WJMK

DB System concepts – 6th
Silberschatz, Korth, Sudarshan

9 Jan, 2023
Tuesday

lec-1

* Relational Data base Management System :- (most commonly used)
(RDBMS)
↳ always store info in form of table

Relational Model

Tabular Data (of ATM)

Customer id	customer name	customer city
192-83		
019-28		

(attributes)

→ records / Tuples

* Database design :-
① logical / conceptual design :-
- like which product to offer
- Table mein kitne rows / columns hain
- which attributes should we record in database ? → Business decision
- Computer science decision : ?

Entity Relationship model :-
(E-R model)
↳ is an enterprise as a collection of entities & relationship.

↳ always store info in form of diagram.

WORK 2
3rd
RDBMS.

① pehle ER modelling hoti hai, fir ki pta chlega kya - 2 attributes kitne hain i.e. RDBMS.

Entity :- thing or object that is distinguishable from other object
↳ like customer, car etc.

② Designs are pictures called, Entity - Relationship diagram.

ER model → proposed by Peter chen (1976)

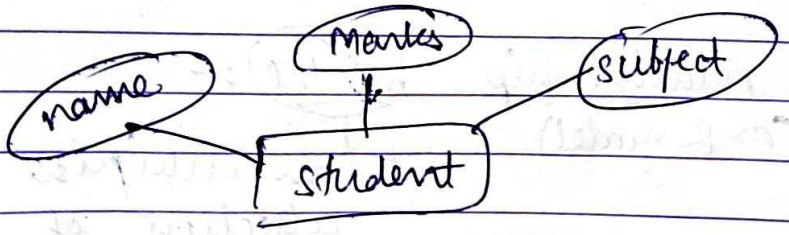
Entity → eg. book, student

Entity set = collection of similar entities
↳ ~~eg~~ whole class ke students ki details.

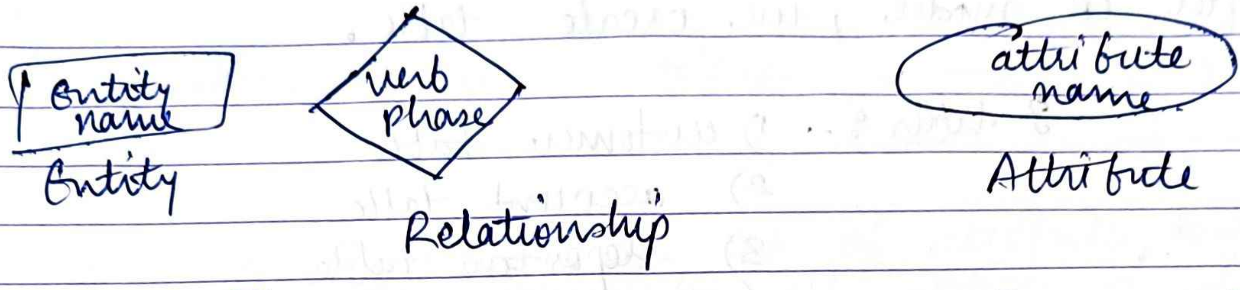
attribute = property of entity set
↳ eg. attendance, roll no, courses ki studied.

Entity set is represented by as rectangle.
Each attribute of an entity set is represented by oval with line to rectangle representing its entity set.

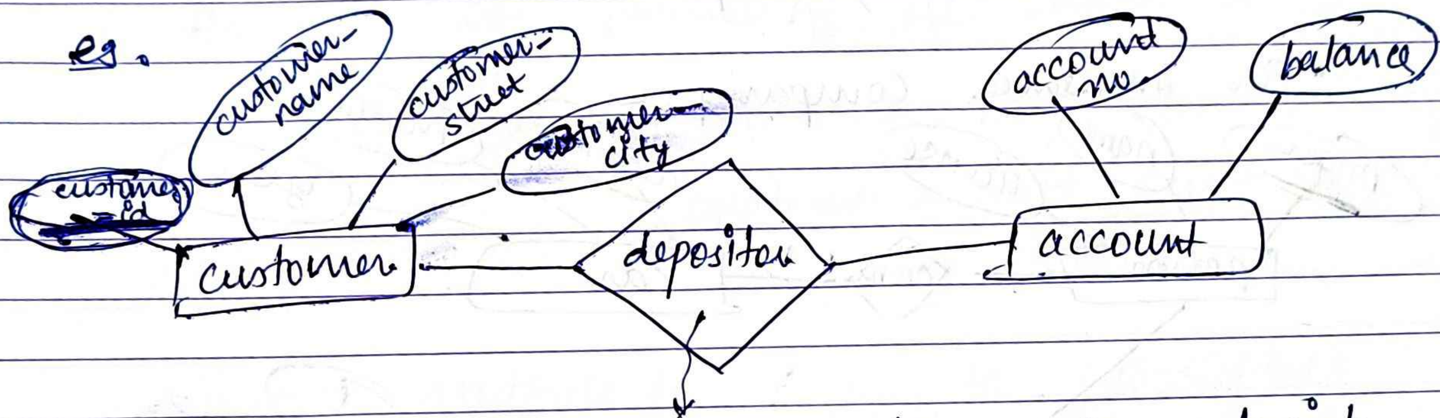
eg:



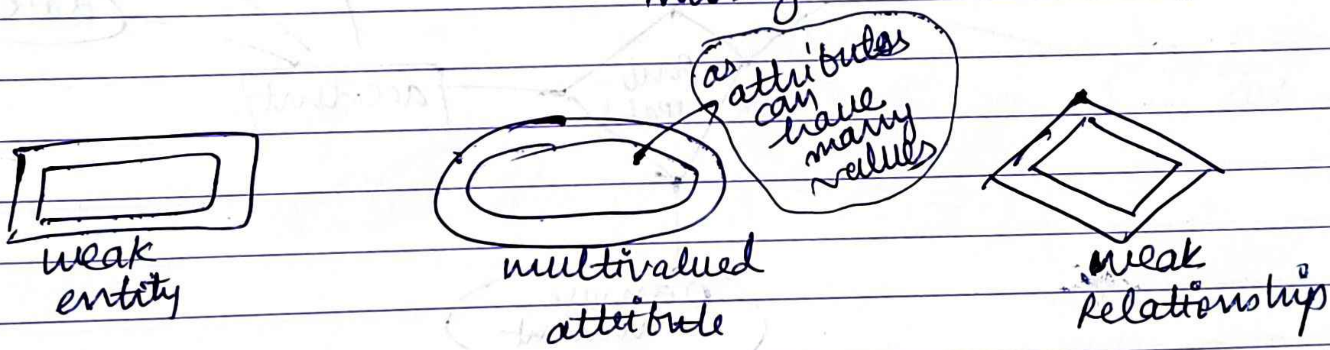
WDMK = 30 = MM.



eg.



means customer can deposit money in account.



main attribute :- attribute which helps in identifying an entity uniquel like. roll no. of student.

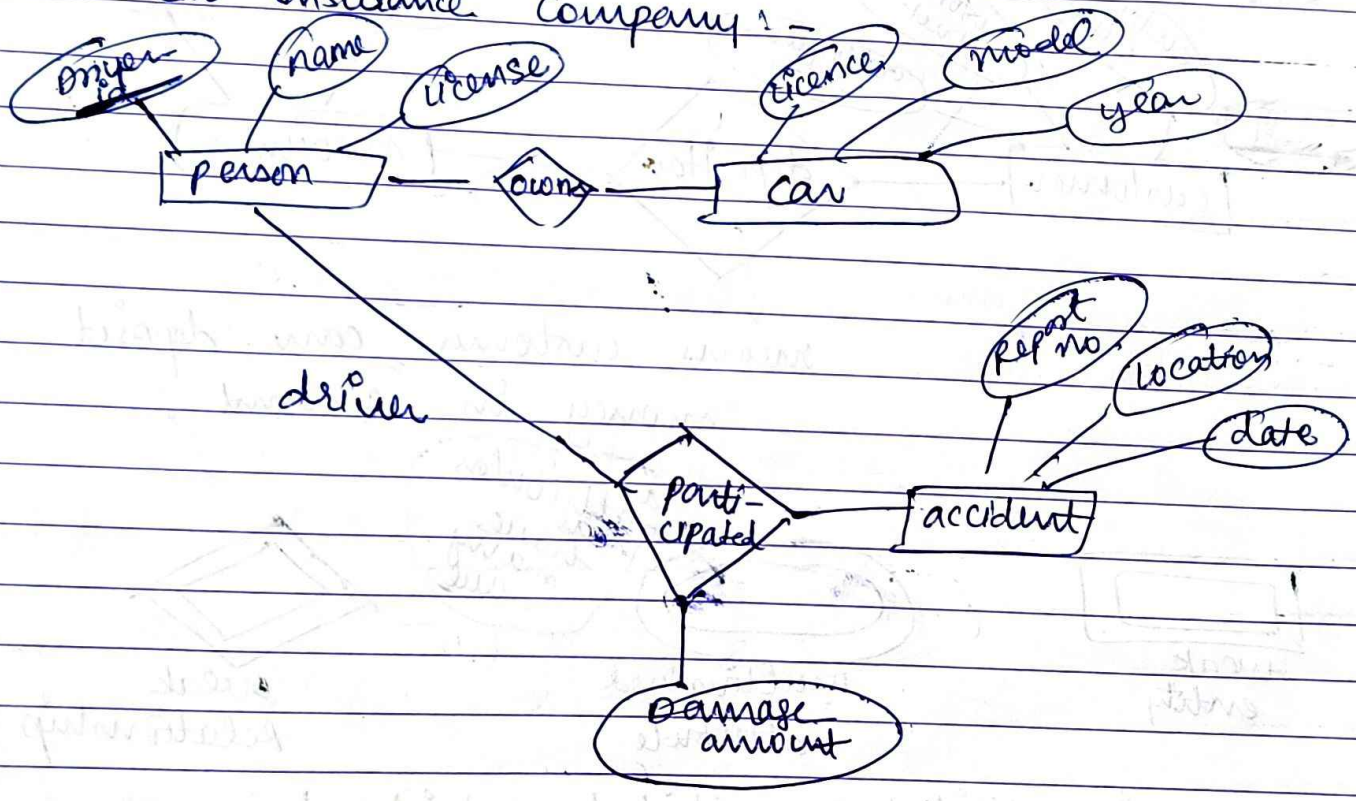
agar na hota, then student to weak entity, as then bhut saare attributes chahihe honge to identify it uniquely.

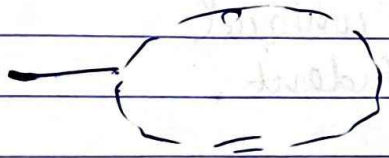
WORK
= 3rd

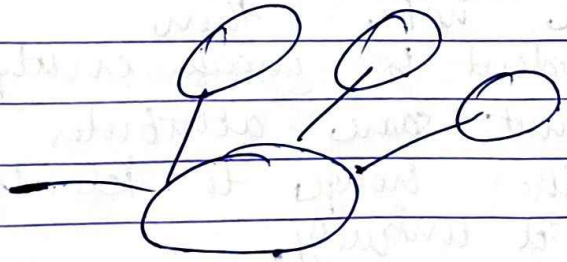
- After ER model, we create table.

3 tables :- 1) customer table
2) account table
3) depositor table

- Car Insurance Company :-




derived attribute (like age today is derived from DOB)


compound / composite attribute (attribute having further attribute info)

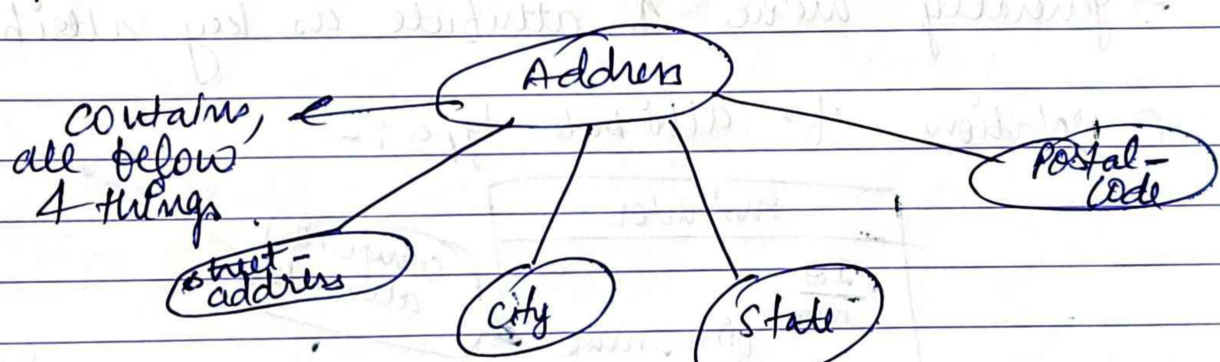

key attribute / main attribute (primary key that defines u uniquely)

W3 MK

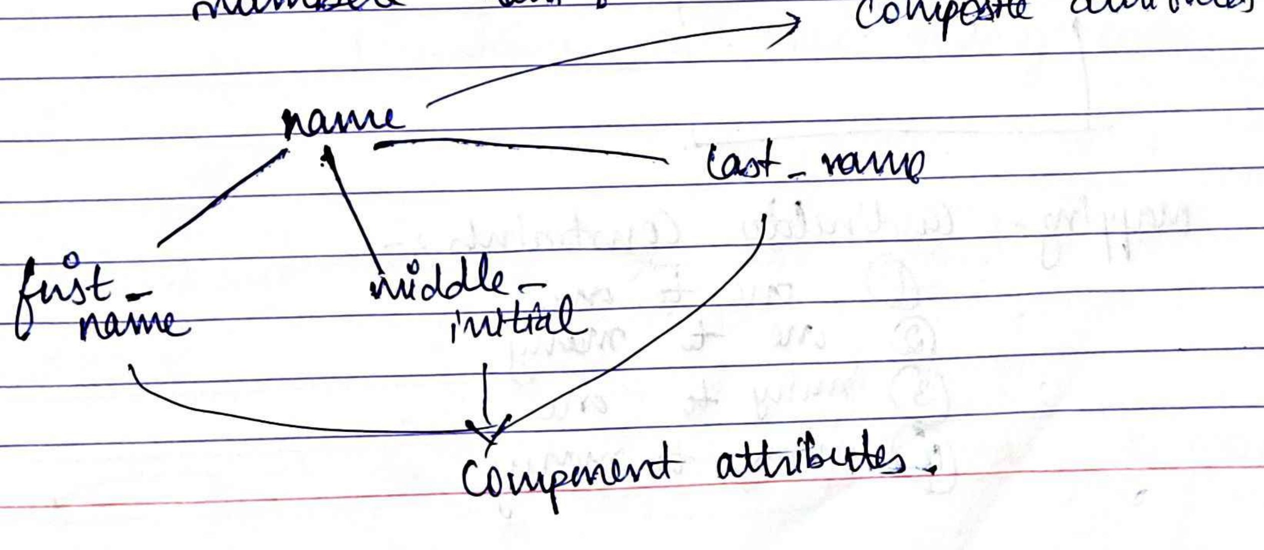
attribute / Normal attribute ——— ⬭

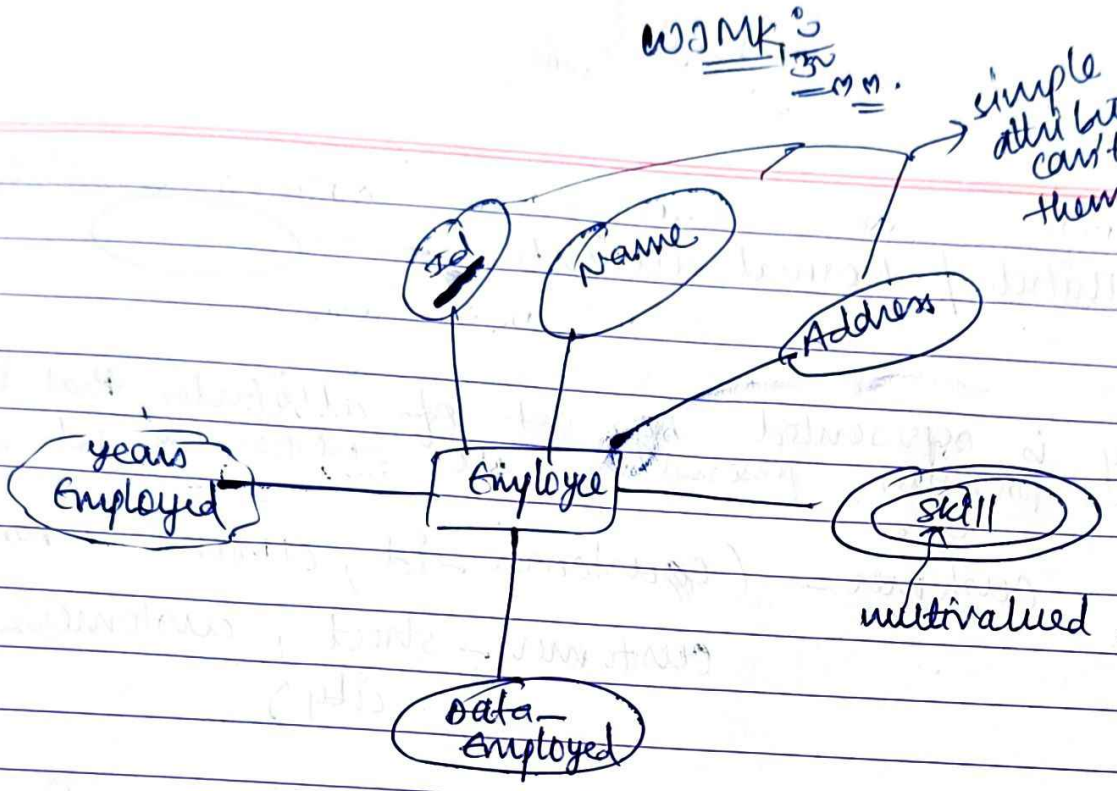
① entity is represented by set of attributes, that is descriptive properties possessed by all entities in set.
eg.
customer = (customer_id, customer_name, customer_street, customer_city).

composite attribute :- (can be sub-divided)



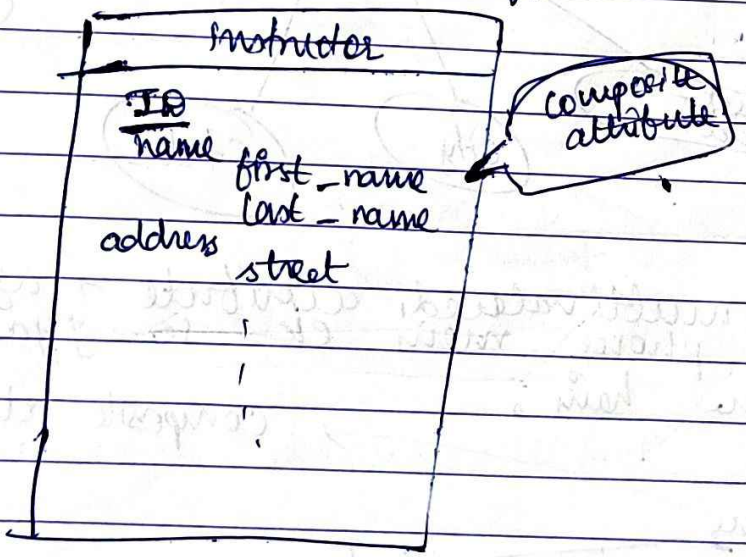
phone no is multivalued attribute → agar aapke phone mein ek se jyada number hain.





→ generally we'use 1 attribute as key attribute.

⑤ Notation of attribute types :-



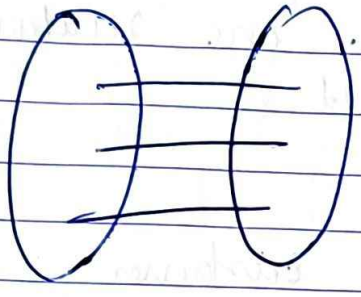
⑥ Mapping Cardinality Constraints :-

1. one to one
2. one to many
3. many to one
4. many to many

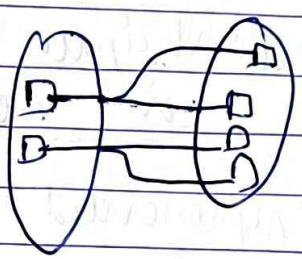
WINK

① 1 customer can have many accounts in bank.

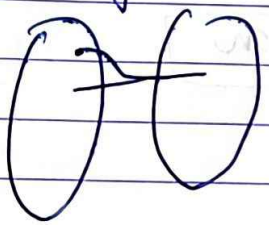
one to one



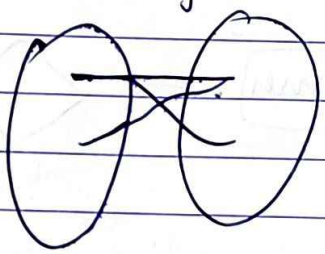
one to many



many to one

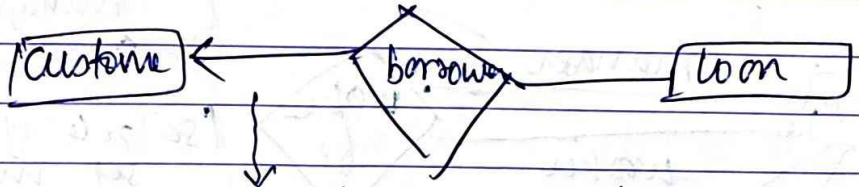


many to many

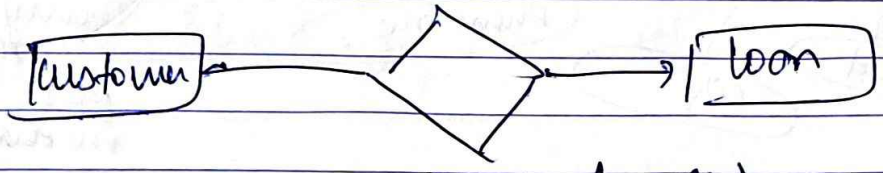


② many → one.

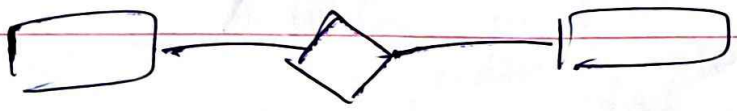
(one to many)



1 customer can have many loan



(many to one)



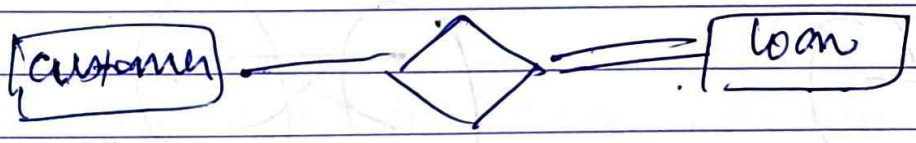
(many to many)

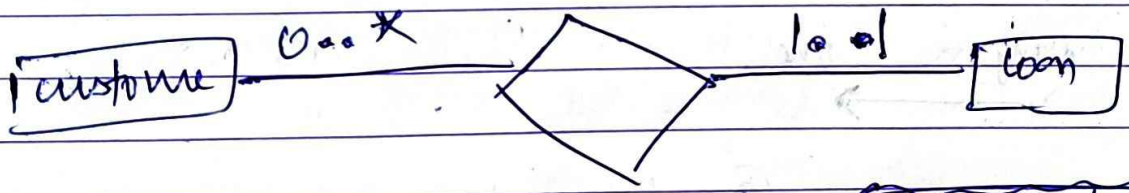
WIMK & SM.

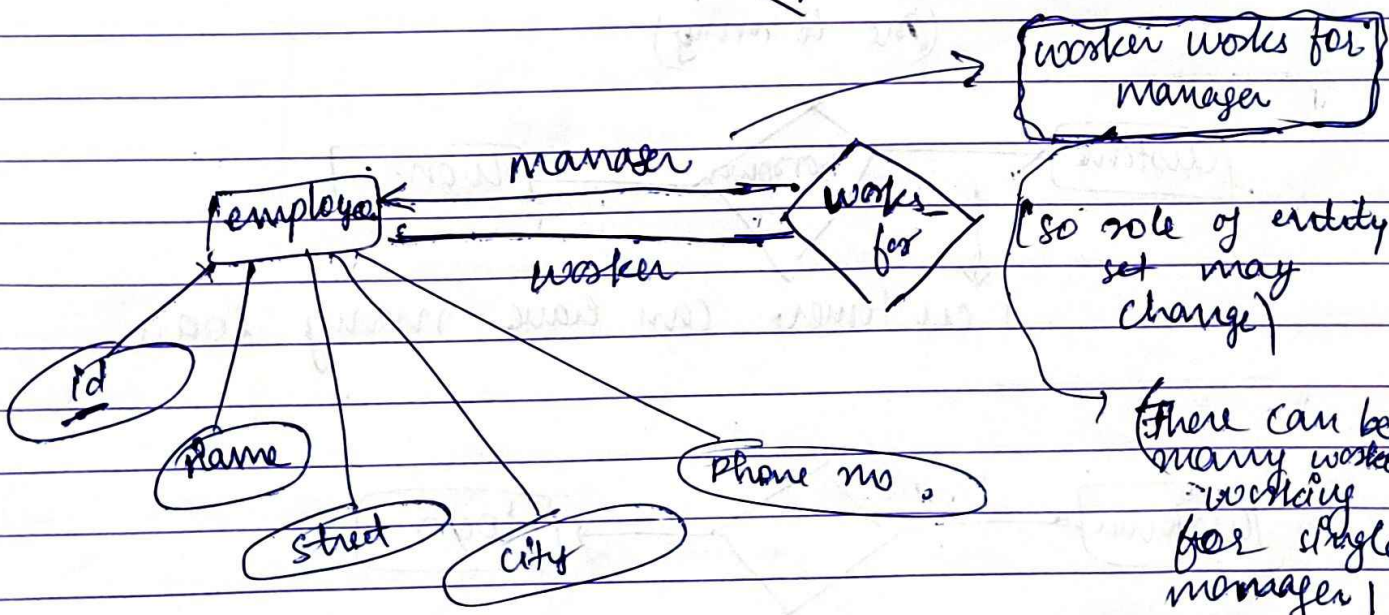
① Participation of entity set in a relationship set.

② Total participation :-
every entity in entity set participates in at least one relationship in relationship set

represented by ═.
loan can't exist w/o customer.





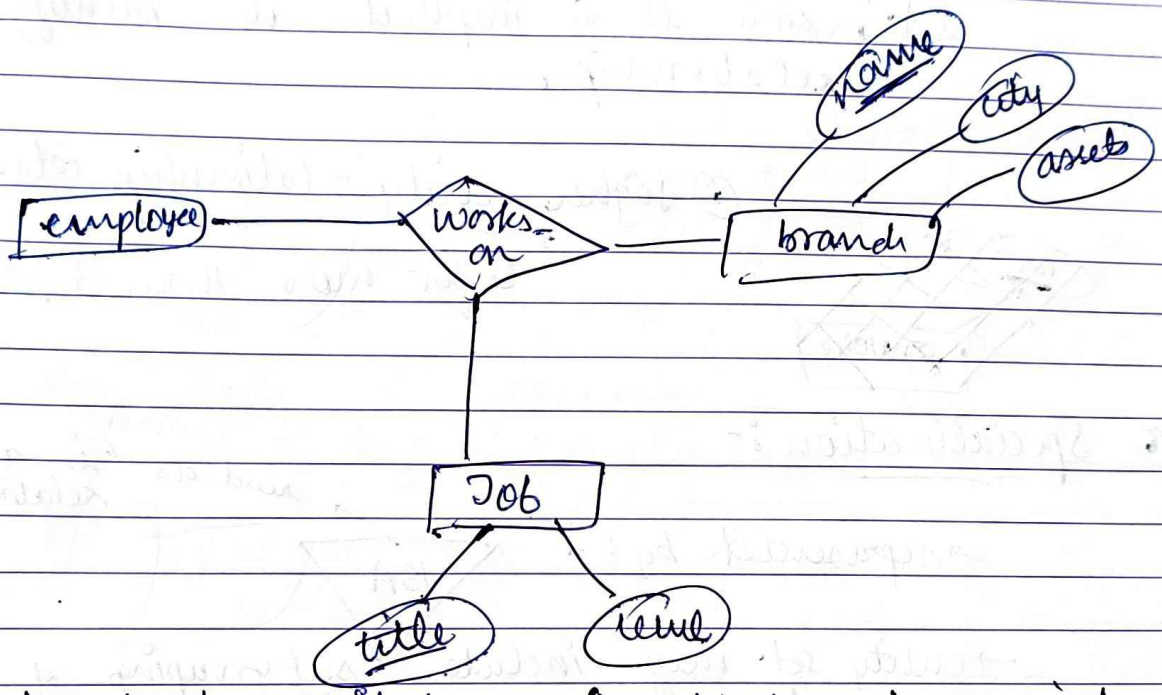


(so role of entity set may change)

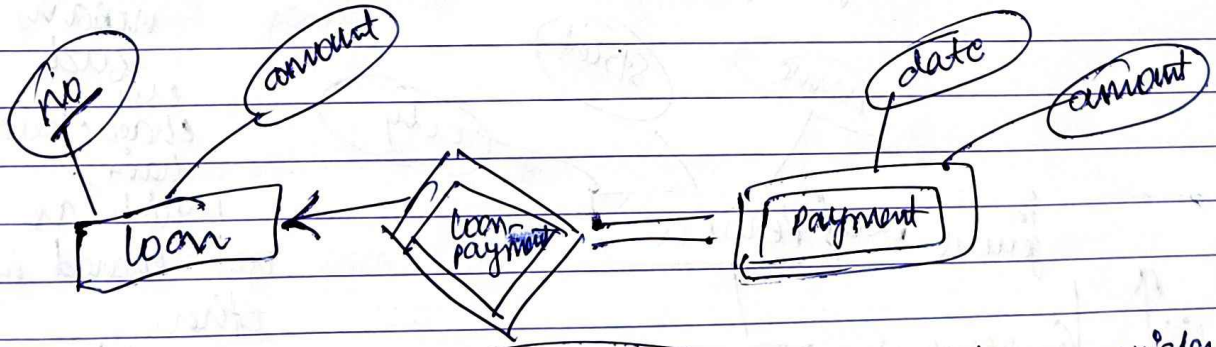
There can be many workers working for single manager

WINK ☺

* ER diagram with Ternary relationship :-



① Redundant attributes :- (attributes not required, agar na ho toh bhi dal jaaye)



② existence of weak entity set depends on existence of identifying entity set.

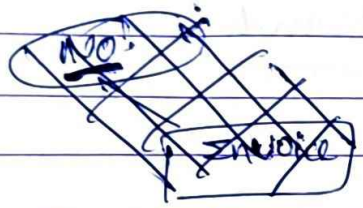
isko primary key is formed by " " of strong entity set on which weak entity set existence dependent, plus weak entity set's discriminator.

WORK @ 3pm.

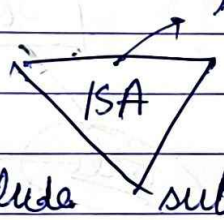
Note :- primary key of strong entity set is not explicitly stored with weak entity set, since it is implicit in identifying relationship.

① higher arity relationship sets..

↳ for more than 3.

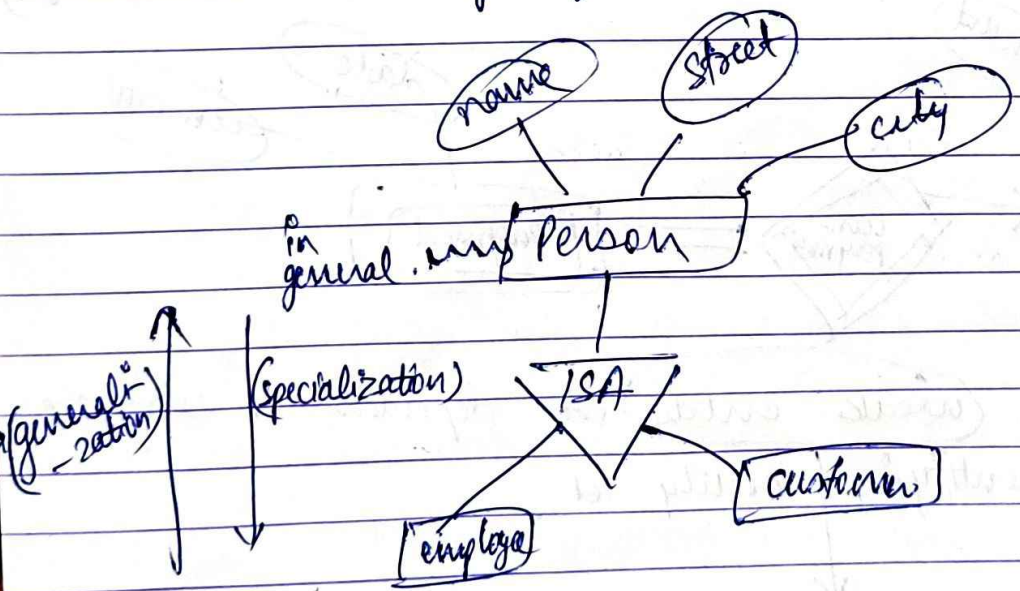


② Specialization :-

→ represented by :- 

read as 'is a' relationship.

→ entity set may include subgroupings of entities that are distinct in some way from other entities in set.



in general, means such essential characteristics have, which are not shared by other entities in set.

ER → RDBMS → code

WOMK :-
= ॐ
= MM.

12 Jan 2023
Friday

lec-2 :-

Today's topic :- Relational model, Relational Algebra, SQL

Relational Modelling :-
we/we

In view, ↑ levels of Abstraction :-

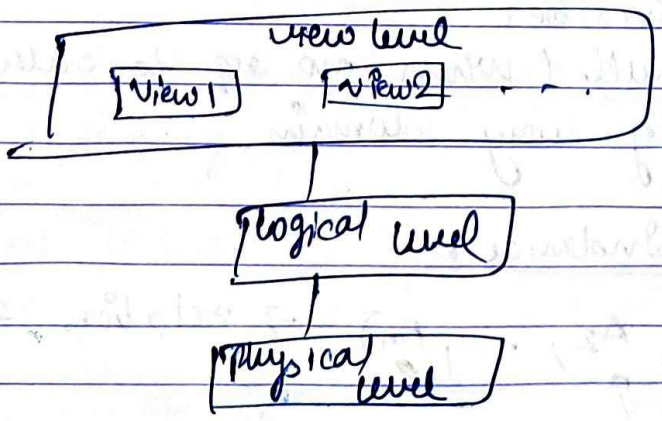
① Physical level :- describes how a record is stored.

② Logical level :-
describes data stored in database & r^n ships among data.

③ Table logical level data for physical level.

Different people have diff view.
like faculty, student, people handling DB have diff-2 views.

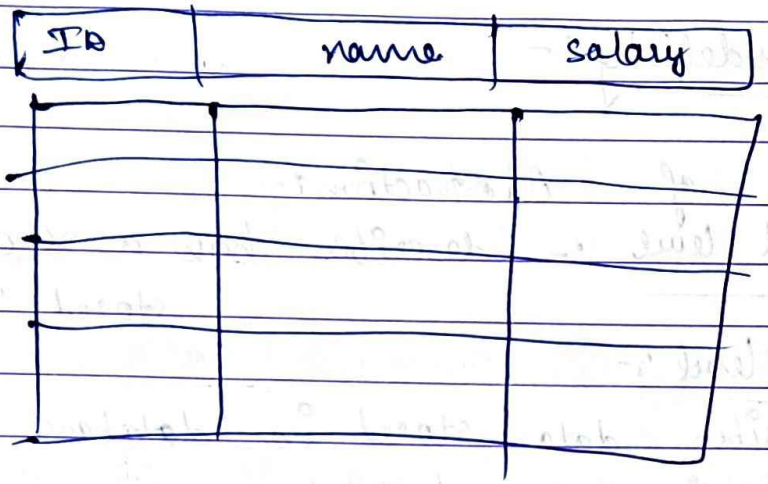
View of Data :-



WJMK ☺

- Physical Data Independence :- ability to modify physical schema w/o changing logical schema
  → ensures that applications depend on logical schema & independent of physical schema.

Example of DB :-



Row represents records (tuples)
Columns " attributes.

Attribute types :-
↳ set of allowed values for each attribute is called domain of attribute.
→ attribute values are req to be atomic (single valued), that is, indivisible.
→ special value null (when no req to enter value) is member of every domain.

Relation Schema & Instance :-

$R = (A_1, A_2, \ldots, A_n)$ → relation schema

attributes.

WJMK

Eg: instructor = (ID, name, salary)

Instructor → complete rn.

ID	name	salary

given sets $D_1, \ldots, D_n$, relation r is subset of $D_1 \times D_2 \times \ldots \times D_n$

- element t of r is a tuple, represented by row in table.
- current values (r^n instance) of a r^n are specified by row.
- R^n's are Unordered (means tuples can be arranged in any order).

Data Consistency :- refers to state of data in which all copies / instances are the same across all systems & databases.
or
→ ensures data is accurate.

Data Integrity :- refers to accuracy, completeness & consistency of data throughout its life cycle.

Keys :- let $K \subseteq R$

① → super key
② → primary key
③ → candidate key
④ → foreign key (???)

WJMK

Roll no ⟶ key

R = Instructor = {ID, name, dep_name, salary}
and K ⊆ R

K → (superkey (SK)) if values for K are sufficient to identify a unique tuple of each possible relation r(R).

eg: {ID}, {ID, name}, ~~both~~ are S.K.
{ID, salary},
{ID, dep_name},
{ID, name, dep_name, salary}

- superkey is candidate key if K is minimal.
  so, {ID} is " ".
- one of the candidate keys is selected to be the primary key.

eg.
{individual_id}, {department_id} both are candidate keys.

Now Database administrator will chose any one of the key as primary key.

⊙ If address is proper address jisse uss bnde tak pauncho skein, then it is also super key.

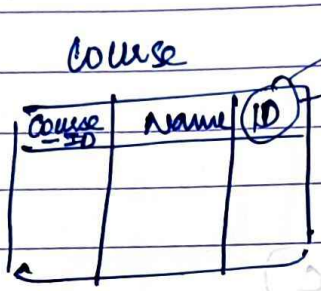
Foreign key:- are set of constraints in DBMS that establish r'ships b/w tables & also ensure consistency & integrity of data.

WIMK
5 mm.

⑤ Foreign key ✗ constraint :- Value in one r^n must appear in another

- referencing r^n
- referenced r^n

course



Instructor teaches course.

⑥ Relational Query Languages :-

- "Pure" languages :-
  1) Relational algebra
  2) Tuple " calculus
  3) Domain " "

- Relation r

A	B	C	D
α	α	1	7
α	β	5	7
β	β	12	3
β	β	23	10

Selection operation — selection of rows. (σ)

$\sigma_{A=B \wedge D>5}(r)$

1st cond^n — 2nd condition. — and operation.

A	B	C	D
α	α	1	7
β	β	23	10

WJMK 30

① Projection operation – selection of columns (attributes).
↳ to retrieve info of "

r :

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

②

$\Pi_{A,C}(r)$

A	C
α	1
α	1
β	1
β	2

=

A	C
α	1
β	1
β	2

↳ duplicate removed

③ Union of 2 r^n's :-

r, s :

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

r ∪ s :

A	B
α	1
α	2
β	1
β	3

↳ can take duplicates all (by using union_all)

WIMK ki Imm.

④ Set difference :-

r - s :

A	B
α	1
β	1

if

r :

A	B
α	1
α	2
β	1

s :

A	B
α	2
β	3

⑤ Set Intersection : $r \cap s = r - (r - s)$

⑥ Joining 2 rs : cartesian - product : $r \times s$

r.C2 s.C2

C1	C2
1	a
2	b

r

×

C2

C3	C4
3	a
4	b

s

=

C1	r.C2	C3	s.C2
1	a	3	a
1	a	4	b
2	b	3	a
2	b	4	b

as where r.C2 ≠ s.C2, wo bhi uthha liya wo value kyun chahiye tha!

problem : (combination of wrong data is entered).

⑦ Renaming a Table :-

↳ allow us to refer a rel (say E) by more than 1 name :

$\rho_x(E)$??

WJMK $\frac{c}{3}$ ♥♥.

- **Composition of operations** :- can build expressions using multiple operations

Eg. $\sigma_{A=C}(r \times s)$

↳ yeh nikaalo

→ fir yeh wale select karlo!

- **Joining 2 r^ns** :- Natural Join

↓

to remove redundant data.

eg :-

(r)

A	B	C	D
α	1	α	a
β	2	γ	a
γ	4	β	b
α	1	γ	a
δ	2	β	b

(s)

B	D	E
1	a	α
3	a	β
1	a	γ
2	b	δ
3	b	ε

→ same, can create problem.

$r \bowtie s$

$\Pi_{A, r.B, C, r.D, E}(\sigma_{r.B = s.B \wedge r.D = s.D}(r \times s))$

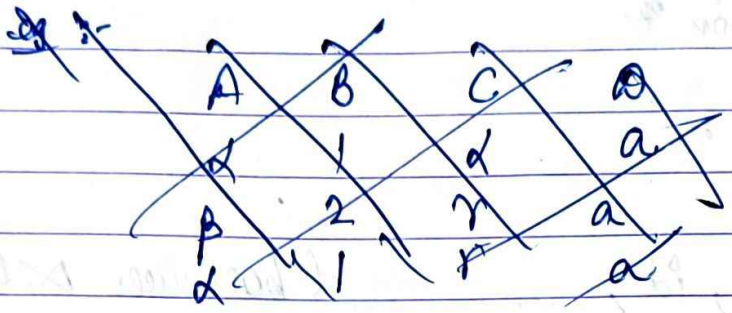
↳ yeh kro.

→ okho sirf wo jo yeh cond^n satisfy krein

A	B	C	D	E
α	1	α	a	α
α	1	α	a	γ
α	1	γ	a	α
α	1	γ	a	γ
δ	2	β	b	δ

(writing once only, dono (r, s) ke alag-2 nahi rakhe as we don't want irrelevant data.)

end mein table mein sirf yeh dikhna.



⑦ Formal relational Query Language :-

Relational Algebra :-

$\sigma_p(r)$, p → selection predicate

$\sigma_p(r) = \{ t \mid t \in r \text{ and } p(t) \}$

Business Idea → ER → Relational model → language → SQL (code)

(Procedure)

$\wedge$ (and) , $\vee$ (or) , $\neg$ (not)

Each term is one of :-

<attribute> op <attribute> or <constant>

eg.

$\sigma_{dept\_name = \text{"physics"}}(instructor)$

Project operation

Union operation

eg. find all courses taught in Fall 2009 sem or in Spring 2010 sem or in both :-

$\Pi_{course\_id}(\sigma_{sem = \text{"Fall"} \wedge year = 2009}(section)) \cup \Pi_{course\_id}(\sigma_{sem = \text{"Spring"} \wedge year = 2010}(section))$

WORK

- Set difference operation
- Natural join operation :-

$\Pi_{customer\_name, id, \ldots}$ (borrower ⋈ loan)

- left outer join, right outer join, full outer join (3)

① Rename Operation :-

ρ (CUST_Name, Π (cust_name) (CUSTOMER))

Iska sirf yeh column project kro

and attribute ko rename krdo isse!

② Tuple Relational Calculus :-

Predicate Calculus formula :-

$x \Rightarrow y$ (If x is true, then y is true)

$x \Rightarrow y \equiv \neg x \vee y$

$\exists t \in r(Q(t)) \equiv$ "there exists" tuple t in r's r s.t. $Q(t)$ is true (??)

WAMK ☺

• find ID, name, dept-name, salary for instructors whose salary greater than $80,000 :-

Query relational language :-

$\Pi_{id, name, salary, dept\_name} (\sigma_{salary > 80,000})$ → (yeh likhne ki zaroorat nhi cas sab schema mein hai (means koi chota set nhi chahiye))

(toh isko hta bhi skte ho)

tuple r.n calculus :-

$\{ t \mid t \in instructor \wedge t[salary] > 80,000 \}$

→ If want ID as output only :-

$\{ t \mid \exists s \in instructor \, (t[ID] = s[ID] \wedge s[salary] > 80,000) \}$

• find set of courses taught in fall 2009 sem or in spring 2010 sem, or in both :

$\{ t \mid \exists s \in section \, (t[course\_id] = s[course\_id] \wedge s[sem] = \text{"fall"} \wedge s[year] = 2009) \vee \exists u \in section \, (t[course\_id] = u[course\_id] \wedge u[sem] = \text{"spring"} \wedge u[year] = 2010) \}$

WOMK e
ನಮ್ಮ.

⊙ Find students who have taken all courses offered in Bio dept department.

Domain Relational Calculus :-

find ID, name, dep-name, salary for salary > $80,000.

⇒ $\{ \langle i, n, d, s \rangle \mid \langle i, n, d, s \rangle \in instructor \wedge s > 80000 \}$

i, n, d, s ⇒ data retrieving (just like projection)

Safety expressions

SQL :-

Create Table construct :-

```
create table instructor (
        ID          char(5),
        name        varchar(20),
        dept_name   varchar(20),
        salary      numeric(8,2))
```

Inserting data in table :-

```
Insert into table_name
        values ( 11 , '...', 'Hi' , 4 )
```

WJMK ॐ MM.

16 Jan 2023
Tuesday

Lec-3 :-

IBM → developed → SQL (IBM sequel language, ab structured query language)

Outer Join :- Common wala content rkho bs!

Left Outer Join :- To avoid loss of info, left outer join ($R_1 \bowtie R_2$) R_1 wali saari info rkhega, jo R_2 mein nahi hogi wo bhi, bs usko null kedega.

Similarly Right Outer Join.

Equivalent Queries :- Same query in different ways

Query 1 :-
$\sigma_{dept\_name = "physics" \wedge salary > 90,000}$ (instructor)

Query 2 :-
$\sigma_{dept\_name = "physics"}$ ($\sigma_{salary > 90,000}$ (instructor))
↳ pehle wo dekhlo jinki salary > 90,000

→ both are equivalent.
(will give same output on any database.)

→ not identical.

SQL :-

Rollup & Cube give same info, but name different.

SQL parts :-
1. DML (Data manipulation Language)
2. Integrity (for accuracy)
3. View definition
4. Transaction Control
5. Embedded & Dynamic SQL
6. Authorization

Wonk & Mm.

DDL (Data definition language)
↳ we create definition of relation
→ includes schema of each relation
→ type of values associated with each attribute eg. int, string.
→ integrity constraints
→ security & authorization

Domain Types in SQL :-

char(n) → fixed length char string

varchar(n) → variable " " ", with user-specified max length (n).

char (20) 'Anjali'
varchar (20) 'Anjali'
6 units of space will occupy
will occupy all 20 units.
(leaving null character naturally)

int, smallint

numeric (p, d) → fixed pt no.; with user-specified precision of p digits, with d digits to right of decimal point. → and 0.2
numeric (3,1) can store 44.5 but not 444.5 or 0.32.

real, double precision

float (n) → floating pt number, with user-specified precision of atleast n digits

WORK ॐ

create table instructor (

ID char(5),
name varchar(20).
salary numeric(8,2)
dept_name varchar(20))

Integrity constraints in create table :

↳ Types :- ① Primary key (A1, ..., An)
② foreign key (Am, ..., An) references r
③ not null

eg : salary hogi kuch na kuch, nahi toh they are not employees.

create table instructor
(ID char(5),
name varchar(20) not null,
primary key (ID))

ID, course_ID both can be primary key ??

No!

course_ID can not be primary key, as 2 students can have same course!

foreign key (dept_name) references department

↳ must be primary key of ↑

insert into instructor values ('10211', 'Smith', 'Biology', 66000);

delete from student → saare tuples remove honge

" " customers WHERE CustomerName = "Alfred";

drop table r;

WAMK → 3rd nov.

⊙ ALTER TABLE customer ADD Password varchar

Basic Query structure :-

select $A_1, A_2, \ldots, A_n$ (attributes)
from $r_1, r_2, \ldots, r_m$ (relations)
where (P) ↳ Predicate or condition

eg.

```
select Name
from Student
where marks > 95;
```

✗ → as semicolon means ending of statement.

⊙ select distinct dept_name
from instructor

⊙ select all dept_name (it will keep all duplicates also)
from instructor

* → means all attributes

⊙ select *
from instructor

⊙ select '437' → no from clause / random name
↳ output : 437 (one column with single row)

⊙ select '437' as FOO
⊙ select 'A' from shippers
↳ To assign particular name ↳ one column & n rows with A in each row.

WJMK ई 30 MM..

select :- can involve arithematic expression : +, -, *, /

select ID, salary/12 as monthly_salary
↳ new name

WJMK ॐ

19 Jan 2023
Friday

[lec-4]:-

① The where clause
select name
from instructor
where dept_name = 'computer science' and salary > 70000
(dept_name → attribute name)

① select A1, A2, ..., An
from R1, R2, ..., Rm

eg. select *
from instructor, teaches } → gives cartesian product instructor X teaches (→ Join)

↑ badh ring the size, but along with it, mismatch of Id's bhi ho gya!
⇑ as instructor.id = teaches.id (hona chahihe tha) (but har jgaah nhi hai!!)
↑ so use Natural Join

So: select name, course_id
from instructor, teaches
where instructor.ID = teaches.ID

Self Join:- select column_name(s) (means → attribute_name(s))
from table1 T1, table1 T2
where condition
→ table1 ko pehle T1 naam se bulaaya, fir T2 naam se bulaaya!

HOMK

- Find name of all instructors who have a higher salary than some instructor in 'comp. sci'.

⇒

```
select name
from instructors
where salary > 75000 or salary > 65000 or
      salary > 92000
```

↳ prblm → data change ho gya then code bhi change krna hoga!

⇒

```
select name
from instructor T1, instructor T2
where T1.salary > T2.salary and T2.dept_name =
      'computer_science';
```

- String Operations:- % → character matches any substring.
  _ → character matches any character.

___ ⇒ means string should contain atleast 3 character
% → can have string length 0 or more.

Find name of all instructors whose name includes substring "dar".

```
select name
from instructor
where name like '%dar%'
```

⑥ match the string "100%" → will give strings starting with 100%

↳ like '100\%' escape '\'

↳ (then escape is must !!)

WOMK ई ज MM.

WOMK ई ज MM.

- string beginning with "Intro"
  ↳ 'Intro%'
- string having exactly 3 characters
  ↳ '___'
- string having atleast 3 characters
  '___%'

#. for concatenation ⇒ ||

↳ Select first_name || ' ' || last_name AS full_name
FROM student

will concatenate first_name & last_name, with space in between, renamed as full_name.

- Select ID, last name, city & length of city :-
- SELECT id, last_name, city, LENGTH (C.city) as length
  FROM employees
- SELECT UPPER (Customer_Name) as UPPERCUSTNAME.
  ↳ sbko upper case mein bdl dega.

Ordering display of Tuples :-

WINK ☺

select distinct name
from instructor
order by name desc (ascending order is by default)

alphabetical order mein dega!

→ will give descending order.

⊙ Sort them by length of city name and then by employee_name

SELECT id, last_name, city, LENGTH(city) as length
from instructor
order by length, last_name

⊙ select name
from instructor
where salary between 90000 and 100000

where (instructor.ID, dept_name) = (teaches.id, 'Bio')

yeh 2nd compare hoga!

yeh compare hoga

⊙ union, intersect, except

eg. (select ... from ... where ...)
union
(select ... from ... where ...)

except → But not.

Null values :-

5 + null returns null

⊛ result of any operation with null returns null.

5 < null → gives unknown.

Aggregate functions :-

```
select avg (salary)
from instructor
where dept_name = 'Computer Science';
```

⊛ Find total no. of instructors who teach a course in Spring 2018 sem.

```
select count (distinct ID)
from teaches
where semester = 'Spring' & year = 2018;
```

⊛ Find avg salary from instructors in each department.

↳ GROUP BY

```
select dept_name, avg(salary) as avg_salary
from instructor
group by dept_name
having avg (salary) > 42000;
```

⊛ Nested Subqueries :-

⊛ <operation> (subquery)

WORK IS WORSHIP.

Set Membership :-

```
select distinct course_id
from section
where sem = 'Fall' & year = 2017 and
      course_id (in) (select course_id
                      from section
                      where sem = 'Spring' &
                            year = 2018);
```

except ke liye (not 'in' likhenge!

OR

```
(select course_id from section where sem = 'Fall'
                                and year = 2017)
intersect
(select course_id from section where sem = 'Spring'
                                and year = 2018)
```

** Set ~~Membership~~ Comparison - "some" clause :-

```
select distinct T.name
from instructor as T, instructor as S
where T.salary > S.salary and S.dept_name =
      'Biology';
```

↑ same query using > some clause!

$F \langle comp \rangle$ some $r \Leftrightarrow \exists\ t \in r$ such that $(F \langle comp \rangle t)$

Where $\langle comp \rangle$ can be: $<, \leq, >, =, \neq$

$(5 < \text{some} \boxed{0\ 5\ 6}) = \text{true}$ → atleast 1 col ek true hona chahiye! (5<0) → true!

$(5 < \text{some} \begin{array}{|c|}\hline 0 \\ \hline 5 \\ \hline \end{array}) = \text{false}$

'All' clause :-

① select name
from instructor
where salary > all (select salary
from instructor
where dept_name = 'Bio');

$(5 < \text{all} \begin{array}{|c|}\hline 0 \\ \hline 5 \\ \hline 6 \\ \hline \end{array}) = \text{false}$

- From clause
- With clause :-

```
with max_budget (value) as
        (select max (budget)
         from department)
select department.name
from department, max_budget
where department.budget = max_budget.value;
```

- Scalar subquery
- Deletion :-

```
delete from instructor
where dept_name = 'Biology';

where dept_name in (select dept_name
                    from department
                    where building = 'Watson');
```

Updates :-

```
update instructor
set salary = salary * (1.05)
```

WORK 2 — WORK 3

20 Jan 2023
Saturday

lec-5

Table → temporary create hoga, database mein store nahi hoga!

→ cond^n jab temporary table bhe??

```
WITH temporaryTable (averageValue) as
    (SELECT avg (attribute1)
     FROM Table)
SELECT Attribute1
FROM Table, temporaryTable
WHERE Table.Attribute1 >
      temporaryTable.averageValue;
```

→ table name → sirf ych attribute hoga

eg:
```
with max_budget (value) as
    (select max(budget)        → gives max
     from department)
select department.name
from   "  , max_budget
where  department.budget = max_budget.value
```

Q. Find all departments where total salary > avg of total salary at all departments.

A:
```
with dept_total (dept_name, value) as
    (select dept_name, sum(salary)
     from instructor
     group by dept_name),
dept_total_avg (value) as
    (select avg(value)
     from dept_total)
select dept_name
from dept_total, dept_total_avg
where dept_total.value > dept_total_avg.value;
```

(1st temp rn) (2nd temp rn)

WORK :-

So, dept_total & dept_total_avg store nahi huye database mein.

Scalar Subquery

Deletion :- ① delete from instructor
where dept_name = 'finance';

~~② delete from~~

② delete from instructor
where salary < (select avg(salary)
from instructor);

Problem :- as we delete tuples from instructor, the avg salary changes.
solution ???

Updates :- update instructor
set salary = salary * 1.05
where salary < 70000;

Case statement for conditional Updates :-

③ update instructor
set salary = case
when salary <= 100000 then salary * 1.05
else salary * 1.03 end;

Update with Scalar ~~and~~ Subqueries :-

Scalar → returning single value.
↳ for eg. select sum (credits)
↓
returning only one value.

<> → means ~~!=~~ != (not equal to)

0 → means null

Tests for Empty Relations :-

exists r ⇔ r ≠ ∅
not exists r ⇔ r = ∅

"exists" clause :- · (correlation name, correlated subquery)

```
Select course_id
from section as S
where sem = 'Fall' and year = 2017 and
exists ( select * from section as T
         where sem = 'Spring' and year = 2018
         and S.course_ID = T.course_ID);
```

NORMALIZATION :-

decomposition : divide rn ~~into~~ into multiple relations

rn with no problem → good rn o/w bad rn.

WEMK 3rm.

Anomalies in Relational model :-

① Insertion anomalies
② Deletion anomalies → if deleting from everywhere
③ Update anomalies

- Features of Good Relational Designs

- Deleting Redundancy
  R → W problematic
  S → W not !!

④ R is not key.

decomposition :-

employee (ID, name, street, city, salary)
into
employee 1 (ID, name)
employee 2 (name, street, city, salary)

→ Iska prblm dikha, wahan decompose karo!

→ prblm yeh arises when have 2 employees with same name.

Lossy decomposition :- decomposition kaha kuch loss huya hai
yaa nahi yeh dekhne ke liye, decomposed
tables ka fir se natural join kro, agar
same table mile → lossless decomposition o/w
lossy decomposition.

WORK ☺
50 MM.

⊙ schema r can be decomposed into R_1 & R_2 and lossless decomposition when :-

$$\Pi_{R_1}(r) \bowtie \Pi_{R_2}(r) = r.$$

and lossy when :-

$$r \subset \Pi_{R_1}(r) \bowtie \Pi_{R_2}(r) = r$$

① $R = (A, B, C)$

$R_1 = (A, B)$ $R_2 = (B, C)$

A	B	C
α	1	A
β	2	B

R_1

A	B
α	1
β	2

R_2

B	C
1	A
2	B

natural join

⇒

A	B	C
α	1	A
β	2	B

as equal ⇒ so lossless.

↓ cartesian product karke dhaan-2 $R_1 \cdot B = R_2 \cdot B$ hoga ⇒ woo rkhlo!

Normalization Theory :-

R → good form rn ⇒ if decompose it into set of rn's ⇒ $\{R_1, \ldots, R_n\}$ such that :

① each rn is in good form
② decomposition is lossless.

WOMK ☺

Functional Dependencies :-

- Functional dependency is generalization of notion of key.

$R \to r^n$ schema

$\alpha \subseteq R$ and $\beta \subseteq R$

functional dependency :- $\alpha \to \beta$ (β can be functionally determined by α)

holds on R $\Leftrightarrow$ for any legal r^n $r(R)$, whenever any 2 tuples t_1 and t_2 of r agree on attributes α, they also agree on attributes β.

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

If $A \to B$ (A is determining B)
and $B \to C$ (B is ——— C)
then $A \to C$

↳ so transitivity holds.

WORK ☺

23 Jan 2023
Tuesday

lec-6

①

A	B
1	4
1	4
3	7
3	2

A and B are not functionally dependent!

as, 3 → 7, 3 → 2

∴ 3 mapped to 2 numbers.

②

T_1

1	a
1	a
2	b
2	b

can write as ⟹

T_2

1	a
2	b

both tables give same answer,
so, T_1 can be reduced to T_2.

L	M
A	A
B	B

⇒ A is determining A,
B is " B

∴ can write like this ?? (like roll no.)
L and M are functionally dependent,

So, L → M
but M is L only
so, L → L

this is Trivial dependency.

③ If A → B , B → C then A → C

If F = { A → B , B → C , . . }
then, A → C , A → A , B → B , C → C ⇒ yeh saare pkka ∈ F.

Where, F^+ = closure of F

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

These are dependencies which are not required, as they are trivial, So can remove them !!

$\alpha \to \beta$ is trivial if $\beta \subseteq \alpha$

⊙ K is superkey for relation schema R $\iff$ $K \to R$

Like roll_no $\to$ {roll_name, name}

⊙ K is candidate key for R iff

$K \to R$

and for no $\alpha \subset K$, $\alpha \to R$

let A is candidate key

then {A, B}, {A, C} ... $\to$ can be super keys !!

⊙ in_dep (ID, name, salary, dept_name, building, budget)

want :- dept_name $\to$ building

ID $\to$ building

not want :- dept_name $\to$ salary

⊙ R $\to$ R_1, R_2 $\Rightarrow$ lossless decomposition hota

If $R_1 \cap R_2 \to R_1$ or $R_1 \cap R_2 \to R_2$??

Q: $R = (A, B, C)$ $F \to \{A \to B, B \to C\}$

$R_1 = (A, B)$ $R_2 = (B, C)$ $\Rightarrow$ lossless as $R_1 \cap R_2 \to \{B\}$ and $B \to BC$

WORK :

① $R_1 = (A, B)$, $R_2 = (A, C)$

lossless

as $R_1 \cap R_2 = \{A\}$ and $A \to AB$

Note :-

$B \to BC$

$\Rightarrow B \to \{B, C\}$.

(A, B, C)

(A, B) (B, C)

↳ whether they preserve dependencies which already exist or not?

↓

Yes !

⇓

$A \to B$, $B \to C$ are preserved!

⇓

So such decomposition ⇒ lossless decomposition with dependency preservation

(A, B, C)

(A, B) (A, C)

⇓

$A \to B$

but $B \to C$ nahi preserved in this

⇒ so dependency preservation nahi hai !

→ It is not necessary, but it must try to preserve the dependencies.

kyi vaar hoga ki table hi itna bda hai ki, preserve kr hi nahi paa she ho preserve !.

work ☺

⊙ Schema :-

A B C

dept_advisors (s_ID, i_ID, department_name)

with functional dependencies :-

i_ID → dept_name ⇒ B → C

s_ID, dept_name → i_ID ⇒ A, C → B

repetition → so decomposition of dept_advisor is required.

(A, B, C)

(A, B) (B, C) → dependency A, C → B not preserved !!

A → BC ⇒ A → B and A → C

but

AB → C ⇏ A → C and B → C

Normal forms :-

⊙ 1NF (1st Normal form)

⊙

when values are atomic in nature, then rn is called in 1st Normal form.

A	B
123	145A
124	15B

⇒ 1st Normal form.

A	
123, 456	

2 diff values in 1 row

↓ not in 1st Normal form!

WORK :

BCNF: (Boyce-Codd Normal form)

A^n R is in BCNF if w.r.t set F of functional dependencies if ∀ functional dependencies in F+ of form $\alpha \to \beta$

where $\alpha \subseteq R$ and $\beta \subseteq R$, at least one of following holds :-

1. $\alpha \to \beta$ is trivial
2. α is superkey of R.

yeh toh jyada tar hota hi^ hai!

agar koi r^n BCNF nahi^, then decompose it into r^n's jo BCNF haun!

means agar $\alpha \to \beta$ hai,
toh α must be superkey (tabhi^ BCNF hoga r^n, o/w nahi^!.

let, R = (A, B, C, D, E)

let,
F = {A → BC, C → B, D → E, E → D}
find F+ and all attributes.

F+ = { A → BC, A → B, A → C, C → B, D → E, E → D,
~~A → A, B → B, C → C, D → D, E → E~~ } (trivial wale mat likho)

for F+, first :- $A^+ = \{A, B, C\}$
$B^+ = \{B\}$
$C^+ = \{B, C\}$
$D^+ = \{D, E\}$
$E^+ = \{D, E\}$

Join :- → cross product
① Cross Join ② Natural Join
③ Conditional Join ④ Equi Join ⑤ self Join
⑥ Outer Join
↙ ↓ ↘ full
left Right

→ hm tab krte hain join 2 tables ko, jab alag-2 dono se jo chahiye wo nahi mil rahi info.

for eg:

Employee ⇒

E_No.	E_name	Address
1	Ram	Delhi
2	Varun	chd
3	Ravi	chd
4	Amrit	Delhi
5	Nitin	Noida

(giving only Employee details)

Department ⇒

Dep_No	Name	E_No → (acting as foreign key, references Employee)
D_1	HR	1
D_2	IT	2
D_3	MRKT	4
D_4	Finance	5

(giving only department details)

⇒ To get address by E-name ⇒ sirf Employee table karafi hai!

eg:
```
select Address from Employee where
    E_name = 'Varun';
```
↳ output : chd

⑥ Find E-name of Emp whose working in HR dept.
↳ Iske liye dono tables ki help chahiye
→ so need to use Join

⑥ Join tabhi lgega agar kuch common hoga in both tables

WJMK ♡
♡♡.

Join : Cross product + select statement (condition)

Natural Join :-

WJMK ॐ नमः

30 Jan 2023
Tuesday

Lec-7 :-

Q. $R1(A, C, B, D, E)$
$F = \{A \to B, \quad C \to D\}$
whether r^n is in BCNF or not?

A. Firstly find superkey, candidate key of r^n.
Find closure of all attributes.

$A^+ = \{A, B\}$, $B^+ = \{B\}$, $C^+ = \{C, D\}$, $D^+ = \{D\}$,
$E^+ = \{E\}$

↑ Now check, kya any attribute is candidate key? ↑ means which can give all attributes!

So here, none of them is candidate key!

Now, we'll try combinations of closure of attributes.

$(A\ B)^+ = \{A, B\}$
$(A\ C)^+ = \{A, B, C, D\}$
$(A\ D)^+ = \{A, B, D\}$
$(A\ E)^+ = \{A, B, E\}$
$(B\ C)^+$
⋮

↑ no one is

So none of them is candidate key, as ~~not~~ giving all attributes.

$(A\ C\ E)^+ = \{A, B, C, D, E\}$

↑ this is giving all attributes!
So, this is a candidate key.

WJMK ☺
35 MM.

and as this is minimal candidate key,
so, $(ACE)^+$ is super key!

$\alpha \to \beta$ → not trivial!
so check, α is superkey or not!
as, A is not ", C is not superkey
in $A \to B$, $C \to D$.
↑ not satisfying condⁿ of BCNF.

as, R1 is not in BCNF, so we'll decompose it! --- tab tak kro jab tak BCNF na ho jaaye!
as, $A \to B$, $C \to D$ (both are violating BCNF)

- we'll decompose R into :→
  1) $\alpha \cup \beta$
  2) $(R - (\beta - \alpha))$
  → ∀ $\alpha \to \beta$ which cause violation of BCNF.

Both FD's violating BCNF.
for $A \to B$
1) $\alpha \cup \beta \Rightarrow R2(A\ B)$
for $A \to B$
2) $(R - (\beta - \alpha)) \Rightarrow$

for overall ⁿ, dependencies are preserved as after decomposition, as dependency toh pehle se thi

R1 (A, C, B, D, E)
→ R2 (A, B)
R3 (A, C, D, E)
dependency preserved
dependency not preserved.

as dekho yeh dono BCNF hain yaa nahi°

for R2 → $A^+ = \{A, B\}$
so A is superkey! (minimal candidate key)
so, $A \to B$
↑ super key!

WJMK $\frac{c}{5}$ mm.

Thus, R2 is in BCNF.

Now R3 (A, C, D, E)

$C \to D$

$A^+ = \{A\}$

$C^+ = \{C, D\}$

$D^+ = \{D\}$

$E^+ = \{E\}$

so, $(A\ C\ E)^+$ is super key.

C is not ", also $C \to D$ is not trivial.

so not in BCNF.

So decompose it further :→

R3 (A, C, D, E) - - - - → (Dependency not preserved)

↙ R4 (C, D) ← BCNF ✓

↘ R5 (A, C, E) ← No FD for this

so ↑ trivial dependecies saari ✓

so ↑ $(A C E)^+$ is S.K.

so ↑ BCNF ✓.

Q2. R (A, B, C, D, E), F = { $A \to E$, $BC \to A$, $DE \to B$ }

check if in BCNF, if no decompose it.

A2.

$A^+ = \{A, E\}$

$B^+ = \{B, A\}$

$C^+ = \{C, A\}$

$D^+ = \{D, B\}$

$E^+ = \{E, B\}$

$(A\ C\ D\ E)^+ = \{A, B, C, D, E\}$

$(B\ C)^+ = \{B, C, A, E\}$ ~~(??)~~

↳ as $BC \to A, A \to E$

so, super key = {C, D, E}, {B, C, D}, {A, C, D}

as $BC \to A$, $A \to E$, $D \to D$

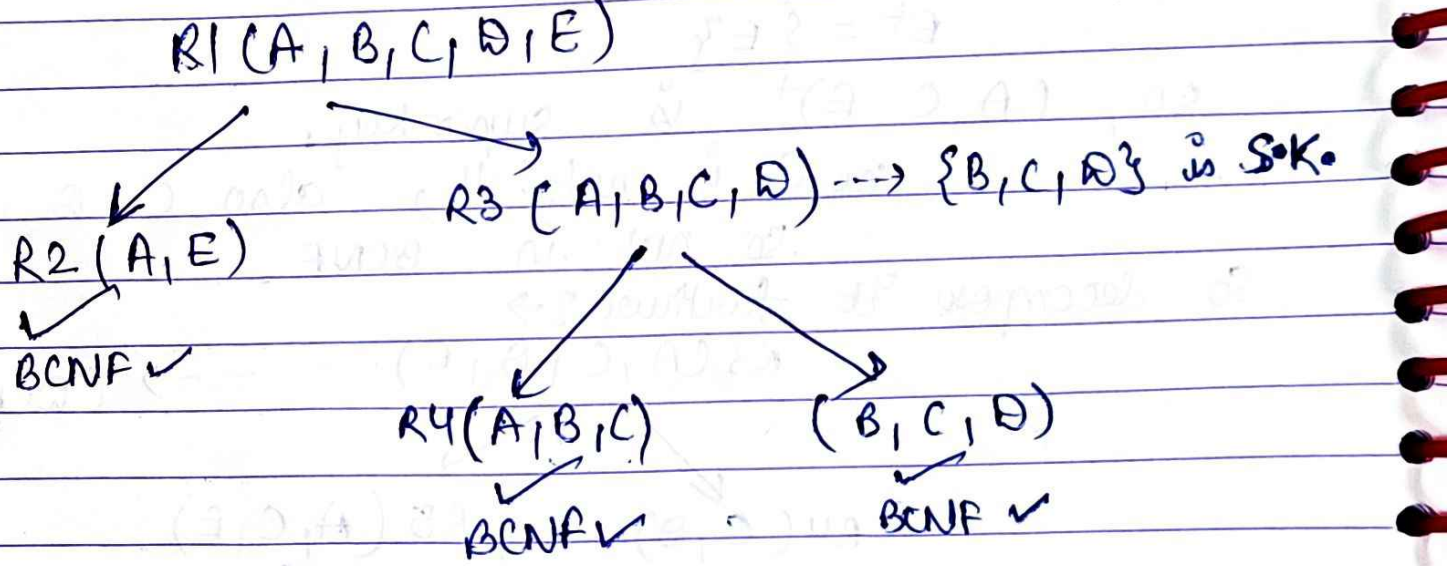
$(A C E)^+ = \{A, E, C, B\}$

WOJMK ☺ 5 MM.

let {B, C, D} is super key!

$\begin{cases} A \to E \\ BC \to A \\ DE \to B \end{cases}$, A not SK.

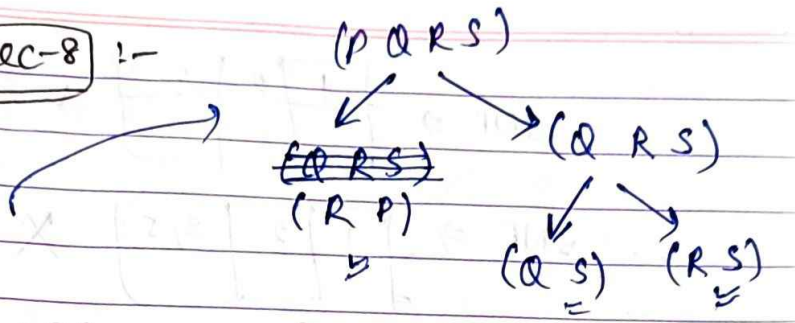
↳ all violating BCNF.



WJMk ☺
35 ♡M ☺

2 Feb 2023
Friday

lec-8 :-



① Relation (P Q R S)
$F = \{QR \to S, R \to P, S \to Q\}$, check if Relation in BCNF.

A. Candidate key, S.K. = {QR, RS}, ~~{R,S}~~ {R, S} again yeh logic, no one will satisfy BCNF

It is not BCNF. as $R \to P$, $S \to Q$ violates BCNF.

② **3rd Normal form (3NF) :-**

→ A relation R is in 3NF if $\forall\ \alpha \to \beta$ in f^+ at least one of the following holds :-

(1) $\alpha \to \beta$ is trivial (i.e. $\beta \subseteq \alpha$)
(2) α is a S.K.
(3) Each attribute A in $\beta - \alpha$ is contained in a candidate ~~key~~ key for R.

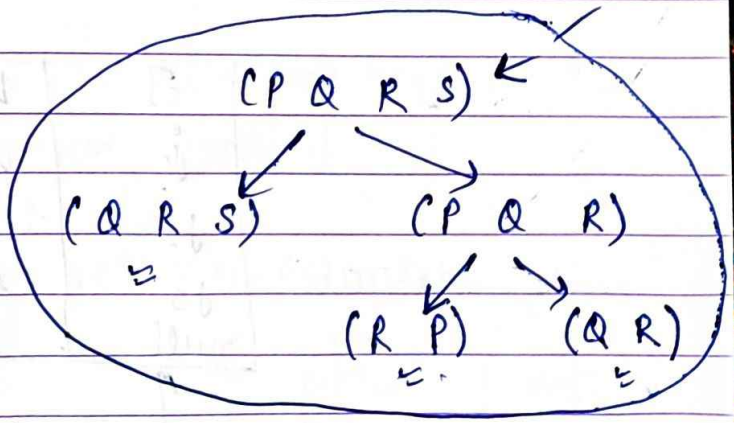
$\beta - \alpha$

attributes in (QP) is Q, P.
they both must be part of candidate key

as, $S \to Q$, $R \to P$, $R \not\to S$

then, $SR \to QPS$
$\beta - \alpha = (QP) \not\to SR$

Thus above R" , not 3NF.



WMK ॐ

1NF ⇒ | 1 | 2 | 3 | ✓

2NF ⇒ | 1 | 2 | 3,5 | ✗

BCNF
↓
3NF
↓
2NF
↓
1NF

redundancy ↑ is ☹

- partially dependent on candidate key?
  ↓
  i_ID → depends partially on candidate key as dept_name is not " " but part of " ".

① R = (I, K, L), F = {IK → L, L → K}

I	L	K
i1	l1	k1
i2	l1	k1
i3	l1	k1
null	l2	k2

kya yeh table satisfies FD or not?

WMK ॐ

If, $\alpha_1 \to \beta_1$
$\alpha_2 \to \beta_2$
If $\alpha_1 = \alpha_2$ then $\beta_1 = \beta_2$ must! → Tabhi $\alpha \to \beta$ FD.

as, $l_1 \to k_1$, $l_2 \to k_2$
so, $L \to K$ ✓

similarly, $IK \to L$ ✓

→ Wrong with Table ⇒ Repetition of Information.
⇒ need to use null value (eg to represent r'ship l_2, k_2, where there is no corresponding value for I).

not in BCNF!!
Candidate keys; SK = {I,K}, {L,I}

for 3NF:-
$IK \to L$ ✓ as IK is SK.

$L \to K$
⇒ K - L = K ∈ candidate key = {I, K}
so 3rd property satisfied
so in 3NF!!
↳ Biggest problm ⇒ Redundancy prblm
↓
repetition of info

So, 3NF hai, but repetition also ☹
↳ bcoz agar BCNF nahi hai toh decompose kete, decomposition kene se redundancy ks, but 3NF hai toh decompose nahi krenge! redundancy rhegi →

WBMk & 3rd NF.

U will use null values & info repetition (major drawback of 3NF)

ID	name	phone
9	David	512-555
9	David	512-556
9	William	512-555
9	William	512-556

↳ same phone

↑ so wrong info !!

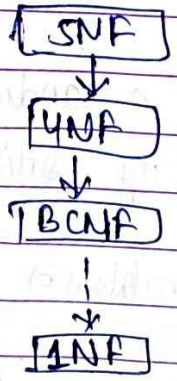
↑ BCNF ✓

{ → still redundancy !!
{ → insertion anomaly !!

Inst_info (ID, name, phone)

To remove this, we go to 4NF

↳ agar isme bhi redundancy ho, go to 5NF ... so on!



$A \to B, B \to C \Rightarrow A \to C$

$CG = GC$

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

$F^+ = \{A \to B, B \to C, A \to C\}$

$\alpha \to \beta \Rightarrow \gamma\alpha \to \gamma\beta$

(a) Armstrong's Axioms :-

(1) Reflexive rule
if $\beta \subseteq \alpha$ then $\alpha \to \beta$

(2) Augmentation rule
if $\alpha \to \beta$ then $\gamma\alpha \to \gamma\beta$

(3) Transitivity rule
if $\alpha \to \beta, \beta \to \gamma$ then $\alpha \to \gamma$

$R = (A, B, C, G, H, I)$

$F = \{A \to B, A \to C, CG \to H, CG \to I, B \to H\}$

$AG \to CG$

then,

$F^+ = \{A \to B, A \to C, CG \to H, CG \to I, B \to H, A \to H, AG \to I, CG \to HI, A \to BC, A \to BCH \ldots\}$

(b) Union Rule :-

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

(c) Decomposition Rule :-

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

(d) $(AG)^+ = ?$ Result $= AG$

$= ABCG$ $(A \to C, A \to B)$

$= ABCGH$ $(CG \to H, CG \subseteq AGBC)$

$= ABCGHI$ $(CG \to I, CG \subseteq AGBCH)$

so, AG is candidate Key.

② from F, can get F^+

(it will have more no of FD than F)

A.F.D ⇒ $F < F^+$

Full form ??

$F^- < F < F^+$

closure of F containing less no. of FD than F.

Eg: AB → C

→ remove B ⇒ (A→C) → (Stronger Result).

bcoz A→C logically implies AB→C but AB→C does not ~~even~~, on its own, logically imply A→C.

(as it may be possible that (B) attribute is Extra)

as, ID, name → Marks

↑ Extraneous attributes! remove it!

but zaroori nahi, agr vaise (AB) combination ki C de paayega ID → Marks.

→ so find all redundant attributes in FD & remove them.

$\alpha \to \beta$

⓪ AB → C

↑ dekhna konsa extra!!

⇒ Now $A \in \alpha$,

F logically implies $(F - \{\alpha \to \beta\}) \cup \{(\alpha - A) \to \beta\}$.

To remove extraneous attribute from left side

WORK SHEET 6

① To remove extraneous attribute from right side :-
   i) $A \in \beta$ and.
   ii) $(F - \{\alpha \to \beta\}) \cup \{\alpha \to (\beta - A)\}$ logically implies F

Example :-

① let $F = \{AB \to CD, A \to E, E \to C\}$

② To check if C is extraneous in $AB \to CD$, we :

(i) Compute $(AB)^+$ under $F' = \{AB \to D, A \to E, E \to C\}$

(ii) closure is $ABCDE$ ($= R$), which includes CD

(iii) This implies C is extraneous.

$F' = (F - \{\alpha \to \beta\}) \cup \{\alpha \to (\beta - A)\}$

$\Downarrow$ $A \to E, E \to C$; $\Downarrow$ $AB \to D$

→ closure of AB nikaala under this F'.

as $(AB)^+ = R$

↑ w/o 'C' → R mil gya, means C was redundant, extra attribute in F → uske bina bhi R mila, so remove it ;

③ Now check whether A is extra or not ?! → for F'

as, F logically implies $(F - \{\alpha \to \beta\}) \cup \{(\alpha - A) \to \beta\}$

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

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

WIMK Example.

Now, $B^+ = BD \neq R$

$AB^+ = ABD$ (in F')

$(AB)^+ = ABCD$ (in F)

$\Downarrow$

as $ABD \neq ABCD$

means A is not extraneous attribute.

WORK is worship.

6 Feb, 2023
Tuesday

Lec-9 :-

① Computing a Canonical Cover :-

$R = (A, B, C)$
$F = \{A \to BC, B \to C, A \to B, AB \to C\}$

- Combine $A \to BC$ and $A \to B$ into $A \to BC$
  So, $F = \{A \to BC, B \to C, AB \to C\}$ → Now B or C or A can be extra!
- A is extraneous in $AB \to C$
  → check if result of deleting A from $AB \to C$ is implied by other dependencies
  - Yes! infact $B \to C$ is present.
  So, $F = \{A \to BC, B \to C\}$
- C is extraneous in $A \to BC$
  → check if $A \to C$ is logically implied by $A \to B$ & other dependencies.
  → Yes using transitivity on $A \to B$ & $B \to C$.
  → can use attribute closure of A in more complex cases.

Canonical Cover is : $A \to B$
$B \to C$

② $R(ABCD)$ $\{A \to B, B \to C\}$ = FD

$R_1(AC)$ $R_2(ABD)$ → $A \to B$ is preserved!

$A \to B$ not preserved here → after decomposition, ~~FD~~ F.D. $B \to C$ is not preserved !!

Dependency is preserved if it is present in any one of the decomposed relation.

WJMK ☺

⊙Q. $R(A\ B\ C\ D\ E)$

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

R^n is in BCNF or 3NF?

lossless decomposition hai yaa nahi?

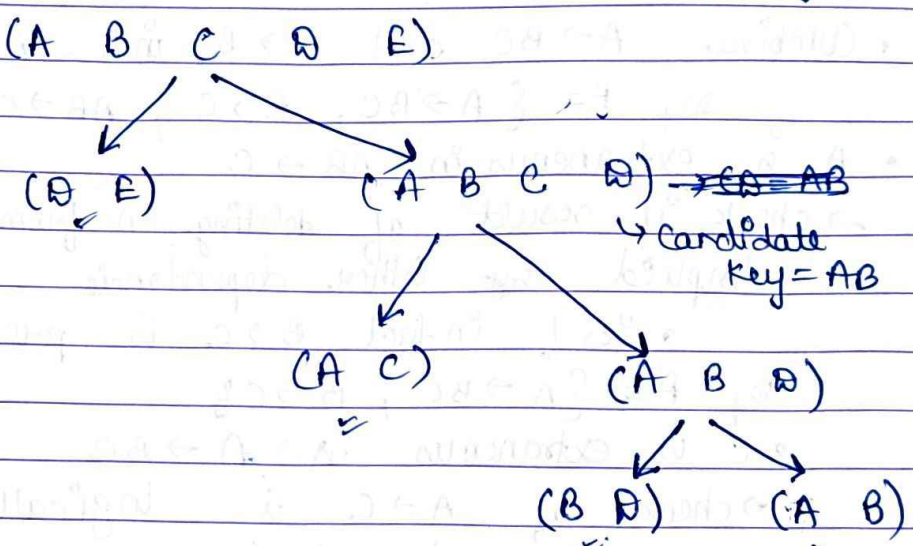
Dependency preserved " " "

A. R^n is not in BCNF nor in 3NF. → (so decompose it!)

Candidate key = S.K. = (A B)

↳ $D \to E$, $A \to C$, $B \to D$

(not part of candidate key)



So,

$ABCDE \to (DE)\ (AC)\ (BD)\ (AB)$ → $AB \to AB$

$D \to E$ $A \to C$ $B \to D$

So, $AB \to CD$ is not preserved.

So, it is BCNF, but dependencies are, ~~too~~ not preserved.

⇒ All BCNF decompositions are lossless decomposition, show it!

→ yeh 3no already hain!!

so, sirf 1st wali dependency, we have to find!

WJMK ≡ 3rd.

can derive $AB \to CD$?

(AB)$^+$ se

$\Rightarrow AB \to AB$ (trivial)

as $A \to C$

$\Rightarrow AB \to CB$

and as $B \to D$

$\Rightarrow \boxed{AB \to CD}$

so, saari FD mil gyi!

Thus lossless decomposition.

Testing for BCNF :-

we are checking FD in F, not in F$^+$, for simplification.

$R = (A\ B\ C\ D\ E)$

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

Candidate key = (A C E)

most important attribute, not present in any FD

so have to include in candidate key.

$R(A\ B\ C\ D\ E)$

$R_1(A\ B)$ — $A \to B$ ✓

$R_2(A\ C\ D\ E)$ (contains no dependency from F)

so we think R_2 is in BCNF

mislead! No! esa nahi hoga!

so sometimes F$^+$ consider krna pdhega, tab voor simplified se kaam nahi chlega

WIMK e 30 ♡♡.

⑥ 3NF decomposition is always lossless & dependency-preserving decomposition.
(as removing some restrictions from BCNF)

Work from home.

9 Feb 2023
Friday

Lec-10 :-

↗ r^n has to be in BCNF, for to be in 4NF.

① 4NF :- (3.5 NF)

MVDs (multivalued dependencies)

inst_child (ID, child_name)
inst_phone (ID, phone_no)
inst_info (ID, child_name, phone_no)

eg:

(99999,	David,	512-555)
("	"	512-556)
("	William,	512-555)
("	"	512-556)

ID	child_name
99999	David
99999	William

ID	phone_no.
99999	512-555
99999	512-556

this shows child having this phone-no.; but actually we want to say that phone_no. are associated with Instructors.

→ so yeh dono likhe !!

This r^n is in BCNF.
as, there candidate key = (ID, child_name, phone_no) = R

problem ⇒ ID, child_name, phone_no. all things are repeating
↳ so repeated info

100 MK ☺ 35 ♥♥.

So redundancy !!
↳ thus decomposition req!
↳ done into inst_child, inst_phone !!

here, ID → C1 → child_name
ID → C2

ID → P1 → phone_no.
ID → P2

ID →→ C } → means multiple child & phone are dependent on ID.
ID →→ P }

	α	β	
t_1	99999	David	512-555
t_2	99999	David	512-556
t_3	99999	William	512-555
t_4	99999	William	512-555

↑ tuples

(These 2 attributes must be independent !!)

$\alpha \twoheadrightarrow \beta$
iff ∀ pairs of tuples t_1 & t_2 in r s.t.
$t_1[\alpha] = t_2[\alpha]$, there exists tuples t_3 & t_4 in r s.t.

$t_1[\alpha] = t_2[\alpha] = t_3[\alpha] = t_4[\alpha]$
$t_3[\beta] = t_1[\beta]$
$t_3[R-\beta] = t_2[R-\beta]$
$t_4[\beta] = t_2[\beta]$
$t_4[R-\beta] = t_1[R-\beta]$

then only MVD exists !!

WORK ON MVD.

⊙ MVD → Tabular representation ?

$Y \to Z$ hai then for $Y \twoheadrightarrow Z$

har attribute value must be same.

~~R Y Z → main bhi same attribute value same.~~

4NF main aim :

① Remove MVD

→ Once R is in BCNF, if there is MVD, then decompose it to make 4NF.

Here BCNF hai ✓, remove MVD !!

Esse decompose kro !!

(A, B, C)

(A, B) BCNF ✓ SK = A

(A, C) BCNF ✓ SK = A

→ atleast 3 columns are req. for MVD to exist.

Q. R = (A, B, C, G, H, I)

F = { A →→ B, B →→ HI, CG →→ H }

• R is not in 4NF as A →→ B, A is not a superkey for R.

↳ must not have MVD, par hai, so 4NF X

↳ thus decompose the table.

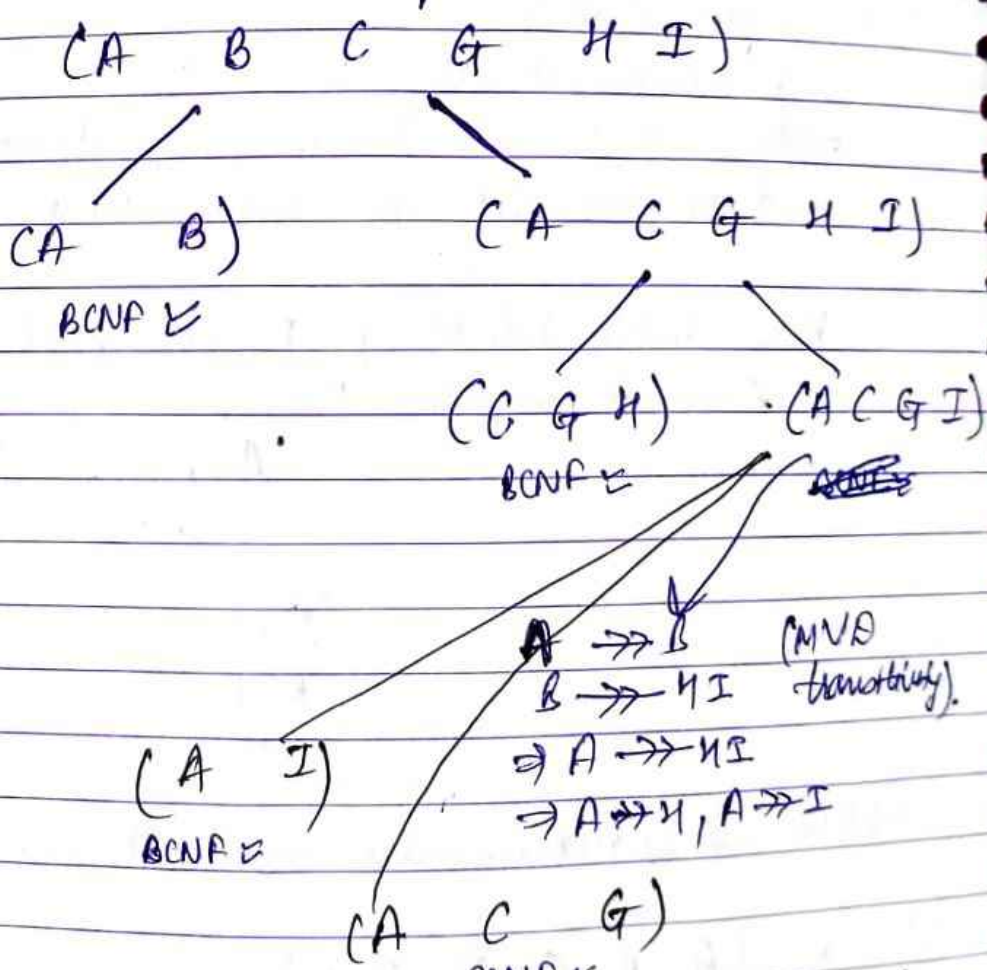
WORK ≡ 50 ≡ mm.

• pehle dekho BCNF mein hai yaa nahi?
what is super key?

SK = { A, ~~B, H, I~~, C, G } = candidate key

$A \twoheadrightarrow B$

not superkey,
so not in BCNF,
so decompose.



(5) PJNF: Project Join Normal Form):- (5th Normal Form).

difficult to identify from table.

so not used generally.

COOMK

BCNF → always in 3NF
3NF → ——— 1NF
4NF → ——— BCNF

- Join dependency (JD) is generalization of MVD.
- ~~A JD → JD(R1, ..., Rk) is said to hold on~~ ~~if:~~

eg :-

Project	skill
Datamining	python
"	Java

R1

roll no	skill
↑	C++
1	python

R2

roll no	Project
1	Datamining
1	computer vision

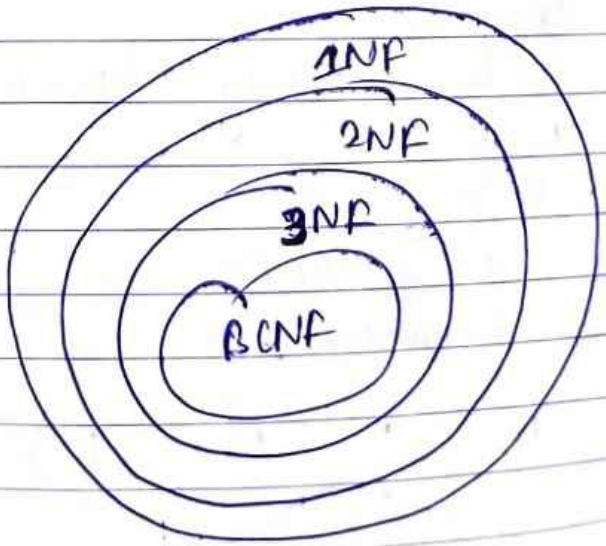
R3

(but JD must hold)

Inka join, must = R.

↓

but while joining → repeated info may come ☹



⑥ Table is in 2NF when :-

i) Table is in 1NF

ii) No non-prime attribute is dependent on proper subset of any candidate key of table.

WOMK ☺
= ♡♡.

Q :- R(A, B, C, D, E)
F = {AB → C, D → E}
CK = candidate key = (A B D) → (then A, B, D are prime attributes as they belong to CK)
C, E → non-prime attributes
Now, AB → C
(AB: depends on proper subset of CK; C: non-prime attribute)

So, AB → C does not satisfies 2NF condⁿ.

Similarly D → E → does not satisfies 2NF condⁿ.

So, Rⁿ not in 2NF.

~~(A B C D E)~~

AB → C
here C is partially dependent on AB

D → E
here E is " " " D.

So not in 2NF
So, decompose s.t. 2NF bn jaaye !!

R(A B C D E)

AB → C
→ R1 (α ∪ β) = (A B C)
d 2NF ≅
AB = CK
AB → C.
(Now C is completely dependent on CK = (AB))

(R − (β − α)) R2 (remove C) = (A B D E)

D → E R21 = (D E)
D = CK
2NF ≅

R22 = (A B D)
2NF ≅
(No FD)

WORK :

Q. R(A B C D E F)

FD = { ABC → D , ABD → E , CD → F , CDF → B , BF → D }

C → B

Find Normal form of this relation.

A. CK = (A B C) , (A C D)

(Umesha check from higher to lower)

~~(A B C D E F)~~

R → not in BCNF as ABD → E is violating !!

ABD → (E) → violating 3NF.

B − α = E → not part of CK, Candidate key !!

(CD) → F → violating 2NF

proper subset of CK = (A C D)

so R is in 1NF.

⊙ Temporal FD X → Y holds on schema R if FD X → Y holds on all snapshots for all legal instances r(R).

transaction ki at 10 am → database mein gyi → wo ek snapshot.

WJMK 5
5 Pm.

13 Feb 2024
Tuesday

Lec-11.

PL/SQL
(Procedural language Extension to SQL)

- SQL do not have procedural capabilities
- SQL does not provide programming techniques of cond^n checking, looping & branching that is req. for data before permanent storage.
- SQL statements are passed to oracle engine one at time, each time an SQL statement will be executed, so each time need to call oracle engine. This will increase traffic on the network.

- PL/SQL is superset of SQL.
- any error that occurs in SQL, oracle engine will display its own error message, SQL has no facility to provide user friendly error.

Adv :-

1. PL/SQL provide facilities of cond^al checking, branching & looping.
2. PL/SQL can be used any where, either in SQL statements or PL/SQL blocks.
3. PL/SQL dealing with error & facilities displaying user-friendly msgs, when error occured.
4. PL/SQL allows declaration of variable, variable, we can use in intermediate result of query processing or calculate values & insert them into table.

WJMK
SMM.

(5) improves performance by adding procedural processing power to Oracle tools.

(6) all sorts of calculations can be done quickly & efficiently without use of oracle engine.

(7) PL/SQL are portable to any OS & Platform on which Oracle runs.
Applications written in

PL/SQL Blocks :-

- PL/SQL code is built of blocks, with a unique structure.

2 types :-

1. Anonymous :- have no name (like scripts)
2. Named

DECLARE (option) → can declare variables you will use in this block.

BEGIN (mandatory)

EXCEPTION (optional) → Exceptional error handle karne ke liye

END; (mandatory)

/

(A) DECLARE SECTION :-

- Code blocks start with a declaration section.

- Declaration of memory variables, constants, cursor etc.
- Once declared, they can be used in SQL statements for data manipulation.

BEGIN :-

→ consists of SQL executable statements as well as PL/SQL statements.

→ SELECT, INSERT, UPDATE, DELETE are supported.

→ CREATE, DROP, ALTER → not allowed.

→ not case sensitive

EXCEPTION :-

- basic char set includes :-
  1. A-Z
  2. 9-Z
  3. 0-9
  4. ( ), * & @ etc.
- words used in PL/SQL blocks are called lexical units.
- can insert free blanks spaces in lexical unit, there will be no effect on PL/SQL block.

⑥ LEXICAL UNITS :- ① Identifiers ② Delimiters ③ literals ④ Comments

Includes letters, numerals, special chars, tabs, spaces, returns, symbols

names given to PL/SQL objects.

WORK 2
= 5 M.

Quoted Identifier:- makes identifier case sensitive, include characters such as space and use reserved words. Example:-
① "begin date" DATE;

Delimiters:- symbols having special meaning

; → terminates PL/SQL statement.

$ nahi hai
^ special symbol hai

:= → assignment operator.
@ → remote access indicator
/* */ → for comments.
= → equality operator
+ → add^n operator.
- → subtraction / negation operator.
<> → Inequality operator
/ → division operator.

LITERALS:-
① Numeric literals
↳ can be Z or float.
② String literal
' hello there '
↑
single quote
③ character literal
(*)

WORKS

① Two hyphen (--) → for single line comment.

② PL/SQL data types :-

1. Number → can write INT / FLOAT also explicitly OR use NUMBER.
2. Char
3. Date
4. Varchar
5. Boolean ↑ not supported by SQL

Variables :-

- Variable must begin with character & can be followed by max 29 other.
- Reserved words can not be used.
- Variables separated from each other by punctuation mark.
- white space can't be used in variable name.

Assignment :-

1. Use :=
2. select or fetch table data values into variable.

CONSTANT :-

keyword CONSTANT must be added to variable name. & assign value immediately.

DISPLAYING USER MESSAGE :-

1. To display msg, SERVEROUTPUT must be set ON.

SERVEROUTPUT [ON/OFF]

[www.sql.oracle.com]

WWMK :
SMM.

```
n int;
y int;
x := 13.6;
y := 12
n/y = ?
```

Example :- SQL > ED
wrote file afiedit.buf

```
DECLARE
    A NUMBER;
    B REAL;
    D VARCHAR(16);
BEGIN
    A := B.6;
    B := 12;
    DBMS_OUTPUT.PUT_LINE ('A/B =' || A/B);
END;
```

↑ concatenation

```
SQL> /
```

PL/SQL procedure successfully completed.

```
SQL>/
PL/SQ  "   "   "
SQL> SET SERVEROUTPUT ON;
SQL>/
A/B = 1.1333. - - - -3
```

Condition Control :-

```
if <cond>
    then <command>
    else if <cond>
        then < >
        else
            < >
    end if;
end if;
```

```
if < >
then
    if < >
    then
        < >
    end if;
else < >
end if;
```

WORK:

Example :-

```
declare
D boolean = null;
begin

If D then
    DBMS_output.put_line ('True');
else if D is null then
                    ('null');
else
                    ('False');
end if;
end if;
end;
/
```

Output :- null

ITERATIVE CONTROL :-

```
SQL> ED
Wrote file afiedt.buf
    declare
        I integer := 0
    begin
        loop
        I := I+2;
        EXIT WHEN I>10;
        END loop;
        DBMS_OUTPUT.PUT_LINE(I);
    END;
SQL>/
12
PL/SQL procedure successfully completed.
```

WHILE LOOP :-

```
WHILE <cond n>
    command.
End loop;
```

eg :-

```
declare
    i integer := 1;
begin
    while i < 10 loop
        DBMS_OUTPUT.PUT_LINE(i);
        i := i+1;
    END LOOP;
```

FOR LOOP :-

```
FOR VARIABLE IN [REVERSE] start .... end
LOOP
    <ACTION>
END LOOP;
```

(can write for decrement)

eg :-

```
declare
  k integer := 0;
begin
    FOR k IN 1 .. 10 LOOP
        DBMS_OUTPUT.PUT_LINE(K);
    END LOOP;
    END;
    /
```

⑥ DATA SELECTION

↳ nothing but just projection

```
BEGIN
 SELECT <col 1>, ... <col n> INTO
 <variable 1>, ... <variable n> → (PL/SQL variables)
        FROM <table_name> WHERE <cond n>
                                END;
```

WOMK

DATA DELETION :-

```
Begin
  Delete
  FROM  <table_name>
  WHERE  <condn>
  END ;
```

DATA UPDATE :-

```
begin
  UPDATE <table_name>
  SET
  <col1> = <val1> , ..<coln> = <val n>
  WHERE <cond'n>
  END;
```

↳ SQL mein toh saara fuse insert kena pdhta tha 😕

DATA INSERTION

work ☺

16 Feb 2024 lec-12 :-
Friday

manipulation
↳ delete, update, insert

⊙ COMMIT ?
⊙ Data can't be inserted ek saath ... ek-ek karke hi kr Skte hain bss !

CURSOR :-
→ oracle engine uses a work area for its internal processing in order to execute an SQL statement.
→ This work area is private SQL's opn & is called cursor.
↑
is temporary memory location
↳ as for complex opn's → we need space (directly can't insert) → kernel.
↳ where DML structure is performed.

Implicit : server will automatically create space
explicit : user will create space
↳ can tell why error.
→ yeh sirf btaayega ki error hai, kya
↳ wo nahi btaayega!

General cursor attributes :

attribute	type	description
%ISOPEN	boolean	evaluates true if open
%NOTFOUND	"	true if most recent fetch does not return row
%FOUND	"	true if most recent fetch returns row
%ROWCOUNT	"	evaluates to total no. of rows returned so far

(How much data you manipulated ?? → wo btaayega!)

Implicit cursor eg:-

① DECLARE var_rows number(3);
BEGIN
UPDATE employee
SET salary = salary * 1000;
if SQL%NOTFOUND THEN
DBMS_OUTPUT.PUT_LINE ('no no. of salaries where updated');
ELSEIF SQL%FOUND THEN
var_rows := SQL%ROWCOUNT;
DBMS_OUTPUT.PUT_LINE ('salaries for' || var_rows || 'employees are updated');
END IF;
END;
/

② DECLARE
ROWS_AFFECTED CHAR(4);
BEGIN
UPDATE EMPLOYEE SET STATUS = 'ACTIVE' WHERE STATUS = 'notactive' AND BRANCH = 'abu';
ROWS_AFFECTED := TO_CHAR (SQL%ROWCOUNT);
IF SQL%ROWCOUNT > 0 THEN
DBMS_OUTPUT.PUT_LINE ('ROWS AFFECTED')
ELSE
DBMS_OUTPUT.PUT_LINE ('NO ROWS AFFECTED');
END IF;
END;
/

WIMK:
3rd M.

① Explicit Cursor :-

① Declare → Publicize cursor into memory
② open → memory is allotted, cursor is opened.
③ Fetch → cursor retrieve data.
④ close → closed to release memory allocation

Declaring Explicit Cursor :-

CURSOR c_cursor_name IS select statement

- cursor names follow the same rules of scope & visibility that apply to PL/SQL identifiers.

Opening :-

OPEN cursor_name ;

Fetch :-

FETCH cursor_name INTO PL/SQL variables,
OR
record ;

FETCH cursorName into variable1, variable2, . .

① Eg: DECLARE
lastname VARCHAR(20),
firstname " ;
accno " ;
salary NUMBER(5) ;
CURSOR C1 IS SELECT lastname, firstname, acc-no, salary from persons;

BEGIN
OPEN C1 ;
IF C1%ISOPEN THEN
DBMS_OUTPUT.PUT_LINE('lastname' || ' ' || 'firstname' || ' ' || 'acc_no' || ' ' || salary);

explicit cursor so SQL variables

WORK SUM.

DBMS_OUTPUT.PUT_LINE('_____');
LOOP
FETCH C1 INTO lastname, firstname, accno, salary;
EXIT WHEN C1%NOT FOUND;
DBMS_OUTPUT.PUT_LINE(LASTNAME||' '||FIRSTNAME||' '||ACC_NO||' '||SALARY);
END LOOP;
END IF; close C1;
END;
/

~~STORED P~~

Where to use these cursors? Benefits?
can't perform complex opers directly, toh wo cursor se hota!
agar sidha INSERT likha, and ager kuch toh hota hai, wo implicit cursor deta!

STORED PROCEDURE / FUNCTION :-
↓
logically grouped set of PL/SQL

made up of :-
1. declarative part
2. executable part
3. optional exception-handling part

for eg: agar 2 insert thodi karenge, toh fn banaalo!

Procedure & Function
↓ ↳ can return only one value.
can return multiple values

WORK 3

① Creating Stored Procedure :-

Syntax :-

```
CREATE OR REPLACE PROCEDURE [SCHEMA]
    <procedure Name>
    (<arguments> {IN, OUT, IN, OUT}, <data type>)
     {IS, AS}
     variable declaration
     Constant declaration
    BEGIN
      PL/SQL statement
    EXCEPTION
      statement
    END;
```

REPLACE → recreates procedure if already exists
schema → oracle engine takes default schema to be current schema

Eg :

```
create or replace procedure "INSERTUSER"
(id IN NUMBER, name IN VARCHAR2)
  is
  begin
  insert into user2 values (id, name);
  end;
  /
  BEGIN INSERTUSER (102, 'Anjali');
DBMS_OUTPUT.PUT_LINE ('record inserted succesfully');
  END;
  /
```

WORK

eg:

① CREATE OR REPLACE Procedure myprocedure
(nameIN Varchar2, msgOut OUT varchar2,
msg2Out OUT varchar2)
IS
msg1 varchar2(100);
msg2 varchar2(100);
begin
msg1 := 'Hello' || nameIn;
msgOut := msg1;
msg2 := 'welcome' || nameIn;
msg2Out := msg2;
display → nameIn,
msg1, msg2,
msgOut, msg2Out
end;
/
declare
msg1var varchar2(100);
msg2var varchar2(100);
begin
myprocedure('Yawin', msg1var, msg2var);
display → msg1var, msg2var
end;
/

Output :- Yawin
Hello Yawin
welcome "
Hello "
welcome "
Hello "
welcome "

WEEK 12.

Function :-

CREATE OR REPLACE FUNCTION [SCHEMA]
<function Name> (<ARGUMENT> IN <data type>)

eg: create or replace function adder (n1 in number, n2 in number)
return number
is
n3 number (8);
begin
n3 := n1+n2;
return n3;
end;
/
DECLARE
n3 number (8);
BEGIN
n3 := adder (11,22);
display → ('Addition is ' || n3);
END;
/

Exception Handling :-

① error occurs during program execution is called Exception

WORK ON MM.

Syntax :-

```
DECLARE
.
BEGIN
EXCEPTION
    WHEN exception1 THEN
    ''    ——— 2   ''
    ''    ——— 3   ''
    - - - -
    WHEN others THEN
END;
```

Exception section :-

default
↳ not user defined exceptions
(These are system defined exceptions)

↓

1. NO_DATA_FOUND
2. payment_overdue
3. OTHERS

① EXCEPTION
WHEN NO_DATA_FOUND
THEN
. - - - ;

② WHEN payment_overdue
THEN
. - - -

WORK :)

① WHEN OTHERS THEN

END;

} If any other exception is encountered, then execute 3rd set of statements.

Example :-

```
DECLARE
  c_id customers.id%type := 8;
  c_name customers.name%type;
  c_addr customers.address%type;
  BEGIN
  SELET name, address INTO c_name, c_addr
  FROM customers
  WHERE id = c_id;
D_O.P_L ('Name: ' || c_name);
D_O.P_L ('Address: ' || c_addr);
EXCEPTION
  WHEN no_data_found THEN
    D_O.P_L ('No such customer!');
  WHEN others THEN
    D_O.P_L ('Error!');
  END;
```

/

② The NO_DATA_FOUND exception is raised under 3 circumstances :-

(1) An implicit query returns no data.
(2) You attempt to reference a row in a PL/SQL table which has not been defined.
(3) You attempt to read past the end of an operating system file.

Raising exceptions :- (User defined exceptions)

```
DECLARE
    exception_name EXCEPTION;
BEGIN
    IF CONDITION THEN
        RAISE exception_name;
    END IF;
EXCEPTION
    WHEN exception_name THEN
        statement
END;
```

DATABASE TRIGGERS :-

↳ insertion / updation / deletion se trigger hota !!

↳ as table pehle bna thya, abh manipulate kar sake ho!

↳ iske help se can verify ki manipulations sahi kar sake hain yaa nahi ??

↳ kahi 'char' wale mein 'int' toh nahi store kar sake ??

TRIGGERS → don't accept parameters, while PROCEDURES can;

↓

are called automatically (executed implicitly by oracle engine) ←

hence to be called by user explicitly !

WOMK B/3rd OM.

best ! → as pehle hi faayega hai dekhne ka !!.

Syntax: -

```
CREATE OR REPLACE TRIGGER <TriggerName>
        {BEFORE, AFTER}
      {DELETE, INSERT, UPDATE [OF COLUMN...]}
  ON <TableName>
     [REFERENCING {OLD AS old, NEW AS new}]
    [FOR EACH ROW [WHEN condition]]
  DECLARE
        <variable declaration>;
        <constant      "      >;
  BEGIN
        <PL/SQL subprogram body>;
  EXCEPTION
  END;
```

eg:

```
Create or Replace Trigger t
Before
      INSERT OR
      UPDATE of SALARY,
      Delete
      ON Employee
   Begin
      Case
   When Inserting THEN
         DBMS_OUTPUT.PUT_LINE ('Inserting')
   When Updating ('salary') then
         D_O.P_L ('updated');
   END CASE
   END; /
```

example :-

```
CREATE TRIGGER AUDIT_TRAIL AFTER UPDATE
OR DELETE
  ON PERSON
  FOR EACH ROW
DECLARE
    OPER VARCHAR2(8);
  BEGIN
    IF updating THEN
       OPER := 'UPDATE';
    END IF;
     IF DELETING THEN
            OPER := 'DELETE';
    END IF;
    INSERT INTO RECORD
VALUES (:OLD.P_ID ; :OLD.LAST_NAME ; :OLD_
          FIRSTNAME ; :OLD.ADD );
END;
/
```

WJMK ☺ ❤

17 Feb 2024
Saturday

lec-13 :-

TRANSACTIONS :-

↳ unit of program execution that accesses and possibly updates various data items.

check balance → use 'select' operation
- 'insert' operation (deposit money)
- 'delete' " (taking out money)

→ all are transactions ??

eg : transaction to transfer $50 from A to B :-

(1) read(A)
(2) A := A - 50
(3) write(A)
(4) read(B)
(5) B := B + 50
(6) write(B)

→ saare transaction ! temporary memory main hote hain & then database mein changes reflect hote hain !

A+B = 1000 (before) → total sum has to be same !
A+B = 950+50 = 1000 (after).

① Two main issues to deal with :
  ① failure of various kinds, such as hardware failures & system crashes
  ② concurrent execution of multiple transactions.
    ↳ for eg : same data
      ↑
      aapke account mein ek daal rha hai, aap nikaal rhe ho at same time !!
      ↳ concurrent execution.

WORK ≡ 30 ≡ M.

Example of fund Transfer :-

① Atomicity Requirement → either all transaction operations are properly reflected in database or not

↳ transaction must be atomic in nature.

→ each transaction has to be complete (total sum, before & after, must be same)

eg: if transaction fails after step 3 & before step 6 then money will lost!, so leading to inconsistent database state.

→ failure can be due to software or hardware.

→ system should ensure that updates of partially executed transaction are not reflected in database.

② Durability Requirement
③ Consistency Requirement
④ Isolation Requirement

if b/w step 3 and 6, another transaction T2 allowed to access partially updated database, it will see an inconsistent database.

→ T1, T2 → must happen in isolation, done ek dusre ko effect na kren, so series mein kro like pehle T1, then T2.

so, run transactions serially.

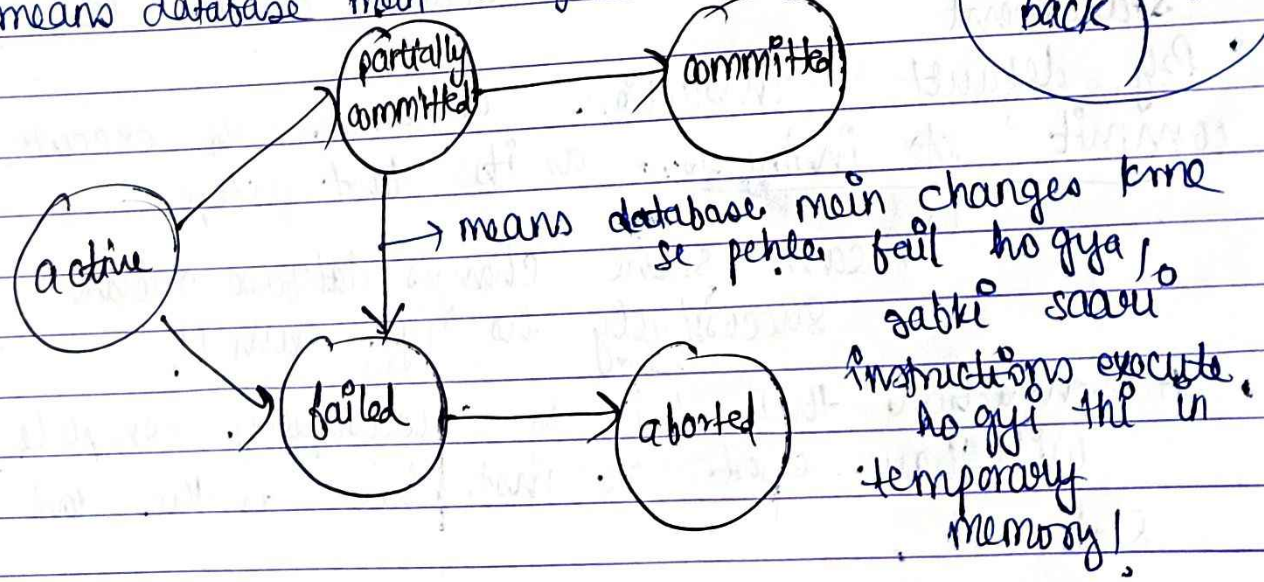
⑤ ACID properties of Transaction

- Atomicity
- Durability
- Isolation
- Consistency

WJMK ॐ

Transaction State :-

1. **Active** :- the initial state, transaction stays in this state while it is executing.
2. **Partially committed** :- (all instructions when executed) after final state has been executed
3. **failed** : after discovery that normal execution can no longer proceed.
4. **Aborted** : after transaction has been rolled back & database restored to its state prior to the start of transaction. (means go back).
   Two options after it has been aborted :
   - Restart transaction
     → can be done only if no internal logical error
   - kill transaction.
5. **Committed** : after successful completion
   ↳ means database main changes ho gye properly!



Concurrent Execution :-

↳ sometimes need!

⊙ like IRCTC mein thatka log ticket can book., essa toh nahi wait kete hahe saare!!.

→ Adv :- ① Red. processor and disk utilization
② Red. average response time

Schedules :-

↓

Sequence of instructions that specify the chronological order in which instructions of concurrent transactions are executed.

- A schedule for set of transactions must consist of all instructions of those transactions.
- Must preserve order in which instructions appear in each individual transaction.
- A transaction that succesfully completes its execution will have a commit instructions as the last statement
- By default transaction assumed to execute commit instruction as its last step

↳ means saare changes database mein succesfully ho gye hain !!.

- A transaction that fails to successfully complete will have abort instruction as the last statement.

WORK 2 5 MM.

Schedule 1

eg:

T_1	T_2
read(A) → 1000	
A := A - 50 → 950	
write(A) → 950	
read(B) → 1000	
B := B + 50 → 1050	
write(B) → 1050	
commit	
	read(A)
	temp := A * 0.1 → 95
	A := A - temp → 855
	write(A) → 855
	read(B) → 1050
	B := B + temp → 1145
	write(B) → 1145
	commit

950 + 1050 = 2000 ✓

→ commit means all changes reflected back in database.

T_1 transfers $50 from A to B, T_2 transfer 10% from A to B.

→ serial schedule in which T_1 is followed by T_2.

Say: before:

A = 1000
B = 1000 } → 2000

after:

A = 855
B = 1145 } → 2000

Work:
=30
=MM.

⑥ A serial schedule where T_2 is followed by T_1

Schedule 2

T_1	T_2
	read(A)
	temp := A * 0.1 → 100
	A := A − temp → 900
	write(A) → 900
	read(B) → 1000
	B := B + temp → 1100
	write(B) → 1100
	commit
	↳ A = 900, B = 1100 ↓
	2000 ✓
read(A)	
A := A − 50 → 850	
write(A) → 850	
read(B) → 1100	
B := B + 50 → 1150	
write(B) → 1150	
commit → 1150 + 850 = 2000 ✓	

Before: A = 1000, B = 1000 → 2000 ✓

After: A = 850, B = 1150 → 2000 ✓

① But serial wale mein toh bhut time lgega!!

Concurrent kro!!

T_1	T_2
read(A)	
A := A − 50 → 950	
write(A) → 950	
	read(A)
	temp := A * 0.1 → 95
	A := A − temp → 855
	write(A)

WORK

read(B)
B := B+50
write(B)
commit

read(B)
B := B + temp
write(B)
commit

Equivalent to schedule 1

Q :-

T_1

read(A)
A := A - 50 → 950

write(A) → ~~855~~ 950
read(B) → 1000
B := B+50 → 1050
write(B) → 1050
Commit

again read(A) = 900 (but already read pe stha 950 wo hi write karega)

T_2

read(A) → 1000
temp := A * 0.1 → ~~95~~ 100
A := A - temp → ~~855~~ 900
write(A) → ~~855~~ 900
read(B) → 1000

→ 1000 + 100
B := B + temp → ~~1145~~ 1150
write(B) → ~~1145~~ 1100
Commit

~~1000~~ 950 ~~855~~ + 1100 ~~2100~~ = 2050 → so this concurrent schedule does not preserve value of (A+B)

WOMK :
= 30 MM :
=

Serializability :→

Basic Assumption

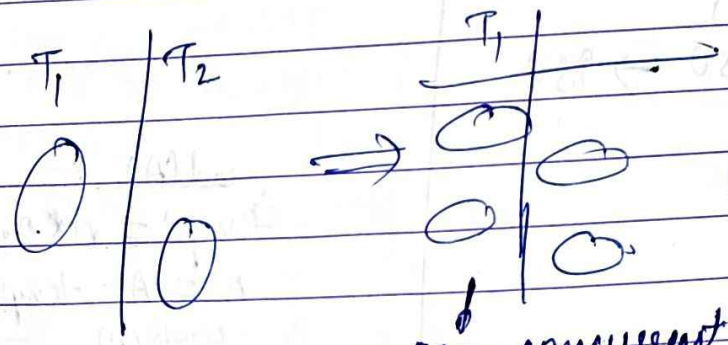
↳ Each transaction preserves database consistency.

① Thus, serial execution of set of transactions preserves database consistency.

② A (possibly concurrent) schedule is serializable if it is equivalent to a serial schedule. Different forms of schedule equivalence give rise to notions of :-

1. Conflict serializability
2. View serializability

③



Agar concurrent mein Apka, toh esa ho ki they will behave like ki serializability mein ho hain !!

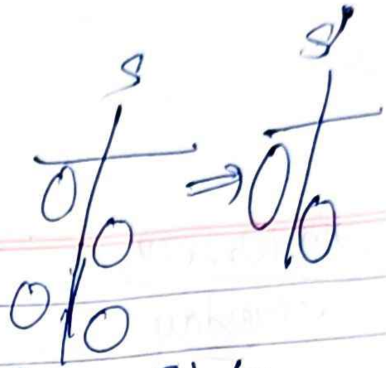
Conflicting instructions :-

- If reading concurrently → then no conflict.

read - write
write - write
write - so read
→ then conflict occurs !!

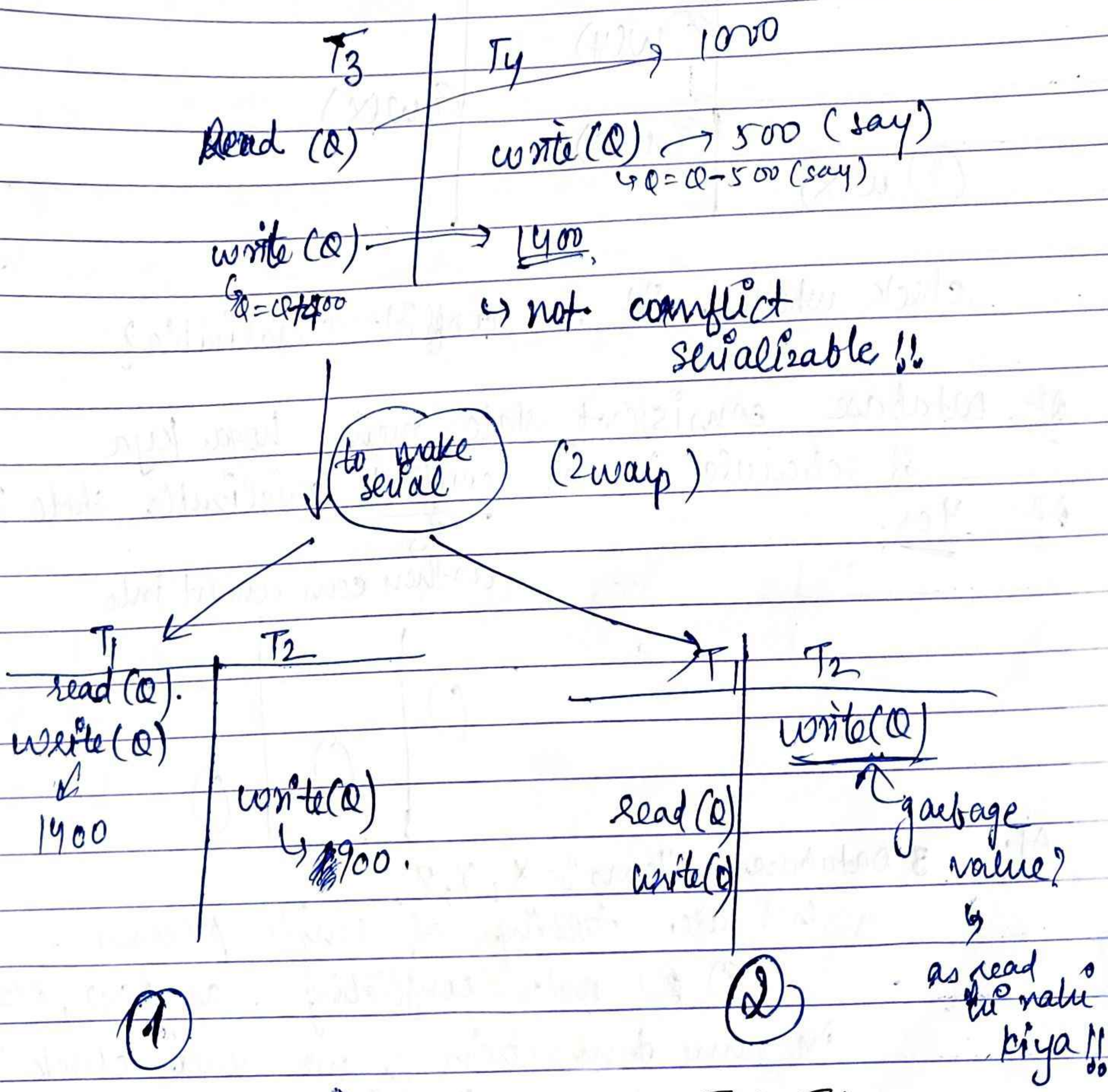
So serializability means, we work on these three !

WDMK = 2 30 = MM.



Conflict Serializability

(0) schedule S can be transformed into S' by series of swaps of non-conflicting instructions, we say S & S' are conflict equivalent.
S is conflict serializable to S', means no conflict !!



T1 > T2
↳ pehle T1 fir T2.

T2 > T1

WJMK ☺

20 Feb 2024
Tuesday

lec-14 :-

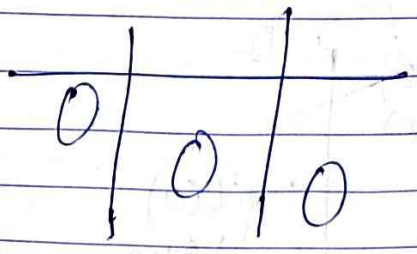
Q1

T_1	T_2	T_3
(1) R(X) → 100		
		(2) R(Z) → 100
		(3) W(Z) → 150 (+50)
	(4) R(y) → 100	
(5) R(y)		
	(6) W(y) → 150 (+50)	
		(7) W(X) → 140 (+40)
	(8) W(Z) → 140 (+40)	
(9) W(X) → 150		

check whether it is conflict serializable?

Q2. database consistent state mein hoga kya
If schedule is in conflict serializable state ??

A2. Yes.
↳ as then can convert into



A1. 3 database items : X, Y, Z
we are talking of single processor.
(1), (2) not conflicting, as R(X), R(Z)
in same transaction, we never check for conflict.
so (2), (3) no conflict!
(3), (4) → no conflict.
↳ W(Z), R(Y)

④, ⑤ → no conflict

⑤, ⑥ → conflict !!

R(y), W(y).

⑥, ⑦ → no conflict.

⑦, ⑧ → no conflict

⑧, ⑨ → no conflict !!

T_1, T_3 → conflicting on dataitem X

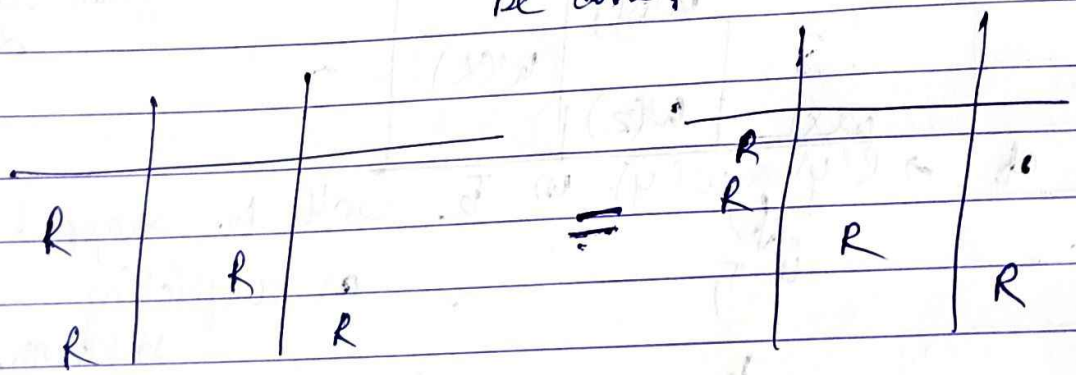
R(X) W(X)

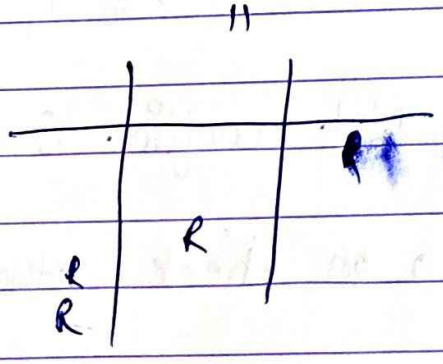
T_2, T_3 → conflicting on dataitem Z.

~~Y ka no conflict → as T2 waiting~~

T_1, T_2 → conflicting on Y.

→ If there are conflicting statements, then no swap can be done!





as they are not conflicting statements.

WOMK ☺ ♡♡.

any of following combo possible :-

$T_1\ T_2\ T_3$
$T_1\ T_3\ T_2$
$T_2\ T_3\ T_1$
$T_2\ T_1\ T_3$
$T_3\ T_1\ T_2$
$T_3\ T_2\ T_1$

to check for conflict serializability !!

R(y) in T_1, R(y) in T_2 can be swapped !!

T_1	T_2	T_3
R(x)		
R(y)		
	R(y)	R(z)
		W(z)
	W(y)	
		W(x)
W(x)	W(z)	

→ as conflicting, so no swap, bhaan hai when solve do !

ab → R(y) of T_1 & W(y) in T_2 can't be swapped as conflicting statements.

↳ so $T_1\ T_2\ T_3$ → combo not possible !!

So, Not conflict serializable !!!

→ so check other remaining combo !!

View Serializability :-

① If schedule is conflict serializable, then it is view serializable. (BUT CONVERSE NEED NOT TO BE TRUE)

3 cond^n's :-

① If in schedule S, Transaction T_i reads initial value of Q, then in schedule S' also T_i must read initial value of Q.

② If in schedule S T_i executes read (Q) & that value was produced by T_j, then in S' also T_i must read (Q) that was produced by same write(Q) operation of T_j.

③ The transaction that performs final write(Q) in S, same " must " " " " " S'.

→ so first read, last write must be same.

eg :-

Work @ 5:30 PM.

S ⇒	T_1	T_2	T_3
	R(Q)		
		W(Q) → blind write	
	W(Q)		
			W(Q)

S' banaya satisfying pichi 3 conditions.

S' ⇒	T_1	T_2	T_3
1st read →	R(Q)		
	W(Q)		
		W(Q)	
			W(Q) ← final write

can swap, as no read

So ~~2nd~~ 2nd cond^n dekhni ka no zaroorat.

↓

So View Serializable ✓

but not conflict "

⊙ Every view serializable schedule that is not conflict serializable has blind writes.

↓

writing w/o reading!!

↳ means over-writing.

Jaise deposit kiye gye → + nhi kiya, overwrite kr diya!!

WSMK ke sawal.

If reading wrong value → dirty read

T_1	T_2	T_3
R(Q)		
	W(Q)	
W(Q)		
↓ fails		R(Q) → dirty read ↳ as reading wrong value !!

Q.

T_1	T_2	T_3
R(X)		R(Z)
		W(Z)
	R(y)	
R(y)	W(y)	
	W(Z).	W(X)
W(X).		

→ so pehla read X ka T_1 krega, Y ka T_2 & Z ka T_3 hi krega!

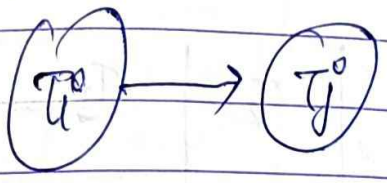
(S) →

(S') →

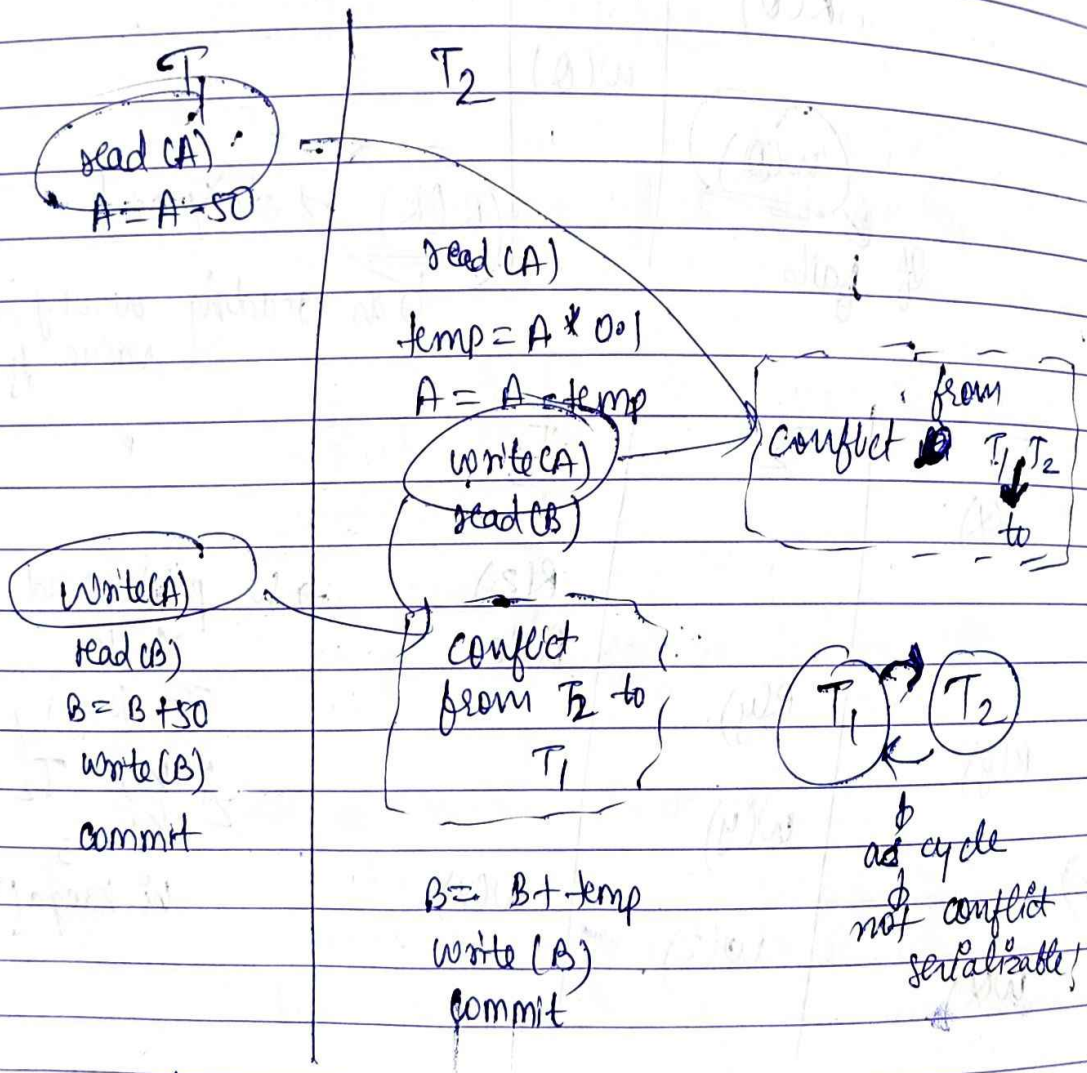
T_1	T_2	T_3
R(X)		
	W(Z)	
W(X)		

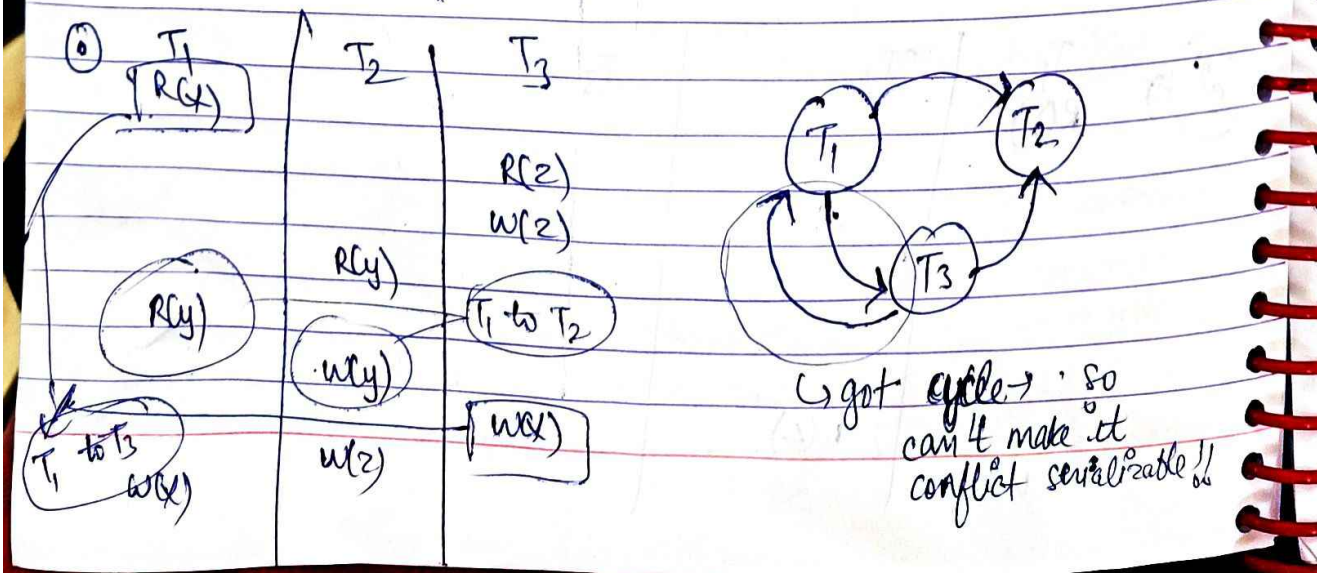
Work is so important.

Precedence graph → can be used to test serializable.



(If T_i, T_j conflicts)





WORK :- 1 MM.

$T_i \rightarrow T_j$

if :- any one holds :-

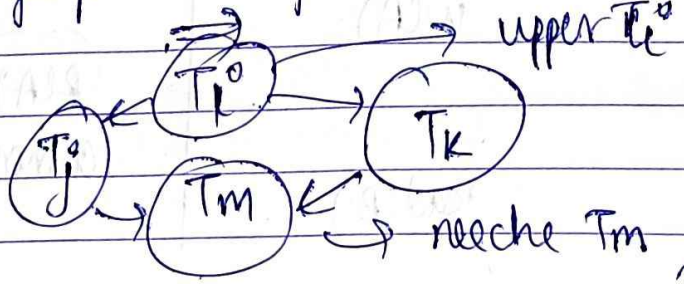
(i) T_i executes $W(Q)$ before T_j executes $R(Q)$

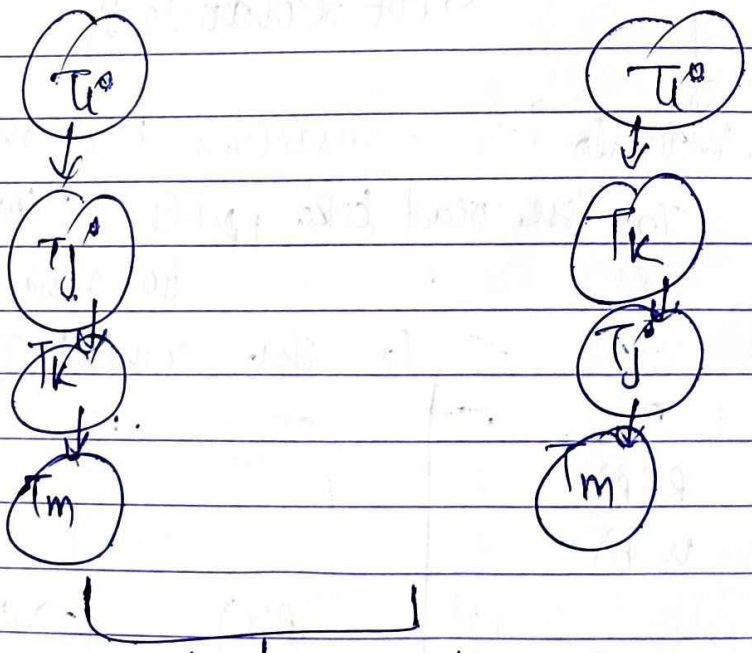
(ii) T_i —— $R(Q)$ —— $W(Q)$

(iii) —— $W(Q)$ —— $W(Q)$

combo bna ske the. hm → wo topological sorting → precedence graph se. kro ji.

if precedence graph is acyclic.





So, only these 2 combo are possible.

Recoverable Schedules :-

① Need to address effect of transaction failures on concurrently running transactions.

② If transaction T_j reads a data item previously written by T_i, then commit operation of T_i appears before commit operation of T_j.

T_1	T_2
R(A)	
W(A)	
	R(A) → can have dirty read!
	commit
read(B)	

↳ not recoverable !!

Maanlo iski transaction beech mein fail ho T_2 isse read kiya, pehle hi commit ho raha hai ⇒ So iska commit T_1 bhi bad ho.

T_1	T_2
R(A)	
W(A)	
	R(A)
R(B)	
commit	
	commit

⇒ at recoverable !!

WORK

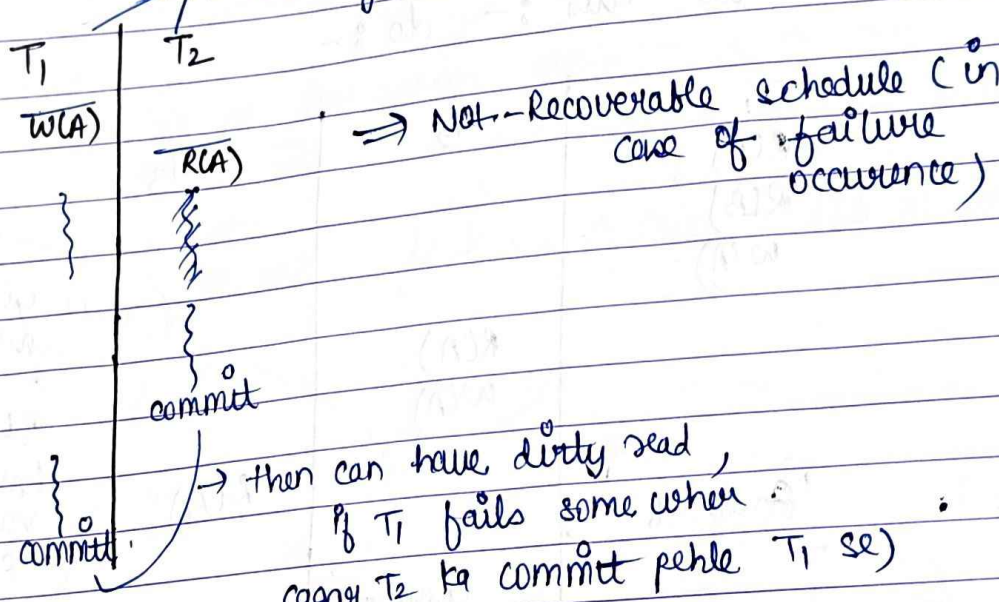
WORK

23 Feb 2024
Friday

lec-15

Recoverable Schedules :-

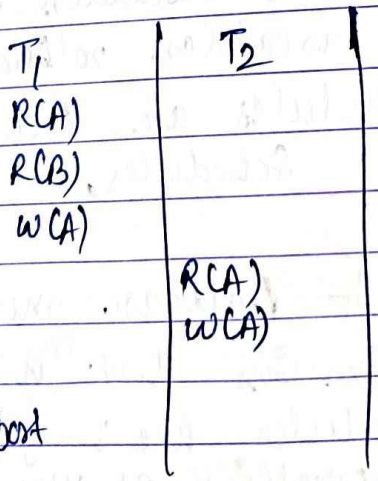
working on same data item



⇒ Not-Recoverable schedule (in case of failure occurence)

→ then can have dirty read, if T_1 fails some where. (agar T_2 ka commit pehle T_1 se)

Cascading Rollback :- a single transition failure leads to a series of transaction rollbacks.

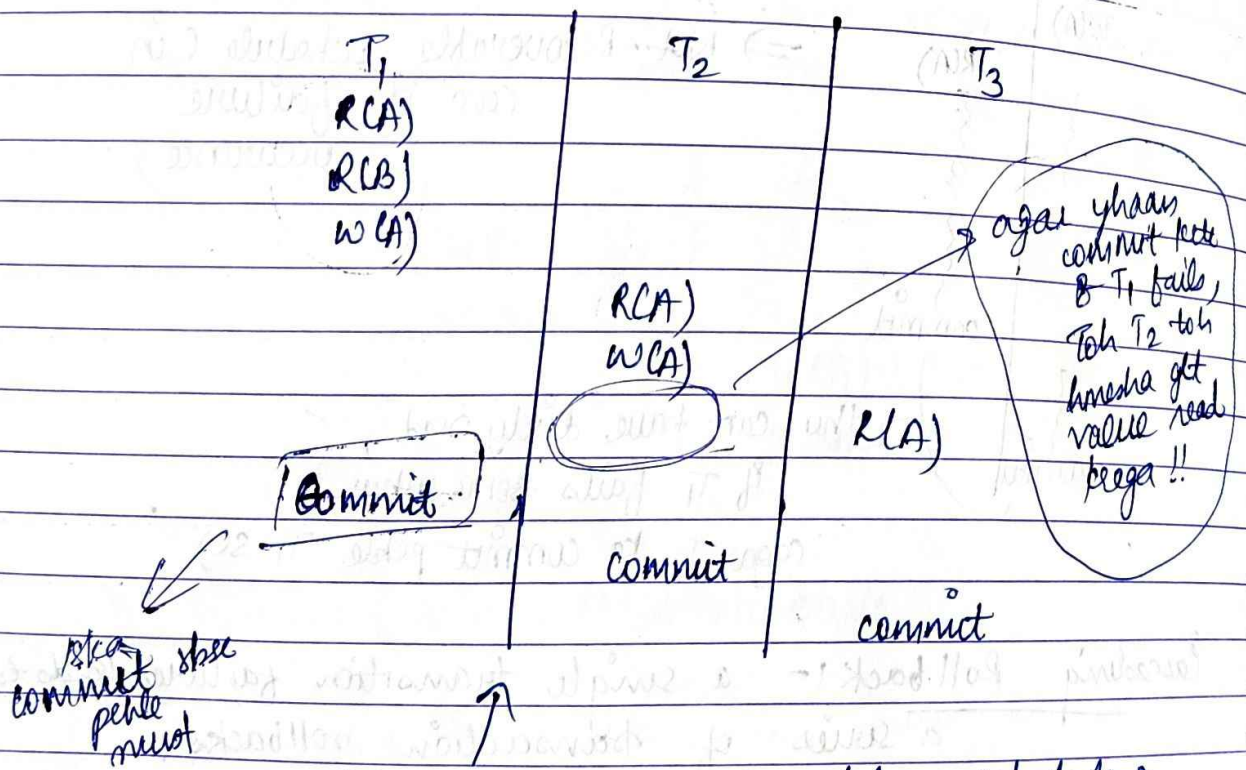


If T_1 fails, T_2, T_3 must be rolled back one by one.
→ can lead to undoing of significant amount of work.

T_1 fails

↳ T_2, T_3 → were reading value of A → so has to roll back to initial state → where database was in consistent state.

To avoid this :- do :-



Then schedule is cascadeless schedule → now no cascadeless rollback,

⓵ All cascadeless schedules are Recoverable schedules.

Concurrency Control :- Database must provide a mechanism that will ensure that all possible schedules are :-

1) either conflict or view serializable
2) either recoverable & preferably cascadeless.

WORK FROM HOME.

② Database will be in consistent state again schedule recoverable / cascadeless / conflict serializable / view serializable.

Goal :- develop concurrency control protocols that will assure serializability.

Implementation of Isolation levels :- (ch-7)

LOCKING :-
~~to get~~ lock on database v/s on items

TIMESTAMPS

multiple versions of each data item

CONCURRENCY CONTROL PROTOCOL :-

LOCKING :- (lock-based protocols)

Shared lock (S)	exclusive lock (X)
→ Data item can only be read	→ Data item can be both read as well as be written
→ requested using lock-S instruction	→ requested using lock-X instruction.
lock-S {Read}	lock-X {Read, write}

→ lock requests are made by concurrency-control manager.

Womk e

① Lock Compatibility matrix :-

	S	X
S	True	false
X	false	false

→ transactions both are requesting shared lock, means both want to read, then compatible, can allow !!

only, shared - shared lock is compatible.

T_1	T_2	T_3
R(A)		
	R(A)	
		R(A)

compatible ✓

locking protocol : set of rules followed by all transactions while requesting & releasing locks.

lock - X (B)

means : lock hogya, T_1 ne lock krdiya, jab tak unlock nahi krega, no other transaction can work on this data item B.

work ☺

T_1	T_2	concurrency-control manager
lock-X(B)		
	(can't write lock-X(B) → as not compatible)	grant-X(B, T_1) → grant ✓
Read(B)		
B := B-50		
W(B)		
Unlock(B)		
	lock-S(A)	
there ← lock-S(A) ✓ → which state can shared-lock compatible by more than 1 transaction)		grant-S(A, T_2)
	R(A)	
	Unlock(A)	
	lock-S(B)	
		grant-S(B, T_2)
	R(B)	
	Unlock(B)	
	display(A+B)	
lock-X(A)		
		grant-X(A, T_1)
R(A)		
A := A+50		
W(A)		
Unlock(A)		

→ locking protocol, enforce serializability by restricting set of possible schedules
↳ how?

Deadlock ?? (Imp) ★

WORK e
EXM.

T_1	T_2
lock-X(B)	
Read(B)	
B := B - 50	
W(B)	
	lock-S(A)
	R(A)
	lock-S(B) → can't assign it, until unlocked by T_1
lock-X(A)	

(To transaction/control manager wait/grant this request)

→ no one can make progress

T_2 has to wait until T_1 release its lock on B

T_1 has to wait until T_2 release its lock on A.

Such a situation is called (DEADLOCK)

to handle it one of T_1, T_2 must be rolled back & its locks released.

(agar lock-S(A) lata, then no conflicting lock → so deadlock)

Starvation → possible, if concurrency control manager is badly designed.

① same transaction is repeatedly rolled back due to deadlocks

② A transaction may be waiting for an X-lock on an item, while a sequence of other transactions request & are granted an S-lock on the same item.

→ toh yeh essi algo bnaaye, that this cond" can be avoided.

कालतंकु कारेचरित्त स्मार गरिजणु,
निरमुक्तेर कालेत्वग अगरिजणु ॥ मम.
wwmk $\frac{e}{3v}$ ♡m.

(•) R

A	B	C
1	2	1
2	2	2
3	3	2

→ R_1 (A (B)) ⟹

A	B
1	2
2	2
3	3

→ R_2 ((B) C) ⟹

B	C
2	1
2	2
3	2

atleast 1 must be common → bcoz agar kuch bhi common na ho, then can't join them agar koi query demand kre

Natural Join (R')

A	B	C
1	2	1
1	2	2
2	2	1
2	2	2
3	3	2

} → spurious tuples

as $R \neq R'$ ← (lossy)

↓ database inconsistent

→ problem bcoz common element should be CK or SK of either R_1 or R_2 or BOTH.

B → duplicates contain krta, can't be CK or SK.

↓

then lossless ✓

↳ cond^n s :- (1) $R_1 \cup R_2 \cup .. \cup R_K = R$

(2) $R_1 \cap R_2 \cap .. \cap R_K \neq \phi$

↓

must be CK or SK of R_1 or R_2 .. or R_K

(•) candidate key

↳ UNIQUE

primary → " + NOT NULL

↑

hm dete hain user ek hi hogi !!.

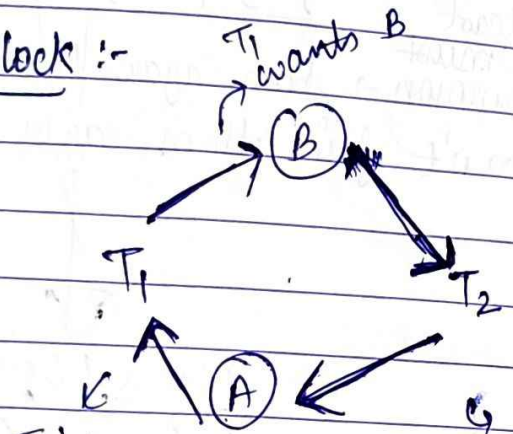
WOMK :)

WOMK :)

2 March 2024
Saturday

Lec-16 :-

① Deadlock :-



↳ no one releasing their acquired item.

↑

So, T_1 waiting for B, T_2 waiting for A for '∞' time.

↑

∞ time ke liye

↑

So situation == Deadlock,

No transaction able to proceed further.

Starvation :-

↑ When u r picking same transaction for everytime to kill to remove deadlock.

For eg: 10 transaction

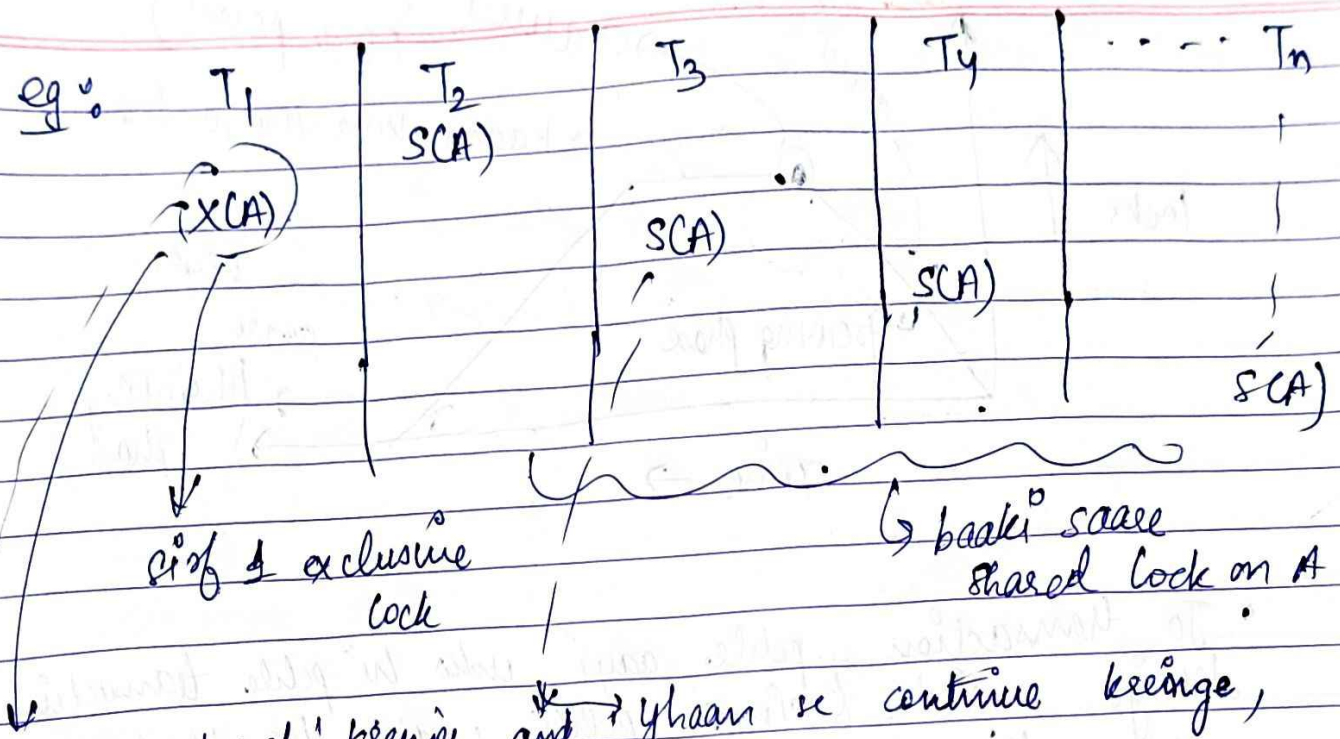
↑

sirf 2-3 mein deadlock → so pick those transactions and kill them!

↓

means rolled back, means that transaction has to restart.

Work in progress.



isko grant nahi krenge and → yhaan se continue krenge, rukenge nahi!

→ isse tum nahi aayegi ∴ T1 ne kiya, T3 ne le liya!

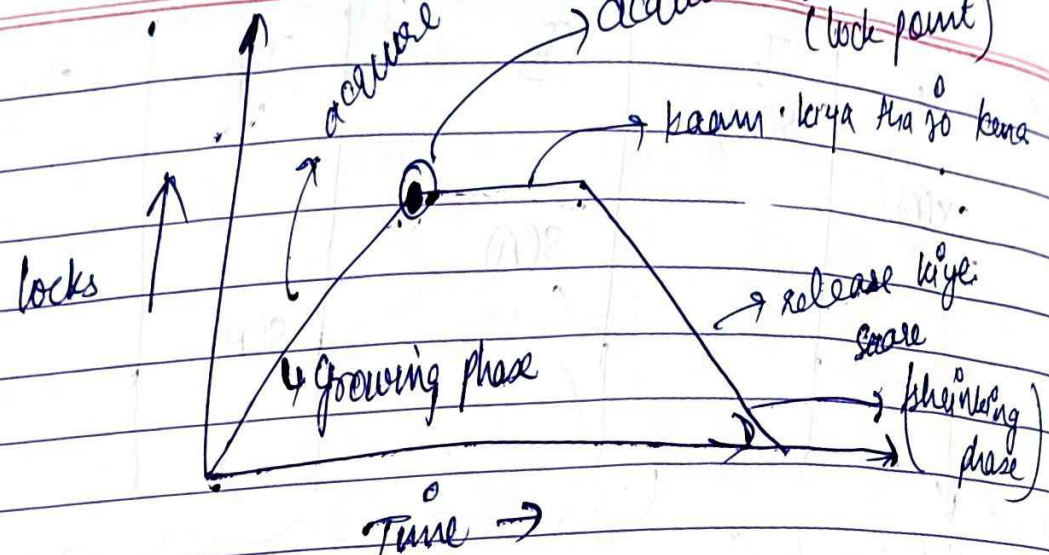
How to avoid ??

By Two phase locking protocol :-

(1) growing phase
- → Transaction may obtain locks
- → Transaction may not release lock.

(2) shrinking phase
- → transaction may release lock
- → transaction may not obtain locks.

- This protocol ensures conflict serializable schedules.
- " " " serializability
- → It can be proved that transactions can be serialized in order of their lock points (i.e. pt. where transaction acquired its final lock).



- Jo transaction pehle aayi, usko hi pehle transaction denge, wait krenge pehli wali ka kratam nahe ki, means T_1 k baad T_2 ko hi denge, ~~won't~~ will not give to T_3.

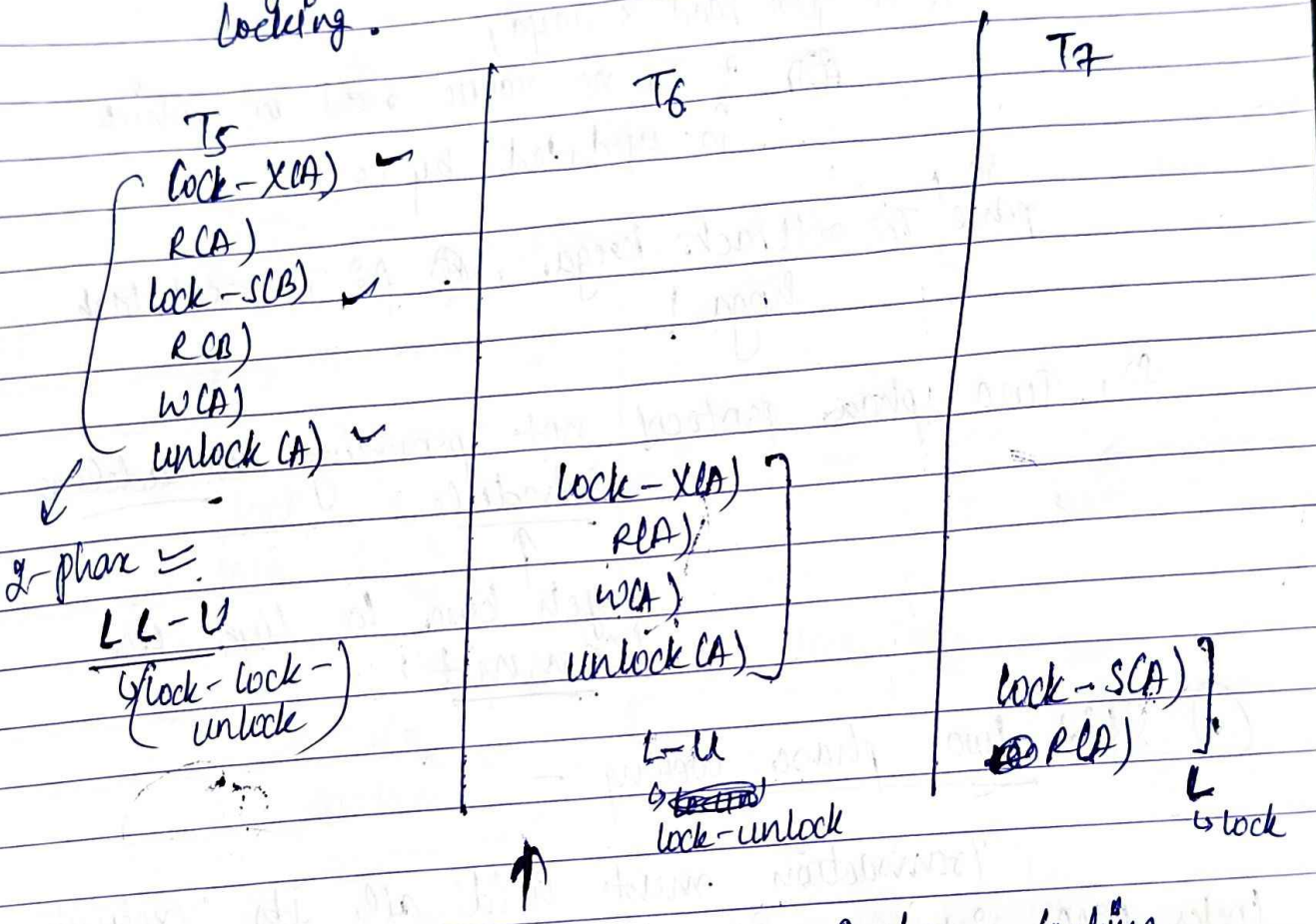
eg: Two phase protocol does not ensure freedom from deadlock.

T_3	T_4
lock-X(B)	
R(B)	
B := B - 50	
W(B)	
.	lock-S(A)
.	R(A)
lock-X(A)	lock-S(B)
.	

T_3, T_4 are 2 phase, but they are deadlocked.

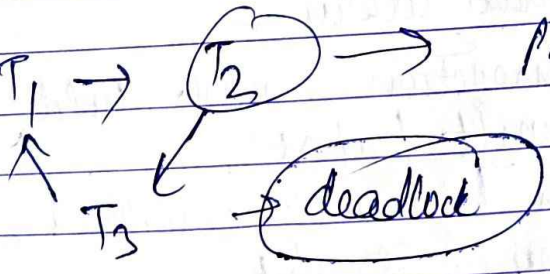
Work in progress.

- Cascading rollback may occur under 2- phase locking.



Each transaction observes 2 phase locking protocol, but failure of T_5 after R(A) step of T_7 leads to cascading roll back of T_6 and T_7.

If



Pick one & kill it, then, T_1, T_3 can utilize resources of T_2 & can complete their task.

bcoz T_5 is waiting for A which is being used by T_6 and T_7. Also if T_5 fails, so T_6 & T_7 ne uss data item par kaam kiya, jo wrong value may contain, so Data

WOMK 30

inconsistent hogya data!

and,
T5 ke just baad T6 aaya,
& T7 ne value read ki which is updated by T6.

so,
pehle T6 rollback krega, fir T7 roll back hoga!

So, Two-phase protocol not providing cascadeless schedule.
↑
yeh krne ke liye use 'commit'

① Strict two phase locking :-

Transaction must hold all its exclusive locks till it commits / aborts.
→ Ensures recoverability & avoids cascading rollbacks.

② Rigorous two phase locking :-

Transaction must hold all locks till commit / abort.
→ Transaction can be serialized in order in which they commit.

LOCKING PROTOCOLS :-

WORK

① Schedule S is legal under a locking protocol if it can be generated by set of transactions that follow a protocol.

T_8	T_9
lock-S(a_1)	
	lock-S(a_1)
lock-S(a_2)	
	lock-S(a_2)
lock-S(a_3)	
lock-S(a_4)	
	unlock(a_1)
	unlock(a_2)
lock-S(a_n)	
upgrade(a_1)	

T_8 : read(a_1)
read(a_2)
.
.
read(a_n)
write(a_1)

T_9 : read(a_1)
read(a_2)
display(a_1+a_2)

Will upgrade lock
S(a_1) → X(a_1)

T_5 → S(a)
X(a)

pehle yeh grant kro,
then release,
then yeh grant kro!
OR
Upgrade the lock.
(means S → X.

downgrade

X → S.

OR

pehle grant, fir release, fir grant.

⊙ Two phase locking protocol with lock conversions :-

Growing Phase :-

can acquire a lock-S on item

X

upgrade

Shrinking phase :-

can release lock-S

X

downgrade.

* This protocol ensures serializability.

⊙ Automatic Acquisition of locks :-

The operation read(D) is processed as :-

if T_i has a lock on D

then

read(D)

else begin

if necessary wait until no other transaction has a lock-X on D ~~grant~~

grant T_i as lock-S on D;

read(D)

end

WORK ☺ 3 com.

Transaction T_i issues standard read/write instruction, w/o explicit locking calls.

⊙ The operation write(D) :-

if T_i has a lock-X on D
then
write(D)
else begin
if necessary wait until no other transaction has any lock on D.
if T_i has a lock-S on D
then
upgrade lock on D to lock-X
else
grant T_i a lock-X on D
write(D)
end;

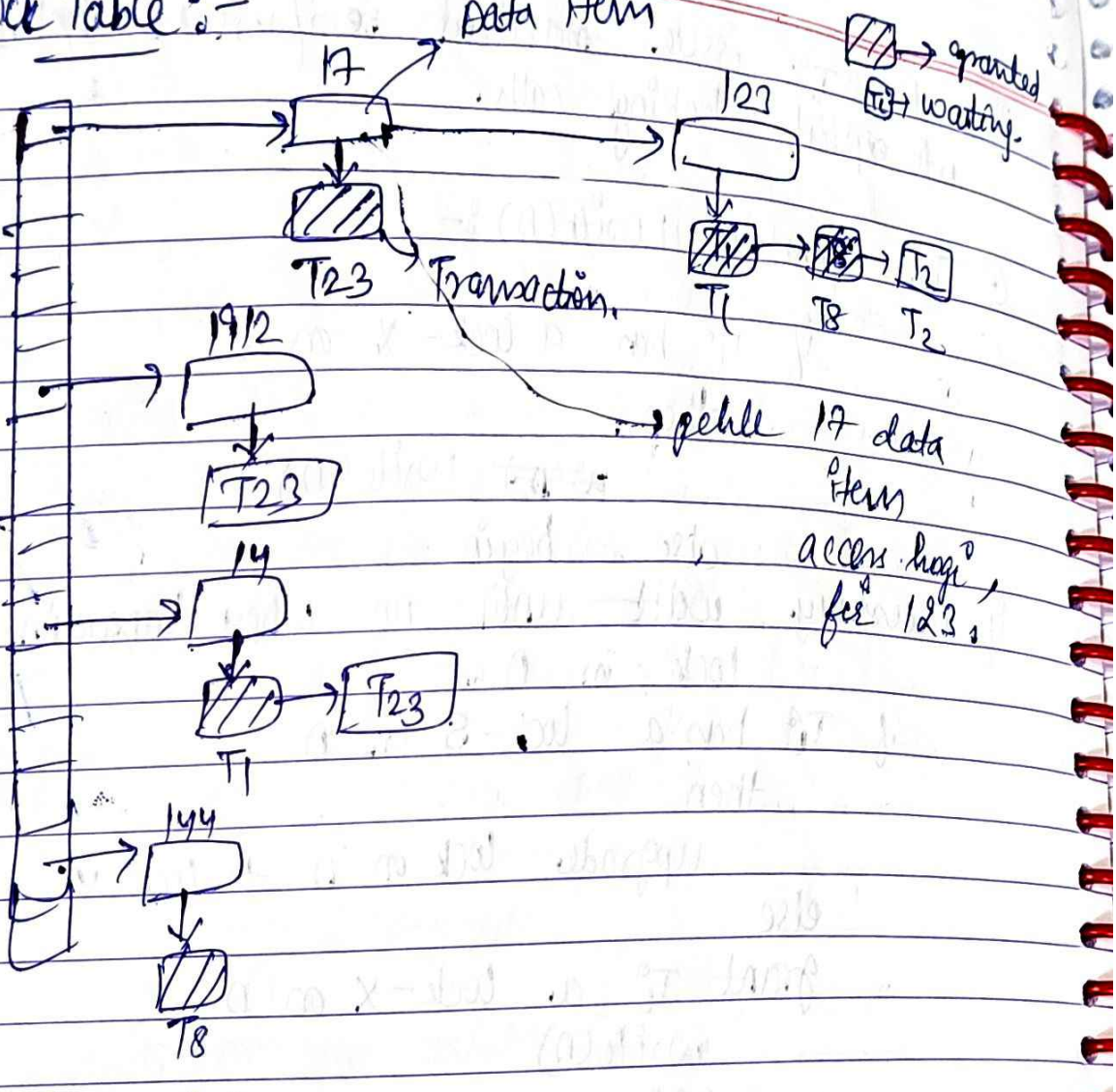
→ All locks are released after commit or abort, obviously.

⇒ all things are done by lock manager.
↑
can be implemented as a separate process.

- lock manager
  - ↳ can send lock or unlock requests as message.
  - ↳ They maintain Table → lock Table
    - ↳ to remember!

WOMK ©

Lock Table :-



(linked list)

⊙ lock-Tables also records type of lock granted or requested.

① New request is added to end of queue of requests for data item.

Graph - Based Protocols :-

→ alternative to two-phase locking.

WORK

① Impose partial ordering → on set $D = \{d_1, d_2, \ldots d_n\}$ of all data items. → partial ordering.

② If $d_i \rightarrow d_j$ then any transaction accessing both d_i and d_j must access d_i before accessing d_j.

③ Implies that set D may now be viewed as directly acyclic graph called database graph.

④ Tree protocol is simple kind of graph protocol.

$D = \{A, B\}$

A coming first

B coming Next.

So, $A \rightarrow B$ (Partial ordering b/w A, B).

ese partial ordering impose kana

again work of → concurrent control manager, in order to maintain consistency of database.

2-phase → Deadlock ☹
cascading rollback ☹

Tree → cascading rollback thoda kam!
↳ as pehle acquiring → fir releasing,

If ∃ circle ⇒ ∃ deadlock,

Tree never contain cycle ⇒ so never deadlock,

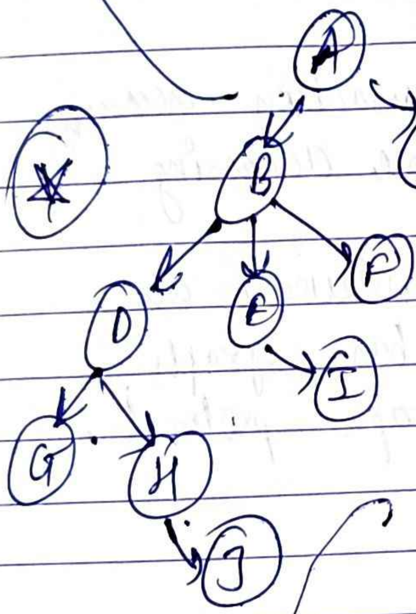
Tree protocol → no deadlock ☺

graph completely based on partial ordering!!

WOMK

A → B
A is parent of B, pehle A use karo, kaam khatam, then B use karo

- only exclusive locks are allowed.
  ↳ as shared mein toh problem hi nahi hoti, so don't consider them!



- first lock by T_i may be on any data item.
  Subsequently, a data Q can be locked by T_i only if parent of Q is currently locked by T_i.
- Data items may be unlocked at any item.
- Data item that has been locked & unlocked by T_i can not subsequently be relocked by T_i.

Case 1

T_1
lock-X(B)
lock-X(D)
lock-X(E)
unlock(B)
lock-X(I)
unlock(D)
unlock(E)
unlock(I)

Case 2

T_1
lock-X(A)
—— B
—— C
unlock(A)
lock-X(D)
unlock(B)
lock-X(G)
unlock(D)
unlock(G)
unlock(C)

for that particular schedule!!

↳ as unlock kiya, yeh soch kar ki future mein nahi chahiye, toh lock nahi kar sakta firse, for that particular schedule.

WORK ☺

⊙ A pehle access, fir B, fir D, fir H, fir J.

⊙ B or C mein se koi bhi access ke, sakte ho, bss dono ko, can be locked only if A is locked / used 1st.

Case 1 → B ✓ locked ✓
D can be locked as B (parent) hai ✓
E ✓
B unlocked ✓
I ✓
D unlocked ✓
E ——— ✓
I ——— ✓

↳ if graph ⇒ (B) → E, D → E, G, H, I, J

satisfy properties of Tree Protocol. ✓

not if previous graph (☆).

Case 2 →
A ✓
B ✓
C ✓
unlock A ✓
D ✓
unlock B ✓
G ✓
unlock D ✓
unlock G ✓
unlock C ✓

eg: want to access G, what sequence has to follow to reach G.
Direct nahi kr skte naa !!
↳ considering graph (★)

↳ so A → B → D → G.
↳ yehi prblm !
can't access directly !
extra access krna pdh raha hai, jab ki 2-phase mein nahi krna pdh raha tha → par tab deadlock situation ↑.

Tree protocol (graph-based protocol)
↳ time consuming
↳ ensures conflict serializability
↳ freedom from deadlock
↳ does not guarantee recoverability or cascade freedom
• need to introduce commit dependencies to ensure recoverability
↳ Transaction may have to lock data items that they do not access.
• increased locking overhead & additional waiting time
• potential decrease in concurrency

↳ Unlocking may occur earlier in tree-locking protocol than in 2 phase.
• shorter waiting times & ↑ in concurrency.

WORK FROM.

- Protocol is deadlock-free, no roll backs are required.

2-phase mein agar
↳ ek Transaction saare data items pehle lock kre, then it is possible ki 'baaki' Transaction wait krti rhegi → thus deadlock may occur.