Database Management system :-

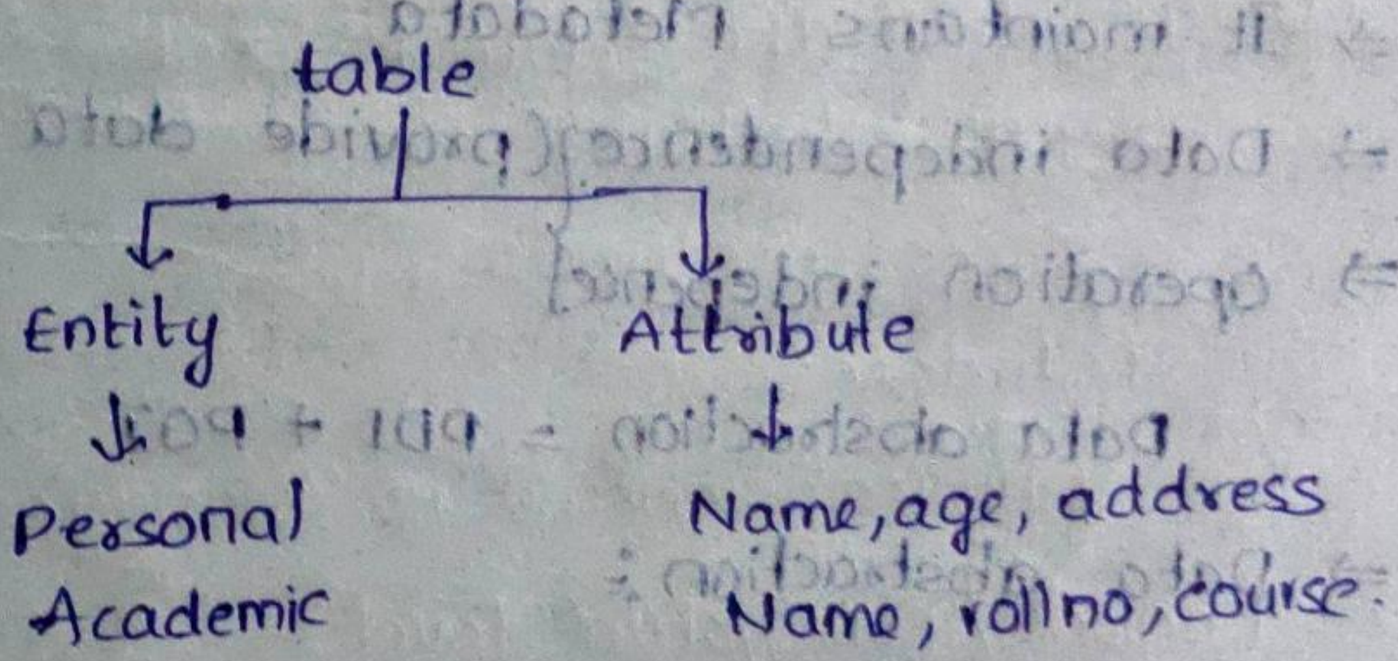
⇒ Data is a raw fact. processed data, is called as information.

⇒ Database is a collection of logically related data and similar data.

i.e., Collection of inter-related data is called database.

⇒ DBMS is a collection of program that manage database structure and controls access to data stored in database.

⇒ Data can be of any size and complexity used to organize data in tables.



Ex:- Microsoft sql server, No sql

Applications :-

Banking

Airlines

Universities

Sales, etc.

Types of Databases & Database Applications :-

Traditional applications:

* Numeric and textual databases.

More recent Applications.

* Multimedia databases (store videos & audios)
* Geographic information systems (GIS) is used to store special data of geographic data (GPS)
* Data ware houses (store analysis)
* Real-time and active databases
  (deals with stock)
  (markets, reservations, transactions online)

Characteristics of DBMS:-

⇒ It maintains Metadata
⇒ Data independence } (provide data abstraction)
⇒ Operation indepence }

Data abstraction = PDI + POI.

⇒ Data abstraction :-

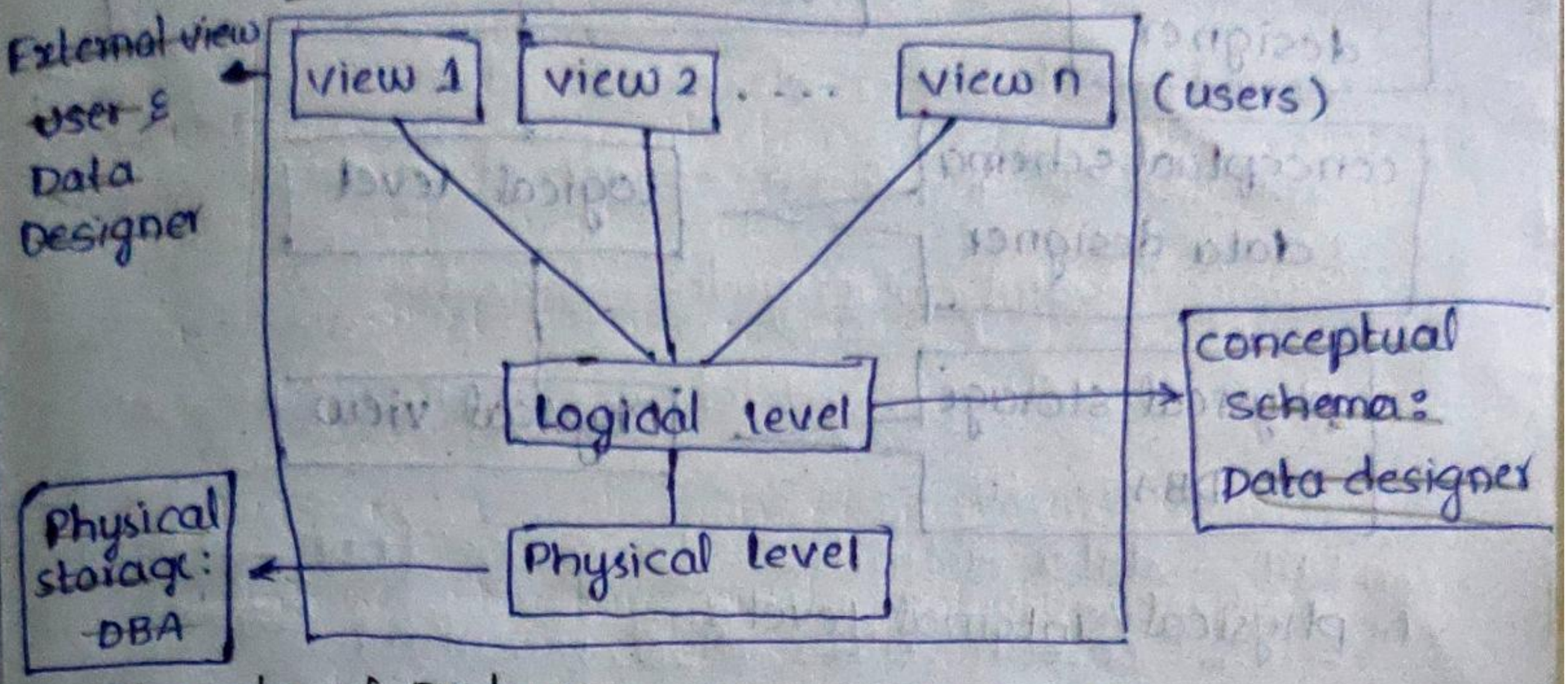
It hides storage details and present the users with conceptual view of database.

⇒ Each user many see a different view of database which describes only data of interest to the user.
⇒ concurrent access & concurrency control through lock mechanism.

⇒ Transaction execution (multi-user transactions)

⇒ Online transaction process.

Three levels of data abstraction



Levels of Data :-

1. Real world data :- type of data exists in real world

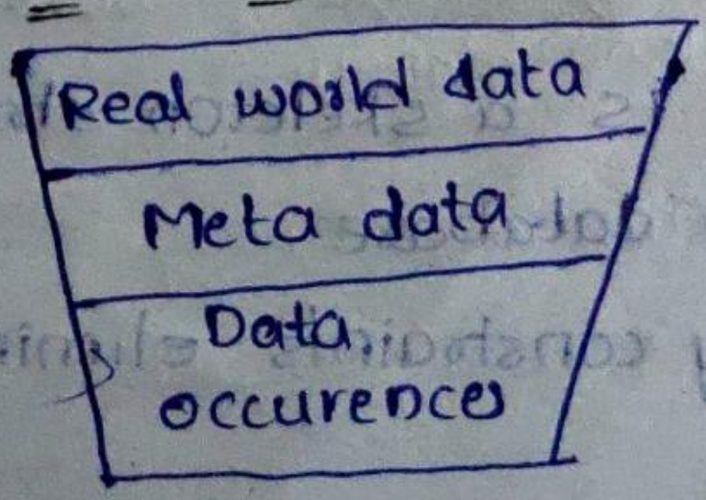
Ex:- any student

2. Meta data :- For storage of data related to any entity or object existing at real world level we define the way the data will stored in database. This is called meta data. It is also known as schema for the real world data.

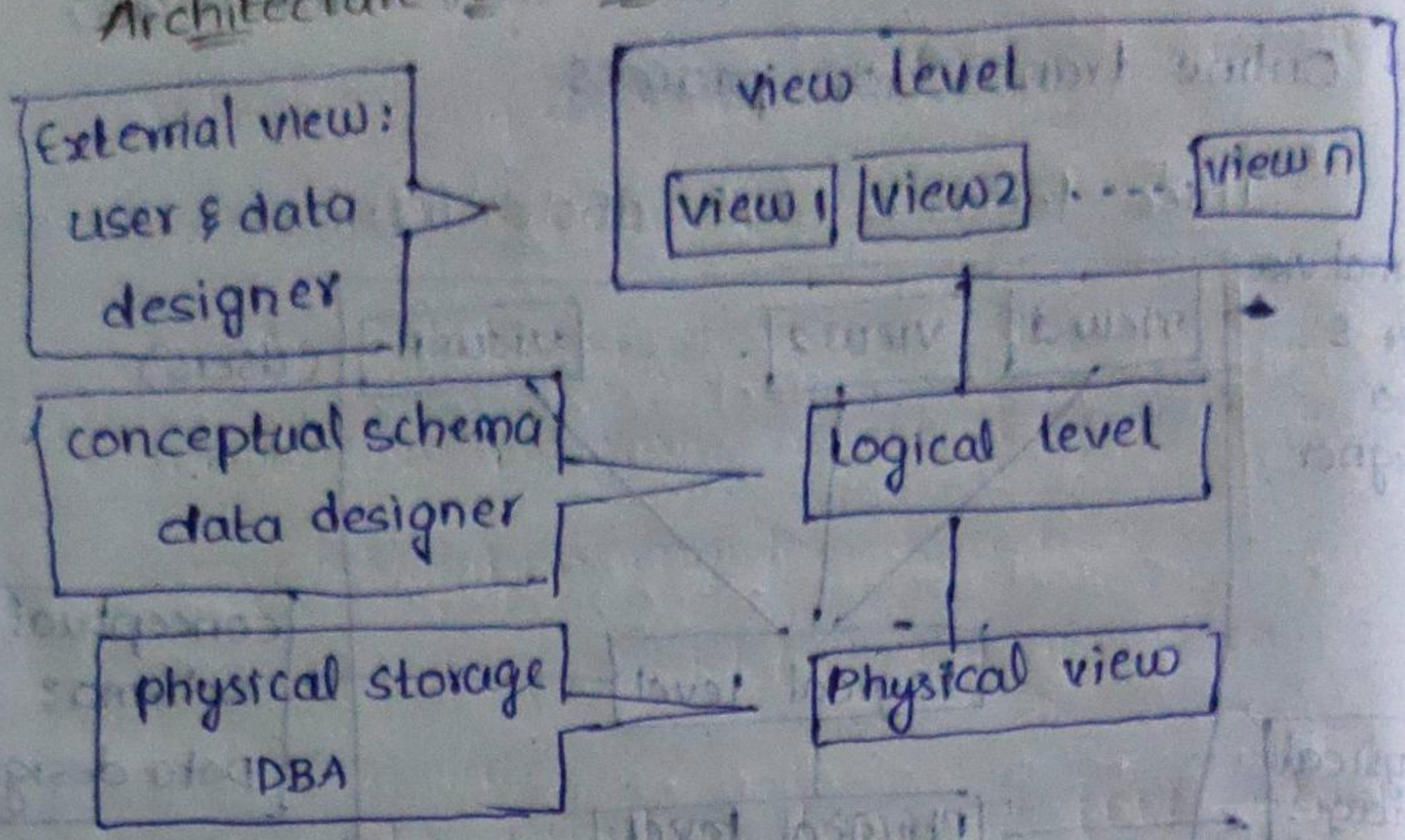
Ex:-

Name	char type	char size
Age	datatype	25 bytes

Levels of data



Architecture of DBMS



1. Physical / Internal level :-

This level describes how the data is stored in the Database.

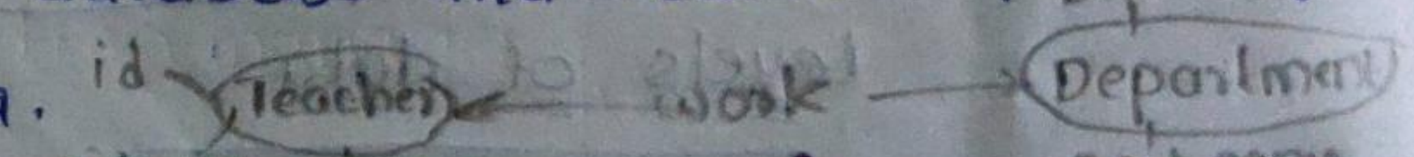
It includes :-

1. where the data is located
2. File structures
3. Access methods
4. Indexes

The physical schema is managed by DBA.

2. Logical / conceptual level :-

This level describes, what data is stored in the database and relationships among the data.



* Schema is a skeleton structure for storing data in database
* Integrity constraints eliminate data redundancy.

view/external level :-

⇒ This level describes that part of database is the relevant to each other

⇒ Each external schema is combination of data tables and views, tailored / tolerent to the needs of single user

⇒ User just access the into or given data.

* various file system in DBMS :-

1. Heap file
(stores data that is of any format / order
2. sequential file (maintains ordering)
3. Hash file (key value type of data)
4. clustered file (It will fetch the data similar to one another.)
(common category in same place)

* Memory storage :-

1. primary (RAM) ↑ speed ↓ storage
2. secondary (HD) ↓ speed ↑ storage
3. teritary ↓ speed ↑ huge volumes of data
(tape libraries)

* Data Access methods :-

1. sequential (data retrieved from table 1 by 1
2. Direct access (file no's - block no's)
access by block no
3. Index access (files - index no)
access by index

Structure of DBMS

⇒ Sophisticated users (analyst) uses query tool to send DML queries.

⇒ Compiler & linker done execution and generate application program object code that generates query evalution engine

(needed to evaluate code)

⇒ DDL - data definition Language (DDL → set of tables (convert))
(create, alter), drop, rename)

- used by DB designers to define schema

→ Eg: SQL

⇒ DML - data manipulation language

- Also known as query language

- insert, update, delete

⇒ DCL - data control language

- grant, revoke

⇒ Storage manager, is, of 4 types:-

1. Transaction manager:-

performs 2 functions

* backup & Recovery
* Concurrency control

Transaction:-

It is a collection of operations that perform a single logical function in DB application

Backup & Recovery:-

Ensure database remains in current state despite failures.

→ System, power, network failures, OS crashes, transaction failures.

Concurrency control:-

Involves managing interaction among concurrent transaction.

2) Buffer Manager

Loads data into main memory disk as it is needed by DBMS and writes it back out when neccessary.

→ It is responsible for:-

* Loading pages of data from disk to mm called buffer; sometime also called cache
* Determines which pages in buffer get replaced writing pages back out to disk
* Decomposition into memory
* page time
* stamps and soon.

3) File Manager:

It is responsible for managing the file that stores data.

⇒ Formatting data files

⇒ Managing free and used space in data

⇒ defragmenting the data files

⇒ Inserting & deleting specific data from file.

4) Authorization & Integrity manager:-

It performs two primary functions

⇒ data security

⇒ data Integrity

File System :-

⇒ A File System is a technique of arranging the files in a storage medium like a hard drive disk, pen drive, DVD etc.

⇒ A file System enables you to handle the way of reading and writing data to the storage medium.

DBMS :-

⇒ DBMS is a software for storing retrieving user's data while considering appropriate security measures

⇒ It consists of a group of programs that manipulate database.

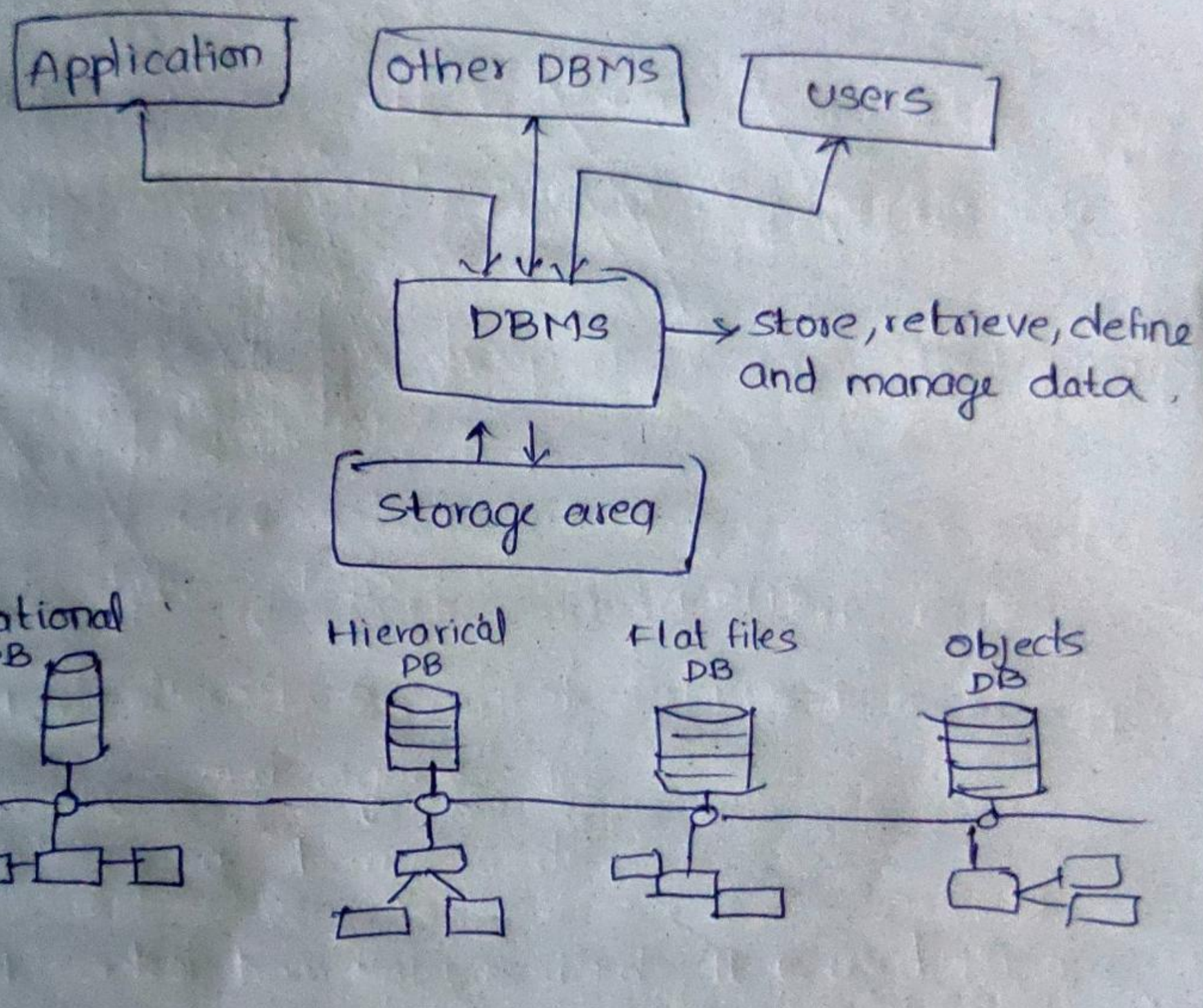
File System	DBMS
⇒ File System is a s/w that manages and organizes the files in a storage medium	· DBMS is a s/w application
⇒ It control how data is stored and retrieved	It is used for accessing creating and managing dB
⇒ The file system provides the details of data representation and storage of data	DBMS gives an abstract view of data that hides the details

⇒ It doesn't offer data recovery processes	⇒ There is a backup recovery for data in DBMS
⇒ the redundancy of data is greather	⇒ low
⇒ incosistency is higher	⇒ low
⇒ less secure	⇒ highly secure
⇒ Not provide support for complicated transactions	⇒ provides
⇒ The centralization process is hard	⇒ easy to achieve.
⇒ No efficient query processing	⇒ you can easily query data in db using SQL Language.
⇒ Applications:- Language specific run-time libraries API programs Managing directories to update the meladata	⇒ Applications:- Accounting system Library system Banking system Airline reservation system Hotel reservation system Finance system.

DB characteristics :-

* self describing nature of DB System.
* DBMS stores a particular description of a particular DB (ex: data structures, types etc)
(mela-data)

DBM Systems

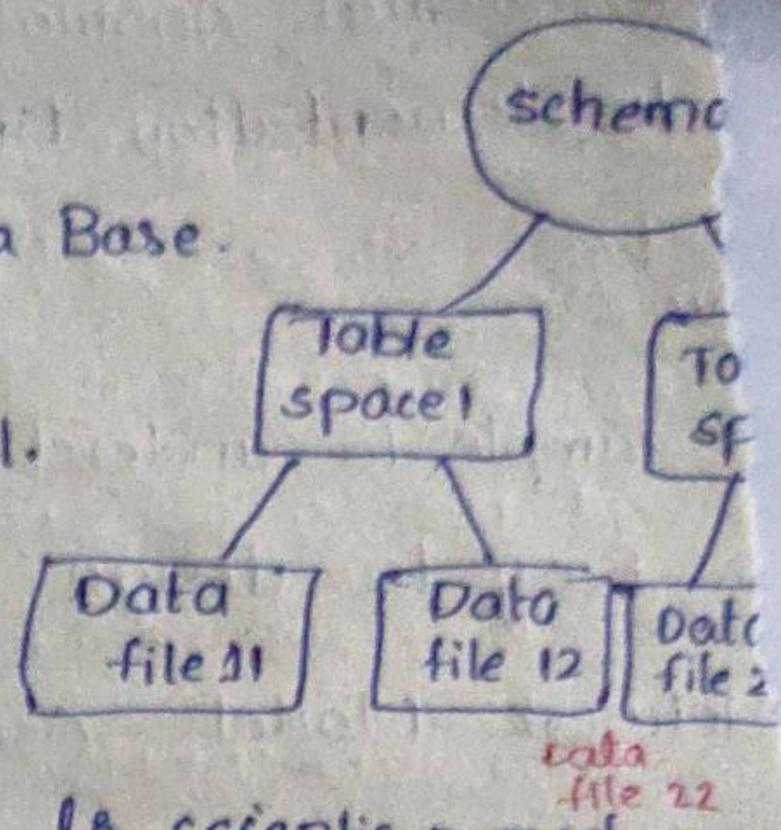


DBMS

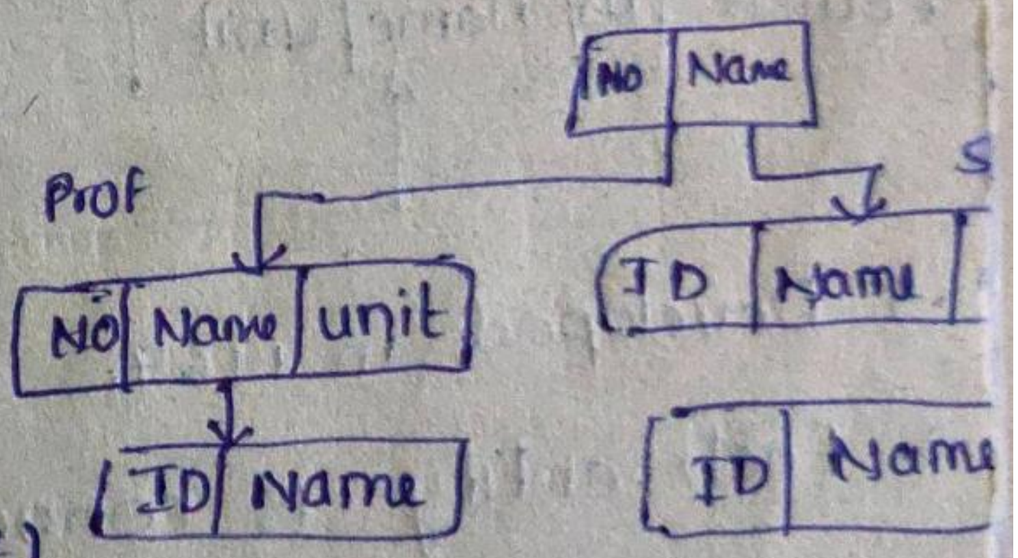
Data Model:-

Logical Structure of Data Base.

⇒ Blue print of the Model.



1. Hierarchical Model:- (1960) for scientic project name
   ⇒ Data is organised as an inverted tree.
   ⇒ Each entity has only one parent but can hav several children.
   ⇒ It was first database model developed to over limitions of the traditional file system.
   ⇒ Deletion of ~~resul~~ parent results in deletion of child records.



Disadvatages:-

1. complexity of Implement-ation (changes making is diff)
2. Difficulty in Management (Moving of data is also c
3. Complexity of programming (find path to access the
4. Poor Portability.
5. Database Management problems.
6. Lack of structural independence. } logical same
7. programs complexity. (data distribution estimati in advanc

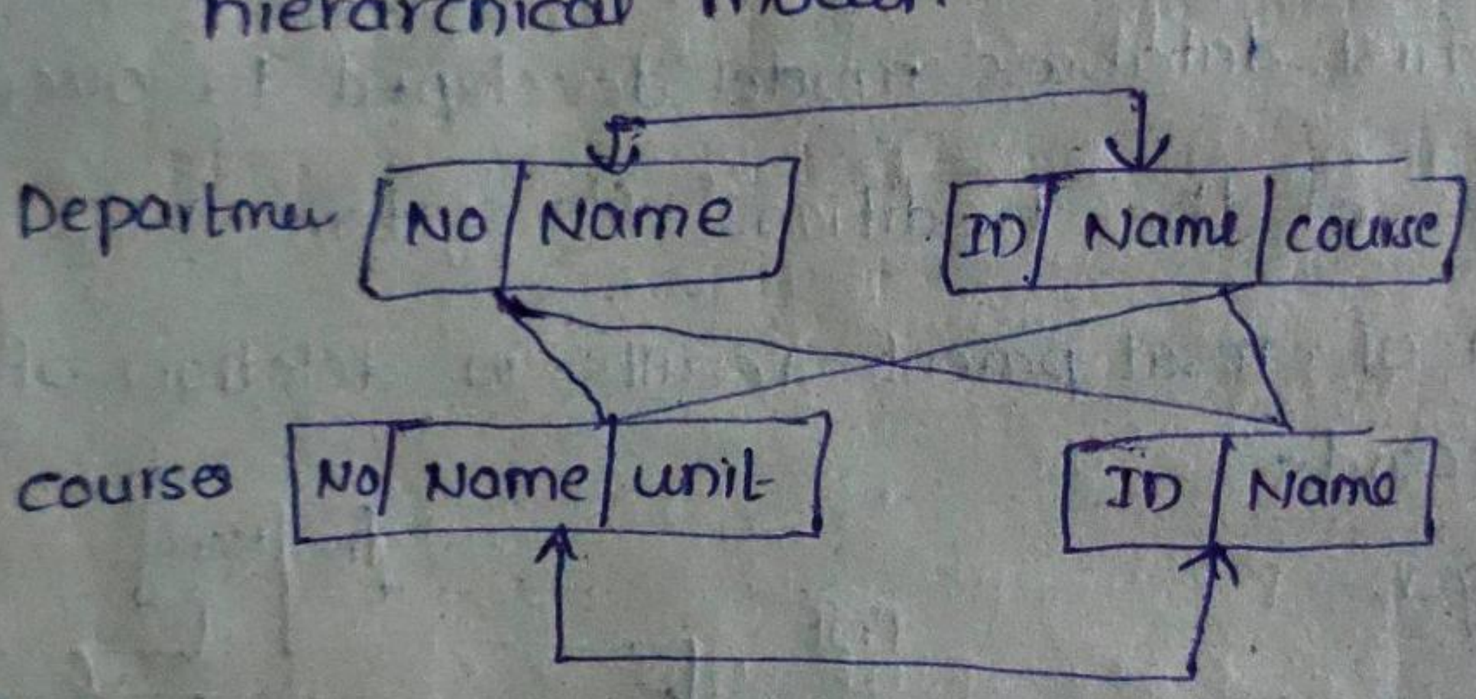
Operational Anomalies. (insert, del etc are diff)
Implementation Limitation. Cone-many relationsh

va

1. Simple to understand
2. Easy to design

Network Model:- (supports many-many relationsh

⇒ Entities are organised in a graph, in which entities can be accessed through several paths
⇒ Developed to overcome the probleme of hierarchical model.



⇒ Developed in 1969.
⇒ Each entity is connected other, entities using pointers.
⇒ might have multiple parents.
⇒ Allow more connection between nodes.

Advantages:-

⇒ Conceptual simplicity

⇒ Ability to handle many relationships

⇒ Easy to access.

⇒ Data integrity (constraints on data

⇒ Data independence

(changes don't effect other entities)

Disadvantages:-

⇒ Complex dealing with pointers (if large, very

⇒ Operational Anomalies :-

(insert, del, update is difficult) bcoz of pointe adj

⇒ Absence of structural independence.

data change — pointer change — address cha

3. Relational Model :-

one-more relations/tabl

⇒ represented by collection of interrelated tabl

⇒ Each table consists :-

row column

tuples/records attribute of entity.

⇒ First proposed by Dr. E.F. Codd of IBM Research 1970.

⇒ Implemented through Relational DBMS.

⇒ many-many.

⇒ Easy to understand

Op vantages:-

Instructural indepence.

Improved conceptual simplicity by concentrating on the logical view

Easier database design.

Ad hoc query capacity - SQL
(creating whenever needed)

Very Powerful DBMS

disadvantages:- (occutionally)

⇒ Substantial h/w and system s/w overhead
(purchase s/w)

⇒ can facilitate poor design and implementation.

⇒ May promote "islands of info" problems.
(unable to connect the relationship among tables)

Entity Relationship Model :-

→ The graphical representation of entities and their relationship among them

Advantages :-

1. conceptually it is very simple
2. better visual represention
3. Effective communication
5. Highly integrated with relation model
6. Easy conversion to any model

disadvantages :-

1. Limited constraints and specification & relationship representation
2. No industry standard for notation.
3. Hidden information.
4. Limited manipulation.

Object Oriented model

⇒ Semantic data model developed by Hammer (meaningful) and McLeod in 1981

⇒ The DBMS developed using this mode is called OODBMS

Adv

* visual representation includes semantic (meaningful) content
* datatypes can be reused (due to inheritance)
* It is quite flexible in most cases.
* easier to extend the design in OO model.

Disadv

* It is not suitable with every database
* Must know OO concept
* not practically implemented in every dB
* Object dB is not popular than RDBMS
* Not many programming language support object databases.
* Object dB do not a standard language.
* Object dB are different difficult to learn for non-programmers.

Object Relation Model / Hybrid Model / Extended Relational mode

⇒ OO Model + Relational Mode

⇒ very efficient model

⇒ allows designers to incorporate obj into familiar table.

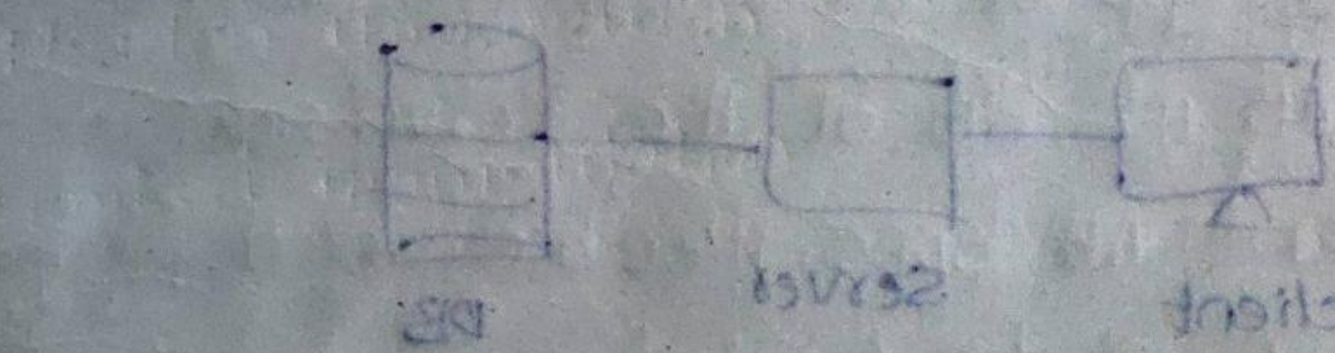
⇒ adopted by any language

Advan

* Reuse and sharing
* Increasing productivity.
* Use of experience in developing RDBMS.

disadv

* Complexity
* Increased costs.



ODBC - Open data base Connectivity.

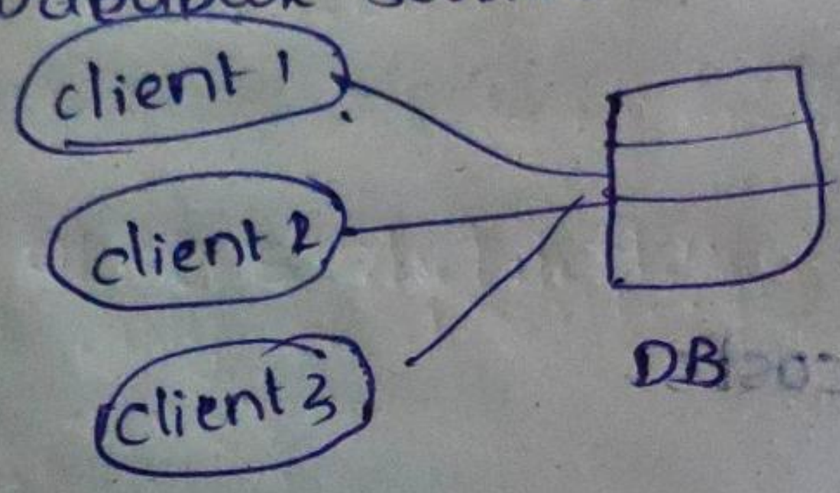
* 1- tier Achitecture

client + server + dB in same machine

Ex:
Installing any dB and writing SQL queries using server.

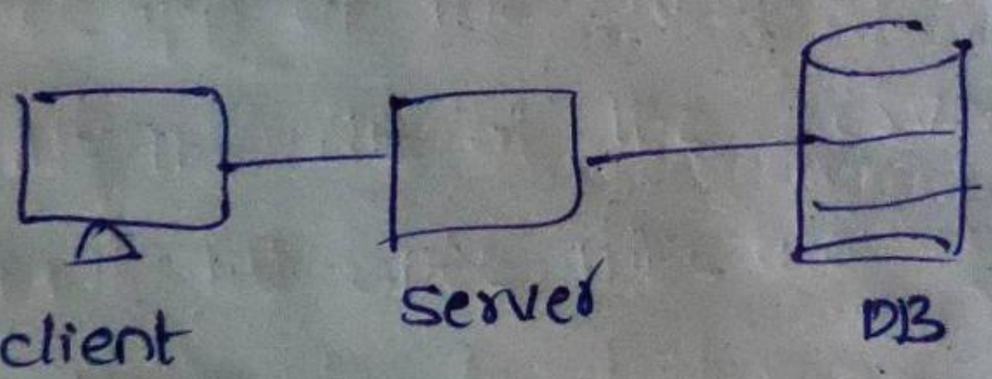
* 2-tier Architecture

1. presentation layer runs on a client (PC, Mobile, Tablet etc)
2. Dababase service



Ex:-
Programming lang UI and executed contact

* 3- tier



1. presentation layer (give data ~~like int~~ through interface)
2. Application layer (server) (mediator blw UI → Bussiness logic & database)
3. Database server.

⇒ Most of the websites use this Architecture.

Ex:- Amazon.

27/1/7

Client - Server Architecture

⇒ It has centralized server

⇒ It is also called as centralized architect-

client :-

* PCs & workstations on which users run applications.
* client rely on servers for resources like files, devices and even processing power

Peer-Peer
(each node has eq responsibi
decentralized
Ex: ubuntu
client-client

*servers

* powerful computers or processes dedicated to managing disk drives (fileservers)
  printers (print servers)
  network traffic (network servers)
* serve the need of the user

⇒ set of services that are provided by servers and a set of clients that use these services

⇒ clients know of servers but servers need not know about clients

⇒ clients and servers are logical processes.

⇒ Reduce traffic congestion

⇒ Fat client (/heavy / rich / thick)

is a Computer in c-s architecture or networks that typically provides rich functionality independent of the central server

Eg:- SMTP, DNS server, FTP (File transfer protocol)
(simple mail transfer protocol) (Domain Naming System)

⇒ Thin client (lean / zero / slim)

an application that depends heavily on another computer (its server) to fulfil its computational roles

Eg: Thunderbird, MS Outlook (used to send mails & chat interfaces)
(mail app developed by mozillah)

Types of C-S Architecture

1. 2-tier

⇒ similar to c-s

⇒ The app at the client end directly comm with the dB at the server side

⇒ API's like ODBC, JDBC are used for intraction.

⇒ 2 types:

1. client application (client Tier),
2. Database (data Client Tier).

⇒ Eg:- SQL queries, but cannot give access multiple clients at a time

Advantages:

⇒ Easy to maintain & can modify

⇒ communication is faster

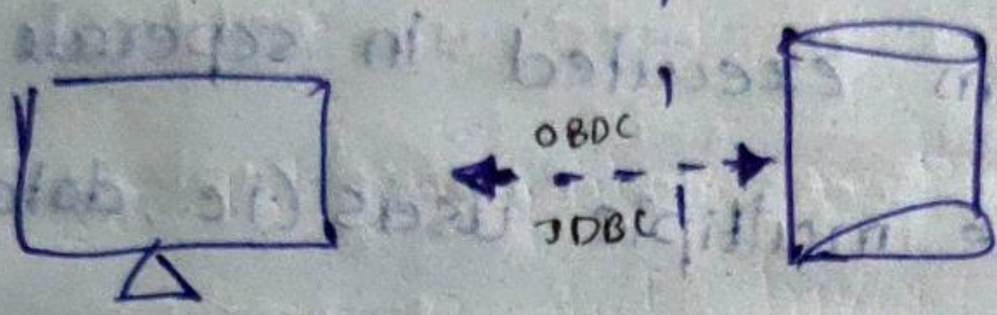
Dis

⇒ app performance ↓ → ↑ users

⇒ cost - ineffective

⇒ limited to simple applications

2. First Tier | 2-Tier



* User Interface	* App services
* presentation services	* Business services
* Application services	* Data services.

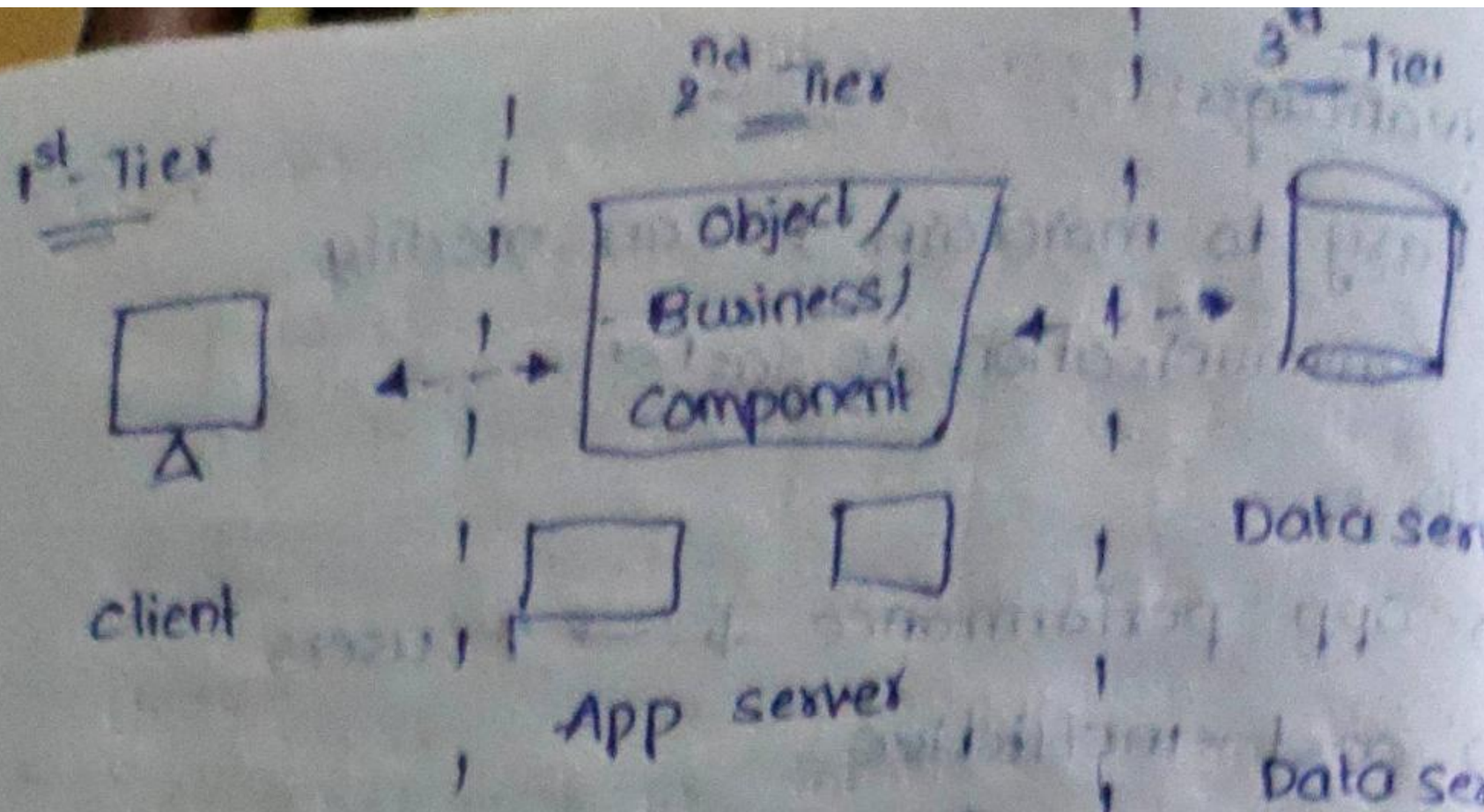
2. 3-tier

⇒ layer b/w client & server called application server, (It will communicates with DB system)

⇒ application server process the queries

⇒ maintains online transactions management.

⇒ used in large Applications / websites.



⇒ Each of them executed in seperate server

⇒ can handle multiple users (i.e., data integrity is there)

Adv

⇒ client only deal with the presentation issue

Advantages of c-s architecture

⇒ All data is stored in servers

⇒ server provide authentication security

⇒ servers can be located anywhere

⇒ we can modify the nodes easily, (nodes are independent) ⟷ request servers

⇒ nodes ie., clients and servers may not be build on the similar platforms yet can faciliate the data transfer

disad

⇒ If no.of clients ↑ at a time then congestion in network takes places.

⇒ If server is failed, all client request will be halted (because all are connected to server)

⇒ The cost of setting and maintaining a client server model is quite high

29/7

2nd Unit

ER Model

used to describe the data involved in a real world enterprise in terms of objects and relationships.

Uses :-

* Can be used in database design by allowing the overall logical structure of a dB
* can graphically represent the data
* can interact the data with entities.

dB design process is divided into 6 steps :-

1. Requirements collection and Analysis :-
   * What type of data can be taken
   * What are the app that we must be built
   * What operations can be used.
     (transactions etc)

   ⇒ gather the info from the user

2. Conceptual dB design :
   ⇒ requirements are converted to ER Model.
   ⇒ Goal of it is a complete understanding of the dB using ER Model

characteristics :-

Expressiveness, simplicity & Understanding
Diagrammatic representation
formality, minimality

3. Logical DB design:

⇒ ER-Model ⟶ table (RDB schema)

⇒ choose DBMS to implement our DB design

4. Schema Refinement:-

⇒ avoid duplication of data

⇒ Identify the retendency

⇒ Using Normalization (i.e., Normal form)

(checking tables for redundancies & anomalies)

5. Physical Database Design:-

⇒ Building indexes on some tables.

⇒ clustering some tables (dividing)

⇒ Redesign of parts of database schema obtained from the previous step

The physical dB design can have

* Response time: elapsed time b/w transaction execution by a dB and receiving a response
* Space Utilization: amount of storage space used by the dB file and their access paths
* Transaction throughput:

avg no of transactions / minute

6. Application & security design:-

⇒ identity different user group & diff roles played by various users

⇒ This will restrict the data to limited users

Student → E.T

ID	Name	Age
S1	Rama	20
S2	Shyam	25
S3	krishna	30

→ Entity :-
thing that exists in real-time distinguishable from others

⇒ Entity Set :-
S1, S2, S3 have same attributes and belong to same table they are called E.S

⇒ Entity type :-
Ex, student → Entity type
same attribute sharing records

⇒ Id, Name, Age are said to be attributes

⇒ Domain :- range for an attribute
If Age : $18 \leq Age < 21$
If months : Jan ≤ m ≤ dec

⇒ Attribute - property given to entity.
↓ represented by
set of attributes

Types of Attributes

1. Simple
which cannot be divisible Ex :- acc no

2. Composite att
can be divided further Ex :- name
& further.

3. single valued :-
single value for Attribute Ex :- age

4. multivalued
   Having multiple values for attribute
   Ex:- degree

5. derived attribute
   can be derived from another attribute
   date of birth ⟶ age
   (can be derived)

6. stored attribute
   canot derived from another stored attribute
   Ex:- date of birth.

* Primary key :-
  Unique data (cannot have null value)
  Ex:- account no.

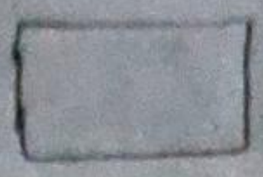
* Weak Entity :-
  can only exist when owned by another o

  Ex:- room (indenpent on space)
  sim, ph, trae car cannot exists without tier

* Strong Entity :-

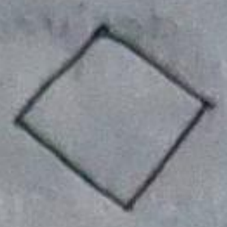
  Ex:- tier can exists without car, soil.

The Symbols that can be used in this model are as follow.

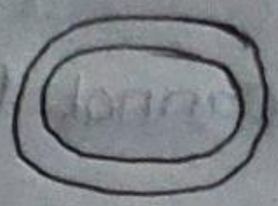
1. Rectangle  : Entity

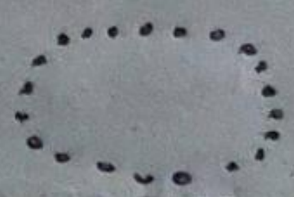
2. Elipses  Attribute

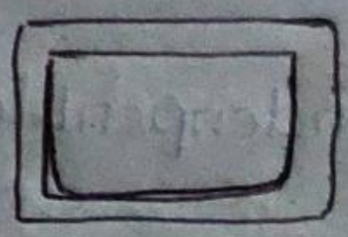
3. lines  Links

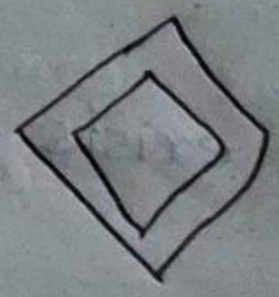
4. Diamonds  relationships

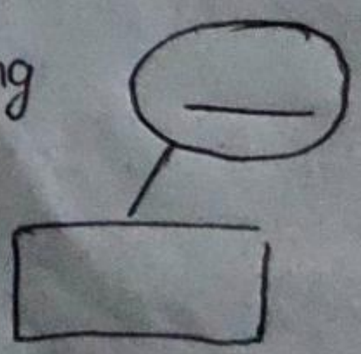
5. Under Elipse line  primary key

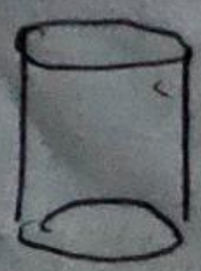
6.  Multi-valued attribute

7.  Derived attribute

8.  weak Entity set

9.  Entity Relationship
Identifying Relationship

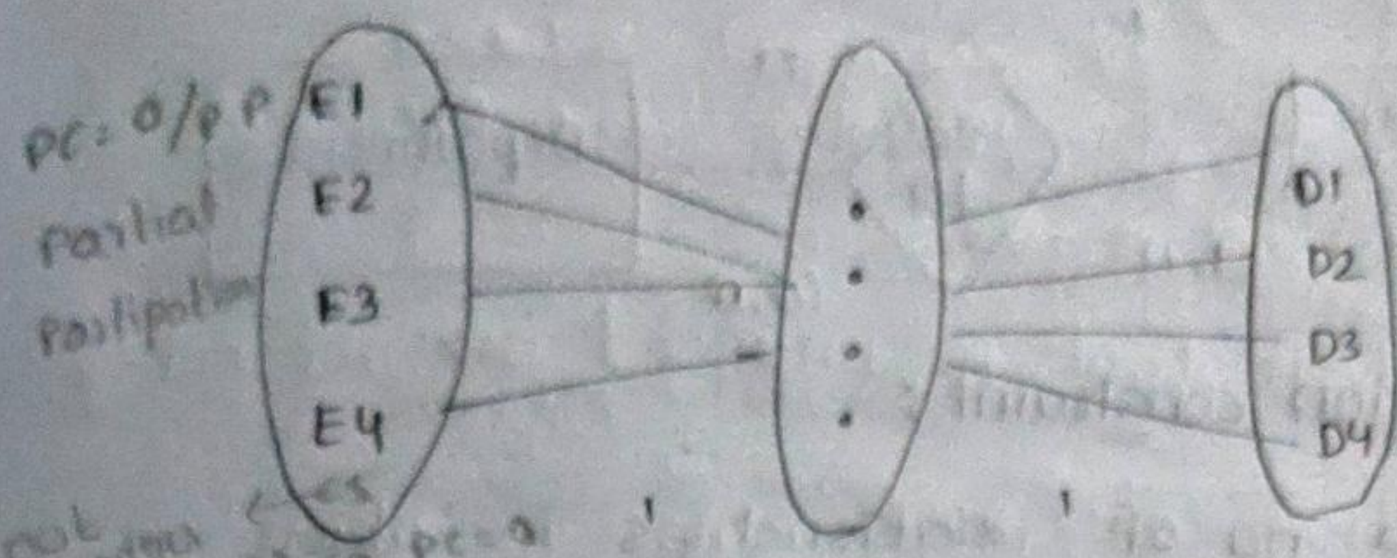
10 Entity set having primary key  Strong Entity set

11.  DB

12  End users.

6/8/20

⇒ Employee — Manages — Department



PC: 0/P.P
Partial participation

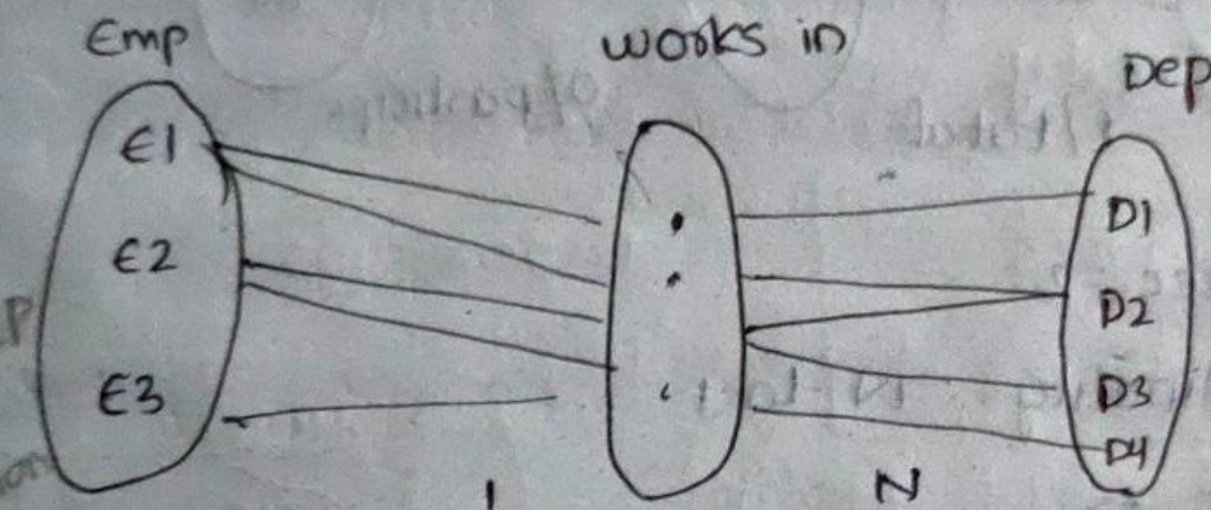
PC: 1/P.P
total participation

⇒ Degree: 2 (no of entities)

Cardinality/C-ratio: maximum no of relationship in which an entity can participate.

⇒ cardinality = 1 to 1 ⇒ participation constraint (PC)

⇒



PC = 1/P.P
total Participation

PC = 0/P.P
Partial Partipation

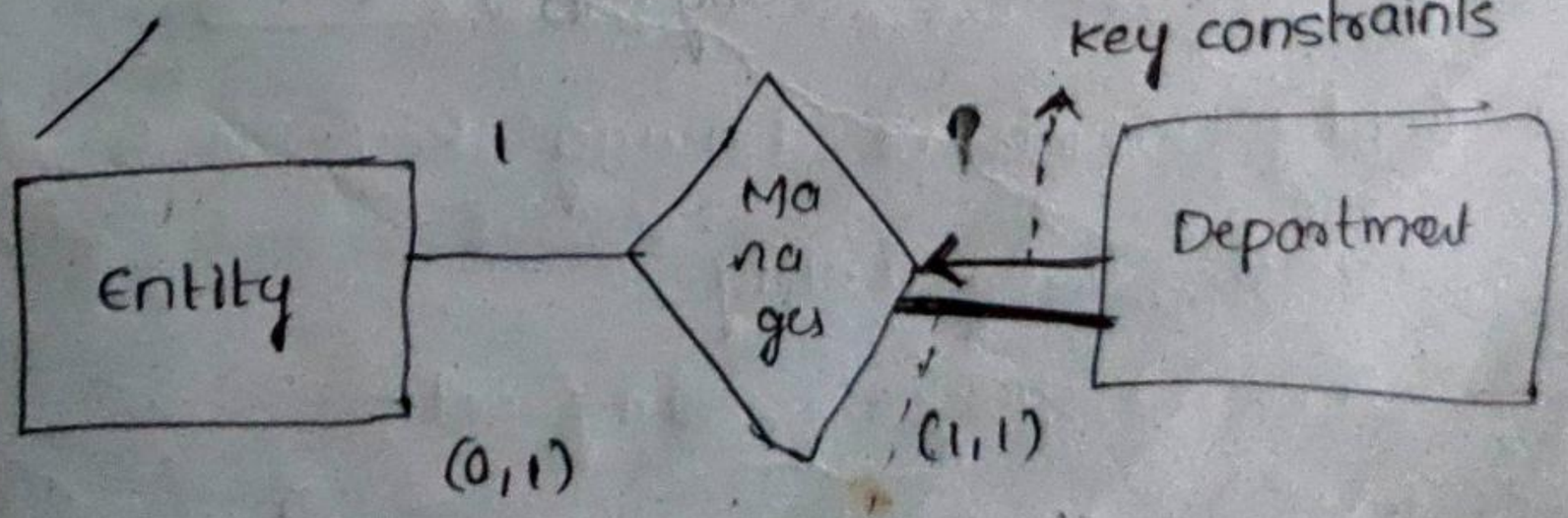
every emp won't be in every department.

Degree: 2

Cardinality = 1 to Many / 1 to N.

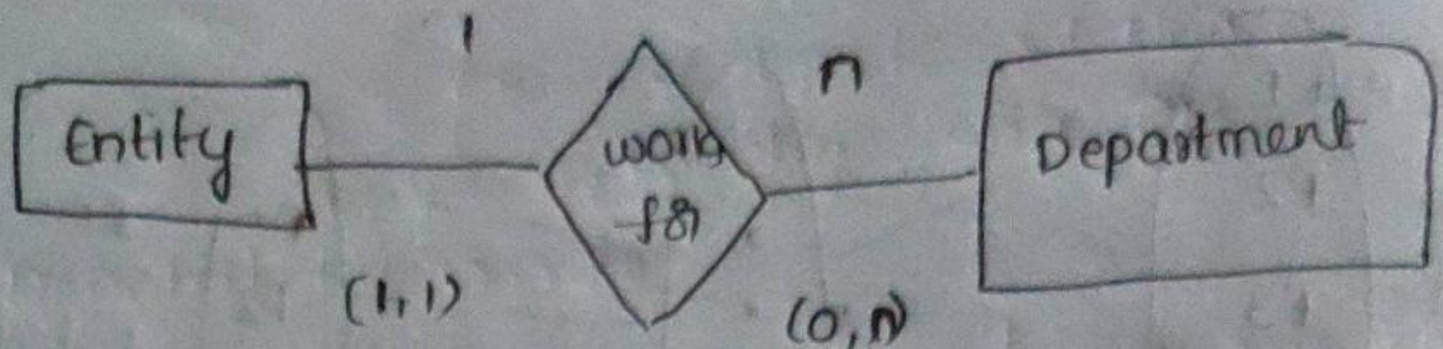
participation constraint =

ER diagram (1-1)



participation constraint
(Bold line)

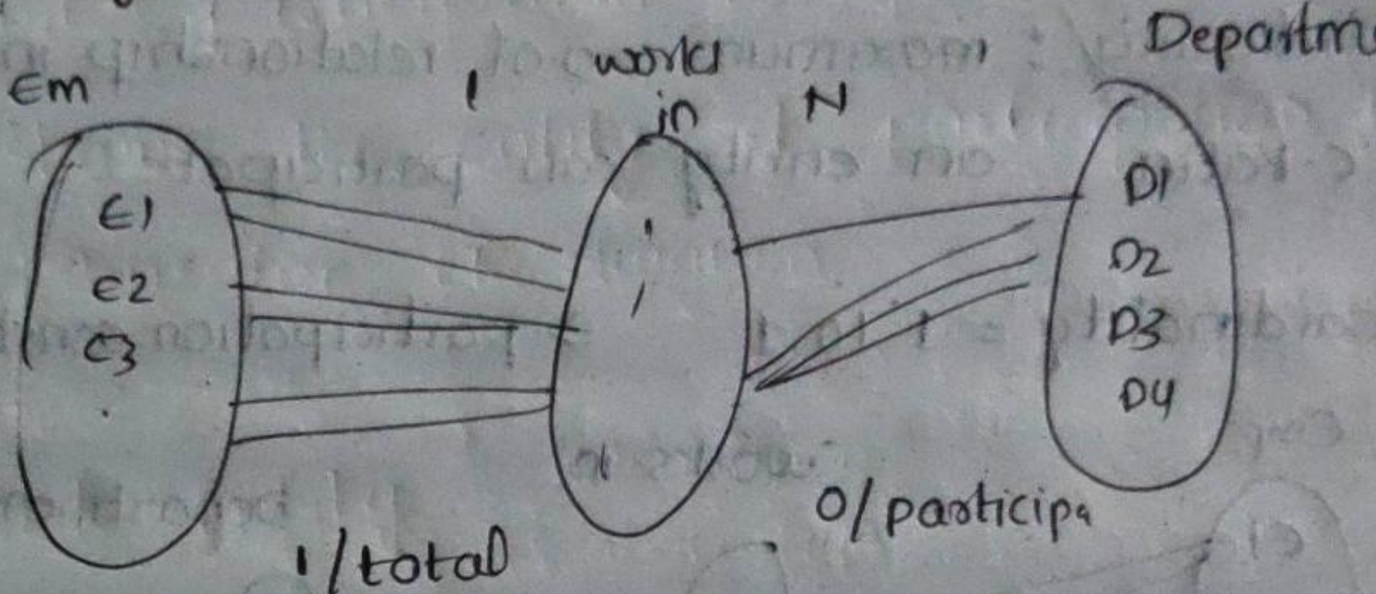
ER diagram :- (1-n)



participation constraint :

Minimum no. of relationships in which an entity can participate.

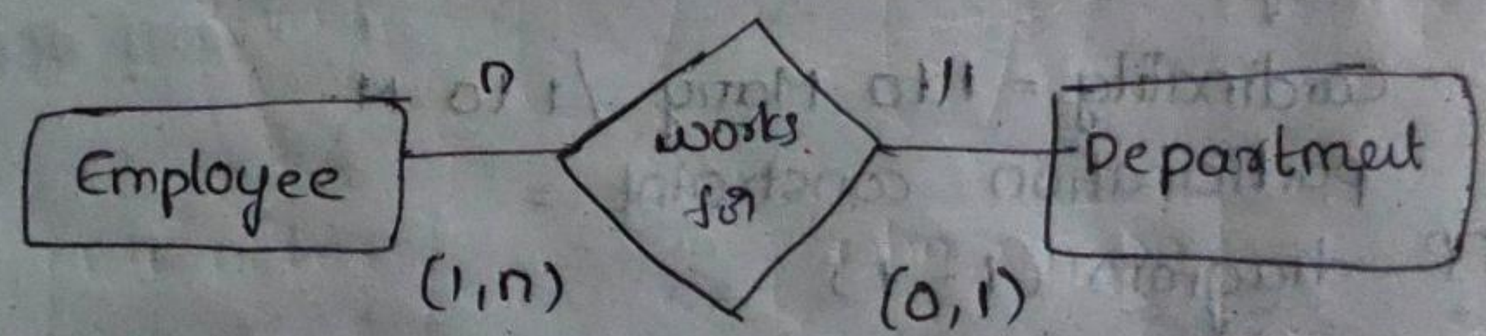
⇒



Degree : 2

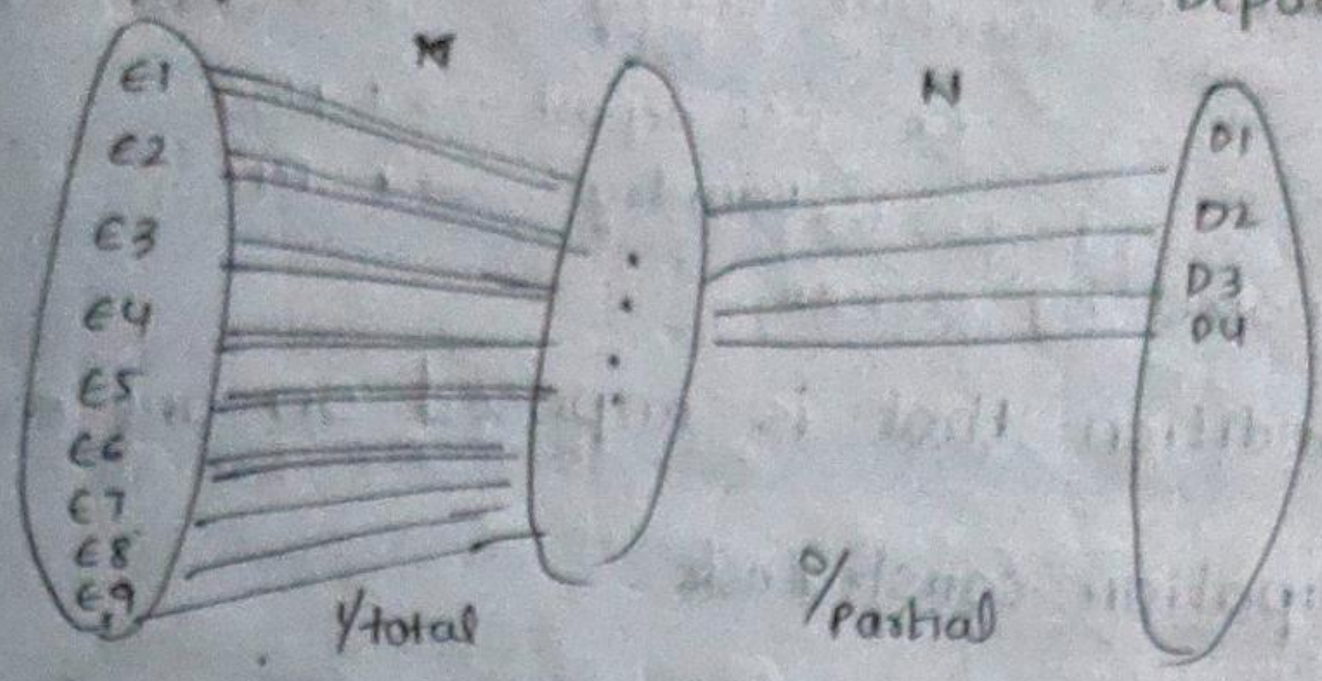
cardinality : 1N to 1

PC ⇒



Er diagram for many to 1

Many - Many



E1, E2, E3 → D1

E1, E2, E4, E5, E6 → D2

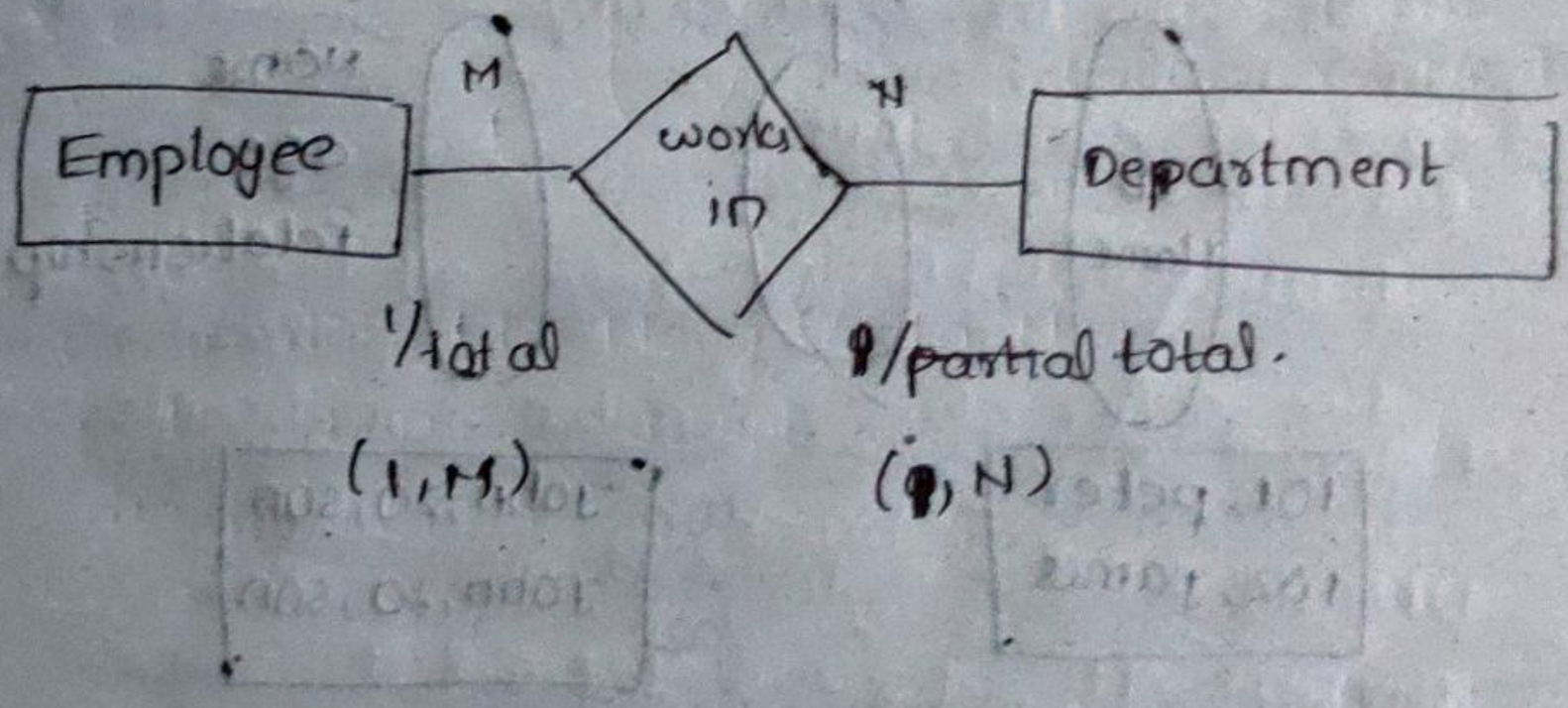
E2, E3, E7, E8 → D3

E9 → D4

Degree = 2

cardinality : M-N

participation constraint;



Key constraints

If particular — One entity → one attribute

Ex:- CSE depart → 1 HOD

Country → 1 PM

⇒ Denoted by arrow

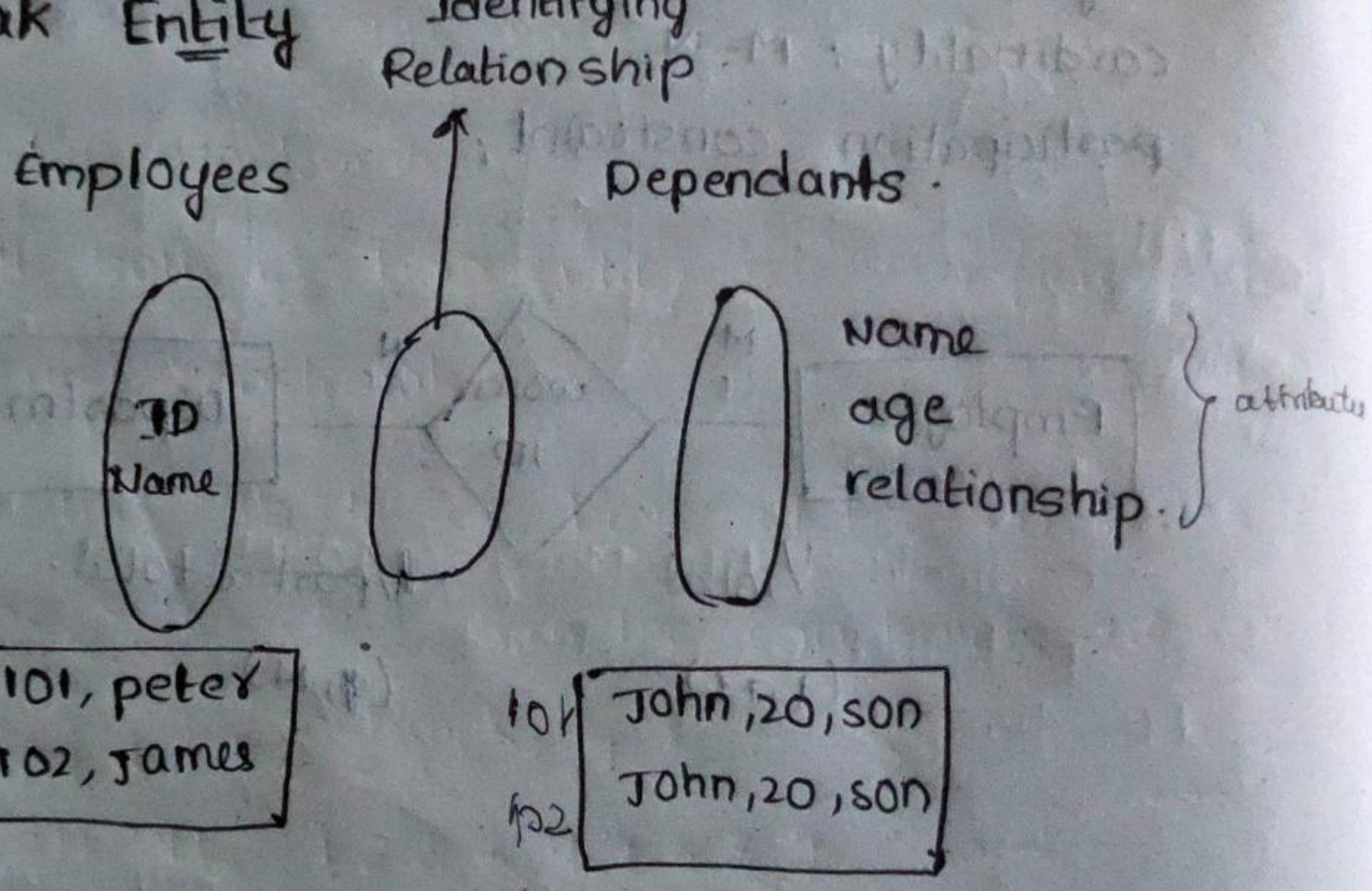
13/8 ⇒ condition that is imposed on one entity set

Participation constraints

⇒ Total participation is represented by bold line

⇒ Shows the entity set participation in a particular ER diagram

Weak Entity



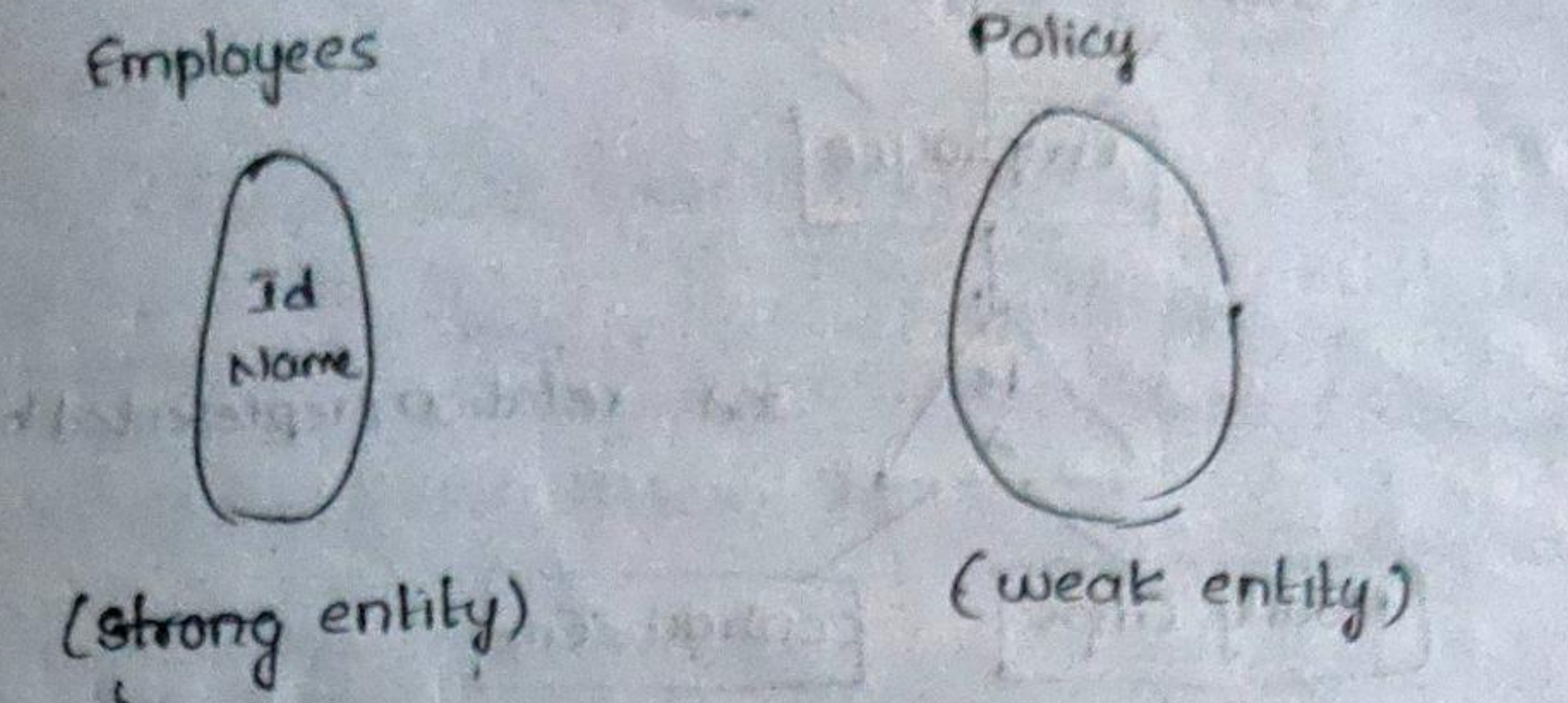
⇒ Entity which is having key attribute, it is called strong Entity

i.e., Employees entity set

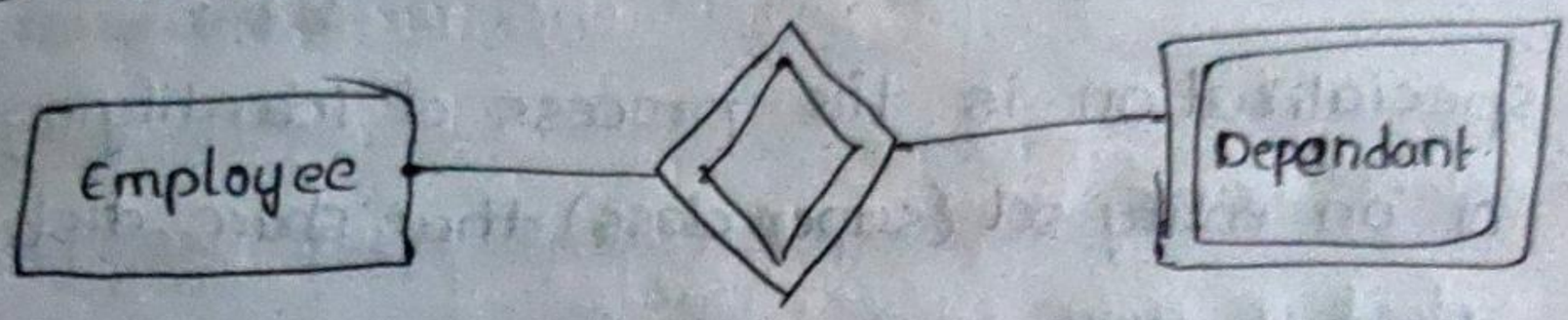
⇒ If any entity set is not have key attribute, that entity is said to be weak entity

i.e., Depedants entity set.

⇒ Weak entity set is dependent on strong entity set.



ER diagram

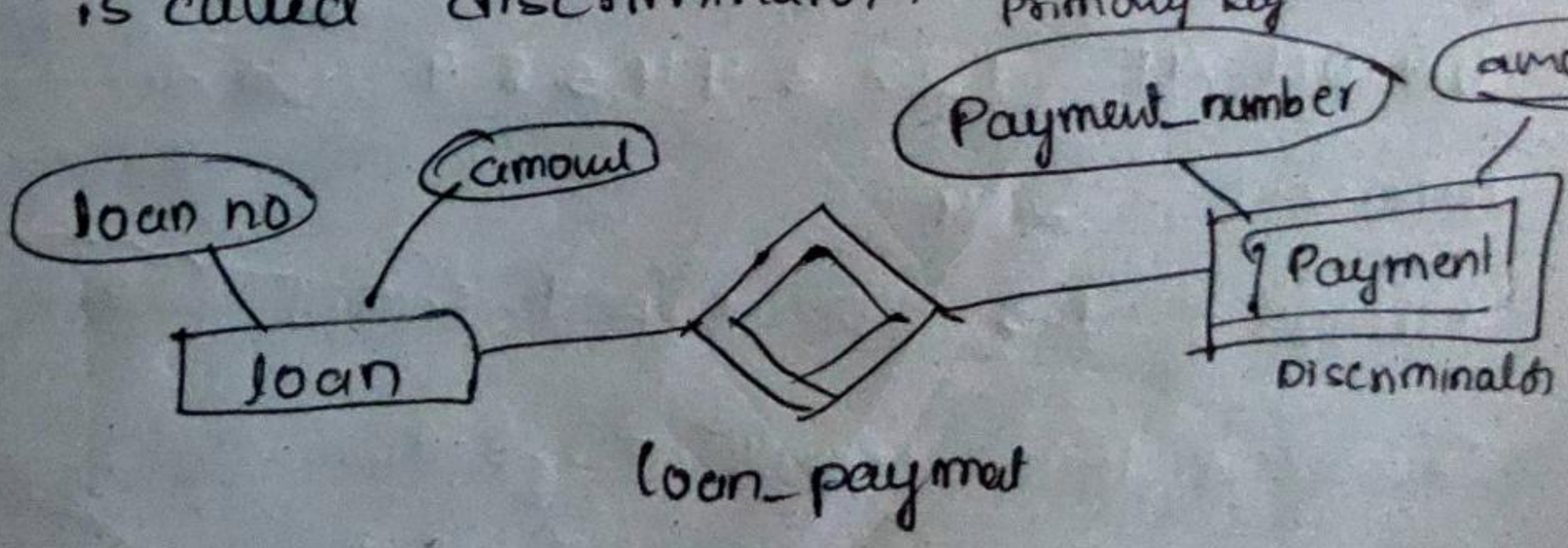


⇒ weak entity can be identified uniquely only by considering the primary key of another entity.

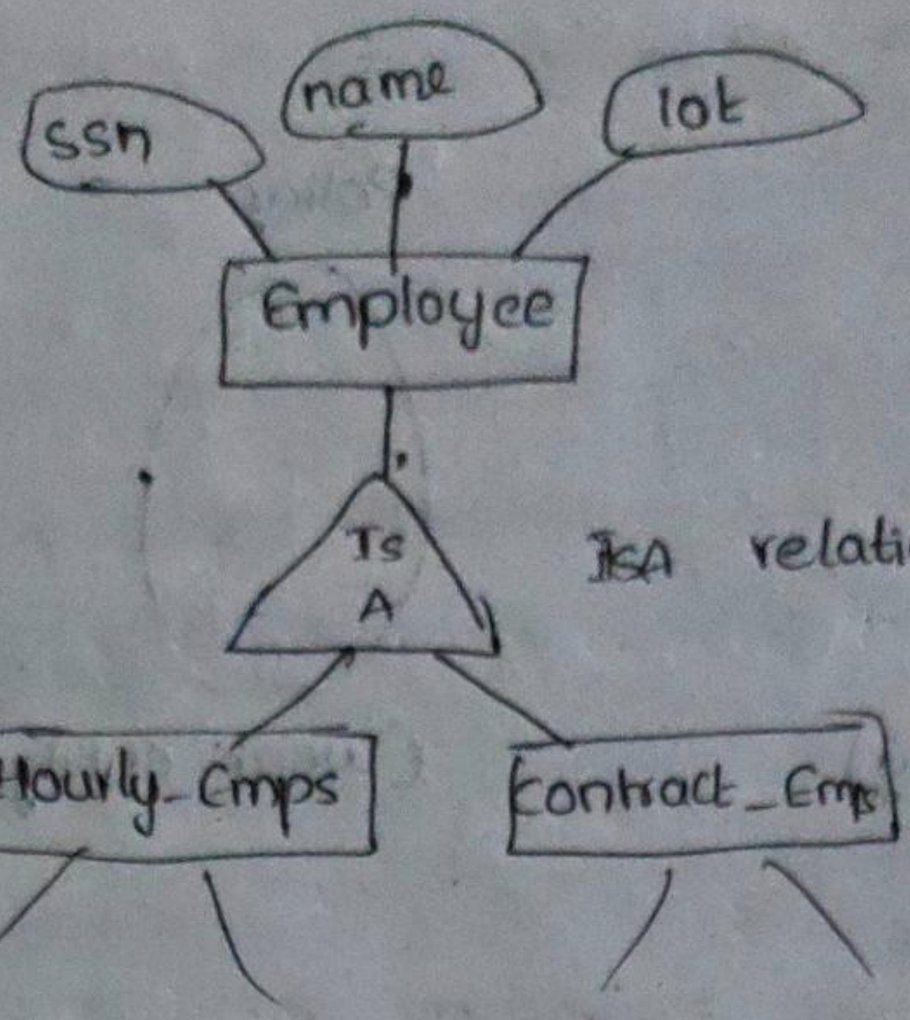
⇒ weak entity set must have total paraticipation in this identifying relationship set.

⇒ discriminator

A weak entity set having the key attribute is called discriminator.



ISA (is a) Hierarchies



ISA relation (represented by Δ)

⇒ specialization is the process of identifying subsets of an entity set (super class) that share distinguishing characteristics.

⇒ overlap constraints

If both ir both entity sets (complete participation may & may not happen)

⇒ covering constraints

If the sub classes have all the attributes of the super class, then the constraints are called covering constraints.

21/8
9/9

* desc emp;
* to insert date into the table.

insert into emp values <501, 'John', TO_DATE <'2020-9-9', 'yyyy-mm-dd'>>;

* select -3 from dual
* select : mid1 + mid2 from student
* conditions. (which)

select mid1 + mid2 from student where percentage>80;

* update

update student set mid1=mid1*1.1 where percentage=-1

* concatenation || & CONCAT function.

select CONCAT ('CONCAT (EName, 'is a ') job) from stud where per>80;

select 'Good' || ' ' || 'morning' || 'Everyone' from dual.
(space)

* add attribute

alter table student add nick char(20);

* value for only one attribute

~~insert into student (nick) values('rr');~~

Update Studet set address = "Hyd" where no=1;

Logical

AND
ARE
Not

Not

select * from emp where not (job IS NULL)
(marks BETWEEN
400 and 500);

AND

Select * From emp where sno=10 AND percentag=90;

OR

10/9

comparsion.

⇒ select, name from emp where name, like 's%';

⇒ select * from emp where name in ('sarath', 'Asish');

⇒ " " " " " " not in ('sarath');

⇒ select age from emp fron where salary is not null;

⇒ All

⇒ Between

⇒ exists

Set Operators

UNION : 1 2 3 4 6 8 9

UNION All : 1 2 2 3 4 4 6 8 9 (consider duplicates also)

Intersect : 2 4

MINUS : 1 3 (or ~~Expect~~ except (not work in java, oracle Server)

Query A has records: 1,2,3,4

Query B has records: 2,4,6,8,9

Queries on sel cpe

desc table_name
structure of table

UNION

select * from

select fn, ln from em
UNION
select fn, ln from en ;

Clauses / Conditions - (3) order by, grup by, having

⇒ used to apply any conditions where

1 Order by

sort the data in some order

ASC - Ascending order (it is by default also)

DESC - Descending order.

⇒ select * from t_name ORDER BY column_name ASC|DESC

(sorting for single column)

⇒ multiple columns

select * from t_name order by c1 ASC|DESC,
c2 ASC|DESC

2. GROUP BY

⇒ used to arrange identical data into groups with the help of some functions.

⇒ placed after where clause.

⇒ Group by is placed after order by (if needed)

⇒ to delete

delete from emp where name = 'As';

→ group by

* select name, sum (salary)

From Student

group by name

* select age, avg(salary)

from student emp

group by ~~name~~ age

* seled subject, year, count(*) from stu group by subject, year ;

(grup by on multiple attributes)

3. Having

is used to place conditions on ~~columns~~ groups

select c1, f_name(c2)

from t_name

where condition

* select name, sum(salary) from emp group by name having sum(salary) > 3000

16/8

=> +

select +3 from dual;

=> -

select -4 from dual;

=> / * +

select sal / 10 are

character Operator

||

=> real datatype for points

=> To avoid duplicates

select distinct sname from sailors;

Multiple tables

=> Find the names of sailors who has reserved boat no 103

* select sname from sailors S, Reservers R where S.sid = R.sid and R.bid = 103

* select sname from sailors, Reservers where sailors.sid = Reserves.sid and reserves.bid = 103.

⇒ Find the id's of sailors who have reserved red boat?

```
seled sid from Reserves R, Boats B where
R.bid = B.bid and B.colour = 'red';
```

⇒ Find the name of sailors who have reserved Red boat?

```
select sname from Reserves R, Boats B,
Sailors S where S.sid = R.sid and
R.bid = B.bid and B.colour = 'red';
```

⇒ Find the colour of the boats reserved by Lubber

```
select ~~sname~~ B.colour from Reserves R, Boats B,
Sailors S where S.sid = R.sid and
R.bid = B.bid and S.sname = 'Lubber'
```

⇒ Find names of sailors who have reserved altcast, one boat

```
Select S.sname from Sailors S, Reserve R
where S.sid = R.sid
```

21/9

A

⇒ select S sname from sailors S, Reserve R, Boats B
where S.sid = and R.Bid and R.bid = B.bid
and B.color = 'red' UNION
select S1.sname from Sailors S1, Reserve R1
Boats B1 where S1.sid = R1.sid and R1.bid =
B1.bid and B1.color = "green';

2. ⇒ select R.sid from Reserves R, Boats B where
R1.bid = B.bid and B.colour = 'red MINUS
select R1.sid from Rserves R1, Boats B1 where
R1.bid = B1.bid and B1.color = 'green';

⇒ select S.sid from Sailors S where S.rating = 0
UNION select R.sid from Reserves R where
R.bid = 103;

⇒ "
R.bid = 104;

Sub-query

4. select S.sname from Sailors S, Reserves R where
S.sid = R.Sid and R.bid = 103;

θ1 {where/ com op} (22)

select S.sname from Sailors S where exists
where S.sid in < select R.sid from Reserves
R where R.bid = 103 >;

7 ⇒ select S.sname from Sailors S where S.sid
in < select R.sid from Reserves R where
R.bid in < select B.bid from Boats B,
where B.color = 'red' >> ;

8 ⇒ exists
select S.sname from Sailor S where
EXISIS < select * from Reserve R where
R.bid = 103 and R.sid = S.sid >;

9 ⇒ NOT EXISIS

10 ⇒ select S.sid from Sailors S where S.rating
> any < Select S1.S.id rating from Sailors S1 where
S1.sname = 'Horatio' >;

⇒ " > all < "

⇒ select S.sqid from sailors S where S rating
>=all <select S1.rating from sailors S1) where

22)?

AVG

⇒ select AVG<age> from sailors

⇒ select AVG<age> from Sailors where rating=D

⇒ select S.sname, S.age from sailors S where S.age =< select Max < S1.age > from sailors S1>;

⇒ select S.sname from Sailors S where S.age> <select Max < S1.age > from sailors S1 where S1.rating=10>;

⇒ select count<DISTINCT sname> from Sailors ?;

⇒ select count<sname> from Sailors;

Min<

⇒ select S.age>, S.rating from Sailors S group by S.rating

⇒ select S.rating, MIN<S.age> as Minage from Sailor S group by Sai S.rating;

⇒ select ~~S.rating~~, Min<S.age> as Minage, S.rating from Sailor S group by S.rating;

⇒ select Min<S.age> as Minage, S.rating from Sailor S where S.age>=18 group by S.rating having count<*>;

Types of join

fetch the data from two or more tables.

* Inner (cross, Theta, Natural)
* Outer (Left, Right, Full)
*
*

⇒ cross join / cartegian product.

select column_name_list

Ex: select * from student class JOIN student_info;

1	tej	1	Hyd
2	dell	2	delhi

⇒ Theta join

1	tej	1	Hyd
2	dell	2	delhi

join Based on Condition

⇒ Equality then Equijoin.

select * from student INNER JOIN student_info where student.id = student_info.id;

⇒ Natural join

column name should have some datatypes

select * from student NATURAL JOIN student_info;

Left → first table full
right → second

⇒ Outer join

left

select * from student LEFT OUTER JOIN student_info ON student.id = student_info.id;

right

Left replace right.

full

Left replace FULL

28/9

view:-

1. create view

create view sail_view as select sname, sid from sailors where rating > 6;

2. view on multiple tables

create view new_view as select Sailors.sname, Sailors.sid, Reserves.bid from Sailors, Reserves where Sailors.sid = Reserves.sid;

ID	Name
1	Abhi
2	Akhil
3	Harika
4	Monal

ID	Address
1	Banglorey
2	Delhi
3	mumbai
4	Hyderabad

Cross join

ID		
1	Abhi	Banglorey
2	Akhil	Delhi
3	Harika	mumbai
4	Monal	Hyderabad

Inner join

1	Abhi	Banglorey
2	Akhil	Delhi
3	Harika	mumbai
4	Monal	Hyderabad

Natural join

1	Abhi	Banglorey
2	Akhil	Delhi
3	Harika	mumbai
4	Monal	Hyderabad

Sol:- Trigger

* create trigger init count Before insert on
students

Declare
count Integer
Begin
count := 0;
end

* create trigger incr count after insert on
students

when (new.age < 18)
For each row / column
Begin
count := count + 1;
end.

Active DB

Event
combine.

UNIT 4

Normalization

⇒ duplicates can be avoided
↓
(data redundancy)

⇒ It is sometime called as schema refinement

schema refinement

⇒ logical structure of dB follows some rules

⇒ avoids data redundancy

Redundancy is in 3 levels:

1. File level ----> f1.txt, f1.txt
2. Entire Record level → sno name marks

sno	name	marks
10	ram	90
20	Hari	100

3. Few Attributes level.

10	raju	c1	c++
20	rani	c1	c++
30	suresh	c2	Java

10	ram	90

(entire)

(few)

⇒ Problem/error in redundancy (dan) is called Anomaly

1. update anomaly :-

If update one record → inconsistency in another

2. Insertion aramdy :-

Inorder to insert some info we should have all the attributes in the table.

Delete Anomaly

we cannot a particular attribute value.

Decomposition

diving the tables which must have common attribute

Functional Dependency

* relations exists with 2 attributes
  - primary key
  - non-key attributes

* $X \rightarrow Y$

  Y dependent on X

  X determines Y

determinent ⟶ dependant
FDO

Types :-

1. Trivial FD $\{A,B\} \rightarrow A$; $A \rightarrow A$, $B \rightarrow B$
2. Non Trivial FD ID → name ; name → dob
3. Fully FD
4. partial FD
5. Multivalued FD car → manf year & car → color

2a/11

⇒ Find names of all sailors whose ratin
select sname from sailor where rati
⇒ find bid whose colour is ≠ red.

⇒ Find all the sailors details whose rating
and age > 35

⇒ Find names of sailors whose age lies i
between 30 to 60

*select * from sailors1 where age (between) 40 and
(select sid from reserves1);

⇒ Find names of boats whose color is
either red & green

⇒ Find name id of sailors whose name
starts with 'h'

select sid from sailors1 where sname
like 'h%';

⇒ find age of oldest sailor

⇒ find