Question-17

(1) **Primary key:-** is a selected candidate key that used to uniquely identify an entity from entity set

appropriate primary keys :-

entity		PK
branch	—	branch_name
customer	—	customerName
Loan	—	loanNumber
borrower	—	NO (or combination)
account	—	account_Number
depositor	—	NO (or combination)

(2) Identifying appropriate **foreign keys:** is a set of attributes that refers to primary key of another table in Relational Database

loan → branch_name from branch
branch_name

borrower → customerName from customer
loan_Number from loan

account → branch_name from branch

depositor → customerName from custome
accountNumber from accout

(c)

(i)

```
SELECT   branch_name
FROM     branch
where    branch_name = "Dhaka" ;
         (city)
```

(ii)

DISTINCT → it ask customen / on any names

```
SELECT ^ customen_name
FROME  ~~bno~~ bonnower, loan
where  bonnowen.loanNumber = loan.loanNumber
       and branch_name = "mirpur";
           (city)
```

→ wrong need to add branch, branch name

OR

using join

```
SELECT   DISTINCT  customen_name
FROM     bonnower
Join     loan  ON  bonnower.loannumber
                   = loan.loannwber
where    loan.branch_name = "mirpur"
```

(III)

```
Select loan_numbers
From loan
where ~~taon~~ amoot > 100000;
```

(IV)

```
select distinct customen_name
from depositor, account
where depositon.account-numben = account.account-numben
      and account.balance > 60000 ;
```

(V)

```
select distinct customen name
from depositon, account, bnench
where depositon.accountnumben = account.account numben
      and account.bnanchname = bnaneck.bnanch name
      and bnanch.bnanch-city = " motijheel
      and account.balance > 60000 ;
```

(2.)

Normalization:- is a database design techniqe to reduce data neduandancy and eliminate undesianc abe chanetenistic (Insention, detele)

1NF: → a table cell need to contain a single value

→ each necond need to be uniqu e

2NF → pantial dependeney Not allowed
(Table has multiple primany key and any nonkey depends or one of primany key)

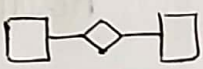
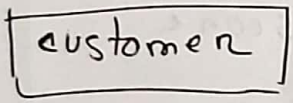
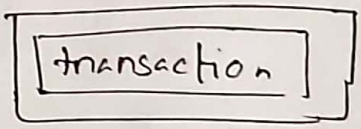
3 NF → transitive dependeney Not allowed
(Non-key depends on Non-key)

member (membershipId, fullname, physial address, salution)
PK NK NK NK
Depends

Solve:-

member (membershipID, fullname, physicaladdness, salutionID)
FK

salutionInfo (salutionID, salution)
PK NK

Strong entity	weak entity
① a strong entity is aentity that doesn't depends on any other entity.	① depends on another to ensure existence
② Independent	② dependent nature
③ has primary key	③ No. but foreign key
④ Relation between two strong entity is nepresented by sole diamond 	④ dual sole diamond 
⑤ Represented by rectangle 	⑤ dual rectangle 

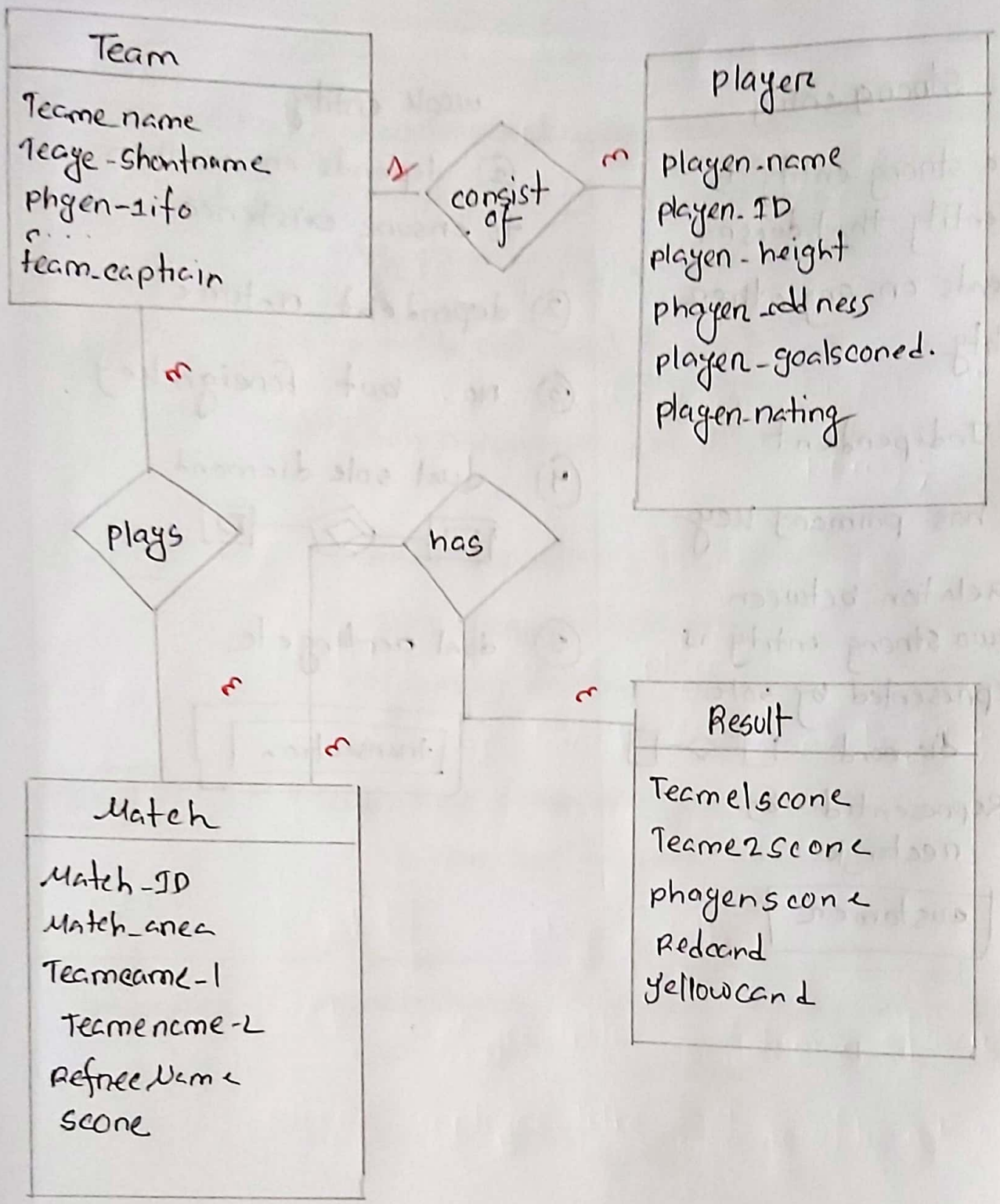


Fig:- ER digram favounite football team

TT-2

① appropriate primary keys

Employee → employee_ID

combination

Bonus → employeeRefID & bonusdate

Titte → employeeRef2D & Affected from

② create Table employee {

employee_ID int PRImary key,
FIRST-name varchar(20),
LAST-name Varchar(20),
Salary Decimal(10,3),
Joining-date DATE
Depantment varchar

);

create Table bonus {

employee-refID int,
bonus_amount Decimal(10,3),
bonous-date DATE,
PRIMARY KEY(employeerefID, bonusdate),
FOReign key (employee-refID) References employee(employee.ID

Foreign key

combination PK

(III)

Insert into employee

(Employee_ID, FIRST_Name, LAST_Name, SALARY,
JOINING_DATE, Department)

Values

(1001, "Abdul", "Qayum", 5000, "1-1-1990",
"HRM");

(IV)

Delete from employee

where department = "marketing";

(V)

Update employee title

Set employee_title = "Junior officer"

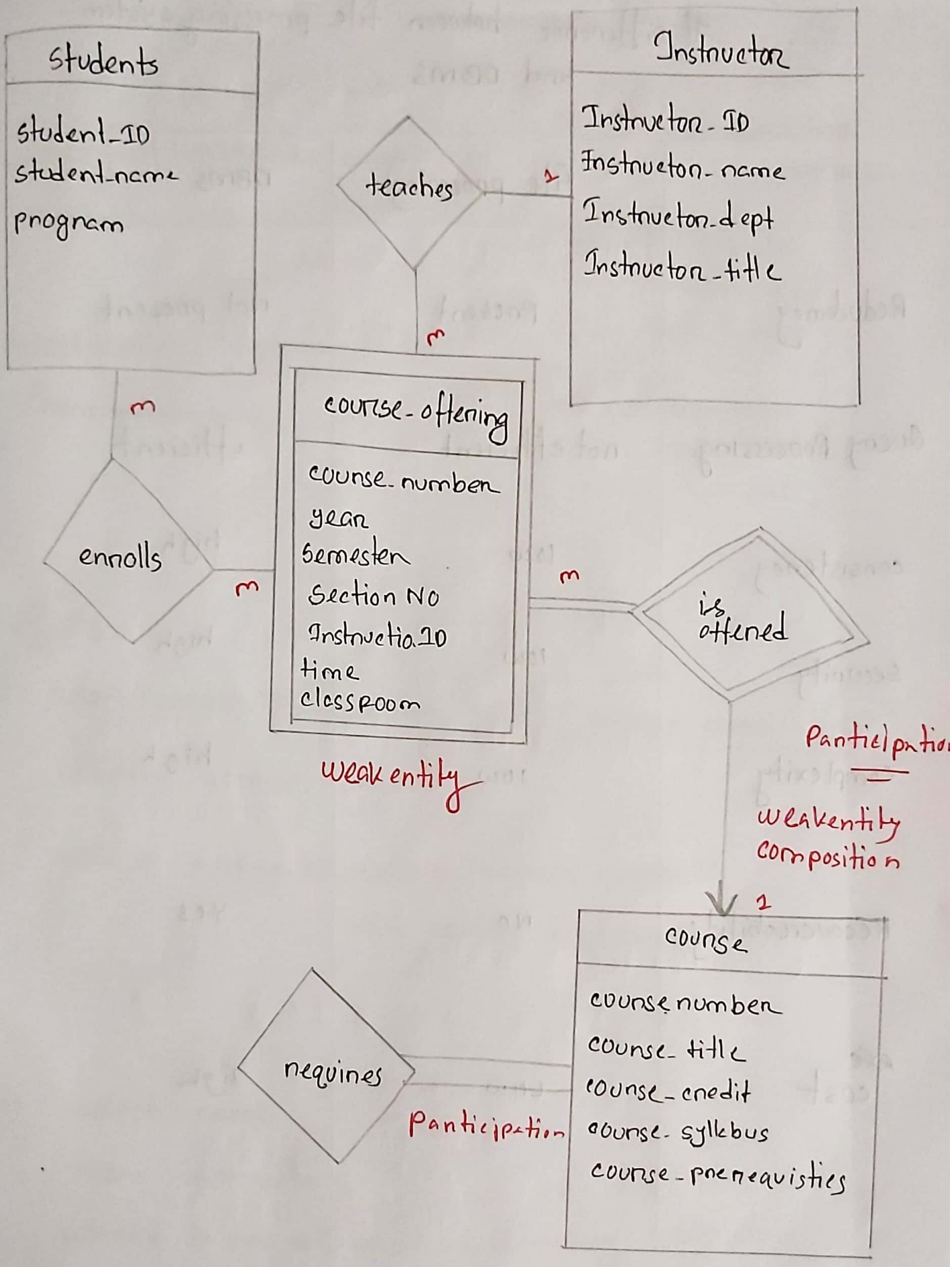
where employee_title = "Jr. officer";

update employee title.

Jr. officer with Junior officer

↓

to.


Students
student_ID
student_name
program
Instructor
Instructor_ID
Instructor_name
Instructor_dept
Instructor_title
teaches
1
m
course_offering
course_number
year
semester
Section NO
Instructio.ID
time
classroom
m
enrolls
m
is offered
m
Participation
weakentity
weakentity
composition
1
course
course number
course_title
course_credit
course_syllabus
course_prerequisties
requires
Participation

16

Difference between file processing system and DBMS

	file processing	DBMS
Redudancy	present	not present
Query Processing	not efficient	efficient
consistency	low	high
Security	low	high
complexity	low	high
Recoverability	NO	Yes
cost	Low	high

(1)

donot remove duplicates

(i) select all dept-name, name
from instructor ;

removedupli

(ii) select distinct deptname, name
from instructor ;

NOTNULL

First_Name varchar
Not NULL

Add column # ALTER Table
Student add
Roll int(5) ;

Delete col # ALTER table
student drop
Roll ;

— 6 —

need to specify

(2) select section. course_ID, section. semester, section. year, course. ~~course~~ title
FROM course, section
where section.courseID = course.courseID
and course.dept name = "compute"

(4) select name
from instructor
where salary > 90,000 and salary < 100,000

salary between 90000 and 100000

③ higher salary than some instructor computer

```
select distinct name
from Instructor
where salary > (
        select max(salary)
        from instructor
        where deptname = "computer"
        );
```

nested query

⑤

```
select dept_name
from depantment
where building like "%watson%" ;
```

match substring

⑥

```
update instructor
set salary = salary * 1.05
where salary < (
        select avg(salary)
        FROM instructor) ;
```

5% ↑

(7)

```
update instructors
set salary = CASE
             when salary > 100,000
                  Then salary * 1.03
             else salary * 1.05
END;
```

salary over 100000, 3% raise
where as allnecive 5% raise

(8)

```
Insert into course
(course_ID, title, dept_name, credits)
values
("CS-437", "Database system", "compoten sciene", 4);
```

(9)

```
Insert into student
(ID, name, dept_name, total_cred)
select ID, name, dept_name, 0
from Instructor
```

Q : Add all instructors to the student relation with total_cred set to 0

(10) Delete all tuples of instructor pertaining to be in finance dept

```
Delete from instructor
where dept_name = "Finance"
```

Delete all tuples of instructor with department located in the watson buiding

(11)

```
Delete from instructor
where dept_name IN (
```

nested

```
    select dept_name
    FROM department
    where building = "watso" );
```

Q :- Find all depantment whene total salary is greater than the avenage total salary at all dept

(12)

```
select dept_name
FROM instructor
Group by dept_name
Having sum(salary) > (
        Select AVG(totalsalary)
        FROM (
            select sum(salary) as totalsal
            FROM Instructon
            GROUP by dept_name
        )
    );
```

CSE

dataset :- A.B.C

Order Index → Highenstorage, (usecase : searching)
sorted, entry all data : A, B, C

sparse Index → maynot sorted, only a subset of entry.
: A, B
Lowstorage,
(usecase : need to focus on specific subset)

—o—

wound-wait → prevents deadlock by ordering timestamp

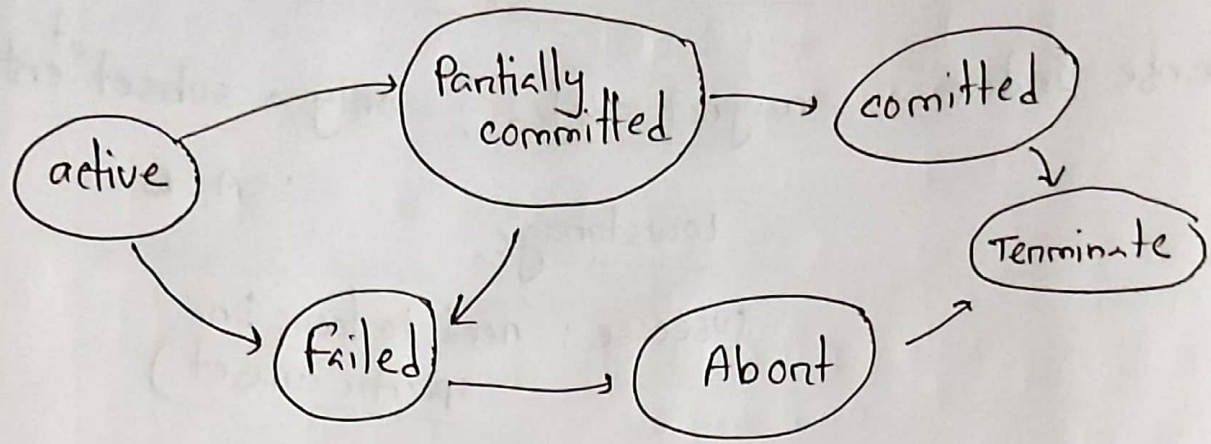
$T_1 \rightarrow 10$
$T_2 \rightarrow 20$
$T_3 \rightarrow 13$
→ Timestamp

Higher > low

T_1 start executing.
T_1 request a ressource held by T_2
$T_2 > T_1$; as T_1 was forced to wait and T_2 use the resource

T_3 comes
$T_2 > T_3$; as T_3 has lower stump. It has to wait untill highenstamp T_2 finis

Transaction state diagname



Pupose of Nomalization:-

1. eliminate data redudancy
2. improve data intregnity
3. reduce anomalies (Insent, delete)
4. effieent stonage & query of data

List of students of 2014 session who have registered for more than 2 course

(1)

```
select *
from student
Join course_registration on student.std_ID = course_registration.stdID
where student.session = 2014
Group by student.stdID
Having count(course_registration.course_ID) > 2
```

(2) Find the student whose name contains 'm' in right most 3rd character

```
select *
from student
where name like "__m%"
```

(3)

```
select reg_no
from student
where dept = "CSE"
ORDER BY reg_no DESC;
```

sorting

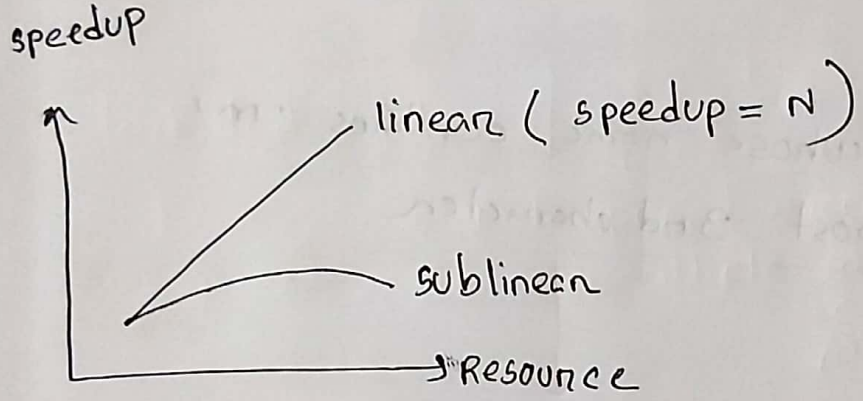
TT-3

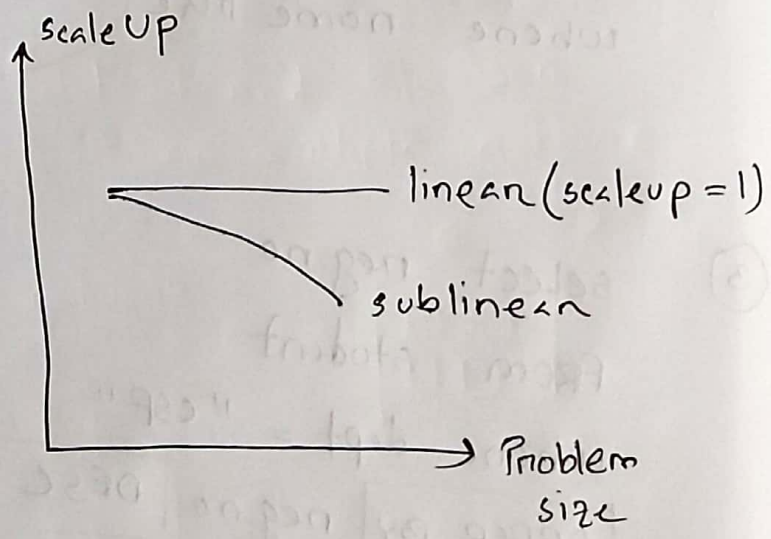
Speed up :-

$$= \frac{\text{small system elasped time}}{\text{large system elasped time}}$$

scaleUp :-

$$= \frac{\text{small system small problem ela ped time}}{\text{large system large problem ela psed time}}$$





Parallel Database :-

→ is refers to database architecture that utilize multiple processors to process multiple data operations co-currently.

Factors that limit speedup & scaleup :-

① **Startup costs :-** costs of starting up multiple process increase computation time

② **Skew :-** increasing degree of parallelisme, increase the variance in service time of parallely executing tasks

③ **Interfenence :-** processes accessing shared resource leads to more waiting time rather useful work

② **2PC** → The two phase commit protocol

→ used to ensure atomicity

(i) each node/site executes transaction untill before commit then leave the decision to coordinator

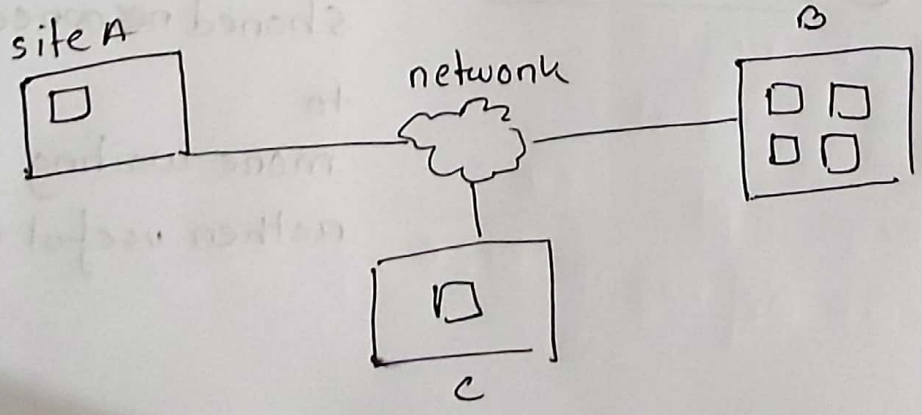
(ii) each node/site must follow the decision of co-ordinator even if there is a failure

Distributed system

→ Data spread over multiple ~~machine~~ machine /node/site

→ machine are connected with network

→ data shared by user on multiple machine



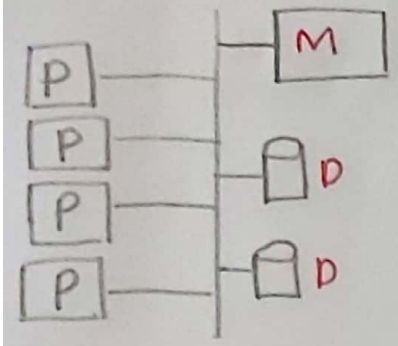
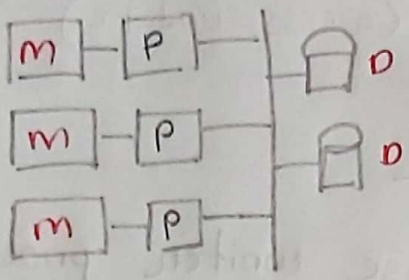
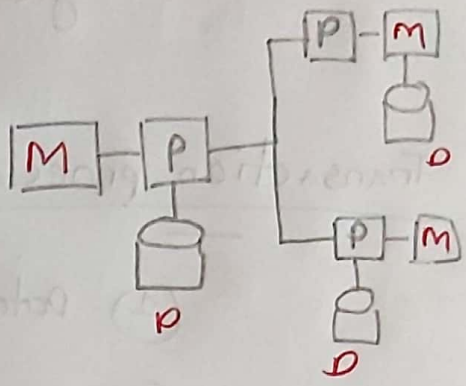
Transaction

↳ is a set of operation (Read, write) to perform a logical unit of work (access, update)

Transaction process

1. Database writen process
2. Log writen process
3. checkpoint process
4. Lock managen process
5. process moniton process

P = Process / Node / site
M = memory
D = Disk

shared memory	shared Disk	Shared nothing
		
multiple process accesses common physical memony spacc	processes have thein own pnivnte memony spacc but shane acess to disk	each process have its own pnivnte memony and d disk
connected by bus on switch	connected by stonage Anea Netwonk	don't shane anything dinectly

Key feature		
→ communication → Data consistency	→ Data access & shaning → fault tolenance → scalability	→ fault tolerance → scalability → Data pantioning → Distribution data → Data parallesim
Issue		
① not scalable ② bottleneck	① bottle neck	② mone costly

CSE

Instance

it refers to actual running or operational representation of a database at a particular point in time

∴ includes record - index - db object

Schema

is a logical container/namespace within a database that holds collection of database objects (tables, views, indexs, procedures, functions)

includes, columns, constraints, datatype relationship

SQL structured Query Language

1. DDL
2. DML

DDL
Data Definition Language

→ use to define and mange the structure and schema of database

→ statement
- create
- ALTER
- DROP
- GRANT
- Truncate

Data Manipulation Language

→ use to manipulate and interact with data within database object

→ statement
- insert
- update
- delete
- select
- menge

—o—

Abstraction level of data

① physical level :- describe how a record is stored

② Logical level :- describe stored data and their relationship

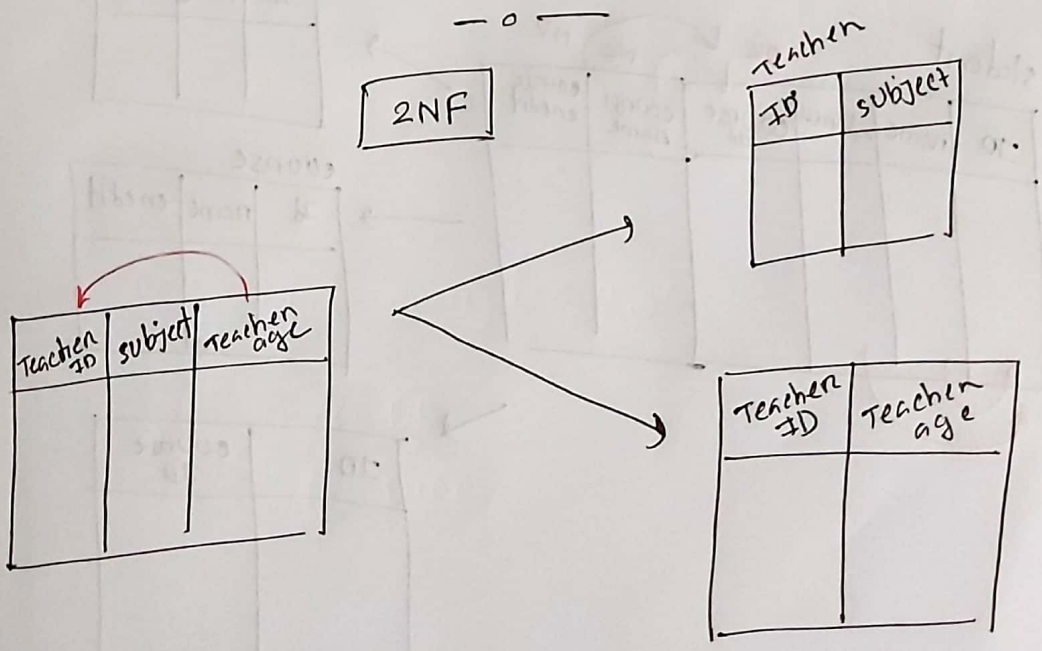
③ view level :- interface

ER model

1. Graphical Representation
2. conceptual level
3. Higher abstraction
4. focus on entites and their relationship
5. define cardinality constraints relationship

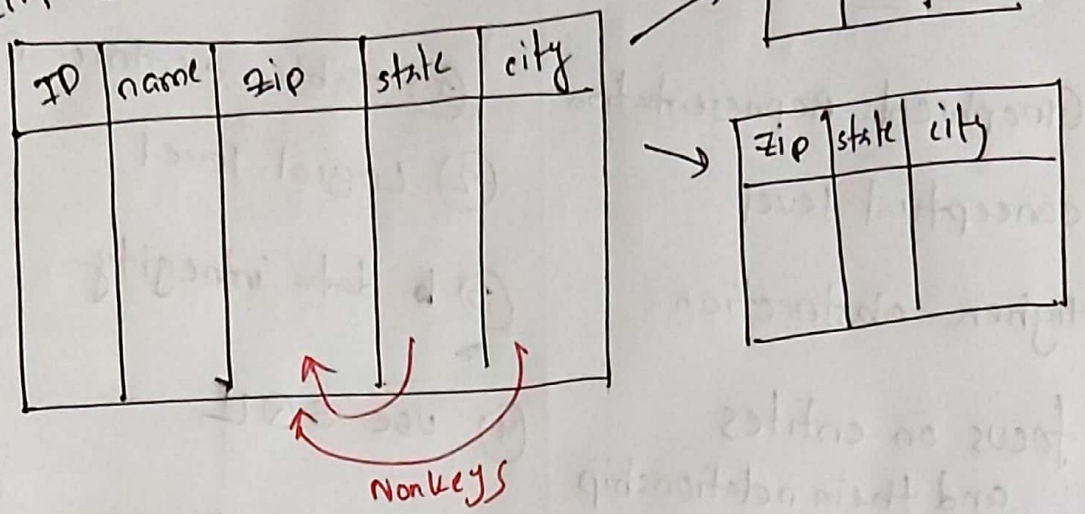
Relational model

1. Table centric Representation
2. Logical level
3. data integrity
4. use SQL
5. normalization
6. widely used



3NF

employe



BCNF

Boyce codd Normal form

multiple 2NF

student