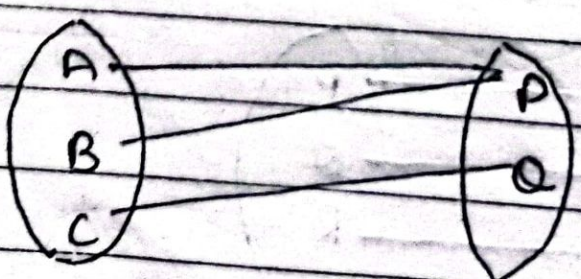
ii) One to many

→ An entity in A can be asso-ciated with more than one entity in B but from entity B one entity can be associated with at most one entity in A.



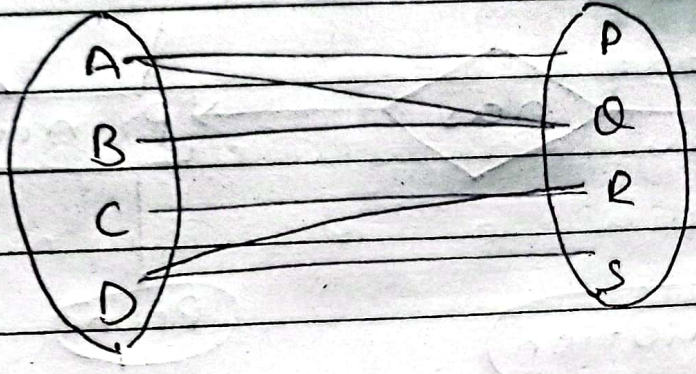
iii) ~~One To~~ more than many to one

→ One entity in A can be associ-ted with at most one entity of B but from B- one entity can be associated with more than one in A.



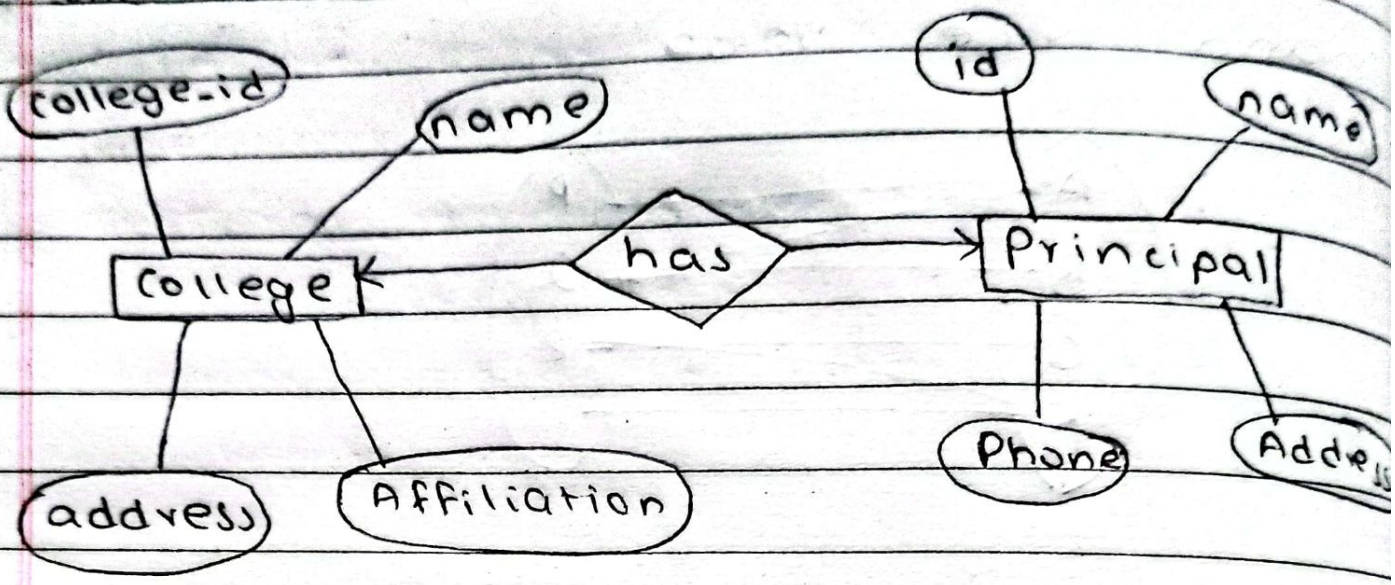
Many to many

→ One entity in A can be associated with more than one entity in B and vice - versa.

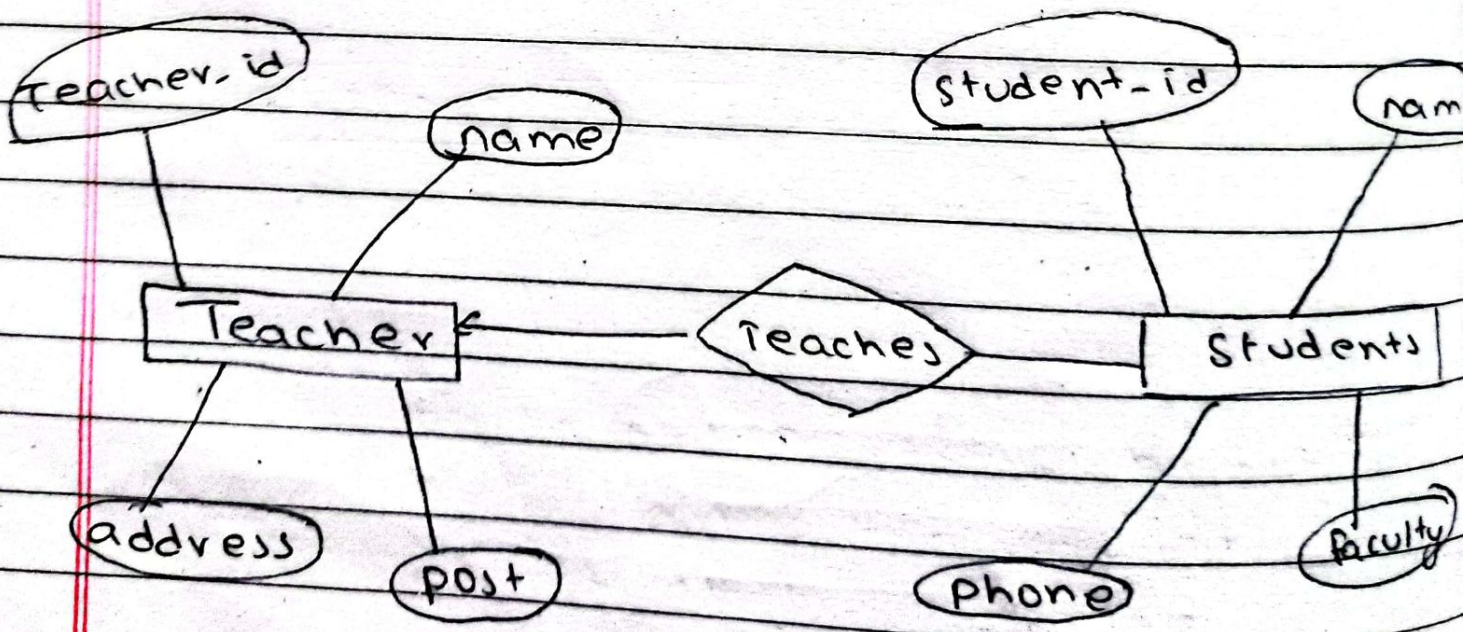


Types of Binary Relationship

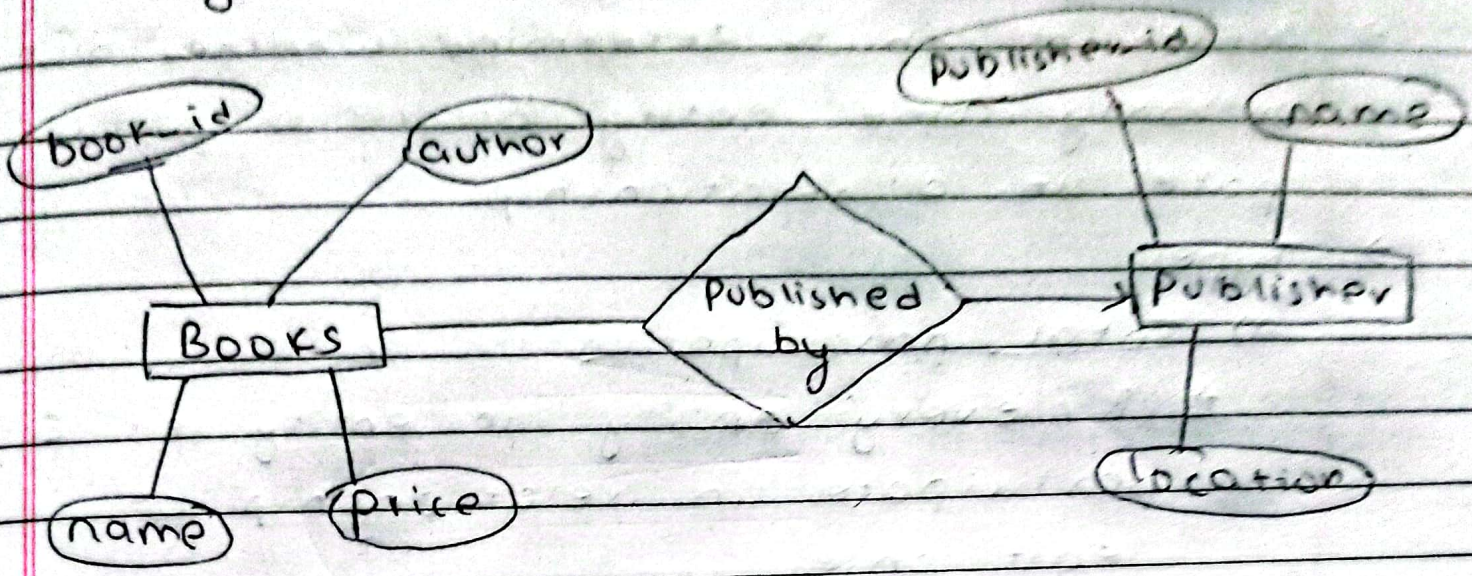
i) One to One Relationship



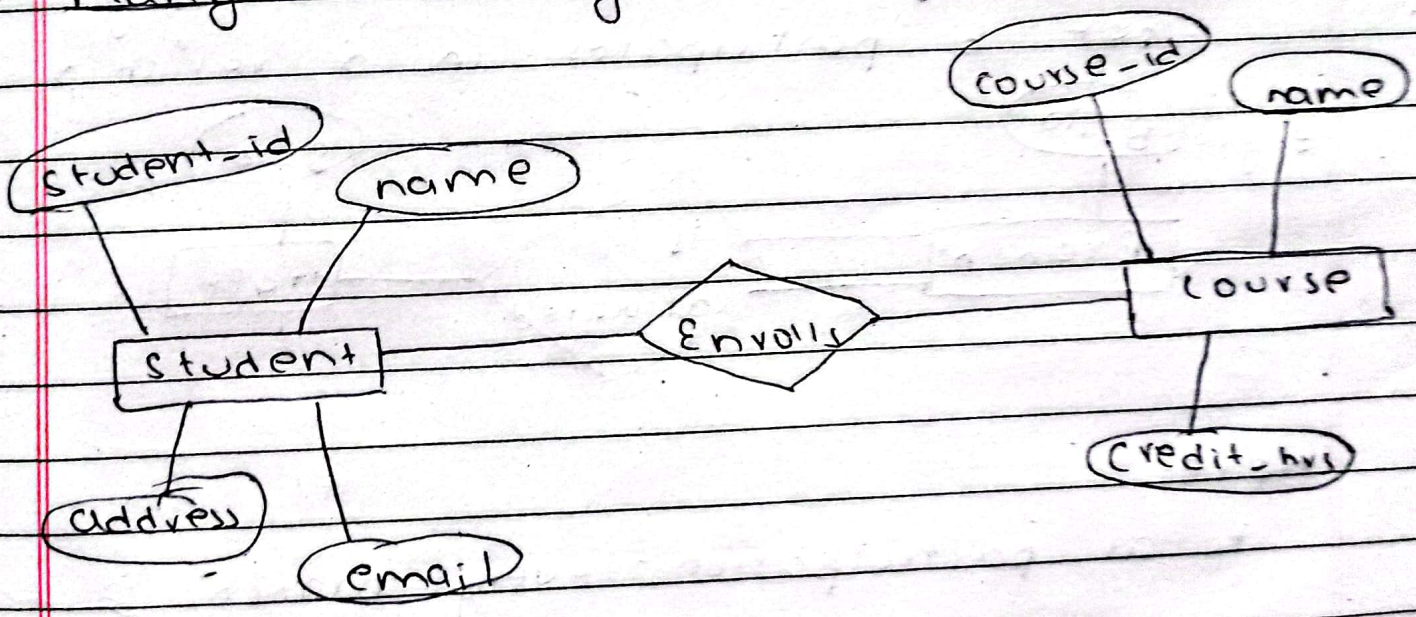
ii) One to many Relationship



iii) Many to one Relationship.



iv) Many to Many Rel^ship



Participation Constraints

→ Constraints that determine whether all or only some entity occurences participate in a relationship.

→ Types:

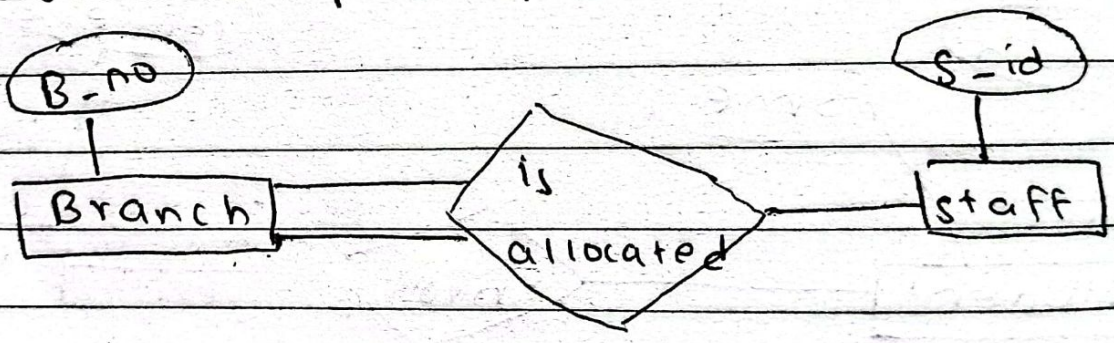
a) Total participation

→ if every entity in entity set E participates in relationship at least once.

b) Partial Participation

→ if only some entities in entity set E participates in a rel^n ship

e.g.



here,

total participation :- every branch is allocated members of staff.

partial participation :- a member of staff need not work at a branch office.

Keys:

- set of one or more attributes whose values are distinct for each individual entity in entity set is called key.

→ one or more columns in db table that is used to sort and for identify rows in a table is called key.

→ is used to fetch records / data from table according to condition.

→ Types:-

i) Super key

- is a set of one or more attribute whose value uniquely determine each ~~to~~ entity.

ii) Candidate key:

- is a minimal super key.

iii) Primary key

- is a candidate key chosen as a principle means of identifying entities within entity set.

ii) Foreign key:

* It is a field in one table that uniquely identifies row of another table.

→ It is used to establish an enforce a link bet^n two tables

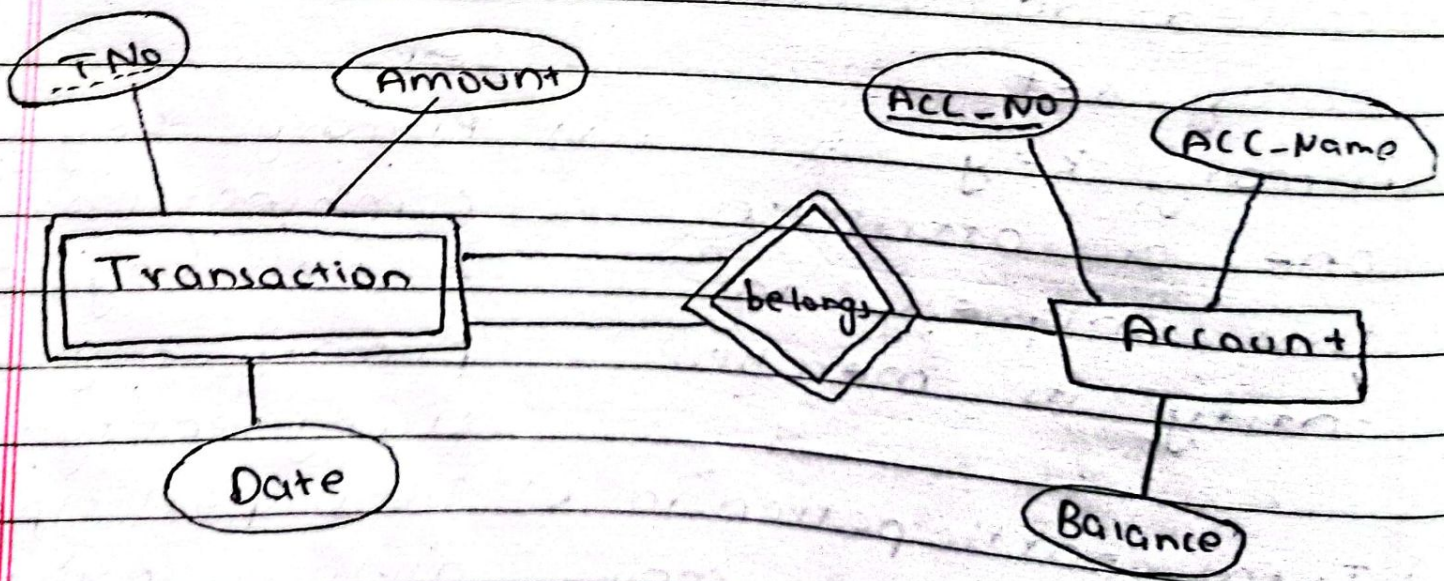
Strong Entity set	Weak Entity set.
i) It has its own primary key.	i) It doesn't have sufficient attributes to form a primary key on its own.
ii) It is represented by rectangle.	ii) It is represented by double rectangle.
iii) It contains a primary key represented by underline.	iii) It contains a partial key or discriminator represented by dashed underline.
iv) Its member is called dominant entity set.	iv) Its member is called sub-ordinate set.
v) Primary key is one of attributes that uniquely identify its members.	v) Primary key is combination of partial key and primary key of strong entity set.
vi) Total participation in rel^ship may or may not exist.	vi) Total participation always exists.
vii) Rel^ship bet^ two strong entity set is represented by diamond symbol.	viii) Rel^ship bet^ one strong and one weak entity set is represented by double diamond symbol.

viii) Single line connects strong entity set with rel^ship.

viii) Double line connects weak entity set with rel^ship.

→ idea of strong and weak entity set is related to the existence dependencies.

→ if the existence of entity A depends on existence of entity B, then A is existence dependent on B.



Process for developing ER diagram

i) Identify entities
ii) Find relationship
iii) Draw rough ER diagram
iv) Fill the cardinality.
v) Identify attributes
vi) Define primary keys.
vii) Map and draw fully attributed ER diagram
viii) Check result.

e.g.
A company has several departments each department has a supervisor and at least one employee.
Employees must be assigned to at least one but possibly more departments.
At least one employee is assigned to a project but an employee may be on vacation and not assigned to any project.
Important data fields are name of departments, projects, supervisor, and employees, as well as

Supervisor and employee no. and unique project number.

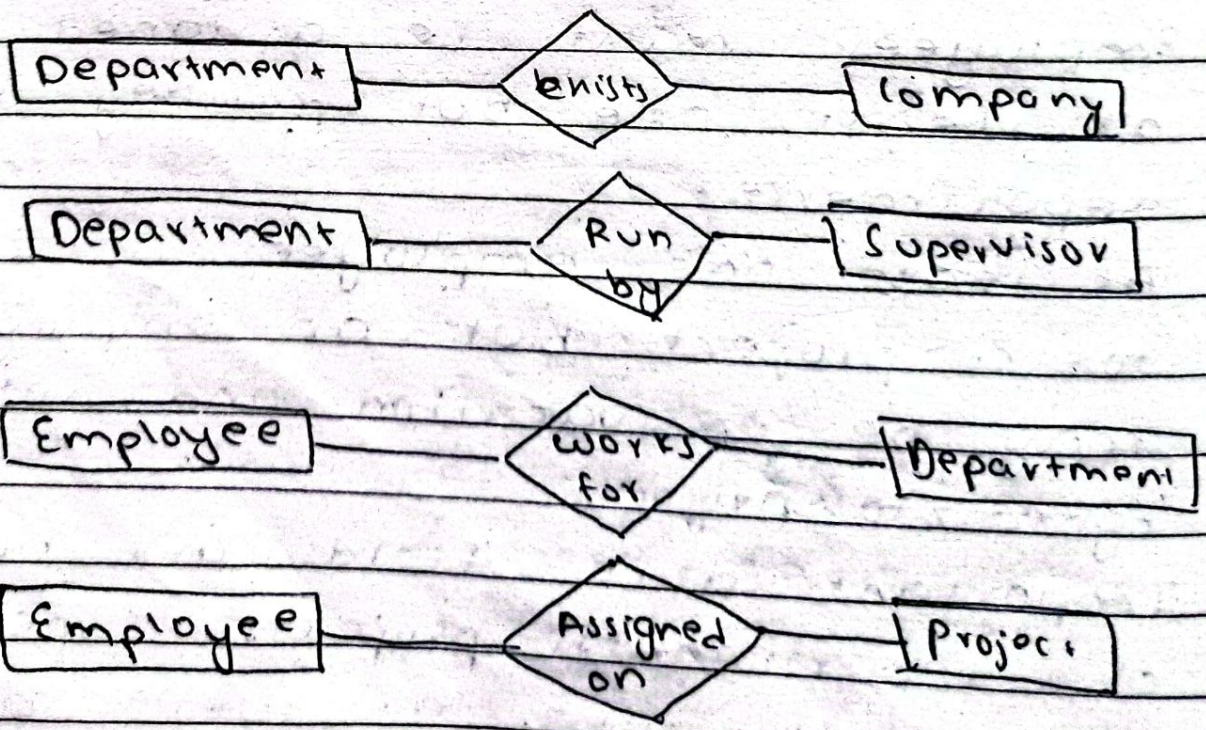
⇒

→ Identify Entities: Company, Department, Supervisor, Employee, Project.

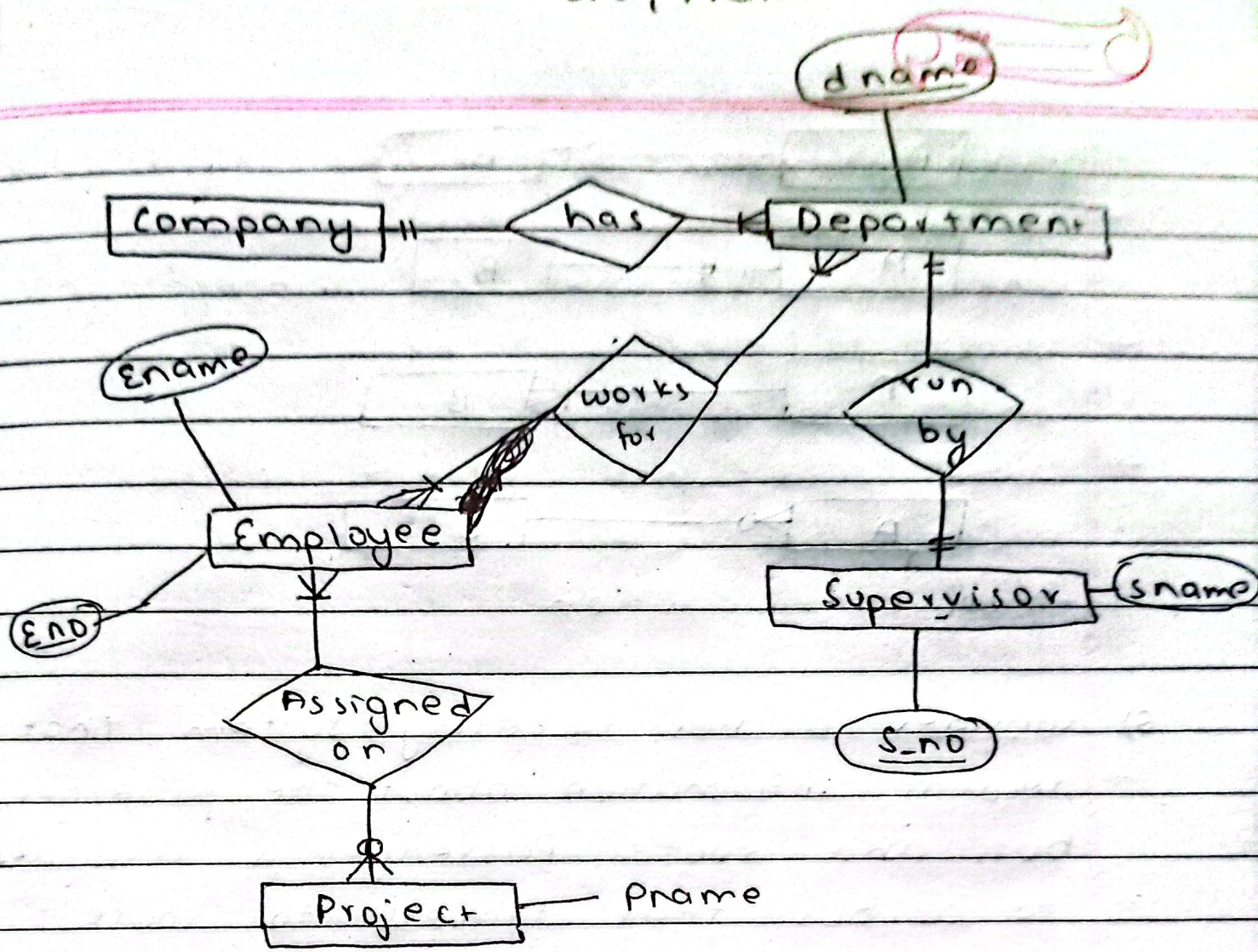
→ Find Relationship:

	Company	Depart.	Supervisor	Emp.	Proj.
Company	-	has	-	-	-
Department	exists	-	has	is assigned	-
Supervisor	-	run	-	-	-
Employee	-	works for	-	-	assign
Project	-	-	-	has	-

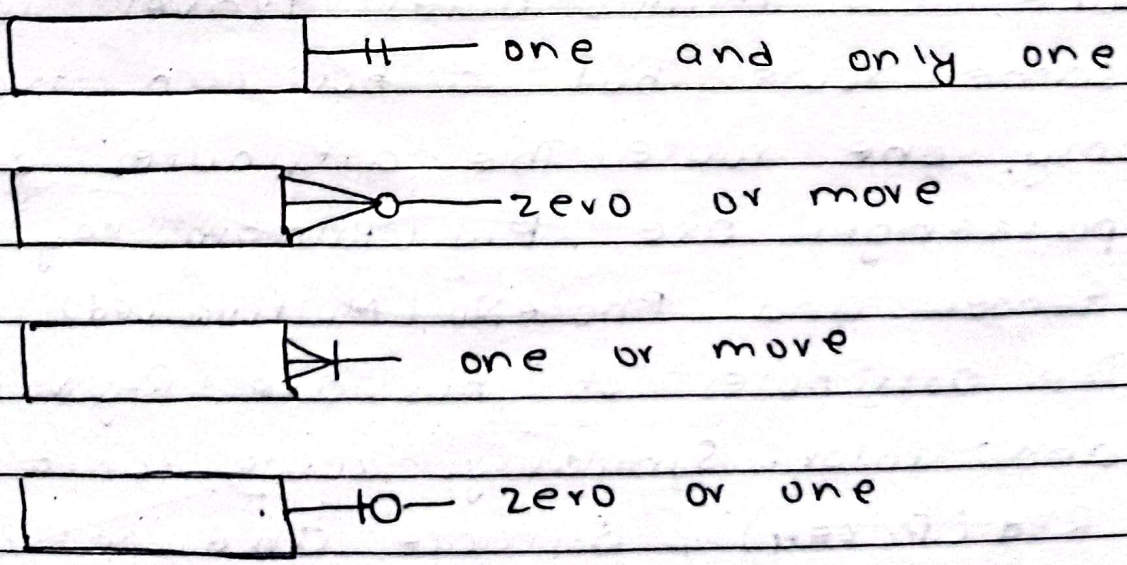
→ Draw rough ER Diagram

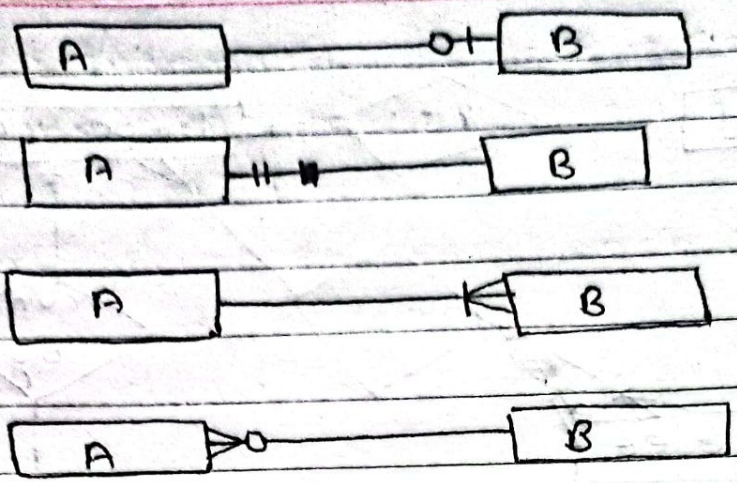


Identify Attributes: dname, Pname, Sname, Ename, Eno, Pname



→ Fill Cardinality



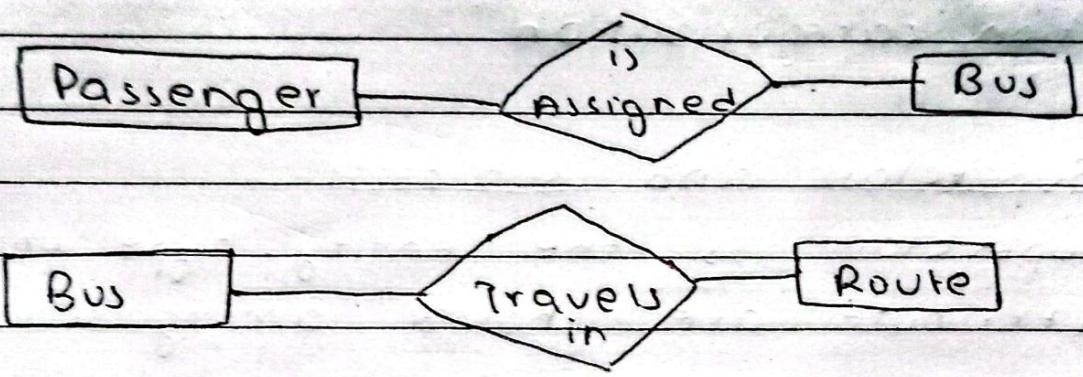


Q) Consider a bus ticketing system that records information about the passenger, bus, and route. Passenger is assigned to a bus that travels to route A bus contains many passengers and a passenger can be assigned into only one bus. Many buses travel in same route but a bus can travel to only one route. The attributes of passenger are Pid (Primary key), Gender and PhoneNo (Multivalued). The attributes of bus are Reg.No (P-key) and color. Similarly, route contains RId (P. key), distance, and rate. Now construct a clean and concise ER diagram with clear cardinality mappings.

→ Identifying Entities: Passenger, Bus, Route.

→ Identifying attributes: Pid, gender, PhoneNo, RegNo, Color, RId, distance, rate.

→ Identifying Relationship.



→ ER diagram

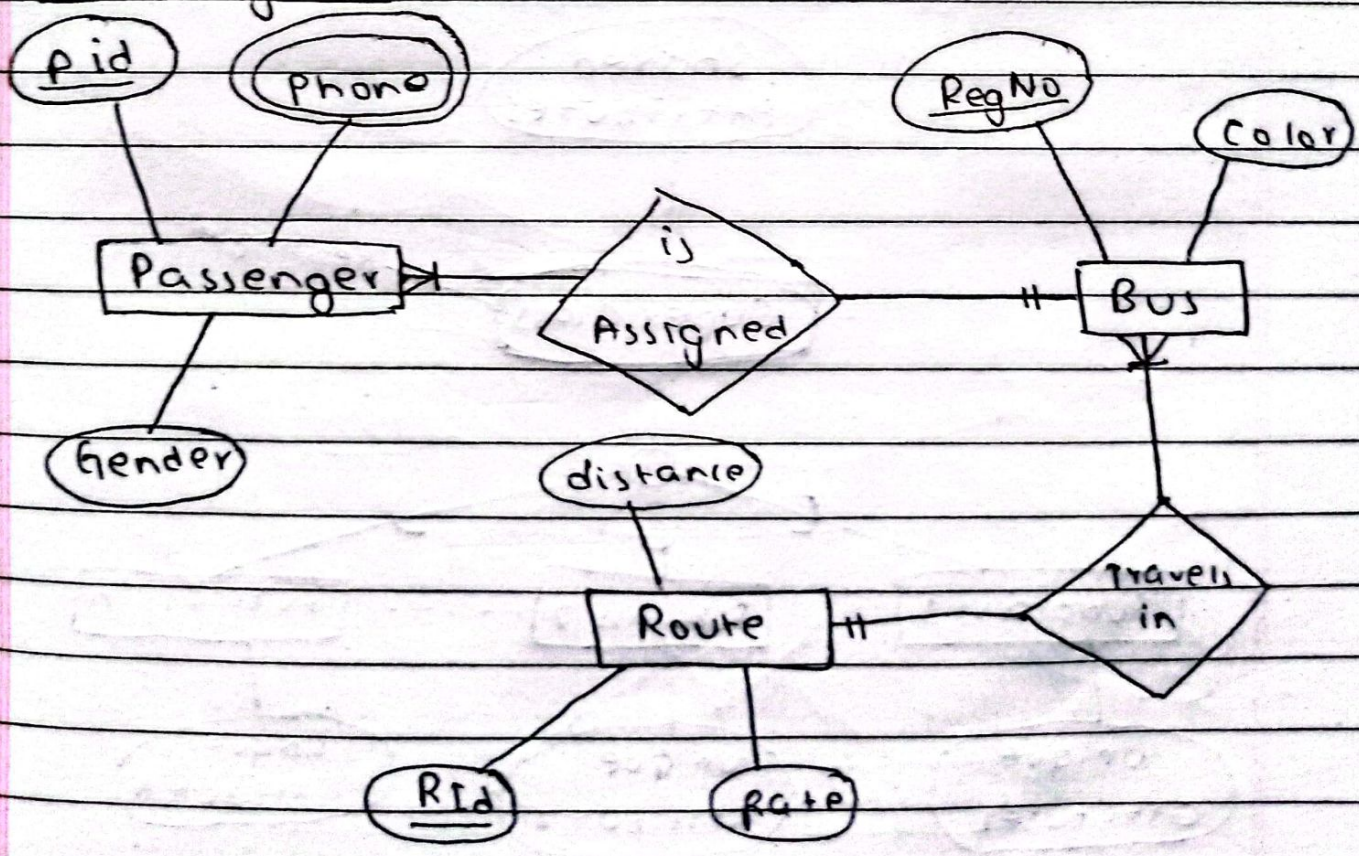


Fig: ER diagram

EER (Extended or Enhanced ER) model

→ is the ER model combined with the additional features of semantic concepts.
→ shows complex relationship between objects in a database.
→ some additional concepts in EER model are: specialization, generalization, categorization.

Superclass and sub class

→ Superclass is the entity type whose attributes are shared among subclass.
→ Subclass is the one whose attributes are unique from other subclasses.

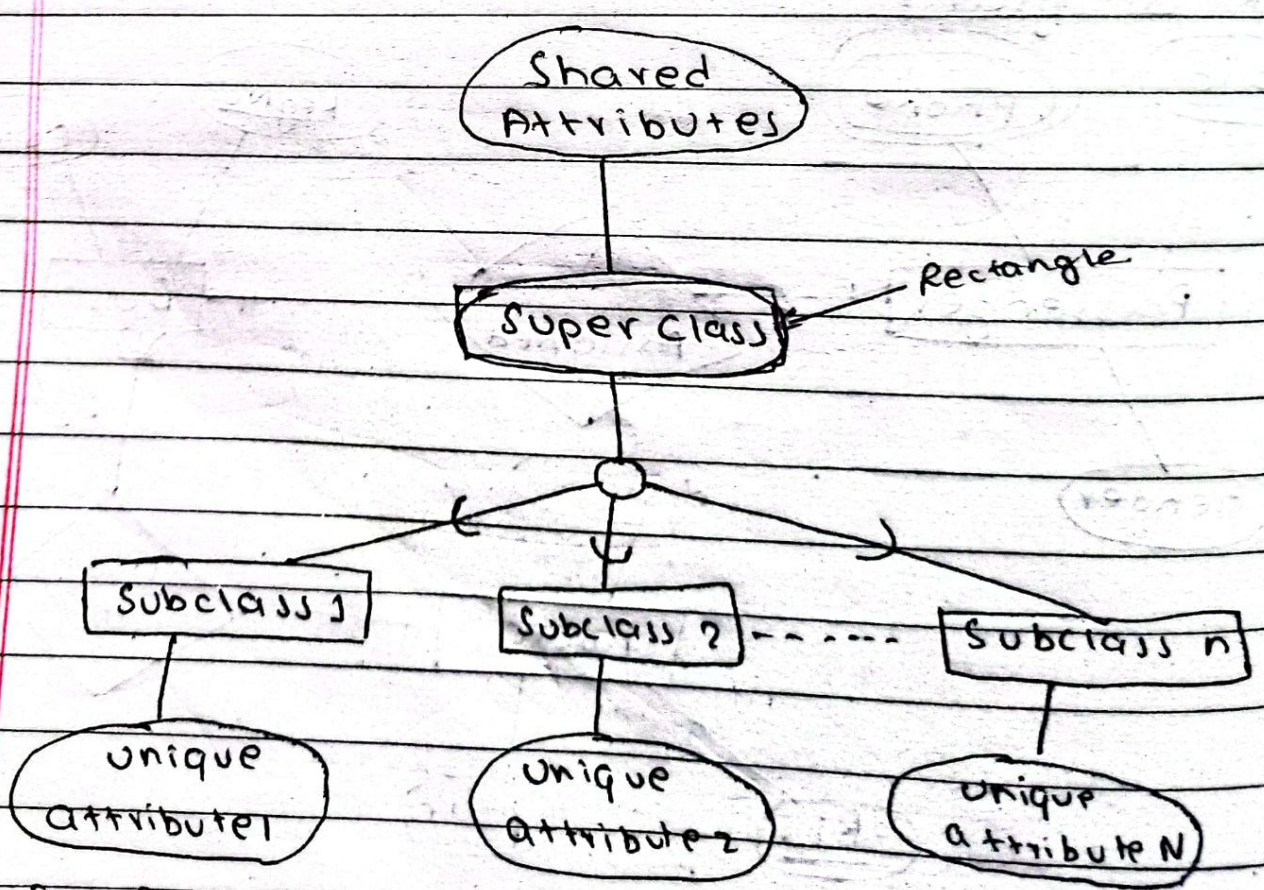
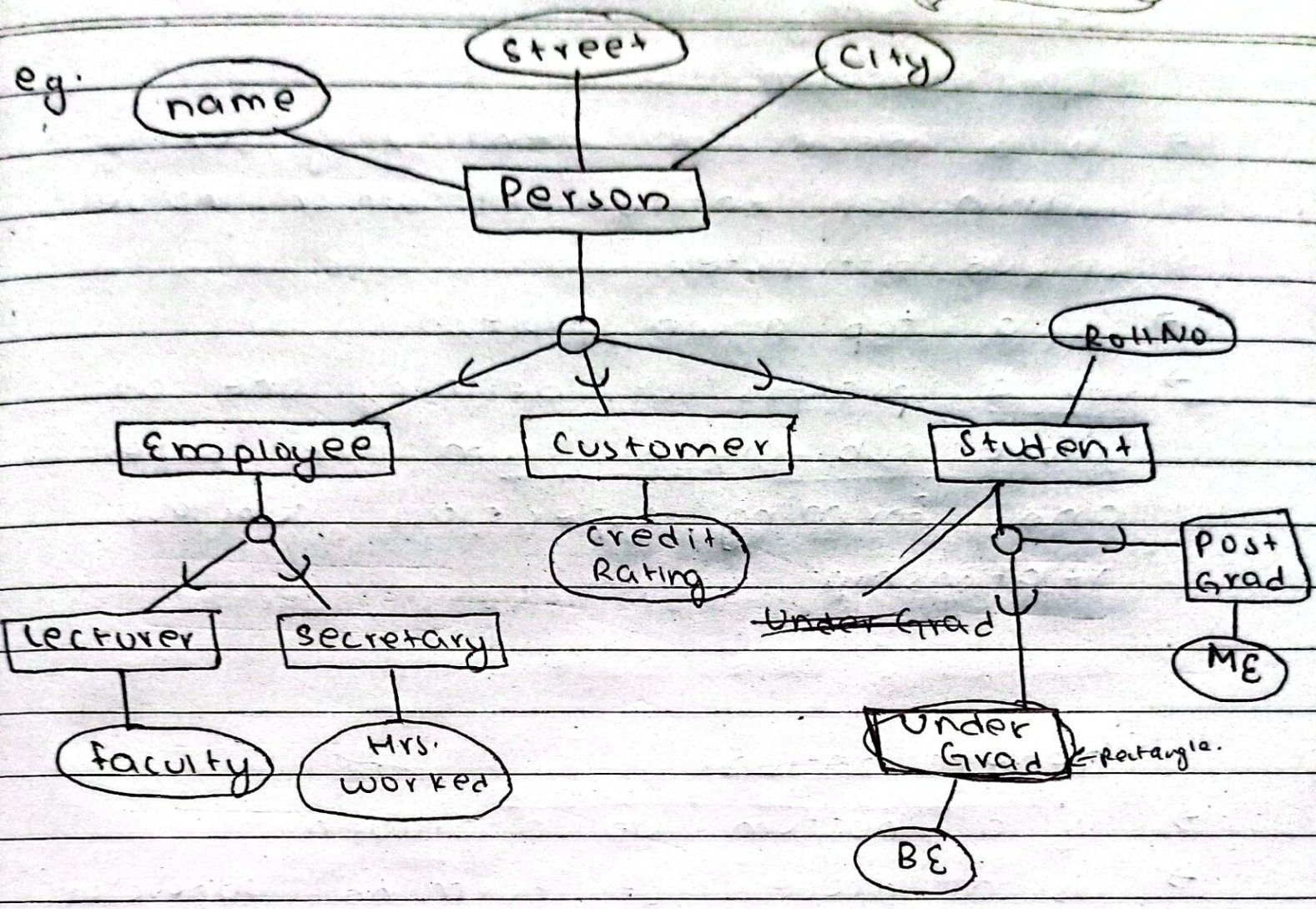


Fig: Basic notation for superclass/subclass relation


eg.
name
Street
City
Person
Employee
Customer
Student
RollNo
Credit Rating
Lecturer
Secretary
Under Grad
Post Grad
ME
Faculty
Hrs. worked
Under Grad
Rectangle.
BE

07/30 Friday

Specialization

→ is a means of identifying sub groups within an entity set which have unique attributes.

→ is a top down process.

→ is a process of defining one or more sub classes from the superclass and forming superclass subclass relationship.

Generalization

→ is a bottom up approach

→ is a process of forming superclass

→ it emphasize the similarities among lower level entity set and hide the difference.

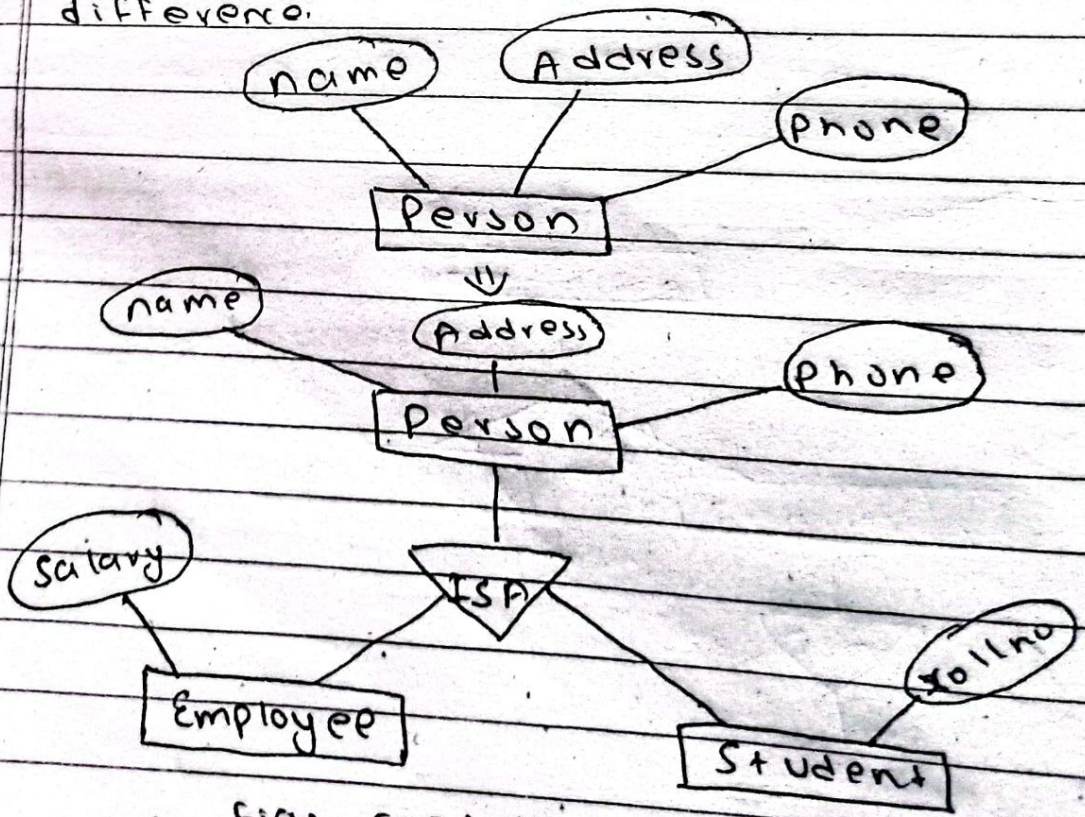


fig: specialization.

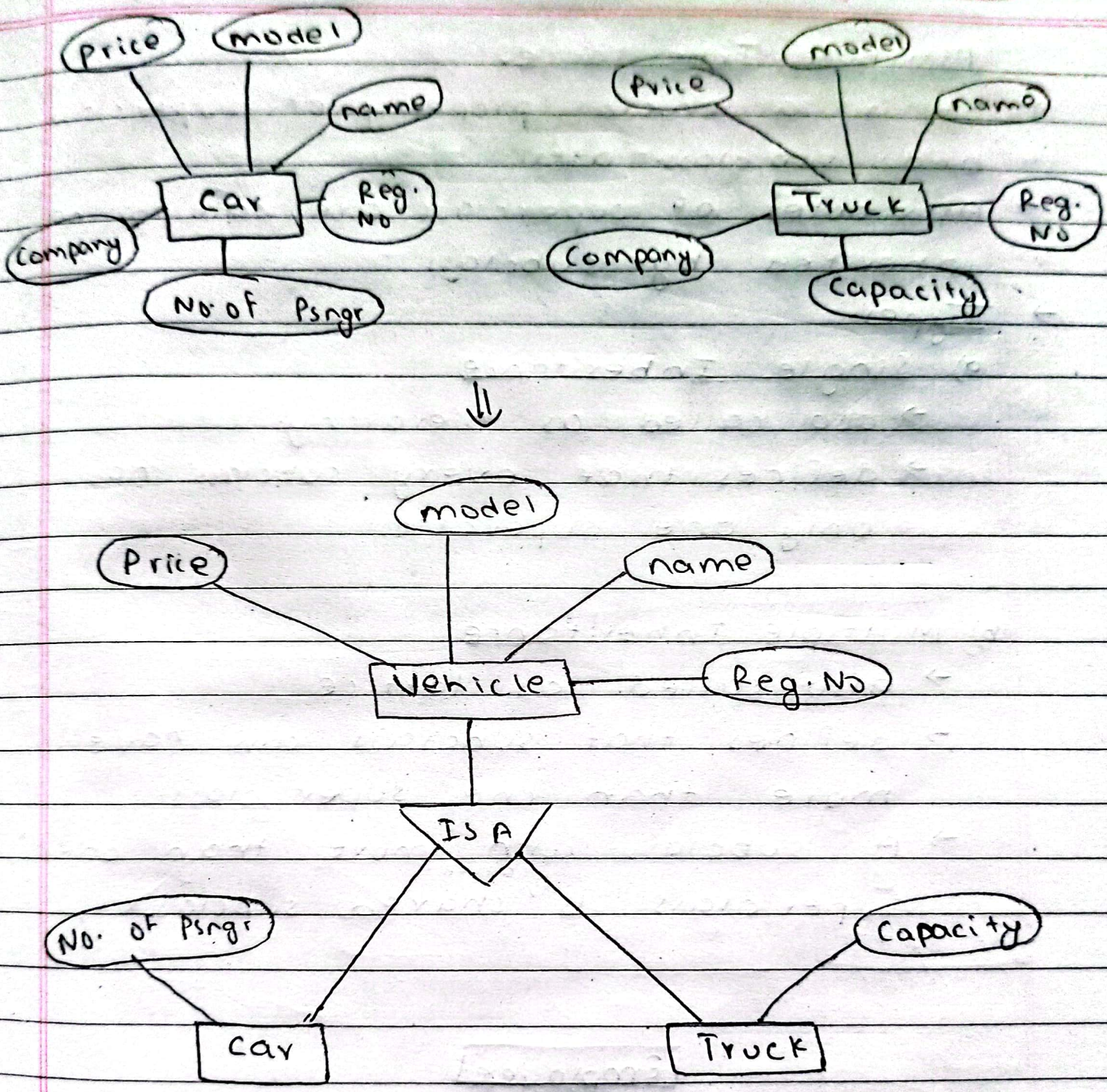


Fig: Generalization.

Attribute Inheritance

→ It is a crucial property of superclass and subclass: attrt

→ Attributes of superclass are said to be inherited by subclass

→ Types

a) Single Inheritance
   → also called as hierarchy.
   → defines that every subclass has only one superclass

b) Multiple Inheritance
   → also called as lattice
   → defines that subclass can have more than one super class.
   → A subclass with more than one superclass is called shared subclass.

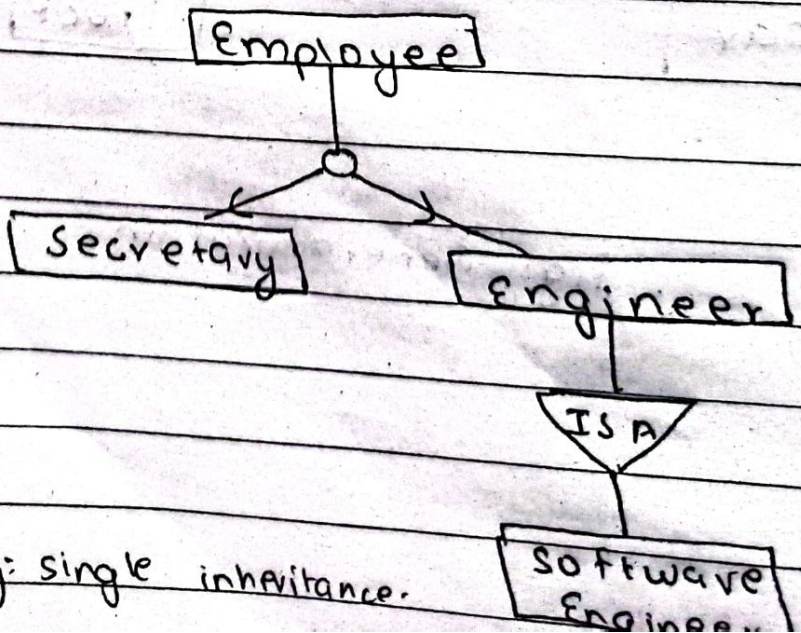


Fig: single inheritance.

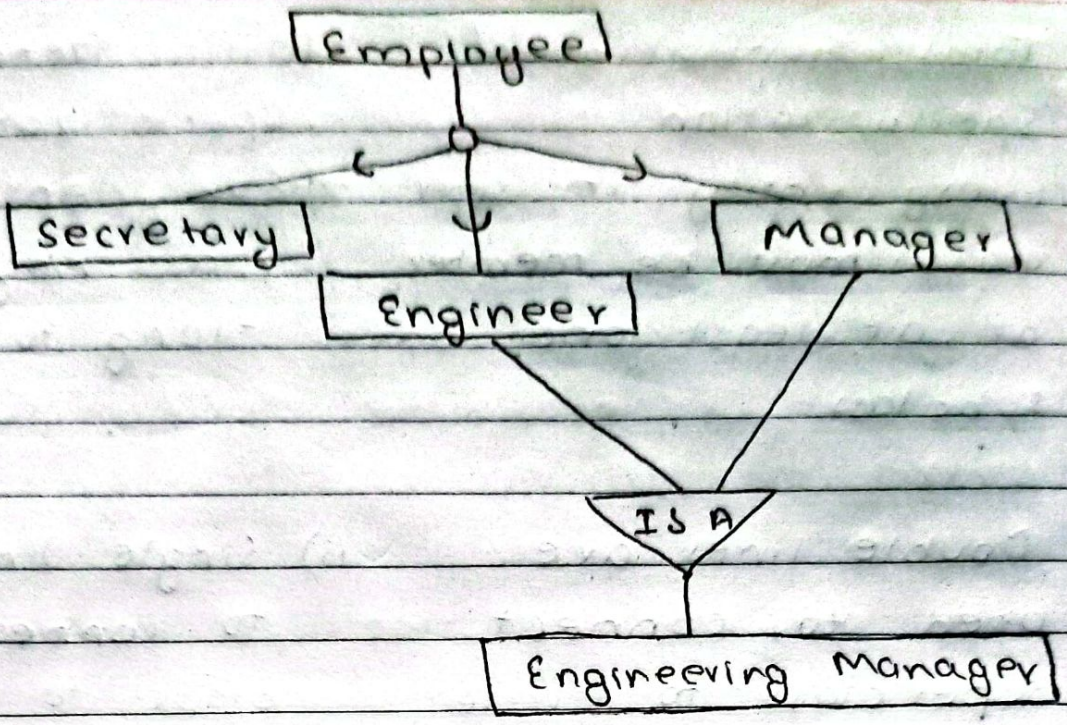


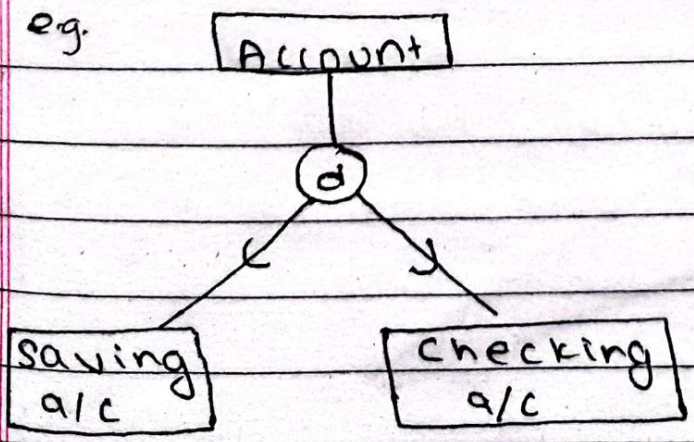
Fig: Multiple Inheritance

Disjoint Constriant

→ Subclass belong to no more than one lower levels entity

→ is represented by placing 'd' in circle.

eg.

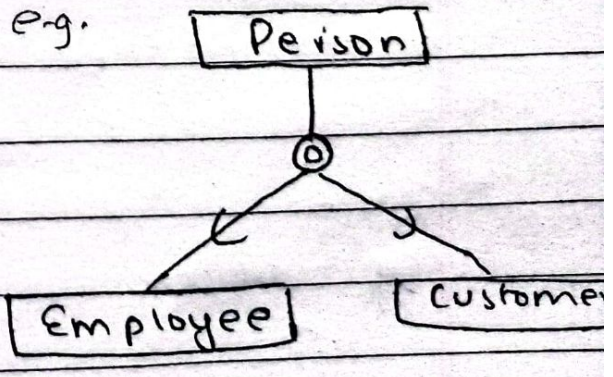


Overlapping Constraint

→ Subclass may belong to more than one lower level entity

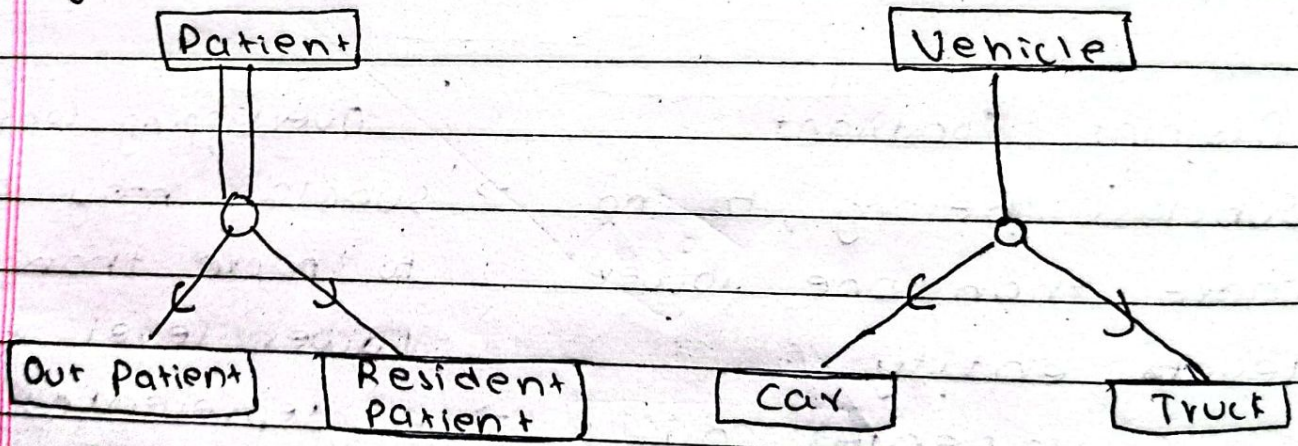
→ is represented by placing 'O' in circ

e.g.



Total Generalization/ Specialization	Partial Generalization/ specialization
→ Every entity in super class must be member of at least one subclass	→ Every entity in super class may not be-long to subclass
ii) Double lines are used to connect superclass to circle	ii) Single line is used to connect super class to circle.

eg.



Categorization (Union Types)

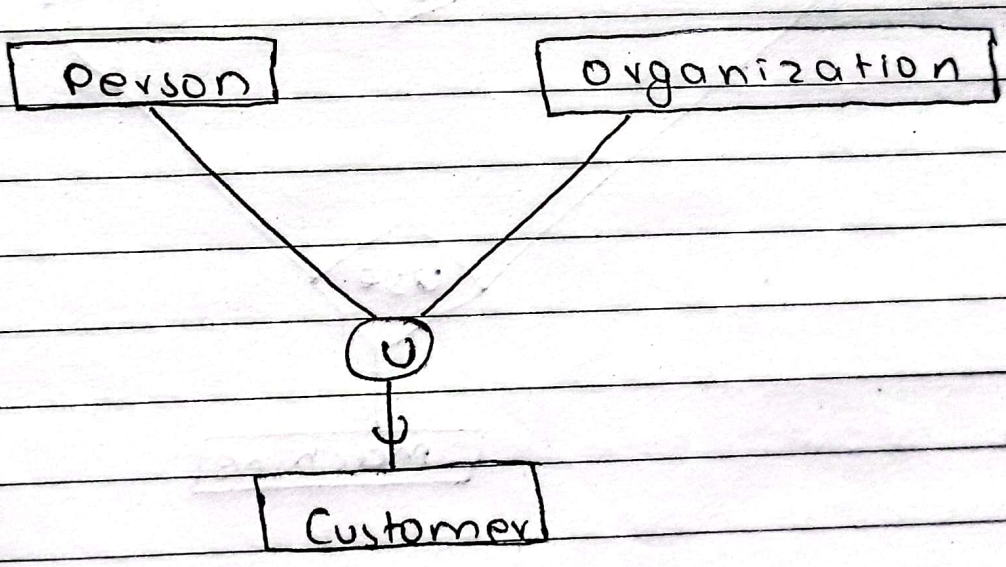
→ models a class / sub class with more than one superclass of distinct entity types.

→ subclass is a category.

→ only one subclass exists in categorization

→ attribute inheritance is selective

→ e.g.



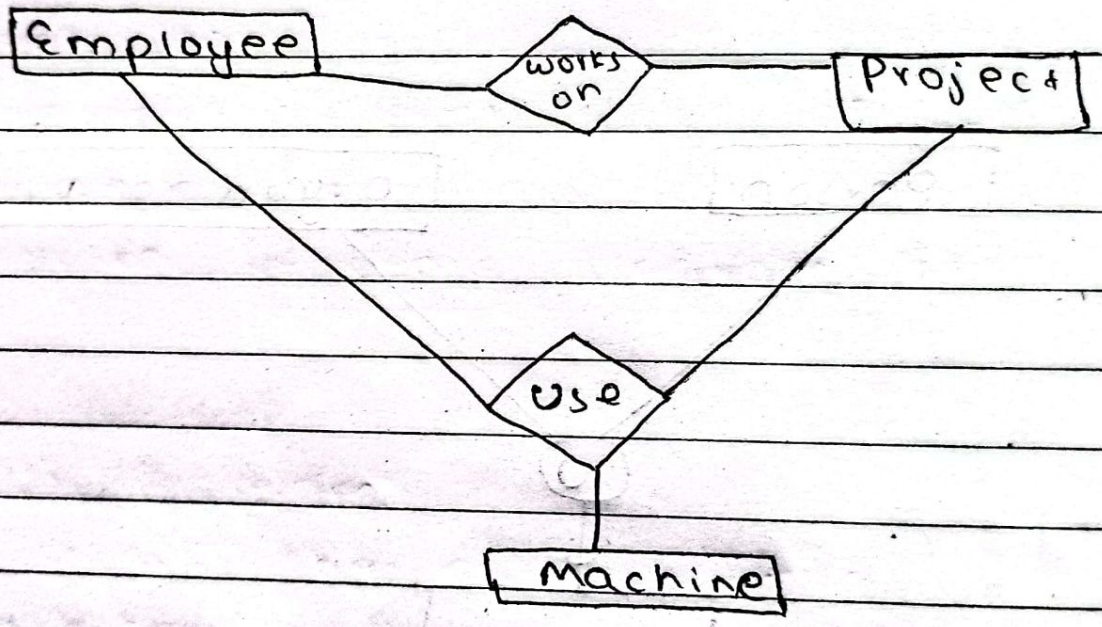
Here, category customer is subclass of union of person and organization.

Aggregation

→ Is an abstraction through which relationships are treated as high level entities.

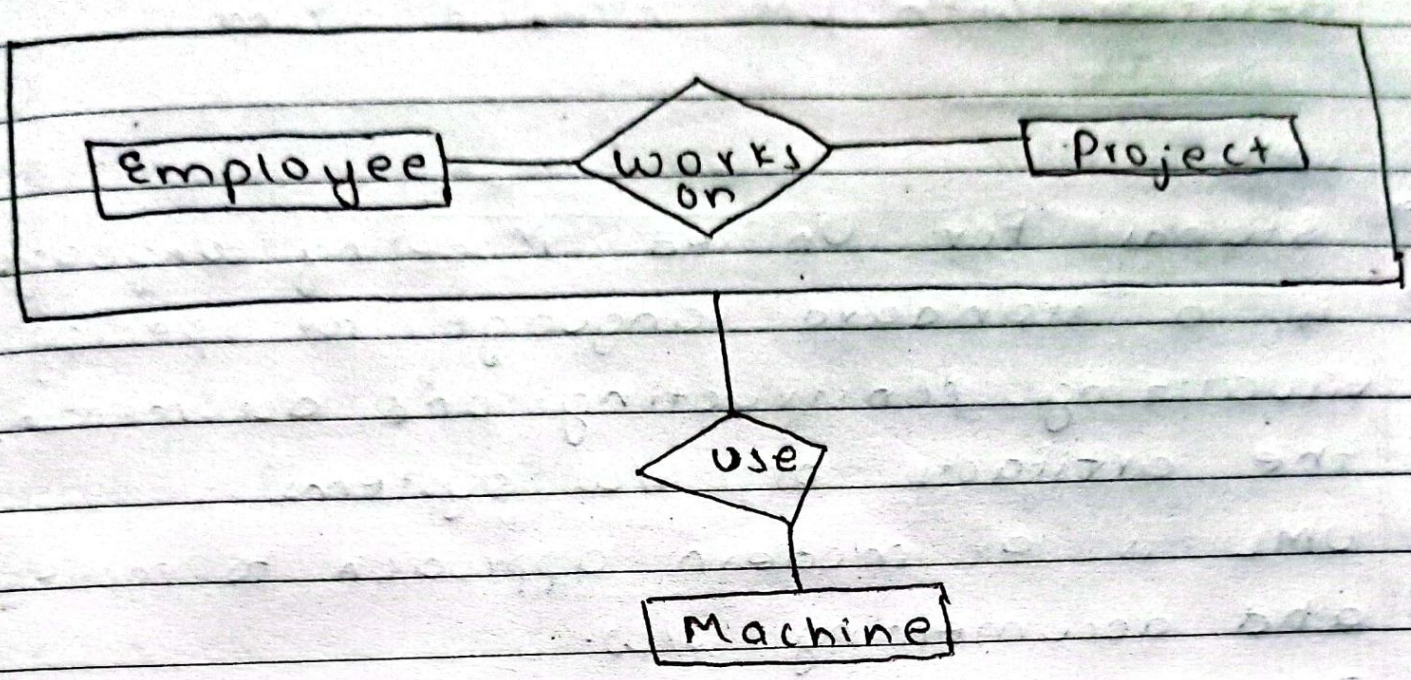
→ e.g. employee works on project and use no of machine during that work.

→ one way of solution is as below.



→ Here, problem is {employee, project} combination that appears in 'works on' and 'use' relationship.

→ New, solution is using aggregation as below:


Employee
works on
Project
use
Machine

08/02 Sunday

Relation with UML class diagram

UML

→ Stands for Unified Modeling language.
→ is a standard language for specifying visualizing, constructing and documenting the artifacts of s/w systems
→ UML is a modern approach to modeling and documenting s/w.
→ It is based on diagramatic repre-sentations of s/w components.
→ There are several types of UML diagrams like class diagram, usecase diagram, sequence diagram, etc.

Class Diagram

→ It shows the classes in a system, attributes and operations of each class and the relationship between each class.
→ in class diagram, name is at the top, attributes in the middle and operations or methods at the bottom

UML notations for ER models

Entities are modeled as a class. The name of class is the entity name and the attributes of class are the entity's attribute.

The name and attributes can be arranged in a box style.

Relationships are shown as a single solid line connecting the two entities. The minimum and maximum cardinalities are shown along the line and verb phrases can be added to completely characterize the nature of the rel^ship.

e.g.

One publisher may be publishing one or more books. One book must be published by one and only one publisher.

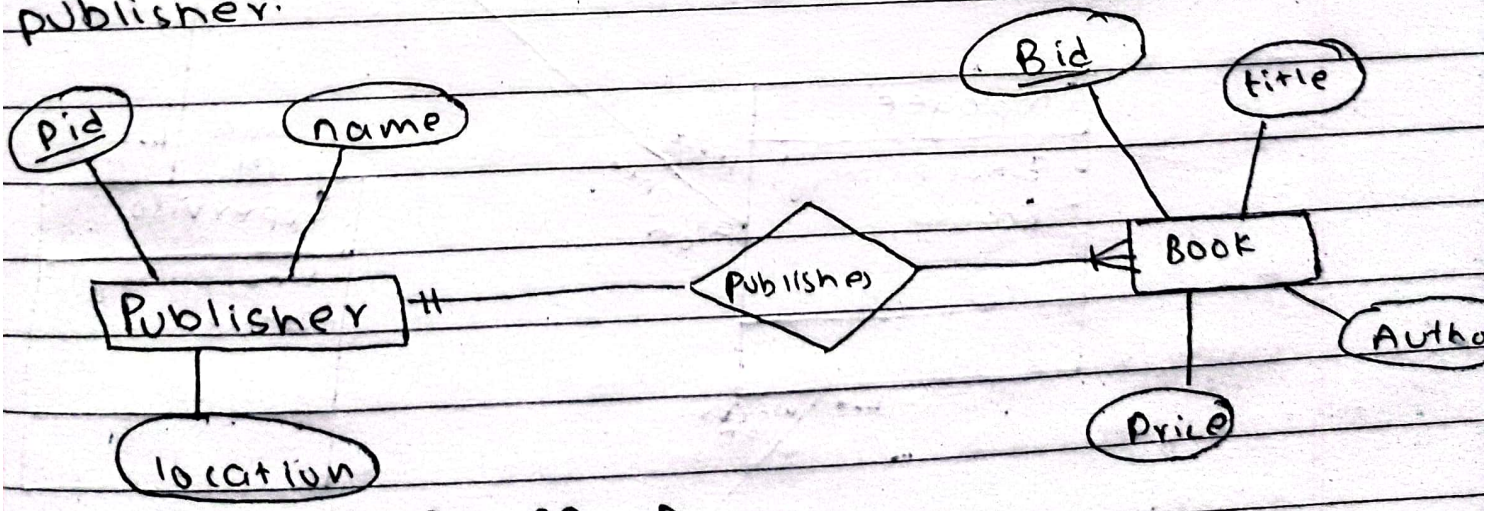


Fig: ER Diagram.

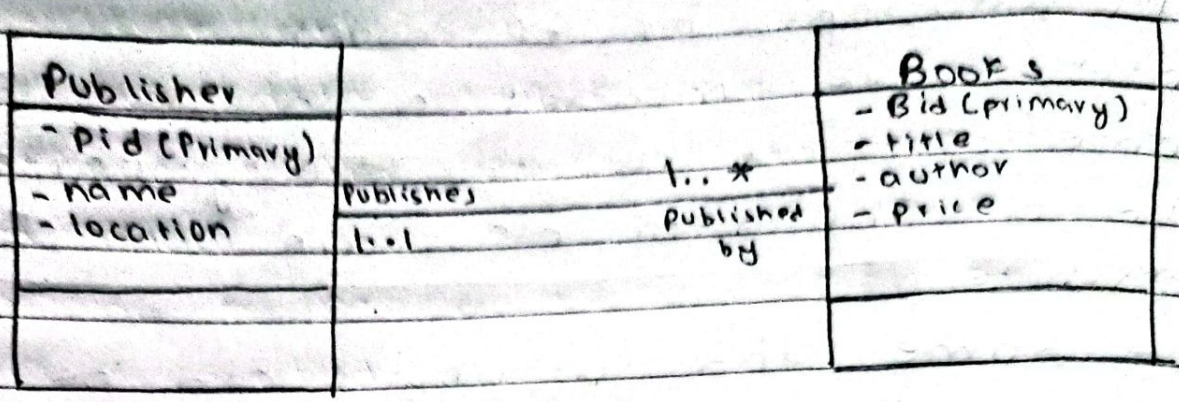
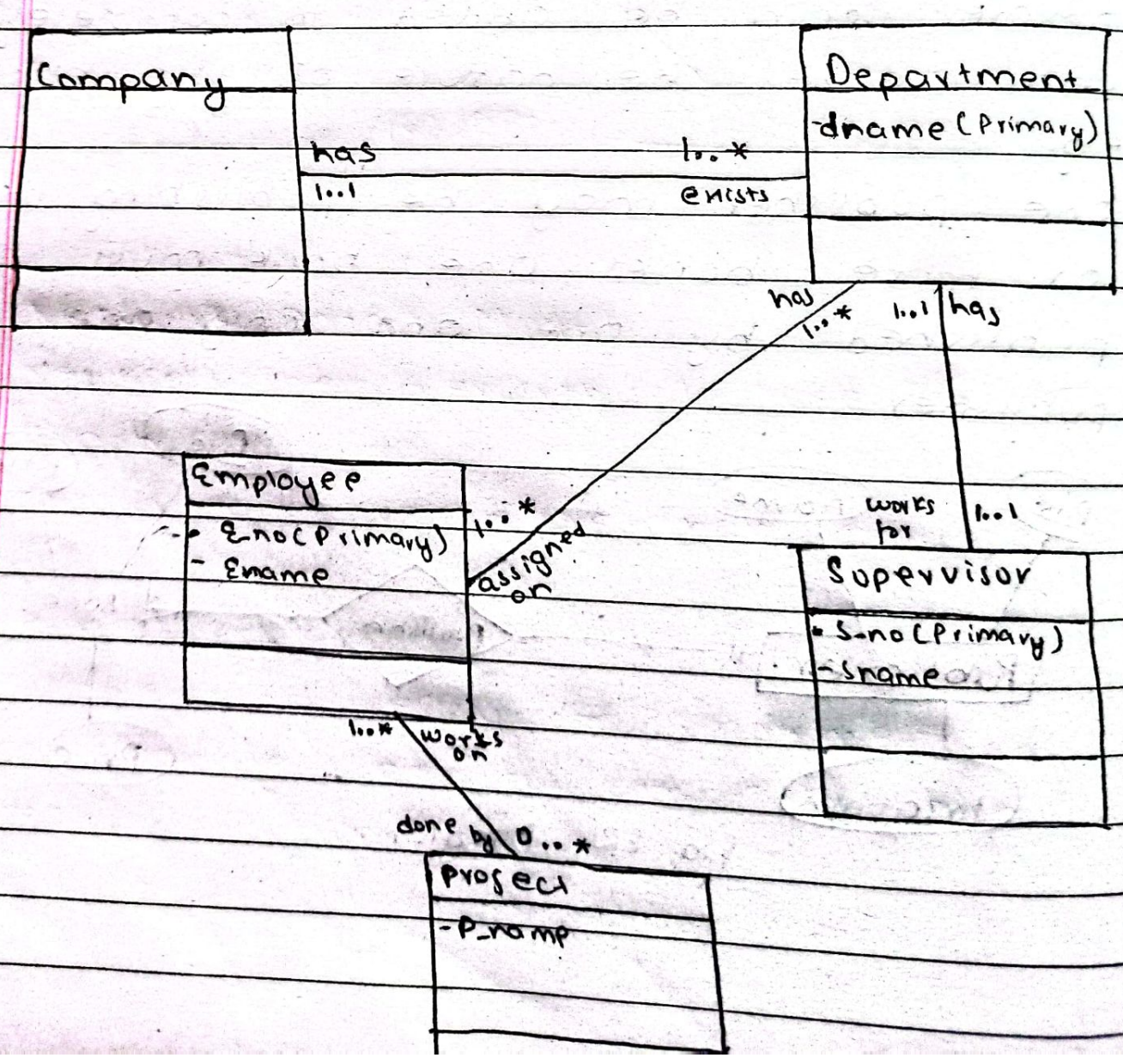


Fig: Class Diagram.

⇒ Class Diagram of ER diagram 1st Question



Alternate Data Models

Hierarchical Data Model

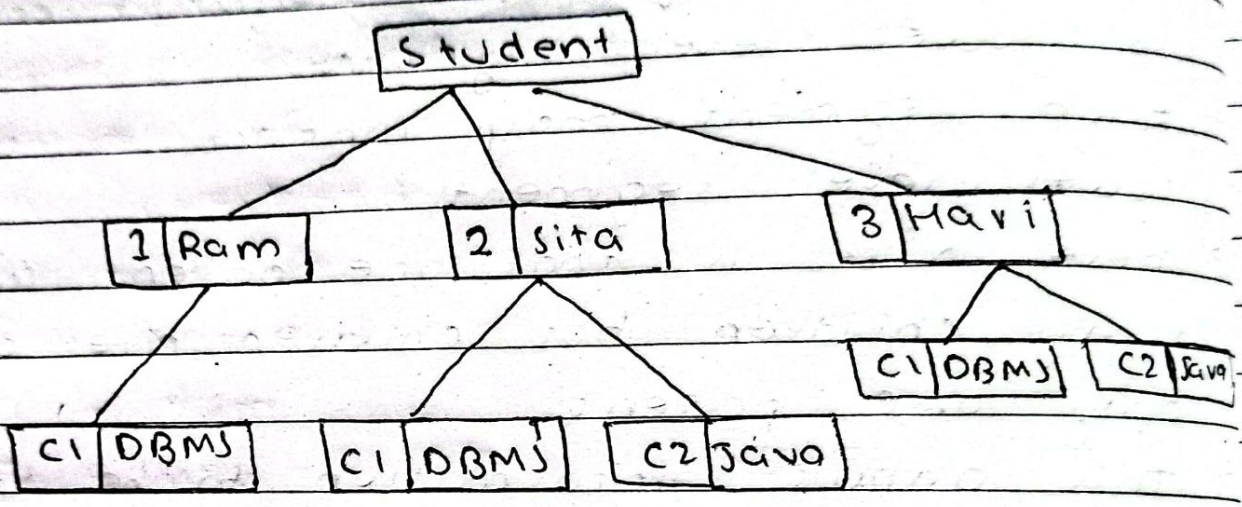
- organizes data in tree structure where there is a hierarchy of parent and child data segments.
- it has a single root segment connected to lower level segments.
- each segment may connect to other lower level segments.
- here rule is that one parent can have many children but children are allowed only one parent.
- This allows information to be repeated through parent child relation.

Advantages

- allows easy addition and deletion of new information.
- fast access to data at top.
- relates well to anything that works on one to many relationships.

Disadvantages

→ allows data redundancy.
→ searching for lower level information is slow.
→ many to many relationships are not supported.



2) Network Data Model

→ replaces the hierarchical tree with graph.
→ allows to have more than one parent

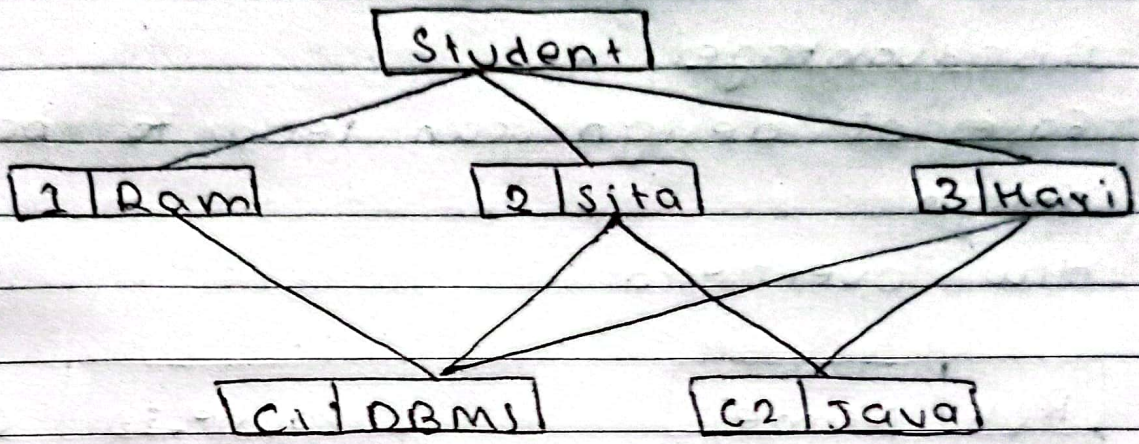
Advantages

→ simple and easy to implement
→ can handle one to one and many to many relationships.
→ data access is easier than hierarchical model

Disadvantages

→ system complexity is difficult to handle and maintain.

→ lack of structural independence i.e change in structure makes change in application too.



3) Relational Data Model

→ all data are stored in tables where each table consists of rows and columns.

→ here, keys are used to order data or relate data to other tables

→ defines set of rules to enforce data integrity, knows as integrity constraints

→ also defines how data are manipulated.

→ defines special feature called normalization to ensure efficient data storage.

Advantages

- easy to use
- Flexibility
- Security
- data independence.

Disadvantages

- Ease of design can lead to bad design
- H/W overhead.

Roll	Name
1	Ram
2	Sita
3	Hari

table 1

Course_ID	Cname
1	DBMS
2	Java

table 2.

StuCourse ID	Roll	Course_ID
1	1	1
2	2	1
3	2	2
4	3	1
5	3	2

table 3

4 Object Oriented Data Model

→ It is based on object oriented prog. language.

→ A core object oriented data model consists of following basic concepts.

i) Object and object identifier
- any real world entity is modeled as object.

ii) attributes and methods
- every object has attributes and methods

iii) Class
- means of grouping all the objects which share same set of attributes and methods

iv) Class hierarchy and inheritance
- derive a new class from existing class.

08/03 Monday

Relational Model

→ is the primary data model widely used for data storage and processing.
→ was provided by Dr. E.F. Codd of IBM in 1970.
→ Data are organized in two dimensional tables called relations.
→ tables are related to each other.
→ DBMS based on relational model is called relational database Management System (RDBMS)

Some terminologies for RDBMS

Record or relation Oriented	Table Oriented
- Relation	Table
- Domain	Set of permitted values
- Tuple, Record	Row
- Attribute, Field	Column
- Cardinality	No. of ~~columns~~ rows
- Degree	No. of columns
- Primary key	unique + NOT NULL

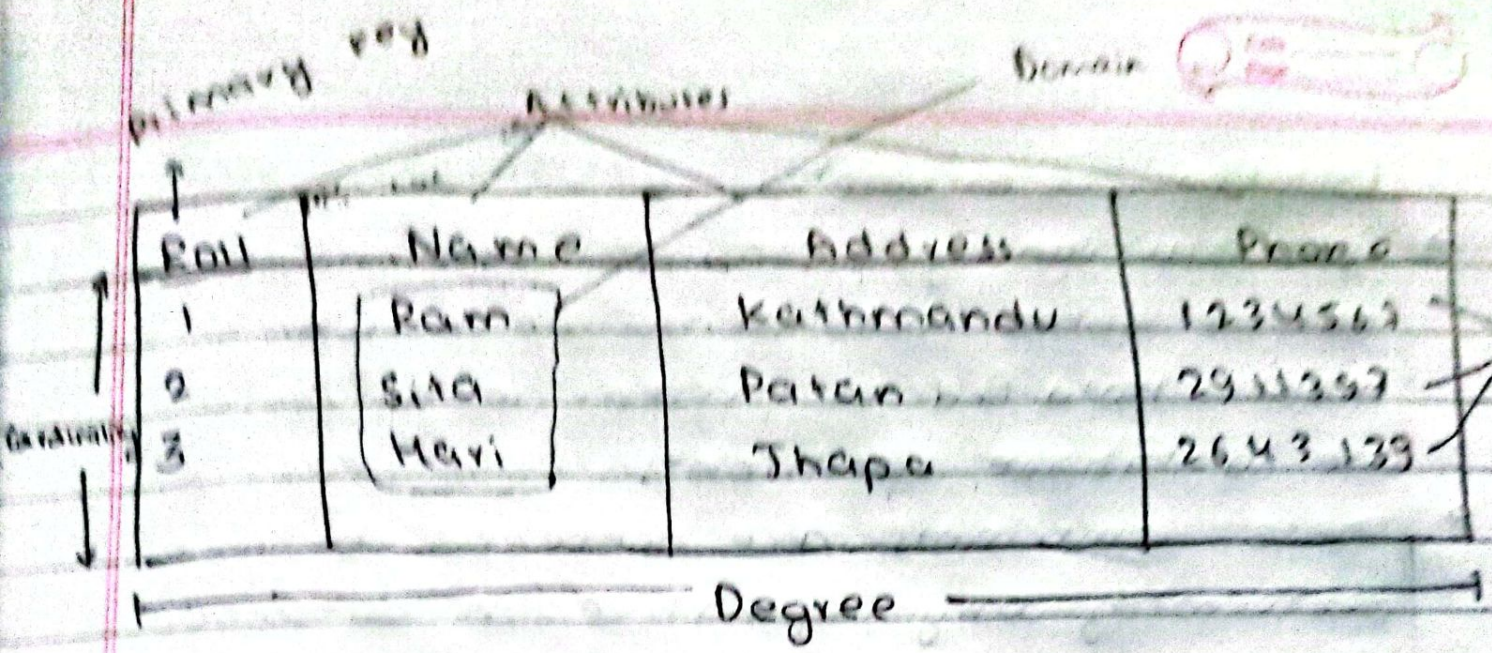


Fig. Student Table or Student relation.

here,

degree of student relation = 4

cardinality of student relation = 3

→ Relation is set of tuples.

→ $t \in R$ denotes that tuple (t) is in relation r.

→ For relation r, domains of all attributes of r be atomic.

Database Constriants

→ while designing relational model, we define some conditions which which must hold for data present in database are database constriants.

→ These constriants are checked before performing any operation in database if there is violation of any constraint, operation will fail.

Integrity constriants

→ integrity refers accuracy or correctness of data in database.

→ Integrity constriants ensure no result in loss of data consistency although changes are made to database by authorized users.

→ Integrity constriants are defined by some key constriants like primary key, foreign key.

→ Two general integrity rules are
  i) Entity integrity
  ii) Referential integrity.

i) Entity integrity

→ Ensures that there are no dobbt duplicate records in the table and the field that identifies each record is unique and NOT NULL.

→ It is mechanism to provide and maintain primary key.

→ It ensures two properties for primary key:

  → Primary key for a row in unique.

  → Primary key is not NULL.

ii) Referential Integrity

→ is concerned with foreign key.

→ it designates a column or combi-nation of columns as foreign key and establishes relationship between primary key.

→ it states that: if in two relations R and S, if an attribute A1 in relation R is primary key and foreign key in S then the value of foreign key in a tuple in S must be either be equal to primary key of tuple in R, or be entirely NULL.

Domain Constriant

→ it can be defined as the definition of a valid set of values for an attribute.

→ is specified on the column of a relation, so that correct values can be entered in the column for each record.

Structure of Relational Database

→ A relation is used to show information about any entity and its relationship with other entities.

→ A relation comprises of a relation schema and relation instance.

→ A relation schema shows the attributes of tables.

→ A relation instance is a two dimensional table with set of tuples.

→ e.g. relation schema:-

- employee(eid, name, address, job, salary)

Relation instance:-

eid	name	Address	Job	Salary
1	Ram	Ktm	Programmer	40,000
2	Sita	Pokhara	Analyst	50,000
3	Hari	ktm	Manager	60,000

Schema Diagram:-

→ Enable database developers to transpose idea to papers

→ provide an overview of entire database.

→ in schema diagram, all database tables are designated with unique columns and special features like primary key etc.

eg. to store information of employee and company and to ~~suit~~ relate employee belonging to company.

employee (id, name, address, job, salary)
company (id, name, address)
works (eid, cid)

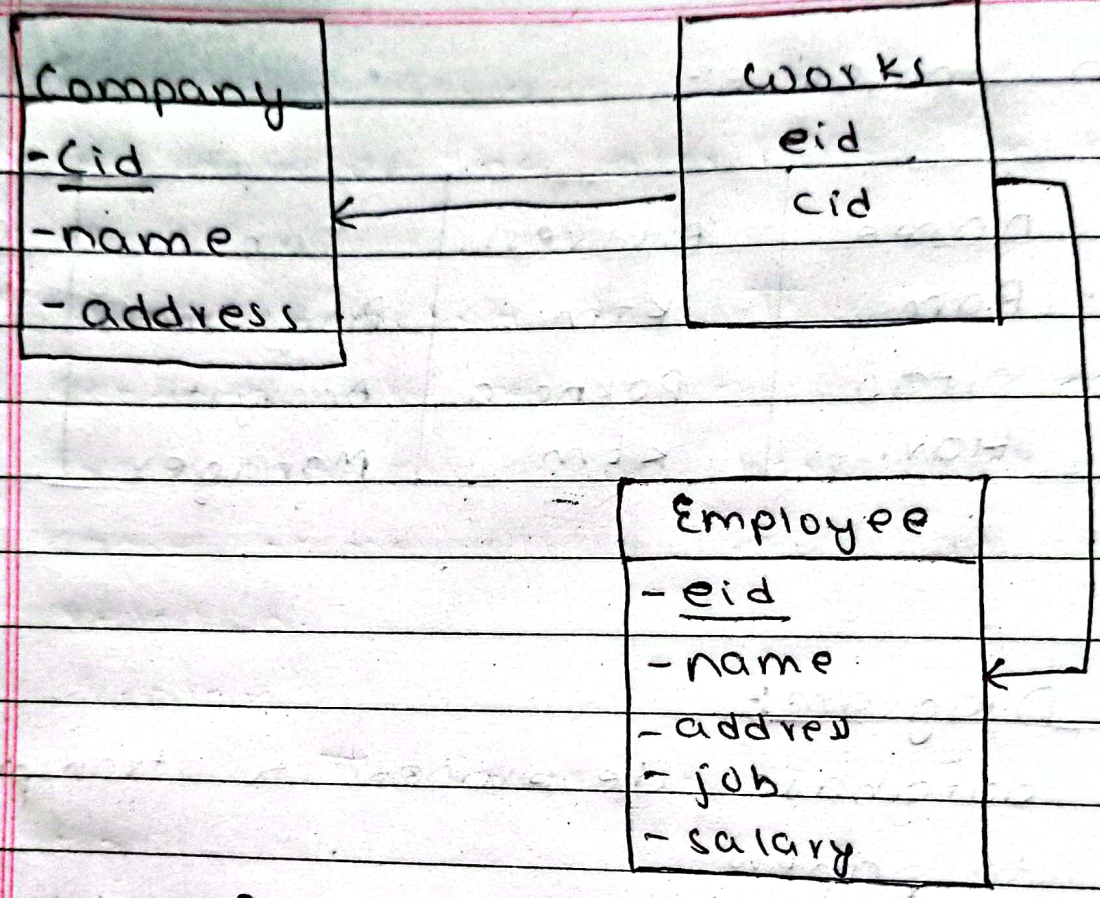


fig: schema diagram.

08/04 Tuesday

Relational Algebra:

→ It is a procedural query language which takes instances of relations as input and gives instances of relation as output.

→ It helps to understand query execution and optimization in RDBMS.

→ Basic operations are:

i) Selection ii) Projection iii) Union
iv) Set Difference v) Intersection
vi) Cartesian Product.

i) Selection

→ it selects a subset of rows from relation that satisfies some condition.

→ it is denoted by σ

→ schema of result is identical to schema of input relation.

→ Result relation can be input for another relational algebra operation.

→ syntax:

$$\sigma_{condition}(relation)$$

→ e.g.

$$\sigma_{salary > 15000}(employee)$$

ii) Projection

→ it deletes the attributes that are not in project list.

→ it is denoted by Π.

→ Schema of result contain exactly the fields in projection lits, with the same names that they had in input relation.

→ Syntax:

$$\Pi_{\text{attribute list}}(\text{relation})$$

→ e.g.

$$\Pi_{\text{name, job}}(\text{employee})$$

iii) Union

→ it builds a relation from tuples appea-ring in either or both of the specified relation.

→ it eliminates duplicates

→ it is denoted by $\cup$.

→ Two relations are union compatible if

- they have same no. of attributes
- attribute domains must be compatible

$\sigma_{job = "Programmer"}$ ^ salary > 60000 (employee)

Syntax:

$\Pi_{attribute\ list}$ (relation1) $\cup$ $\Pi_{attribute\ list}$ (relation2)

$R \cup S = S \cup R$

Intersection

→ results tuples that appear in both relations

→ It is denoted by $\cap$.

→ Relations must be union compatible.

→ Syntax:

$\Pi_{attributelist}$ (relation1) $\cap$ $\Pi_{attributelist}$ (relation 2)

→ $R \cap S = S \cap R$

Set Difference

→ it results tuples present in first relation but not in second relation.

→ it is denoted by $-$.

→ union compatibility should exist.

→ Syntax:

$\Pi_{attributelist}$ (relation1) $-$ $\Pi_{attributelist}$ (relation 2

→ $R - S \neq S - R$

vi) Cartesian Product

→ it builds a relation with concatenation of every row in first relation with every row in second relation.

→ Also known as cross product or cross join.

→ it is denoted by X.

→ Syntax:

R X S

vii) Rename

→ Allows to rename the output relation

→ It is denoted by ρ.

→ Syntax:

$\rho_x(E)$ where result of expression E is stored with name x.

e.g.

$\rho_{emp(name, phone)}$ (employee)

eg.

id	name	address
101	Ram	ktm
102	Sita	Patan
103	Gita	Pkr

Relation 1

id	name	address
101	Ram	ktm
201	Gopal	Patan
202	Shyam	Dharan

Relation 2

1) Relation1 ∪ Relation2

id	name	address
101	Ram	ktm
102	Sita	Patan
103	Gita	Pkr
201	Gopal	Patan
202	Shyam	Dharan

2) Relation1 ∩ Relation2

id	name	address
101	Ram	ktm

3) Relation1 − Relation 2

id	name	address
102	Sita	Patan
103	Gita	Pkr.

4) Relation1 x Relation2

id	name	address	id	name	address
101	Ram	ktm	101	Ram	Ktm
101	Ram	ktm	201	Gopal	Patan
101	Ram	ktm	202	shyam	Dharan
102	Sita	Patan	101	Ram	ktm
102	Sita	patan	201	Gopal	Patan
102	Sita	Patan	202	shyam	Dharan

08/23 Sunday

Join Operation (⋈)

→ allows users to combine two relations in a specified way.

→ The join operation can be stated in terms of a cartesian product followed by selection operation.

→ is very important as used frequently while specifying database queries

→ Three types of join
  i) Theta Join (condition Join)
  ii) Equi Join (inner join)
  iii) Natural Join

Theta Join (condition Join)

→ joins two relations together on basis of some comparision operator.

→ θ is a predicate and consists of one of the comparision operator (=, <, >, <=, >=, <>) and specifies a join condition.

→ defined as a cross product followed by selection.

Equi Join (Inner Join)

→ If θ is '=' then join is called equi join.

→ The = condition is performed by primary and foreign keys.

→ The result must include two attributes with property that values of those two attributes are equal every tuple in relation.

e.g.

Sid	bid	date
22	101	2020
58	103	2021

R1

sid	Sname	rating	age
22	Ram	7	45
31	John	9	28
58	Hari	8	30

~~R2~~ S1

Now,

$S1 \bowtie_{S1.sid < R1.sid} R1$

After finding cartesian product, result is:

⇒

sid	Sname	rating	age	Sid	bid	date
22	Ram	7	45	58	103	2021
31	John	9	28	58	103	2021

S1

sid	sname	rating	age
22	Ram	7	45
22	Ram	7	45
31	Joln	9	28
31	John	9	28
58	Hari	8	30
58	Hari	8	30

R1

sid	bid	date
22	101	2020
58	103	2021
22	101	2020
58	103	2021
22	101	2020
58	103	2021

Equi join

$R1 \bowtie_{R1.sid = S1.sid} S1$

Cartesian Product.

R1 sid	bid	date	S1 sid	sname	rating	age
22	101	2020	22	Ram	7	45
22	101	2020	31	John	9	28
22	101	2020	58	Hari	8	30
58	103	2021	22	Ram	7	45
58	103	2021	31	John	9	28
58	103	2021	58	Hari	8	30

Result is.

sid	bid	date	sid	sname	rating	age
22	101	2020	22	Ram	7	45
58	103	2021	58	Hari	8	30

Natural Join

→ is represented by ⋈

→ only relates those rows that are equal on their attribute names.

→ Natural join can be defined formally as below

Let R and S be any two relations and R = {A, B, C, D, E} and S = {A, B, F, G} are attributes of given relations then their natural join is denoted by R ⋈ S and is defined as follows:

$$R \bowtie S = \pi_{R.A,\ R.B,\ C,\ D,\ E,\ F,\ G}\left(\sigma_{R.A = S.A \wedge R.B = S.B}(R \times S)\right)$$

→ So natural join can be computed by using fundamental operations: cartesian product, selection, projection.

e.g.

empid	name	deptname
101	Ram	IT
102	Sita	Finance
103	Mari	IT

deptname	manager
IT	Rita
Finance	Shyam
Marketing	John

employee ⋈ department.

$\Pi_{empid, name, deptname, manager}(\sigma_{emp.deptname = dept.deptname}(emp \times dept))$

⇒ Solution

emp x dept.

empid	name	deptname	deptname	Manager
101	Ram	IT	IT	Rita
~~101~~	~~Sita~~	~~Finance~~	~~IT~~	
101	Ram	IT	Finance	Shyam
101	Ram	IT	Marketing	John
102	Sita	Finance	IT	Rita
102	Sita	Finance	Finance	Shyam
102	Sita	Finance	Marketing	John
103	Hari	IT	IT	Rita
103	Hari	IT	Finance	Shyam
103	Hari	IT	Marketing	John.

Result of employee ⋈ department is:

empid	name	deptname	Manager.
101	Ram	IT	Rita
102	Sita	Finance	shyam
103	Hari	IT	Rita.

Assignment operation

→ provides a convineint way to express complex queries.

→ allows us to express large and complex queries in series of small and simple queries so that they can be more easily condensed.

Extended Relational Algebra Expression

→ are used to express the request that cannot be performed with original relational algebra operation.

→ enhance expressive power of original relational algebra.

→ different operations are

i) outer join operation

ii) generalized projection

iii) Aggregate Function.

Outer Join Operation

natural join operation returns only those tuples that have matching value in common attributes of both relations. natural join eliminates tuples that donot have matching value in common attributes.

Outer join is used to display rows in the result that do not have matching values in join columns.

different outer join operations are:

i) Left Outer Join

→ In left outer join of two relations R and S, matching tuples from both relations as well as tuples from R that do not have matching values in common in S are included in result relation.

→ denoted by.

$R \bowtie S$

ii) Right Outer Join

→ In right outer join of two relation R and S matching tuples from both relation as well as tuples from S that do not have matching values in common in R are included in result ~~operation~~ relation.

→ denoted by: $R \bowtie S$

iii) Full Outer Join

→ Here, matching tuples from both relations as well as tuples tuples from both relation (R and S) that do not have matching values in common are also included in result relation.

→ denoted by: $R \bowtie S$

e.g.

Bid	Title	Price	Year
B01	DBMS	500	2018
B02	C++	650	2017
B03	JAVA	800	2020
B04	SEP	700	2019

Book

Rid	Bid	Rname	Rating
R01	B01	Ram	8
R02	B02	Hari	9
R03	B03	Gita	8
R04	B04	Sita	7

Review By

Generalized Projection

→ It extends the projection operation by allowing arithmetic functions to be used in projection list.

→ Syntax:

$$\Pi_{F1, F2 \ldots Fn}(E)$$

Here,

F1 ... Fn represents arithmetic expressions

E represents relational algebra expression.

eg.

Display name, job, and 10% of salary of all employees from employee relation.

⇒ $\Pi_{name, job, 0.1 * salary}(employee)$

Aggregate Functions (Operations)

→ SUM(), MAX(), MIN(), COUNT(), AVG() are aggregate functions used in relational algebra.

→ Syntax:-

$$g_1, g_2 \ldots g_n \mathcal{G} F_1(A1), f2(A2) \ldots Fn(An) (E)$$

Here,

- E represents relational algebra expression.
- each F_i is aggregate function
- each A_i is attribute
- $g_1 \ldots g_n$ is a list of arithi attribute on which to group (can be empty).

eg.

1) Find Average salary of employee.

$= \mathcal{G}_{AVG(salary)}(employee)$

2) Find maximum salary of Programmer

$\Rightarrow \mathcal{G}_{MAX(salary)}(\sigma_{job = "programmer"}(employee))$

3) Find total no. of employee jobwise.

$\Rightarrow$

$_{job}\mathcal{G}_{COUNT(eid)}(employee)$

Modification of Database

→ Mainly three operations for modification of database are insertion, deletion and update.

→ All these operations can be expressed using assignment operator.

① INSERTION

Syntax:

$$r \leftarrow r \cup E$$

where, r represent relation
E represent const. tuple or relational algebra expression.

e.g.

To insert a record {10, "John", "Dharan", "Developer", 35000} in employee relation.

⇒ employee ← employee ∪ {10, "John", "Dharan", "Developer", 35000}

② DELETION

→ Syntax:

$$r \leftarrow r - E$$

where, r represent relation
E represent algebra expression or const. tuple.

eg. To delete records of employee having salary less than 30000 from employee relation.

⇒ employee ← employee − $\sigma_{salary < 30000}$ (employee)

UPDATE

⇒ Generalized Projection is used.

⇒ Syntax:

i) $r \leftarrow \pi_{F1, F2, F3, \ldots Fn}(E)$

ii) $r \leftarrow \pi_{F1, F2 \ldots Fn}(\sigma_{predicate}(r)) \cup \sigma_{non\ predicate}(r)$

eg.

1) UPDATE salary of all employee by 10%

employee ← $\pi_{eid, name, address, job, salary = 1.1 * salary}$ (employee)

2) Update salary of manager to 80000

employee ← $\pi_{eid, name, address, job, salary = 80000}$ ~~(employee)~~

$(\sigma_{job = "manager"}(employee)) \cup (\sigma_{jobs \neq "manager"}(employee))$.

Consider following database

i) employee (eid, name, address, job, salary)

ii) department (did, dname, block)

iii) works (eid, did)

Write relational algebra for following.

1) Find name of employee who works as manager

⇒ $\Pi_{name}(\sigma_{job="manager"}(employee))$

2) List department name and blocks.

⇒ $\Pi_{dname, block}(department)$

3) Display total salary and average salary of employee jobwise

⇒ $_{job}\mathcal{G}_{SUM(salary), AVG(salary)}(employee)$

4) Display employee name and department name

⇒ $\Pi_{name, dname.}(department \bowtie (employee \bowtie works))$

department

5) Find name of employee who works in software ^

a)

$\pi_{name}(\sigma_{dname:"software"}(department \bowtie (employee \bowtie works)))$